Title: Assessing Multiple Software Process Changes Using Simulation and DEA

Course Title: Capstone
Course Number: EMGT 589
Instructor: Dr. Anderson
Term: Summer
Year: 1998
Author(s): Song Ji

ETM OFFICE USE ONLY
Report No.:
Type: Student Project
Note:

Abstract

Assessing Multiple Software Process Changes Using Simulation and DEA

S. Ji

EMP-P98078

EMGT589
SUMMER 1998

CAPSTONE PROJECT REPORT
ADVISOR: Dr. TIMOTHY R. ANDERSON

ASSESSING MULTIPLE SOFTWARE PROCESS CHANGES USING SIMULATION AND DEA

Song Ji

MAY 28, 1999

ASSESSING MULTIPLE SOFTWARE PROCESS CHANGES USING SIMULATION AND DEA

Abstract

In order to achieve better performance on quality, schedule and cost for a software project, many companies are seeking ways to improve their software processes by introducing process changes into their existing ones. The opportunities of process change can be many. Before implementing a potential process change, management needs to prove that the process change deserves a "go" decision through a cautious cost-benefit analysis.

One of the areas which process changes can be introduced into is inspection process. Management is interested in the following questions:

1. How much benefit can we get from a formal inspection?
2. If we skip the inspection in a situation that meeting schedule is so compelling, what'll be the consequences?
3. What if we conduct a "compact" inspection instead of a "full" inspection?
4. Is it wise to implement multiple inspection changes in a same project?
5. How much do those process changes cost us?

All the above questions are to be addressed with respect to the three key dimensions of a software project, quality, schedule and cost which are designated by remaining errors, project life cycle duration, and life cycle effort in this study.

In this study, we identified a set of combinations of multiple inspection process changes within a waterfall process. Based on an early research, we modified an existing simulation model and adapted it to this study. We conducted the experiments using this modified model, obtained the performance measures for each combination which is a process candidate each, and then post-processed all process candidates including the AS-IS case via Data Envelopment Analysis (DEA).

The first DEA model is a basic DEA model, which identifies a set of potentially best process candidates. The second model is a weights-restricted model. The weights limitation ranges were determined based on the piece-wise utility functions in an early research. To enhance the results of the second model, a third model which includes the standard deviations of the input variables was developed. This model, extending from the first two ones, can further provide more useful information about the risks associated with each process candidate.

The basic DEA model identifies FFFF, CCCC and FCFF as efficient processes. Later in DEA analyses incorporated weights restriction and/or input variable standard deviations, FFFF outshine all other process candidates. CCCC, even provides excellent result on development effort, but rated as inefficient after a thorough analysis when compared to FFFF.

The main contributions of this study lie in the following areas:

- To present a viable approach by which simulation and DEA are combined to assess a large size process improvement problem;
- To propose a new procedure to elicit weights for DEA model;
- By incorporating standard deviations of each input variable into DEA model, it is enabled to provide more useful information about the risk associated with each process candidate.

Future improvements would be done on the simulation model and a few DEA issues.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Literature Review

The software industry is experiencing increasingly large pressures on developing products with higher quality, lower price during a shorter cycle time. More and more companies are driven by competition to achieve better performance on software development projects, which have been notoriously of poor quality, large overruns on cost and schedule for decades. Besides adopting new technology, the companies are seeking ways to operate more efficiently and effectively, so as to achieve their project goals. In the courses of pursuing higher levels of Capability Maturity Model (CMM) (Paulk 1993), the companies are investing much more than ever on software process improvements.

In order to improve software processes, a software project manager may be interested in a number of questions such as (Raffo 1997):

- What development phases are essential? Which phases could be skipped or significantly minimized in order to reduce costs or cycle without sacrificing quality?
- Are inspections worthwhile?
- What is the value of applying tools to support development activities? Will these tools be worth the cost?
- How do we predict the benefit associated with implementing a new process before a substantial commitment of resources and effort is made?
- Given a number of potential process alternatives, how do we prioritize them for implementation?

To answer these questions, quantitative models which deal with process level issues are required. The purpose of this research is to provide a new set of quantitative tools to the software industry for helping them evaluate process improvements.

Raffo (Raffo 1997) divides the quantitative models used to predict software project performance into three classes: Analytic Summary Models, Analytic Structural Models, and Process Models.

Representative examples of Analytic Summary Models (defined in (Kellner 1988)) are COCOMO I (Boehm 1981), and SLIM (Putnam 1980). This class of models consist of mathematical relationships between selected model variables. These models focuses on high-level quantitative relationships between input and output factors of the software development process. Details of the process are usually considered to be a "black box" and are not explicitly considered. Typical output factors include manpower requirements, and schedule data. Typical input factors include: estimated project size (lines of code), personnel capability, experience with the development language and environment, development environment volatility, and availability of software tools. Constraints on the product or process such as execution time, requirement reliability, schedule, and required database size are also captured.

As described above, the Analytic Summary Models view the software process as a "black box" that is not specified in detail. As a result, these models do not capture many important details which are related to process alternatives.

Analytic Structural Models capture the interrelations, dependencies, and structure of software development at a deeper and more detailed level than Analytic Summary Models. If the

latter consider the software process as a "black box", then the former consider the process as "several black boxes making up a system". Analytic Structural Models can be quite rich in the system dynamics, interactions, and structural relationships captured, as shown in a representative research (Abdel-Hamid and Madnick 1991). However, few process details are included. Using system dynamics approach, Madachy's work (Madachy 1994) and other work by Tvedt and Collofello (Tvedt 1995; Tvedt and Collofello 1995) have extended the work of Abdel-Hamid (Abdel-Hamid and Madnick 1991) to include process steps in their continuous simulation framework. The work (Tvedt 1995; Tvedt and Collofello 1995) model a detailed inspection process. Madachy [Madachy, December 1994 #6] uses system dynamics to model the effect of performing formal inspections. The cost, schedule and quality impacts were compared for varying degrees of conducting inspections throughout the life cycle. It modeled the interested flows of tasks, errors, and personnel throughout different phases, detailed inspection activities, and the model was calibrated to industrial data. The strength and weakness of the above extensions (Madachy 1994; Tvedt 1995; Tvedt and Collofello 1995) will be discussed further after we describe the last class of predictive models, Process Models.

Process Models support process improvement by providing operational guidance regarding the critical sequence of process steps, information flows, and organizational responsibilities. In addition, these models have the capability to check the integrity of the process(Dieters and Gruhn 1991). These models are distinct from analytic summary and analytic structural models because they delve into the details of the process used to develop software (Dieters and Gruhn 1991; Kellner 1991; Lehman 1991; Raffo 1996). Beyond developing methods for modeling software processes qualitatively, Kellner's work (Kellner 1991) shows how process models could be used to support management planning and control activities, and how process issues are quantitatively analyzed and assessed. Later, Raffo (Raffo 1996) describes a quantitative approach, Process Trade-off Analysis Method, to process improvement. This research contributed many substantial extensions and improvements to the techniques originally reported in (Kellner 1991).

Raffo's research (Raffo 1996) is a fundamental research in quantitative modeling approach for software process improvement. In his dissertation, Raffo develops the Process Tradeoff Analysis Method which is a systematic and quantitative approach for analyzing the impact of process improvements *a priori*. PTAM is composed of two phases - a modeling phase and an analysis phase. The purpose of the modeling phase is to predict the performance of software development process improvements. In Raffo's dissertation, he shows how state-based Process Models can be used to predict the performance of software development projects in terms of the three critical performance measures of development cost, project schedule, and product quality. The purpose of the analysis phase is to take the performance output generated by the predictive models and use it to make a rationale decision about which process improvement to implement. In Raffo's dissertation, he used utility functions and financial measures such as return on investment to justify the selection of one process change. Raffo's work provides a foundation by developing an approach and technique for comparing alternative processes based on estimated quantitative performance from predictive software process models.

Both Analytic Summary Models and Analytic Structural Models are often large grained models that do not capture many of the significant process intricacies and junctures that can cause a project to go out of control (e.g. code requiring more testing loops than anticipated etc.). Consequently, most analytic summary models and structural models do not capture the details which can support specification and analysis of process alternatives.

Earlier we stated that the system dynamics framework (belong to analytic structural model) is extended to capture detailed process steps as shown in the researches (Madachy 1994; Tvedt

1995; Tvedt and Collofello 1995). These work utilize system dynamics approach. System dynamics is essentially the continuous simulation approach in software engineering context. Its distinct advantages appear to be primarily related to capturing project level or "systems" related aspects of the project. Many of these higher-level issues and aspects seem to lend themselves most naturally to a continuous simulation approach. Some examples of these issues include: the systems treatment of productivity and learning curve, the effects of schedule pressure, loses due to communication and motivation, willingness to change workforce level, among others.

However, at the detailed process level, tasks and activities are more naturally broken out into distinct discrete units. This is because modern software is modularized and the cycle time for these modules is in days rather than minutes. And certain important quantities (e.g. number of errors injected, detected, or fixed) take on discrete, not continuous, values. As a result, it seems that further work is necessary to discern the usefulness and validity of extending the continuous modeling approach to such a detailed level. At this level of process detail, it seems unclear whether the continuous assumptions which are at the foundation of this approach would be satisfied. Moreover, when using continuous simulation approach, it can be cumbersome and difficult to include stochastic random variables which are needed to capture the uncertainty and risk associated with actual software development projects. But the risk consideration is important for many software managers as they evaluate process alternatives.

In this study, our goal is to develop a new set of quantitative tools to assess a number of process proposals for improvement. Here, by process proposal we mean a complete software life cycle process configuration. Even though the specific process changes proposed in one process proposal might be local, it is expected the introduced changes would have non-trivial impacts on the overall performance of the whole software development process. Hence, we would prefer to evaluating each process proposal on a level of a complete life cycle process. Thus, in order to prioritize a set of process proposals, it's desired to evaluate each process proposal in terms of project cost (e.g. effort in person hours consumed in the whole project, including process change implementation cost), product quality (e.g. the number of remaining errors when the product is released to customer), and project duration (e.g. duration in total hours of the project), which are most commonly used performance dimensions of a software project. Rather than treating the process characteristics "statically" and obtaining deterministic estimates of these performance measures, we prefer to treating the process characteristics stochastically and obtaining stochastic performance measures.

The aforementioned Analytic Summary Models cannot be used to achieve our goal stated above because this type of models are not able to handle process level issues and consequently not able to assess process alternatives. Among Analytic Structural Models, even the enhanced system dynamics models (Madachy 1994; Tvedt 1995; Tvedt and Collofello 1995) deal with process level issues, its weaknesses have been described before as: i) continuous treatment of variables with discrete nature (e.g. number of errors, and other events with discrete nature in a software project); ii) difficulty in handling stochastic variables, which is of great importance for a software project manager. Therefore, we prefer to discrete simulation in modeling software processes in this study. The state-based process simulation approach, reported in (Kellner 1991; Raffo 1996), is discrete approach in nature. Being aware that a software process usually involves important issues that are naturally discrete yet others that are continuous, some pioneer research (Kocaoglu 1998) has been done to propose an integration of both techniques within a single model, provided the availability of such simulation tool, i.e. *Extend*.

In a recent work, Raffo et al. (Raffo 1998) present research utilizing state-based stochastic simulation models of a large-scale, real-world, software development process. The goal of the

work was to provide a quantitative analysis of proposed process change alternatives in terms of key performance measures of cost, quality, and schedule. Their model also supports a quantitative assessment of risk or uncertainty associated with each alternative. The approach is viewed as being key to successfully achieving higher level CMM practices related to Quantitative Process Management and Software Quality Management (CMM Level 4) as well as Process and Technology Change Management (CMM Level 5).

Other researches that have been done recently in software process improvement arena with different focuses (Christie 1998; Pfahl 1998; Powell 1998; Raffo 1998; Scacchi 1998). In Pfahl and Lebsanft's research (Pfahl 1998), they present a new approach, IMMS (Integrated Measurement, Modeling and Simulation) and this approach helps to address the need for a well defined and repeatable procedure for generating and using information based on experience and empirical evidence. Powell et al (Powell 1998) presents a method that utilizes a system dynamics simulation model to evaluate the use of advanced software life-cycle techniques for accelerating software development schedules. The above researches reflect some new applications in software improvement arena. As Christie (Christie 1998) pointed out, simulation can be used to support process improvement at all five levels of the CMM. As process maturity improves, simulation is increasingly tied to operational metrics, to validate one's understanding of process behavior and improve the simulation's predictive power, and this predictive power allows new processes to be tested off-line and introduced with much greater confidence.

The aforementioned PTA Method in (Raffo 1996) is an effective framework for addressing process improvement problems and will be used as an overarching method for this study. As described earlier, the purpose of the analysis phase of PTAM is to take the performance output generated by the predictive models and use it to make a rationale decision about which process improvement to implement. In Raffo's dissertation, utility functions and financial measures such as return on investment are used to justify the selection of one process change. In this work, we show how an alternative approach, DEA, can be used to efficiently evaluate a set of process proposals. We also compare this evaluation to the utility function calculations and identify where these analyses provide different recommendations and discuss these implications.

Data Envelopment Analysis (DEA), as a powerful tool for dealing with multi-attribute benchmarking problem, provides a potential alternative for the analysis technique within the above PTA Method. DEA, initially developed by Charnes et al, (Charnes, Cooper et al. 1978) is intended to assess the efficiency of a set of homogeneous units (Decision Making Unit, in the DEA terminology) by comparing each DMU against a frontier which is constructed within the same set of DMUs via linear programming technique, with respect to the levels of inputs consumed (less is desired) and the levels of outputs produced (more is desired).

When DEA is constructed in a ratio format, the definition of efficiency is specified as the ratio of the weighted sum of outputs over the weighted sum of inputs of a DMU. A linear programming model is solved for each assessed DMU which seeks to derive weights for the inputs and outputs that would maximize its efficiency (Allen, Athanassopoulos et al. 1997). DEA in its purest form (Charnes, Cooper et al. 1978) allows total flexibility in the selection of weights such that each DMU will achieve the maximum efficiency rating. However, such unbounded flexibility may prove to be flawed in many real world applications since some weighting schemes identified by the above purest DEA can be unacceptable. Thus the weight-restricted DEA model is flourished to address this problem.

In this study, we found that a software development process fits naturally a production process which is the object of common DEA study. A software process can be viewed as a

production process which consumes certain resources(inputs) and generate certain outcomes(outputs). Regarding to software process resources, it's natural to consider manpower and other expenses consumed by the project as inputs. The volume (lines of code or number of function points) and quality (can be measured by e.g. the number of errors remaining in the product) of the final software product can be viewed as outputs of the process. A few researches (Banker and Kemerer 1989; Banker, Datar et al. 1991; Elam 1991) have been done on applying DEA to evaluate the efficiency of software production. The reader is referred to (Paradi, Reese et al. 1997) for a review on different software production models which represent different selection of inputs/outputs.

The determination of an appropriate software production model depends on the purpose of DEA analysis and the availability of data. In this study, the purpose of DEA study is to evaluate the efficiency of each process proposal by taking account of project cost (i.e. effort in person hours consumed in the whole project, including process change implementation cost), product quality (i.e. the number of remaining errors when the product is released to customer), and project duration (i.e. duration in total hours of the project). From a DEA perspective, all the above three measures are "bads" for a software process, which means the less they are, the more desired the process is. Rather than treating the project cost as production model input but quality and duration as outputs as done in (Paradi, Reese et al. 1997), we treat all three performance measures as model inputs. In doing so, we avoid the necessary transformation for quality and duration when they are treated as model outputs, without losing any useful information for this study.

Another issue of DEA modeling is how to incorporating value judgements of the management about the relative importance of the three performance measures in assessing the efficiency for each process proposal. The user of the DEA analysis, in this case a software project manager, needs to decided how s/he is willing to trade-off among the three performance measures and then these value judgements need to be incorporated into DEA analysis. Given that the unbounded flexibility of the selection of weights in a "purest form" DEA (Charnes, Cooper et al. 1978) is unrealistic, it's necessary to develop the DEA models with weight restrictions. The details of the development of DEA models used in this study are presented later in Chapter 3.

The three performance measures for each process proposal can be obtained from state-based process simulation reported in (Raffo 1996). In this study, we modified the original model developed in (Raffo 1996) and adapt it according to the purpose of this study. The mean, standard deviation, minimum, and maximum values of each performance measures for each process candidate can be obtained from simulation experiments. For DEA analysis, the mean values are used as model inputs to assess the efficiency of each process candidate in the first stage. In the second stage analysis, value judgements carried in the utility function developed in (Raffo 1996) are used to determine the weight restrictions in DEA model. In the third stage analysis, the standard deviation of each performance measure for each process candidate is further included in DEA inputs so as to assess the uncertainty associated with each process candidate.

Based on our earlier discussion about analytic summary models and analytic structural models, it is clear that both these type of quantitative models wouldn't support the need of generating performance measures data in the first phase of PTAM. Even though the enhanced analytic structural models reported in (Madachy 1994; Tvedt 1995; Tvedt and Collofello 1995) can provide the performance measures data for each process candidate, the difficulty in handling discrete and stochastic variables make this type of models not appropriate for this study.

DEA has been widely applied to solve the benchmarking problems in a variety of areas (Seiford 1997). Among the large amount of DEA literature, there is an intersection on simulation

and DEA. A number of researches used simulation techniques to assist the evaluation of the goodness of certain DEA modeling. To name a few, they include (Orme and Smith 1996; Anderson and Sharp 1997; Smith 1997). Meanwhile, a few researches are found on the real world applications of using DEA to benchmark the DMUs whose performance measures are generated from simulation experiments (Shafer and Bradford 1995; Shang and Sueyoshi 1995; Chang, Sueyoshi et al. 1996; Talluri and Sarkis 1997).

Shafer et al (Shafer and Bradford 1995) illustrates how DEA can be used to measure the efficiency of alternative machine component grouping solutions which are generated from computer simulation, given the existence of multiple inputs and/or outputs in the transformation process with no *a priori* weightings. This study shows that DEA can be a viable tool to evaluate the simulation results corresponding to alternate plant layouts. Talluri et al (Talluri and Sarkis 1997) presents an extension for Shafer's study by applying cross evaluation.

In Chang's research (Chang, Sueyoshi et al. 1996), DEA is used to measure multiple performance criteria for 42 dispatching rules in a job shop environment for the purpose of production scheduling. Seven performance measures are considered in the evaluation. Without pre-assigning weights to any performance measure, DEA evaluates the efficiency of each dispatching rule relative to the other rules. After running a large number of experiments, their research results show that 2 extreme subgroups of dispatching rules perform consistently.

Shang (Shang and Sueyoshi 1995) proposed a unified framework to facilitate decision making in the design and planning of the most appropriate flexible manufacturing system (FMS) for a manufacturing organization. In his research, an AHP model examines the nonmonetary criteria associated with corporate goals and long-term objectives, while a simulation model is employed to analyze the tangible benefits. Both AHP and simulation models are used to generate the necessary outputs for the DEA, whereas the accounting procedure determines the required inputs, such as expenditures and resources, for realizing the potential benefits.

This research differentiates itself from the above researches because i) this research is being applied to the area of software engineering; ii) we are using weight-restricted DEA model to assess total 82 process candidates; and iii) the incorporation of standard deviations of performance measures provides useful information about the uncertainty associated with each process candidate, which is of value on rationale decision making. Also, we present a procedure for DEA weight elicitation in this context. In section 3.1, we will further discuss a variety of researches on DEA with weight restrictions.

In this study, we use a state-based process model to generate a set of process change alternatives, and then use a weight-restricted DEA model to assess the efficiencies of the set of process alternatives. In doing so, we are trying to provide a new set of quantitative tools in process improvement arena. Finally, the DEA model developed in this study shows potentials in fitting the need of the PTAM by providing a viable, easy-to-use, and informative analysis tool. DEA's advantages on handling such a large size, multi-dimensional benchmarking problem are apparent. Currently, its weakness appears to be that the DEA analyses done in this study still have difficulty in interpreting the benchmarking results with statistical confidence, given that all the performance measures obtained from simulation experiments are stochastic. As an extreme point technique, DEA is handy when only mean values of performance measures are taken account of in this context.

Chance constrained DEA techniques are developed to attack the problem of dealing with the stochastic type data. The reader is referred to (Sengupta 1987; Cooper, Huang et al. 1993; Olesen

and Petersen 1995) for a few representative researches on chance constrained DEA. To further improve the DEA analysis, developing chance constrained & weight restricted DEA models appears to be the right direction. To our best knowledge, the integration of chance constraint and weight restriction approaches has not been reported yet. The incorporation of standard deviation shown in this study is our initial effort to address the problem of handling stochastic variables. Future effort on incorporating statistical methods may help to enhance the current analyses, too.

1.2 The Research Questions

One of the interesting areas which process changes can be introduced into is inspection process. Madachy (Madachy 1994) already used system dynamics to model the effect of performing formal inspections and demonstrate promises of using simulation in this context.

Usually, management is interested in the following questions:

1. How much benefit can we get from a formal inspection?
2. If we skip the inspection in a situation that meeting schedule is so compelling, what'll be the consequences?
3. What if we conduct a "compact" inspection instead of a "full" inspection?
4. Is it wise to implement multiple process changes in a same project?
5. How much do those process changes cost us?

All the above questions are desired to be addressed quantitatively with respect to the three key dimensions of a software project — quality, schedule and cost.

Here, a formal inspection is similar to a so-called Fagan inspection (Fagan 1976) (Fagan 1986). But a "compact" inspection is different from a *walkthrough* described in (Fagan 1976) (Fagan 1986). In our study, we re-defined the formal inspection ("Full" inspection) and the "Compact" inspection in Chapter 2 based on our process models. In our study, a "Compact" inspection is a well-defined and repeatable inspection process.

Regarding to the issues raised above, the companies diversify their strategies to a large extent in practices. Potentially, if process improvements are in order, there are a number of proposals on the inspection management. In this study, we are still using PTA Method but trying to use a new benchmarking tool within PTAM on assessing a large number of process improvement proposals.

1.3 The Approach

A procedure of introducing process changes can be the following:

1. Identifying and defining potential process changes. If multiple process changes are potentially beneficial, a variety of combinations of process changes are identified and defined;
2. Experimental design. Designing a benchmarking scheme for all process-change candidates and the existing AS-IS process;
3. Developing a simulation model for the purpose of benchmarking. Collecting data from existing process and company processes history. Adopting reasonable assumptions; Verifying and validating the model.
4. Conducting experiments via the developed model;
5. Post-processing(benchmarking) the simulation results via benchmarking tools;
6. Analyzing benchmarking results. Sensitivity analysis. Recommending the best process

proposals;
7. Implementing proposed process change(s).

The above procedure is consistent with PTAM in (Raffo 1996). In this study, we identified a set of combinations of multiple inspection process changes within a waterfall process. Based on the early researches (Kellner 1991; Raffo 1996), we modified an existing simulation model and adapted it to this study. We conducted the simulation experiments for all combinations using this modified model and evaluated the whole set of combinations via Data Envelopment Analysis (DEA).

1.4 The Implications

The benefits from inspection have been appreciated in both academia and practices. But in practice there are still large variations regarding to inspection strategies during the companies' software processes. Some company have well-established inspection process others may mainly rely on unit tests, functional and system tests. Through this study, we'll be able to comprehensively research the impact of those different types of inspection policies (and their mixes) via the simulation model. The results from a simulation model can hardly be precise forecasting for the future, given the greater uncertainties inherent in our real world than in our model, but they do serve as a good foresight to assist rational decision-making. More importantly, adopting simulation for assistance in software project management will help a company continuously pursue higher CMM Levels' practice.

DEA possesses an inherent ability to handle large-size, multi-attribute decision- making problems. In this study, we use DEA as an evaluation tool for assessing the simulation results. The way we develop the weight-restriction and the DEA model incorporating standard deviations holds potentials for wider application.

1.5 Overview of this report

In this chapter we introduce the topic, approaches and implications of this study. Chapter 2 describes the process model used in this study, how the process changes are defined and simulation model is adapted and verified. The third section of Chapter 2 presents the results from simulation experiments. Chapter 3 describes the development of the DEA model and results of DEA analysis. The last section of Chapter 3 presents a proposed procedure for eliciting weights for DEA modeling. Chapter 4 compares the results from DEA and directly applying utility function on the same set of data, and discusses the sensitivity of DEA model to its input variable selection. Chapter 5 concludes this research.

Chapter 2 Process Model, Process Changes Definition and Simulation Experiments

In this chapter, we describe in section 2.1 that the source process models which are modified in this study and the performance measures that we are using to evaluate a software process. In section 2.2, we define the process change proposals analyzed in this study. In section 2.3, we present the results from simulation experiments based upon our design in section 2.2. And finally in section 2.4 we describe how we verified the modifications done on the source process models.

2.1 The Process Models and Process Performance Measures

2.1.1 The Process Models

Figure 2-1 shows the main process model SQL employed in this study. The backbone process studied essentially followed a Waterfall Process Model including the major phases of Functional Specification (FS), High Level Design (HLD), Low Level Design (LLD), Coding (Code), Unit Test (UT), Functional Verification Test (FVT), and System Verification Test (SVT). Inspections of Functional Specification, High Level Design, Low Level Design and Coding were also included. Field Support and Maintenance phase, and process segments for developing test plans and writing test cases for Functional Test and System Test were also included. More details about this model can be found in (Raffo 1996).

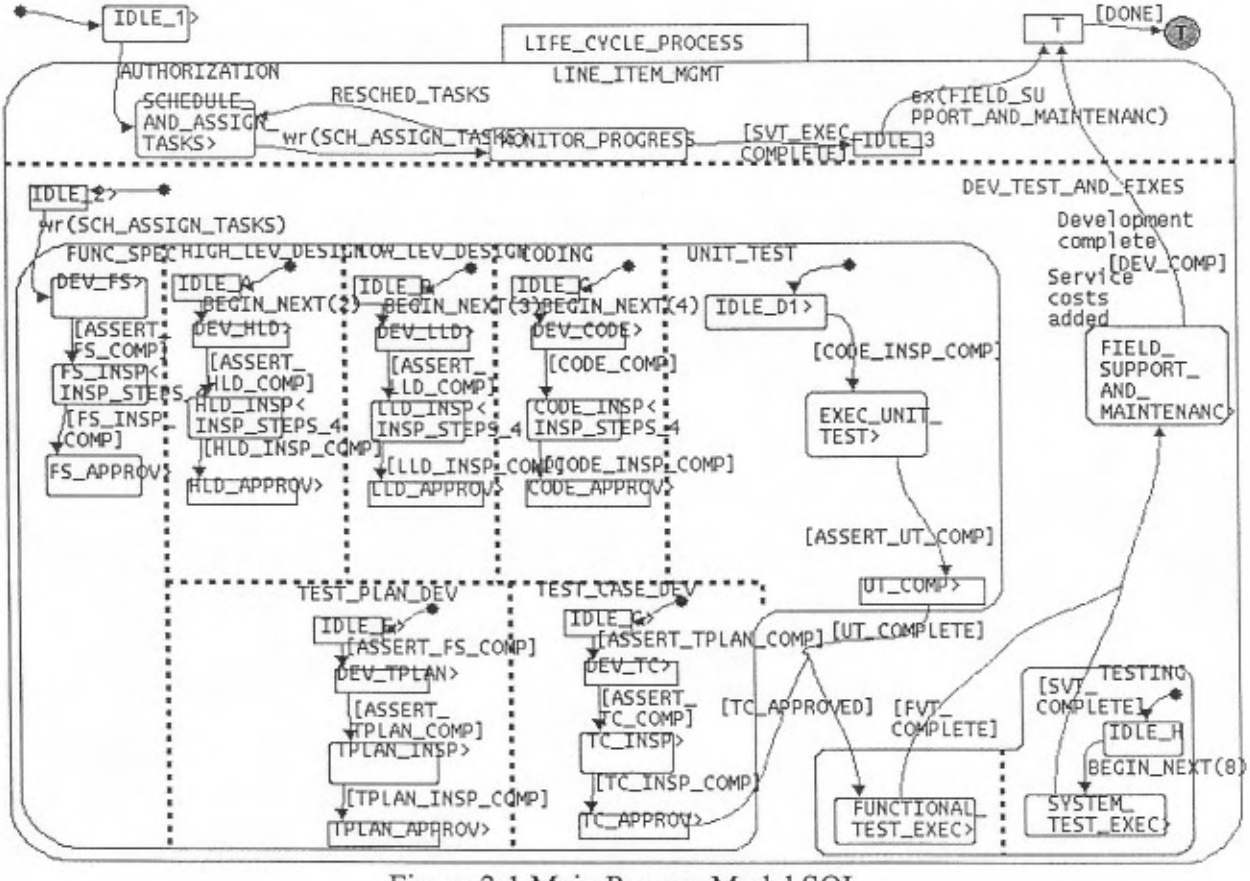

Figure 2-1 Main Process Model SQL

Figure 2-2 shows the inspection model INSP, which models the details of a "generic" inspection process and is the focus of our study. Here "generic" means the inspection model essentially captures all possible steps and issues raised in an inspection process for this particular business case, and allows a variety of ways of utilization. The inspection model is called on and utilized by the main model SQL during different phases of execution, namely Functional Specification, High Level Design, Low Level Design and Coding.

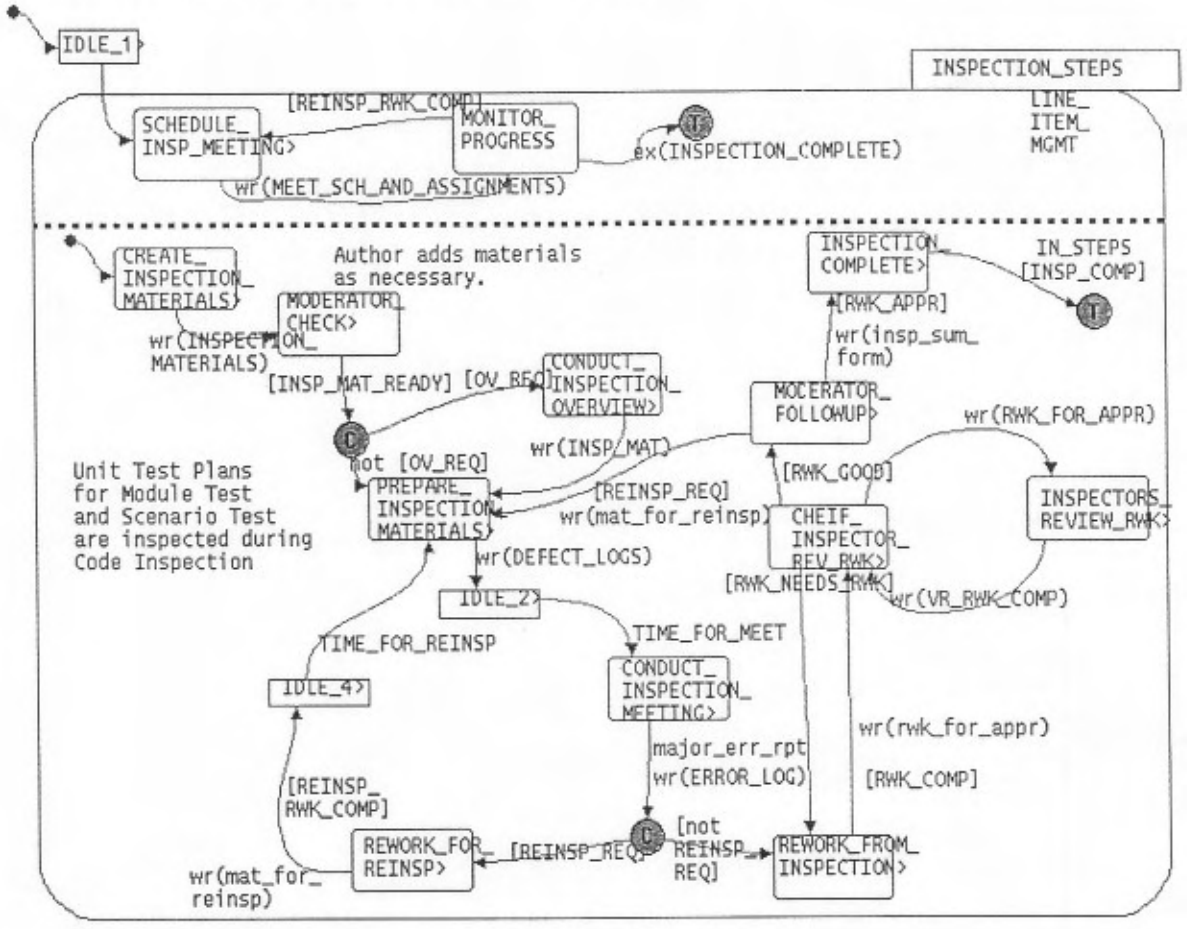

Figure 2-2 Inspection Model INSP

Following is a brief description of all the steps involved in the inspection model. Again more details can be found in (Raffo 1996).

1. The model is entered through the IDLE_1 step. This block is used to initialize the inspection model to the appropriate development phase. The state then immediately transitions into SCHEDULE_INSP_MEETING in the Line Item Management portion of the model and to the CREATE_INSPECTION_MATERIALS in the Inspection Steps portion of the model.
2. Inspection Materials are created quickly and the MODERATOR_CHECK step is then conducted.
3. If it is an inspection of the Functional Specifications, then an overview meeting of the functionality is called (CONDUCT_INSPECTION_OVERVIEW).
4. Once the inspection material has been distributed, inspectors prepare the material and create defect logs. This takes place during the PREPARE_INSPECTION_MATERIALS

step.

5. IDLE_2 is a holding step which captures the slack time between the PREPARE_INSPECTION_MATERIALS step and the CONDUCT_INSPECTION_MEETING.
6. During the CONDUCT_INSPECTION_MEETING step, inspectors meet to discuss the artifact being inspected. This step is a decision point for the inspection process. If the number of detected errors is too high, a reinspection is called.
7. If a reinspection is required, the process passes through a rework step REWORK_FOR_REINSPECTION where errors are corrected and injected, and an idle step before inspectors review the work again. The PREPARE_INSPECTION_MATERIALS state is then entered and the reinspection is begun.
8. If the material does not require a reinspection, the REWORK_FROM_INSPECTION state is entered. Here errors are corrected, new errors are injected, and some errors escape.
9. Follow-up from the inspection is handled by several people at the company, where this model is developed originally. Once the rework is complete, it is submitted to the Chief Inspector (CHIEF_INSPECTOR_REV_RWK). The Chief Inspector then distributes the rework to the various technical experts who were involved in the inspection meeting. They review the rework (INSPECTORS_REVIEW_RWK) and return it with their comments to the Chief Inspector.
10. If the rework is rejected, it is returned to the authors for additional work. The rework must then be reviewed again according to the steps described above.
11. If the rework is accepted, the error logs, rework, and reviews are submitted to the moderator (MODERATOR_FOLLOWUP) who approves that the inspection is complete and closes the record out of the management system. The inspection is now complete.

Note: During Low Level Design Inspection or Code Inspection phases, the moderator may call a reinspection of the material if a large previously undetected error surfaces which would cause significant rework to the functional specification or design. This is shown by the line connecting the MODERATOR_FOLLOWUP and the PREPARE_INSPECTION_MATERIALS. However this occurs very infrequently.

In our study, both the main model SQL and the inspection model INSP are to be modified to adapt to our research design. But the inspection model INSP is the one that will carry our major re-design. We'll explore the re-design in next section.

2.1.2 The Process Performance Measures

The number of remaining errors, the amount of staff effort in person hours, and the amount of duration in hours, are three key performance measures by which we used to evaluate a software process in this study. To be more specific, Remaining Error , Life Cycle Effort (and Total Effort), and Life Cycle Duration are the key measures output by the simulation model employed in this study and they will be treated as inputs for follow-up DEA analysis.

In our study, Remaining Error(RE) is the number of remaining errors or the number of defects which remain in the product and are passed to the customer. These defects generate field trouble reports. Developers from the project team are required to replicate and correct these defects as a preemptive priority in their work. The effort associated with correcting these defects is an essential component of field serve effort.

Life Cycle Effort(LEFF) is the staff effort used required to execute the Functional Specification through System Test operations of the process. Total Effort(TEFF) is Life Cycle Effort plus Field Service Effort.

Life Cycle Duration(DUR) is the task duration from the start of the Functional Specification operation through the completion of all phases of developing and testing. It does not include the amount of time spent correcting defects which were discovered after the product was released to the public.

We feel the error, effort, and duration capture the key dimensions of performance of a software process and comply with the theory and practice of project management in software engineering.

2.2 Defining Process Changes

In this section, we'll define the process changes that we proposed. The process changes including two aspects: configuration and set of process characteristics associated with each configuration.

2.2.1 Configuration Definition

As described in section 2.2, our research object is a waterfall process that consists of Functional Specification (FS), High Level Design (HLD), Low Level Design (LLD), Coding, Unit Test, Test Plan Design, Test Case Design, Functional Test, System Test, and Field Support & Maintenance. During the end of each phase among FS, HLD, LLD and Coding, there is an inspection sub-process.

Inspection and test processes are the vehicles by which the errors, injected during the design phases of FS, HLD, LLD and Coding as well as rework along the whole process, can be detected and corrected. Even though it is widely accepted among the academics that inspection plays a key role in achieving the targeted project goals on quality, schedule and cost, in practice the companies diversify their inspection policies in many ways. A company may conduct decent inspection process in each of the above four design phases where errors are mainly injected. Or a company may do a rush job on inspection and rely on unit test and other test processes to meet quality standards. Even a company may skip inspection during some particular software development phases while they conduct inspections during other phases.

In this study, we define three modes of inspection, which are "Full", "Compact" and "None" versions inspection. A "Full" inspection consists of more inspection steps including more rework, occupies more inspectors and has higher detection capability (a ratio designated that the fraction of errors detected out of the errors injected, which is probabilistic). A "Compact" inspection consists of moderate inspection steps which consumes less effort than "Full" inspection, occupies moderate number inspectors and has moderate detection capability. A "None" version inspection simply means no inspection at all. Then for each key design phase among FS, HLD, LLD and Coding, we have three options on its inspection mode: Full, Compact, or None. Therefore, we hypothesize 81 combinations of possible inspection mode for each key design phase. For example, a combination FCNC means a "F" inspection for FS phase, a "C" inspection for HLD phase, a "N" inspection for LLD phase, and a "C" inspection for Coding phase. In this study, a combination is also called a process configuration, a process candidate, or a process proposal.

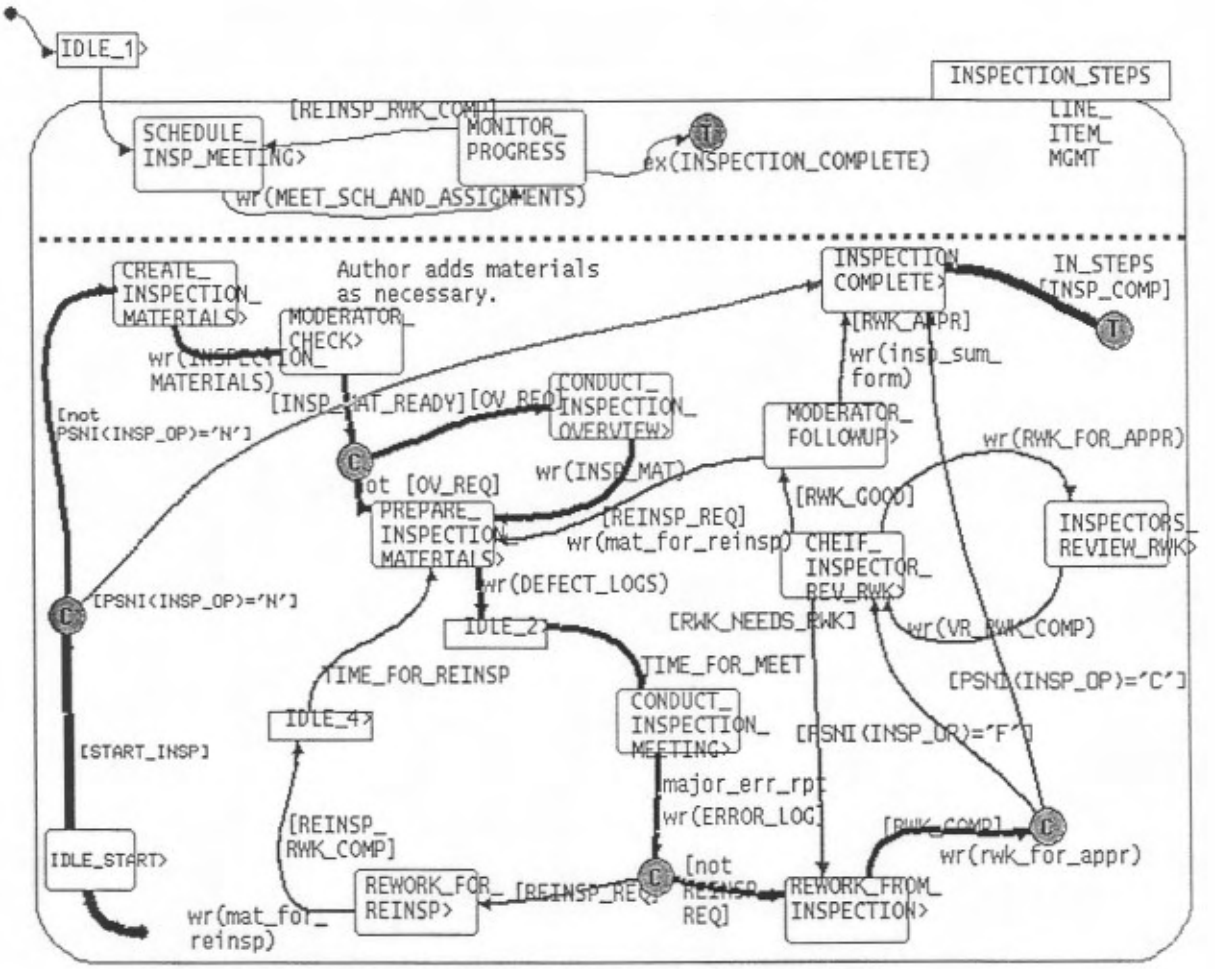

Figure 2-3 Modified Inspection Model INSP_NEW

Figure 2-3 depicts the visible changes that we make to the original inspection model INSP described in section 2.1. In the model INSP, all phases inspection sub-processes do not differentiated with one another, which means as long as INSP is called by main model SQL at any phase among FS, HLD, LLD and Coding, the same set of steps from 1 to11 described in section 2.1 will be gone through. The new model INSP_NEW depicted in Figure 2-3 redefined this assumption by differentiating "F", "C" and "N" inspection modes as follows:

"F" inspection: steps sequence is 1→2→3→4→5→6→7→8→9→10→11;
"C" inspection: steps sequence is 1→2→3→4→5→6→8→11;
"N" inspection: steps sequence is 1→11.

It can be noted that in model INSP_NEW, the technical steps in the section IN_STEPS (the big box below the horizontal dashed line) starts from state "IDLE_START" instead of state "CREATE_INSPECTION_MATERIAL" which is the starting point for section IN_STEPS in model INSP. Here the state "IDLE_START" inserted between "IDLE_1" and "CREATE_INSPECTION_MATERIAL" just serves as a start point for the section IN_STEPS. Also visually, the widest connectors in the section IN_STEPS — e.g. the lines connect the two ends of state "IDLE_START" — represent the paths shared by all three inspection modes. The second widest connectors represent the paths shared by "F" and "C" modes. The third widest

connectors represent the paths occupied by each mode for their own. (Red for "F", blue for "C", green for "N", brown for all, and purple for "F"&"C").

In the model execution, "F", "C" and "N" follow different paths through the inspection steps, obeying the above definitions of the steps sequence for each mode itself. However, the "F" and "C" have important different policies at step 6, where a "F" inspection will decide whether to proceed ahead to step 8 or head back to step 7 for a reinspection, depending on whether the number of errors detected during previous inspection steps. But a "C" inspection will simply choose to proceed ahead to step 8 and finish the current phase inspection. Besides the difference on execution path, it can be expected that some process characteristics will differ for these three inspection modes, too. These issues are discussed in next section 2.2.2.

As mentioned earlier, here a "full" inspection is similar to a so-called Fagan inspection (Fagan 1976) (Fagan 1986). But a "compact" inspection is different from a *walkthrough* described in (Fagan 1976) (Fagan 1986). In our study, a "Compact" inspection is a well-defined and repeatable inspection process, but a *walkthrough* described in (Fagan 1976) (Fagan 1986) is an unstructured and unstable inspection process.

The aforementioned 81 process configurations are proposals for process improvement. In this study, these 81 process configurations will be treated as potential process candidates to be benchmarked against the AS_IS process, which is abstracted in the original model SQL and sub-model INSP before our modification described above. According to the original models SQL and INSP, the AS_IS process is a repeatable process with unsatisfactory performance on inspection. The data for four performance measures of this AS_IS process can be referred to Appendix I-1.

2.2.2 Process Change Data

Given the process re-designed in section 2.2.1, some process characteristics need to be re-considered, such as detection capability in an inspection, number of inspectors involved.

A "Full" inspection is more probable to achieve higher detection capability than a "Compact" inspection, while the detection capability of a "None" inspection is zero. A "Full" inspection should involve more number of inspectors than a "Compact" inspection, while the number of inspectors in a "None" inspection is zero.

Accordingly, we modified or inserted the relevant parameters in both SQL and INSP_NEW to implement the above assumptions. The definitions of these parameters are the following:

Detection Capability	The percentage of current errors detected at a development phase among FS, HLD, LLD, and Code. Determined based upon historical detection capability rates.
Number of Inspectors	The number of inspectors involved in a phase inspection.

And Table 2-1 presents the key parameters for three versions of inspection.

Table 2-1. Key Parameters for Three Versions of Inspection

Parameter	Development Phases	To-BE values in new model		
		Full Inspection	Compact Inspection	None Inspection
Detection Capability	FS	Uniform(0.3, 0.7)	Triangular(0.2,0.4,0.6)	0
	HLD	Uniform(0.3, 0.7)	Triangular(0.2,0.4,0.6)	0
	LLD	Uniform(0.3, 0.7)	Triangular(0.2,0.4,0.6)	0
	Code	Uniform(0.3, 0.7)	Triangular(0.2,0.4,0.6)	0
Number of Inspectors	FS	6	2	0
	HLD	6	2	0
	LLD	6	2	0
	Code	6	2	0

The above assumptions were developed in consultation with a senior software engineer at a leading software development company. An interested reader is referred to (Fagan 1986; Russell 1991; Weller 1993) for a few important researches on inspection. It should be noted that there is considerable variation on detection capability in different practices.

2.2.3 Implementation Costs

In order to implement a process change, resources need to be allocated to designing, communicating, championing, and implementing the process change. These resources may come from the development staff, and/or, much of the work should be done by members of the site Software Engineering Process Group (SEPG). A complete description of how the implementation costs are collected and collated from field site can be found in (Raffo 1996).

The project under research is a one with 60 staff members and in this study we used 5KLOC as the beginning size of the module developed under this process, which is 1/6 of the whole project (30 KLOC). Typically the size of the software under development is the most important factor that essentially impacts on all dimensions of software project, effort, quality and schedule.

In this study, we considered implementation costs are divided into the following components: kick-off meeting, staff training, follow-ups, maintenance and documentation costs. The kick-off meeting is participated by all the project staff members, which announces the process changes to be made and typically lasts two hours. The staff training may consist of about 10 hours per engineer and overall about 6 hours follow-up in implementing a "Full" inspection. The same activities may be only total 3~4 hours per engineer in implementing a "Compact" inspection. The amount of maintenance and documentation performed by SEPG for the whole project is estimated to be about 400~480 hours in implementing a "Full" inspection and about 80~100 hours in implementing a "Compact" inspection.

Based on the abovementioned information, we can estimate the implementation costs incurred by each of the 81 process candidates that we proposed. Obviously for the AS_IS process, no implementation costs incurred for process change at all.

Appendix I-2-2 shows our calculation on the implementation costs for each process candidate. It can be noted that we initially compute the implementation costs for the whole project (the complete software) and then discount the amount in accordance with the module size under this study, which means we divide the initial costs results by 6.

Because the load on any one developer was not great, we consider that the implementation of process change only impacts on Effort measure of the project rather than impacts on Effort and Schedule both. This may deserve a slight adjustment in a real application.

2.3 Results from Simulation Experiments

The results from simulation experiments for total 81 configurations, which resulted from the design described in section 2.2, and from the default process AS_IS, are shown in Appendix I-1. 50 simulation replications are executed for simulating each configuration. In Appendix I-1, the average, standard deviation, minimum, and maximum values based on the 50 replications are listed for each configuration on each performance measure.

In this study, the mean value of each performance measure for each configuration will be used as the inputs for DEA analysis. To obtain the performance measures for each configuration, the implementation costs (described in section 2.2.3) for each configuration are added into both the Life Cycle Effort and the Total Effort (the mean value). And the Remaining Errors, the Life Cycle Duration, plus the Life Cycle Effort and the Total Effort with implementation costs for each configuration are listed separately in Appendix I-2-1. In all quantitative analyses in this study, which will be shown in the rest of this report, both the Life Cycle Effort and the Total Effort include implementation costs.

In section 2.3.1, we pick up a sample configuration NNNC and test the normality of its performance results. And the correlation among the four performance measures is investigated in section 2.3.2.

At a first glance at the results in Appendix I-1 or I-2-1, it's hard to prioritize these process candidates. In Chapter 3, all the data for total 82 process candidates are evaluated via DEA models in order to identify the process candidates with outstanding performance.

2.3.1 The Normality of Data

Figures 2-4 ~ 2-7 show the normal-scores plots for each of the four performance measures of the configuration NNNC. The four performance measures are Remaining Error (RE), Life Cycle Effort (LEFF), Life Cycle Duration (DUR), and Total Effort (TEFF). The performance data for 50 simulation replications can be found in Appendix I-4.

It can be seen that in each of Figures 2-5, 2-6 and 2-7, all points tend to follow a straight line, which indicate that normal distributions are plausible. The plot in Figure 2-4 is discontinuous with regularity. This is because the performance measure Remaining Error (RE) is in integer values and consequently the normality assumption doesn't apply. However, it can be noted that the overall pattern still follow a straight line. It is expected that if RE results were not rounded during simulation execution, the plot would show the data is nicely normal.

Further, we use the *w/s* test to test the normality of the above data. Table 2-2 shows the calculated *w/s* values for each performance measure. For $n = 50$ and we choose $\alpha = 0.005$, then we found the critical values are 3.56 for lower boundary and 5.93 for upper boundary. The *w/s* values for all performance measures (4.21 for RE, 4.42 for LEFF, 4.98 for DUR , 4.17 for TEFF) are well within the range 3.56 ~ 5.93, which suggest the null hypothesis that the distribution is

normal can not be rejected. Hence, through the normal score plots and the *w/s* tests, it is shown that normal distributions are plausible, which are consistent to the results of the normality test done for the "AS-IS" process in (Raffo 1996). The reader is referred to (Kanji 1993) for the description of the *w/s* test and the referred critical values.

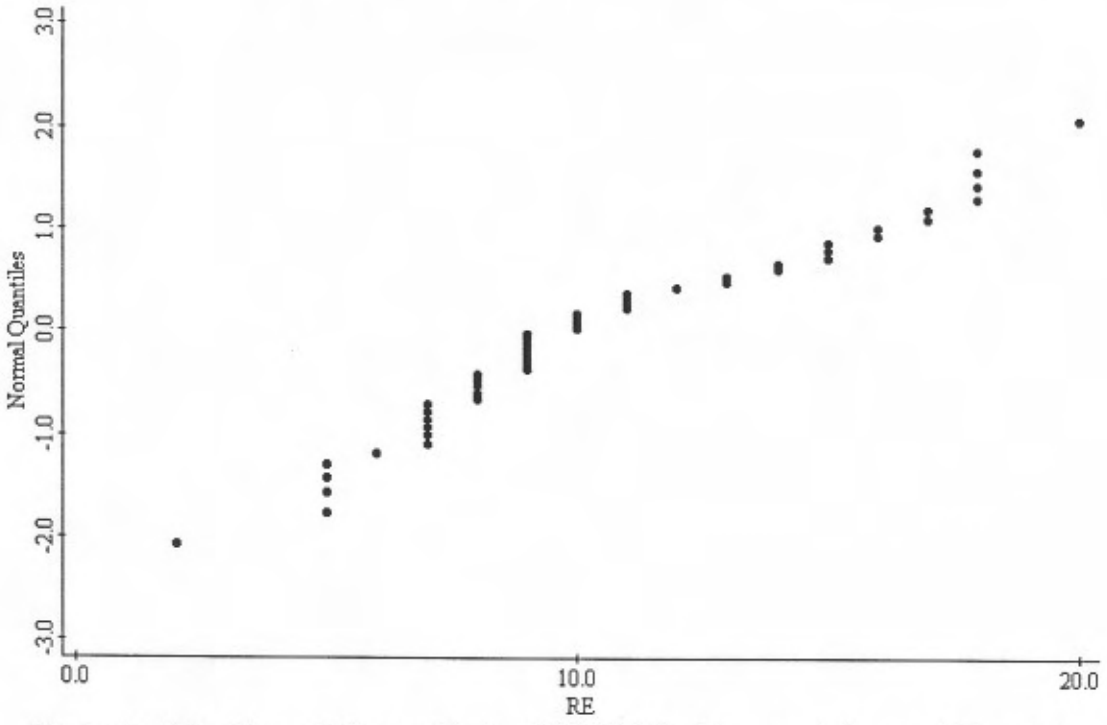

Figure 2-4 The Normal-Scores Plot for NNNC's Performance Measure "Remaining Error"

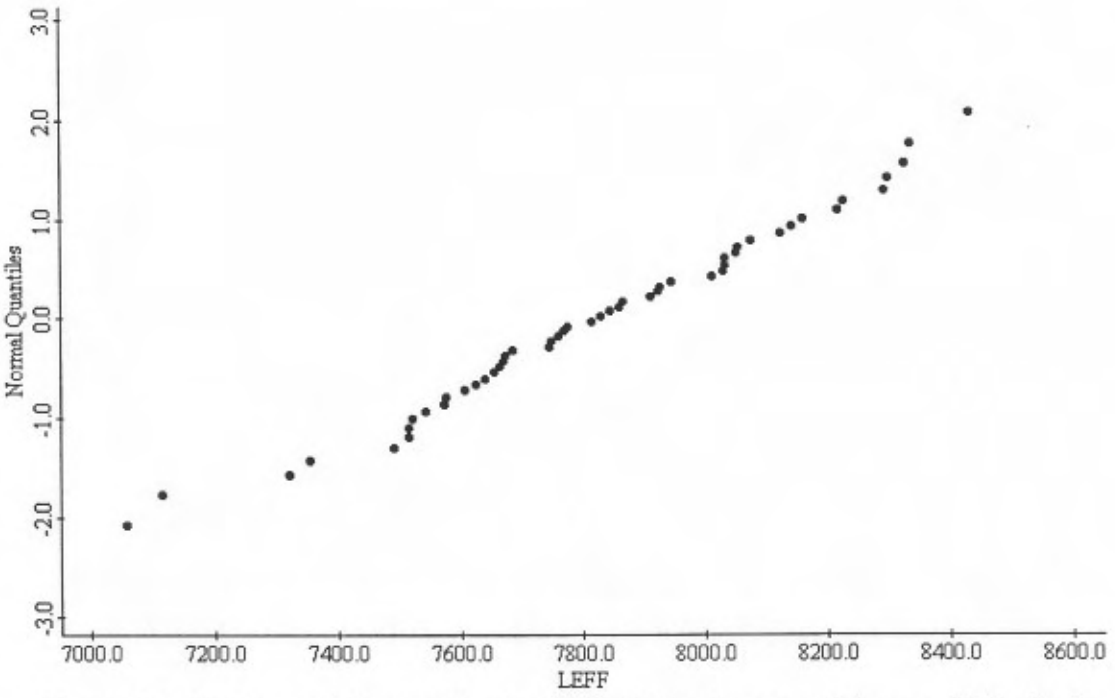

Figure 2-5 The Normal-Scores Plot for NNNC's Performance Measure "Life Cycle Effort"

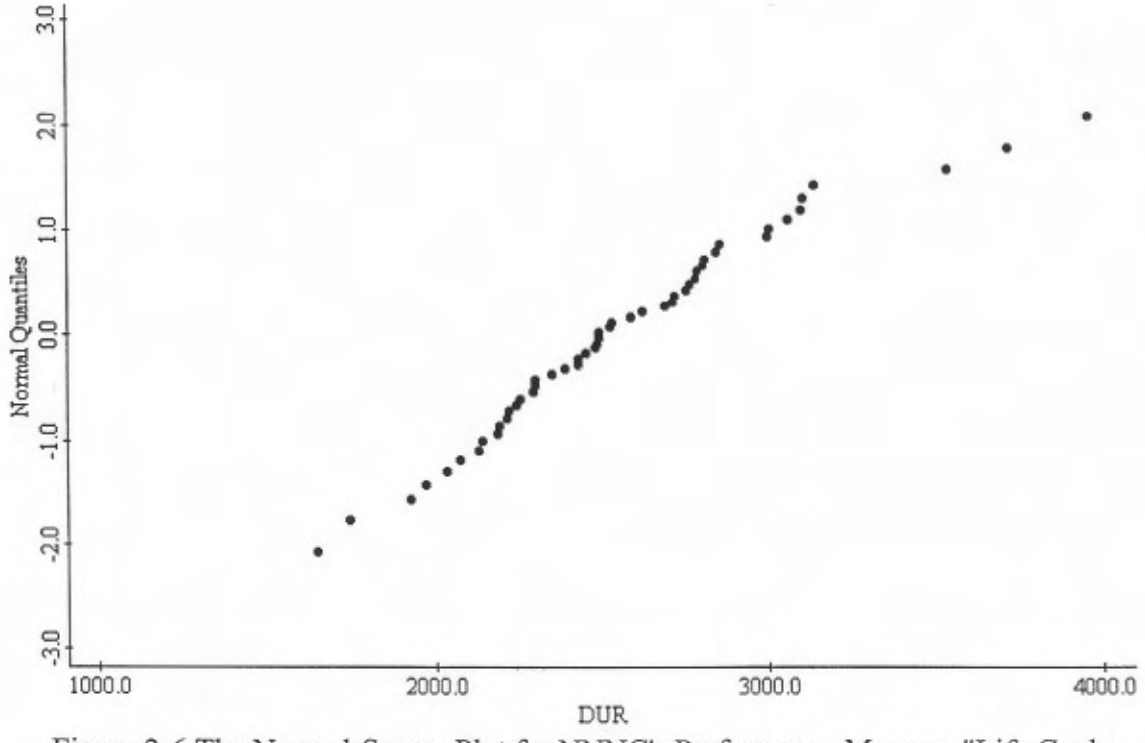

Figure 2-6 The Normal-Scores Plot for NNNC's Performance Measure "Life Cycle Duration"

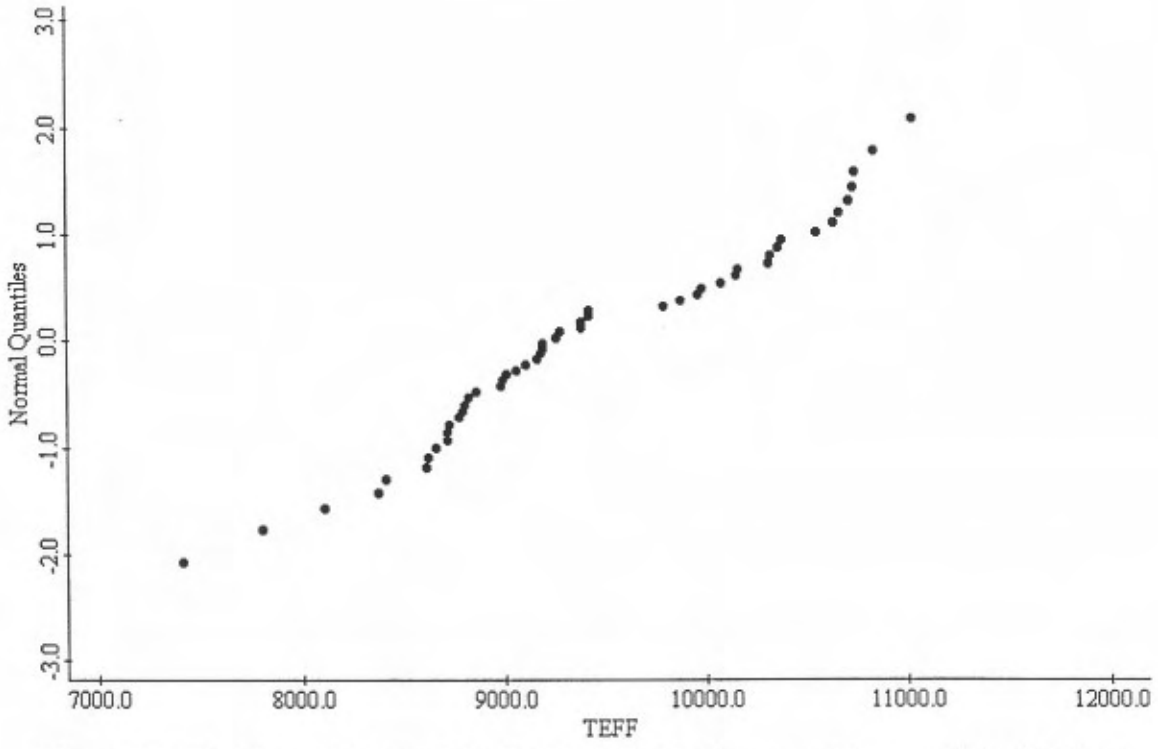

Figure 2-7 The Normal-Scores Plot for NNNC's Performance Measure "Total Effort"

Table 2-2 The Normality Test (w/s Test) for Configuration NNNC

	RE	LEFF	DUR	TEFF
Range (w)	18	1377.35	2312.80	3614.15
Std (s)	4.28	311.78	464.09	866.03
w/s	4.21	4.42	4.98	4.17

Where:

RE is Remaining Error, the number of remaining errors or the number of defects which remain in the product and are passed to the customer. These defects generate field trouble reports. Developers from the project team are required to replicate and correct these defects as a preemptive priority in their work. The effort associated with correcting these defects is an essential component of Field Serve Effort.

LEFF is Life Cycle Effort, the staff effort used required to execute the Functional Specification through System Test operations of the process.

DUR is Life Cycle Duration, the task duration from the start of the Functional Specification operation through the completion of all phases of developing and testing. It does not include the amount of time spent correcting defects which were discovered after the product was released to the public.

TEFF is Total Effort, the Life Cycle Effort plus Field Service Effort.

2.3.2 *The Correlation Among Data*

Table 2-3 presents the correlation coefficients among the four performance measures and Table 2-4 shows the corresponding *t*-test values. For $n = 82$, the critical value for significance $\alpha = 0.005$ is 2.576. Thus, these two tables demonstrate that all four performance measures are highly correlated with each other with very strong statistical significance. This reflects our perception of the real world environment, which are captured inherently in the simulation models.

Table 2-3 The Correlation Coefficient Matrix

Correlation coefficient	RE	LEFF	DUR	TEFF
RE	1.00	0.92	0.98	0.99
LEFF		1.00	0.90	0.97
DUR			1.00	0.97
TEFF				1.00

Where: the legend is the same for all column headings as shown in Table 2-2.

Table 2-4 The *t*-test Values For The Correlation Coefficients

t	RE	LEFF	DUR	TEFF
RE	Infinity	20.53	50.98	58.94
LEFF		Infinity	18.83	33.59
DUR			Infinity	38.38
TEFF				Infinity

Where: the legend is the same for all column headings as shown in Table 2-2.

2.4 *Model Verification*

In this study, the source models SQL and INSP are modified to be two new models SQL_NEW and INSP_NEW. In this section we'll show how we verify the modifications made and described in last section 2.2.

2.4.1 *Model Verification I — Configuration Verification*

The modifications done for configuration re-design is shown in section 2.2.1. These modifications focus on the paths which the INSP_NEW model execution follows. The animation capability of Statemate make it very straightforward to verify the path changes made for different inspections Full, Compact or None. Based on our observations on simulation execution, we found that the inspection model proceeds exactly as what we designed in section 2.2.1. i.e.

"F" inspection: steps sequence is 1→2→3→4→5→6→7→8→9→10→11;

"C" inspection: steps sequence is 1→2→3→4→5→6→8→11;

"N" inspection: steps sequence is 1→11.

2.4.2 *Model Verification II — Manual Checking*

Besides modifying the inspection configuration, we also modified some key parameters such as Detection Capability and the Number of Inspectors. Does the model perform according our modifications and generate correct results? In this section, we verify the computation performed

by the simulation model via manual checking.

The approach we follow is to choose a representative configuration, and have the simulation model execute this configuration once to generate one replication. Meanwhile, we construct a spreadsheet, which implements all the computations performed by the simulation model for executing this configuration. The initial values and the random numbers generated during the simulation are retrieved from the simulation execution through parameter passing. Given these initial values, intermediate random numbers and formulas copied from simulation model, "spreadsheet model" can generate outputs for each performance measure. This process is equivalent to using our brain to execute the process model with spreadsheet's assistance on numerical computation.

Finally we compare the results from simulation model and those from spreadsheet. If the results are identical, we then consider that the simulation model does what it is supposed to do.

In this study, Our modifications are essentially done on inspection model INSP. To make sure that our modifications are made correctly, we need only to check whether the modified inspection model INSP_NEW performs in accordance with the design described in section 2.2. Corresponding to three project performance measures Remaining Errors, Life Cycle Effort and Life Cycle Duration, there are three key performance measures during inspection, which are Corrected Errors, Inspection Effort, and Inspection Duration.

Among them, Inspection Effort is added into the Life Cycle Effort directly, Inspection Duration is added into Life Cycle Duration based on the scheduling algorithm applied in the simulation model, and the Remaining Errors is negatively impacted by the number of Corrected Errors during an inspection. Therefore, given that our modifications are done only inspection, we need only verify the simulation outputs on Corrected Errors (in an inspection), Inspection Effort, and Inspection Duration are identical to the results from the above "spreadsheet model".

In this verification we choose NNNC as the representative configuration. The reasons are twofold: 1. This configuration has only one inspection during Code phase, which makes our verification simpler without losing validity since all four key development phases FS, HLD, LLD, and Code utilize a same inspection model except that the initial values are different. 2. "C" inspection is the inspection which embodies our main process change design described in section 2.2. A "N" inspection is simply skipping the inspection. And a "F" inspection is essentially the AS_IS inspection except that the key parameter "Detection Capability" is changed, as described in section 2.2.2.

Upon verifying NNNC configuration for one replication, we believe that the modifications are done correctly.

Appendix I-3-1 shows the "spreadsheet model" and its results which we built up based upon the formulas applied in the simulation model. In implementing the "spreadsheet model", we manually interpret the execution conditions, performing judgements and the calculations. In short, we manually "simulate" what the inspection simulation model does, using spreadsheet.

As mentioned before, all the initial values and intermediate random numbers are retrieved from simulation execution. In this verification, we sampled total 73 data points from the simulation execution. Appendix I-3-2 shows the parameter mapping used in this verification and Appendix I-3-3 shows the relevant formulas for a "C" inspection which are used by our manual model.

As can be seen in Appendix I-3-1, the left top of the sheet shows the simulation model results on Corrected Errors During Code Inspection (Corr E - Code), the Effort During Code Inspection (Code Insp Eff), and the Duration of Code Inspection (Code Insp Dur), which are 56, 16.1496, and 51.5764, respectively.

Below the top two rows is the spreadsheet model. The sheet area are horizontally divided into three sections of Error, Effort and Duration, and vertically divided into 13 sections designating 13 inspection steps (in Statemate they are 13 individual states). The final results of a "C" inspection can be read from the last row of step "RFI" which is the last step during a "C" inspection with active reactions (reaction is a Statemate terminology, and here can be understood as the actual computation about the process performance). These results are 56 for Corr E - Code, 16.14964 for Code Insp Eff, and 51.5764 for Code Insp Dur. We found that these results are exactly the same as the aforementioned outputs from the simulation model execution. The small difference on Code Insp Eff, 0.00004 is seen as a rounded error which can be ignored.

Therefore, we verify that our modifications are implemented in accordance with our design in section 2.2.

Chapter 3 DEA Analysis

3.1 Development of the Multiplier Model with Weight Restrictions

In this section, we introduce the basic DEA multiplier model developed by Charnes et al. (Charnes, Cooper et al. 1978) and review a few techniques for incorporating weight restrictions into the multiplier model. Finally we present the development of DEA model with weight restrictions in this study.

3.1.1 Introduction of the Multiplier Model

Charnes et al. (Charnes, Cooper et al. 1978) developed the first DEA model based on an earlier work by Farrell (Farrell 1957). DEA is intended to assess the efficiency of a set of homogeneous units (Decision Making Unit, in the DEA terminology) by comparing each DMU against a frontier which is constructed within the same set of DMUs via linear programming technique, with respect to the levels of inputs consumed (less is desired) and the levels of outputs produced (more is desired). Utilizing linear programming techniques, a DEA model is capable to assess the relative efficiency of each DMU in the whole set.

If we consider a set of n DMUs, where DMU j consumes a vector of m inputs $\boldsymbol{x}_j$ and produces a vector of s outputs $\mathbf{y}_j$, the efficiency of a DMU_{j0}, is assessed by choosing weights vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ so as to:

$$\text{Maximize} \quad h_0 = \frac{\sum_{r=1}^{s} u_r y_{rj_0}}{\sum_{i=1}^{m} v_i x_{ij_0}}$$

$$\text{Subject to} \quad \frac{\sum_{r=1}^{s} u_r y_{rj}}{\sum_{i=1}^{m} v_i x_{ij}} \le 1 \qquad j = 1, \ldots, n,$$

$$-v_i \le -\varepsilon, \qquad i = 1, \ldots, m,$$

$$-u_r \le -\varepsilon, \qquad r = 1, \ldots, s.$$

where:

y_{rj} is the amount of the r th output produced by DMU j;
u_r is the weight given to r th output;
x_{ij} is the amount of the i th input consumed by DMU j;
v_i is the weight given to the i th input; and
ε is a non-Archimedian infinitesimal.

The above model is often called ratio model. When DEA is constructed in a ratio format, the definition of efficiency is specified as the ratio of the weighted sum of outputs over the weighted sum of inputs of a DMU. The ratio model is not a linear model that can be solved by using linear programming technique. However, it is readily linearized as Model 3-1. The Model 3-1 is also called the multiplier model. By solving this linear program for each DMU in the set, the relative

efficiency of an assessed DMU is obtained.

MODEL 3-1

$$\text{Maximize} \quad \sum_{r=1}^{s} u_r y_{rj_0}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} v_i x_{ij_0} = 1,$$

$$\sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} \le 0, \qquad j = 1, \ldots, N,$$

$$-v_i \le -\varepsilon, \qquad i = 1, \ldots, m,$$

$$-u_r \le -\varepsilon, \qquad r = 1, \ldots, s.$$

In our study, a process configuration is a DMU. Now we let x_{1j}, x_{2j}, and x_{3j} be the three performance measures Remaining Error (RE), Life Cycle Effort (LEFF), and Life Cycle Duration (DUR), respectively, and y_1 be a dummy constant output 1 (only one output, namely $r = 1$, and by y_1=1 we mean a process has been completed), for any process configuration j. Among all process configurations, j_0 is the one under evaluation. The multiplier model will allow process configuration j_0 to choose its most favorite weight scheme in DEA evaluation to determine its own efficiency score. Noted here $i = 1$, 2, or 3, $r = 1$, and $j = 1...82$. However, a weight scheme chosen by a specific process configuration may be totally unacceptable in reality. E.g. We use the multiplier model to evaluate a set of software processes and our goal is to come up with a "general score" for each software process. We use the performance measures for Remaining Error, Life Cycle Effort, and Life Cycle Duration as inputs for each process. If in assessing process j, DEA assigns process j's most favorite weight scheme such that 60% of the sum of all "output * weight" products is for "RE * RE's weight", 40% of the sum of all "output * weight" products for "LEFF * LEFF's weight", and 0% for "DUR * DUR's weight", the efficiency score resulting from this scheme may not be accepted because it totally denies the significance of the third dimension, project duration.

3.1.2 A Review of Techniques for Weight Restrictions

It is argued that the decision to include a factor (input or output) in a DEA model represents an implicit judgement that the factor has a non-trivial weight, and that it seems perverse to allow DEA to assign a trivial weight to that factor in assessing the efficiency of a DMU (Pedraja-Chaparro, Salinas-Jimenez et al. 1997). Therefore there is a strong case for imposing restrictions on factor weights. Allen (Allen, Athanassopoulos et al. 1997) argues that the incorporation of value judgements (weights restrictions) was motivated by applications of the method in the real life organizations, and the application driven development of the methods has led to a number of different approaches in the literature which have inevitably different uses and interpretations.

Allen (Allen, Athanassopoulos et al. 1997) divides the current approaches on imposing weight-restrictions into three categories: direct restrictions on the weights (Thompson, Singleton et al. 1986; Dyson and Thanassoulis 1988; Banker and Morey 1989; Thanassoulis, Boussofiane et al. 1995), adjusting the observed input-output levels (Charnes, Cooper et al. 1990; Cook, Kress et al. 1992), and restricting the virtual inputs and outputs (Beasley 1990; Wong and Beasley 1990). Among these three categories, there is significantly more literature falling in the first category than the other two.

In the first category "direct restrictions on the weights", there are two sub-category approach, namely Assurance Regions, and Absolute Weights Restrictions. The Absolute Weights Restrictions was first developed by Dyson and Thanassoulis (Dyson and Thanassoulis 1988). The approach is to impose simple numerical limits on each of the weights v_i in a model like Model 3-1, of the sort:

$$Q2_i \le v_i \le Q1_i,$$

where $Q2_i$ and $Q1_i$ are identified constants. And equivalent for outputs:

$$P2_r \le u_r \le P1_r,$$

where $P2_r$ and $P1_r$ are identified constants.

This method suffers from the possiblity that it may lead to a linear program with no feasible solutions. Weights can usually only be set after examining the results from an unbounded DEA, and may have to be varied from one DMU to another. As a result, it is difficult for the users of a DEA analysis to sense the implications of the weights restrictions.

Thompson et al (Thompson, Langemeier et al. 1990) have developed the concept of an "assurance region" (AR) to circumvent the problem of infeasibility. Their approach sets boundaries for the ratios between the various weights of two inputs (or two outputs), using available information and expert opinion. The AR constraints are of the form:

$$K_r \le \frac{u_r}{u_1} \le L_r$$

$$G_i \le \frac{v_i}{v_1} \le H_i$$

where K, L, G, and H are supplied by the user. The above constraints are then readily linearized as the following:

$$K_r u_1 \le u_r \le L_r u_1$$
$$G_i v_1 \le v_i \le H_i v_1$$

Equivalently, the AR method has the form as:

$$u_r \le \alpha_{rs} u_s \qquad \forall r \ne s$$
$$v_i \le \beta_{ij} v_i \qquad \forall i \ne j$$

where $\alpha_{rs} = L_r / K_s$ and $\beta_{ij} = H_i / G_j$.

Thompson et al pointed out that the AR approach allows one to refine the DEA analysis in the 'eyes of the user', who is the ultimate 'consumer' of the modeling product. More applications of AR approaches can be found in (Thompson, Dharmapala et al. 1994; Thompson, Brinkmann et al. 1997).

For the second category of weights restriction methods — adjusting the observed input-output levels, Charnes et al. (Charnes, Cooper et al. 1990) developed the "cone-ratio" approach and Golany (Golany 1988) developed another method. In their approaches, transformed input-output data is used to simulate weight restrictions. This type of approaches allows the use of DEA software which does not otherwise offer weights restrictions facilities and they allow zero or even negative observed data levels to be used. The disadvantage is that the data must be transformed and then once results are obtained it must be transformed back to the original form in order to interpret the results, which may prove more cumbersome than the direct application of weight restrictions.

For the third category of weights restriction methods — restricting the virtual inputs and outputs, Wong and Beasley (Wong and Beasley 1990) suggested a method based upon the use of proportions. Conceptually, $V_iX_{ij}/\sum V_iX_{ij}$ represents the importance of input variable i to DMU j, and can be considered a "virtual" input weight. Wong and Beasley propose that the user or analyst should set boundaries, $[a_i, b_i]$, for the importance of input variable i to DMU j. In input space, the constraint determined by this method is of the form:

$$a_i \le \frac{V_i X_{ij}}{\sum_{i=1}^{m} V_i X_{ij}} \le b_i,$$

and a similar set of constraints might be imposed on outputs. This approach is sensible in the sense that an input (or output) should contribute at least a certain portion to the total budgets (or benefits). So far, this type of approach of weight restrictions has received relatively little attention.

Allen (Allen, Athanassopoulos et al. 1997) noted that different approaches are likely to prove more appropriate in different contexts while no overall approach to setting weights has been identified. In next section, 3.1.3, we will show why and how the Assurance Region method is used to determine the weight restrictions for this study.

3.1.3 Development of the Multiplier Model with Weight Restrictions

In this study, in order to have the DEA model yield reasonable results, we have to pose restrictions on the model based on the judgements about the relative significance of the different input variables such as Remaining Error, Life Cycle Effort, and Life Cycle Duration. These judgements may come from the knowledge of the analyst and more likely, they come from the teamwork of the analyst and the user, namely the decision-maker in evaluation. In our study, the decision-maker in evaluating software process is the software project manager.

In the study (Raffo 1996), Raffo elicited the weights of Remaining Error, Staff Effort, and Task Duration in developing a piece-wise utility function for Process Trade-off Analysis. The utility function is as below:

$$\text{UTILITY} = \begin{cases} 40*\Delta RE + 2*\Delta EFF + \Delta DUR, & \text{when } \Delta RE = [0, 2),\ \Delta EFF = [0, 20), \text{ and } \Delta DUR = [0,40); \\ 24*\Delta RE + \Delta EFF + 6.5*\Delta DUR, & \text{when } \Delta RE = [2, 6],\ \Delta EFF = [20, 800], \text{ and/or } \Delta DUR = [40,160], \text{ and excluding any } \Delta RE > 6,\ \Delta EFF > 800, \text{ or } \Delta DUR > 160; \\ 48*\Delta RE + \Delta EFF + 2*\Delta DUR, & \text{when } \Delta RE > 6,\ \Delta EFF > 800, \text{ and/or } \Delta DUR > 160. \end{cases} \quad (3\text{-}1)$$

Here ΔRE represents the change in the number of Remaining Errors, ΔEFF represents the change in the hours of staff effort, and ΔDUR represents the change in the hours of project duration. Here the change means the incremental change (improvement) between the To-BE process and the AS-IS process.

Clearly, the above utility function contains relative price information between RE, EFF and

DUR. For each piece of the above utility function, we can determine a set of price relationships between error, effort, and duration. e.g. According to the first piece utility function, in order to have the TO-BE process increase 1 util from AS-IS, we can let either ΔRE = 1/40, or ΔEFF = 1/2, or ΔDUR = 1 (assuming that when we give incremental change to one dimension, the other two dimensions remain unchanged). In a sense, we say the change of 1/40 remaining error "equals" ½ person hours in effort or 1 hour in duration. Now we have three pieces of utility functions so it is easy to come up with the ranges of these prices. This information is well suited for determining weight restrictions for DEA model.

In this study, we examined different techniques on weight restrictions mentioned in last sub-section. Among the first category of directly restricting weights, the Absolute Weights Restrictions approach causes infeasible linear programs for the data used in this study. The second category of weight restriction methods — adjusting the observed input-output levels, is cumbersome and does not seem to offer visible benefits for this study. The third category of weights restriction methods — restricting the virtual inputs and outputs, requires user judgements regarding to the boundaries $[a_i, b_i]$, for the importance of input variable i to DMU j, as described earlier. This *a priori* information is not available. We found the Assurance Region (AR) method, belong to the first-category weight restriction approach, is feasible for this study and matches the need of analysis.

In this study, the information captured by the above utility function is used in developing the weights for DEA weights restriction model via an Assurance Region approach. Based on the first piece of the utility function, we have the following weight restrictions on ΔRE, ΔEFF, and ΔDUR:

$$\frac{v_1}{v_2} = \frac{40}{2} = 20, \qquad \frac{v_3}{v_2} = \frac{1}{2} = 0.5, \qquad \frac{v_1}{v_3} = \frac{40}{1} = 40,$$

where v_1, v_2, *and* v_3 are the weights for ΔRE, ΔEFF, and ΔDUR respectively. Similarly for the second piece of the utility function, we have:

$$\frac{v_1}{v_2} = \frac{24}{1} = 24, \qquad \frac{v_3}{v_2} = 6.5, \qquad \frac{v_1}{v_3} = \frac{24}{6.5} = 3.69,$$

and for the third piece of the utility function, we have:

$$\frac{v_1}{v_2} = \frac{48}{1} = 48, \qquad \frac{v_3}{v_2} = \frac{2}{1} = 2, \qquad \frac{v_1}{v_3} = \frac{48}{2} = 24.$$

These relationship equations represent conversion equality which vary, depending upon the magnitude of performance changes under consideration. Three relationship equations for each pair of weights reflect this variability due to the magnitude of performance change, and thus reasonably represent the range of the variability. By allowing DEA model to be flexible within this elicited range on assigning weights for each performance measure in evaluating each process candidate, the weight-restricted DEA analysis give us a balance view in assessing the large pool of process proposals within which there will be large variability on performance change.

The above so-called conversion equality may vary over time, too. e.g. At the beginning or within the ending period of a project, a software manager may have largely different judgements about trading-off among the three performance dimensions. But in this study, we assume we make our evaluation at a specific time point, namely the beginning of the project. Thus we don't deal with the variability due to time in this study.

Studying the above relationship equations across the three pieces of utility function, we have the following series:

$v_1/v_2 = 20, 24, \text{ or } 48;$
$v_3/v_2 = 0.5.\ 6.5, \text{ or } 2;$
$v_1/v_3 = 40, 3.69, \text{ or } 24.$

The upper and lower limits for v_1/v_2 , v_1/v_3, and v_3/v_2 are found to be:

$$20 \le \frac{v_1}{v_2} \le 48, \qquad 0.5 \le \frac{v_3}{v_2} \le 6.5, \qquad 3.69 \le \frac{v_1}{v_3} \le 40.$$

By incorporating these weight restrictions into the model in 3-1, we develop the following multiplier model for this study:

MODEL 3-2

$$\text{Maximize} \quad u_1 y_{1j_0}$$

$$\text{Subject to} \quad \sum_{i=1}^{3} v_i x_{ij_0} = 1,$$

$$u_1 y_{1j} - \sum_{i=1}^{3} v_i x_{ij} <= 0, \qquad j = 1, .., 82,$$

$$20v_2 \le v_1 \le 48v_2,$$
$$0.5v_2 \le v_3 \le 6.5v_2,$$
$$3.69v_3 \le v_1 \le 40v_3,$$
$$-v_i \le -\varepsilon, \qquad i = 1, 2, 3$$
$$-u_1 \le -\varepsilon,$$

(here ε is a non-Archimedian infinitesimal.)

where v_1, v_2, and v_3 are the weights for RE, LEFF, and DUR respectively.

The DEA implication for such restriction is discussed below.

It can be noted that there are jumps between applying different pieces of the utility function. e.g. For two process proposals, process 1 with ΔRE = 6.0, ΔEFF = 5, ΔDUR = 8 and process 2 with ΔRE = 6.1, ΔEFF = 5, ΔDUR = 8. The two process proposals have the same levels of ΔEFF and ΔDUR within the range confined by the first piece of the utility function. But the ΔRE = 6.0 of process 1 falls within the first piece of utility function and the ΔRE = 6.1 of process 2 falls within the second piece of utility function. For process 1 we use the first piece of the utility function then we have $v_1/v_2 = 24$, but for process 2 we use the second piece utility function then $v_1/v_2 = 48$. The difference between ΔRE = 6.0 and ΔRE = 6.1 appears to be trivial but the difference between $v_1/v_2 = 24$ and $v_1/v_2 = 48$ is nontrivial. As a result, the difference between the results of utility for these two processes is Δutil = (util of process 1 – util of process 2) = 258 – 203.4 = 54.6.

If we apply only the first piece of the utility function for both processes 1 and 2, then we have the difference between the results of utility for these two processes is Δutil' = (util of process 1 – util of process 2) = 258 – 262 = – 4. Or, if we apply only the second piece of the utility function for both processes 1 and 2, then we have the difference between the results of utility for these two processes is Δutil" = (util of process 2 – util of process 1) = 201 – 203.4 = – 2.4.

It's likely that process 1 with ΔRE = 6.0, ΔEFF = 5, ΔDUR = 8 and process 2 with ΔRE = 6.1, ΔEFF = 5, ΔDUR = 8 are essentially perceived as equally good by a software project manager. But, as shown above, when applying the first piece and second piece of the utility function for process 1 and 2, respectively, the difference of the utility results can be as large as 54.6(21.1% of the utility of process 1, or 26.8% of the utility of process 2). This difference is nontrivial. When applying the same piece of the utility function for these two processes both, the difference is either 4 (1.55% of the utility of process 1, or 1.53% of the utility of process 2), or 2.4 (1.19% of the utility of process 1, or 1.18% of the utility of process 2), which is trivial and more acceptable in this context.

Therefore, the large jumps for the weights assessment of ΔRE, ΔEFF, and ΔDUR between the adjacent pieces of the utility function seem to be arbitrary as discussed above. The DEA Model 3-2 such discontinuity in price judgement but maintaining its efficacy by allowing each configuration to have certain degree of flexibility in choosing its weight scheme.

In the DEA study, we use the absolute number of Remaining Errors, person hours of Life Cycle Effort, and hours of Life Cycle Duration as the performance measures, rather than using the incremental changes (ΔRE, ΔEFF, and ΔDUR) between 81 process configurations versus the AS-IS process. But these weight schemes used for RE, LEFF and DUR are actually the weights schemes developed based for ΔRE, ΔEFF, and ΔDUR, as described in this section. This treatment is an approximation but the weights derived here still convey the information about the relative "prices" of three dimensions of performance measures. In this context, this treatment seems the best solution and its effect is left to be a meaningful topic for future research.

3.2 Analysis and Results

In this section, we conducted extensive DEA analysis based on the simulation results obtained from section 2.3. First we used a DEA model without weight restrictions (Model 3-1) to obtain the preliminary results. Then we incorporate weight restrictions into the first model to develop Model 3-2 and used this model to obtain the results for the second round analysis. And finally we further incorporate standard deviations into Model 3-2 to account for the risks associated with each configuration. These three round analyses unfold by sections 3.2.1, 3.2.2 and 3.2.3.

3.2.1 Analysis I — DEA without Weight Restrictions

The DEA model used in this analysis is Model 3-1. This model allows DEA with unbounded flexibility on weights selection in evaluating each process candidate. Appendix II-1 demonstrates the results from this analysis.

Table 3-1 presents the efficiency scores, ranks, and chosen weights of those top 15 process configurations with highest efficiency scores and AS_IS process, based on Appendix II-1.

Table 3-1 Selection of The Results from Analysis I: The Ranks and Efficiency Scores of The Top 15 Process Configurations and AS_IS Process

RANK	CONFIG	RE	LEFF	DUR	Weight - RE $\times 10^{-3}$	Weight - LEFF $\times 10^{-4}$	Weight - DUR $\times 10^{-5}$	Efficiency Score
1	CCCC	6.6	7560.3	2055.7	0.000	1.255	2.502	1.0000
1	FCFF	3.7	7604.1	1860.3	1.972	1.305	0.000	1.0000
1	FFFF	3.3	7615.9	1776.9	3.825	1.296	0.000	1.0000
4	CFFF	3.6	7610.0	1829.0	1.052	1.278	1.304	0.9998
5	CCFF	4.0	7601.4	1897.2	1.971	1.305	0.000	0.9997
6	NFFF	4.1	7607.1	1904.0	1.969	1.304	0.000	0.9988
7	FCCF	4.5	7603.2	1955.4	1.969	1.304	0.000	0.9985
8	CCCF	4.9	7607.8	2006.3	1.966	1.302	0.000	0.9971
9	NCCC	7.1	7587.0	2136.8	0.000	1.318	0.000	0.9965
10	NFCF	5.0	7614.6	2013.2	1.964	1.300	0.000	0.9961
11	CFCF	4.4	7625.1	1938.9	1.964	1.300	0.000	0.9958
12	NCFF	4.6	7622.7	1965.5	1.963	1.300	0.000	0.9957
13	FFCF	4.1	7634.6	1881.1	1.047	1.272	1.298	0.9954
14	NCCF	5.4	7629.9	2069.8	1.959	1.297	0.000	0.9932
15	CNCC	7.7	7636.6	2222.5	0.000	1.309	0.000	0.9900
38	AS_IS	10.0	7820.9	2426.9	0.000	1.279	0.000	0.9667

Where:

RANK	is the efficiency ranking of each configuration in this analysis.
CONFIG	is the abbreviation of "Configuration".
RE	is the mean value of Remaining Error for 50 simulation replications, one of the performance measures used for benchmarking processes.
LEFF	is the mean value of Life Cycle Effort for 50 simulation replications and include implementation costs, in person hours. One of the performance measures used for benchmarking processes.
DUR	is the mean value of Life Cycle Duration for 50 simulation replications, in hours. One of the performance measures used for benchmarking processes.
Weight -RE	is the weight for RE resulting from DEA analysis.
Weight -LEFF	is the weight for LEFF resulting from DEA analysis.
Weight - DUR	is the weight for DUR resulting from DEA analysis.
Efficiency Score	is efficiency score obtained from DEA analysis.

We have the following findings based on this analysis:

1. Configurations CCCC, FCFF and FFFF determine the efficiency frontier. Among the 82 process configurations, FFFF has the best performance on two measures — Remaining Error Mean, and Life Cycle Duration Mean. FFFF is not superior on Life Cycle Effort Mean among the top process candidates listed in the above table. Its moderately higher Life Cycle Effort is explained by its relatively high implementation costs (500 person hours) associated with it against the AS-IS based on our calculation described in section 2.2.3.

   Efficient candidate CCCC excels by demonstrating the lowest Life Cycle Effort Mean. This result is noteworthy because it supports the management expectation that a "Compact"

inspection will save development effort. However, this candidate has notably high Life Cycle Duration Mean and its Remaining Error Mean (6.6) is as many as twice of that of FFFF (3.3). Also it can be noted that CCCC's lowest Life Cycle Effort is partly contributed by its relatively low implementation costs (120 person hours).

It appears that the other efficient candidate FCFF excels by demonstrating a "best mix" of Remaining Error Mean, Life Cycle Duration Mean, and Life Cycle Effort Mean. This result is very interesting because it appears to help on addressing a management question like "Can we compress the inspection effort on High Level Design given that we have already done an excellent job(inspection) on its preceding phase, namely Functional Specification?". Similar questions has been raised by software project managers such as "What if we compress inspection for Low Level Design since we have already done an excellent job(inspection) for its preceding phase High Level Design?".

Based on the simulation results, the FCFF process performs well on the inspection at the first development phase (FS) so that it leaves fewer errors remaining in the work-in-progress, which relieves the pressure for the successive quality assurance activities (inspections/tests) after FS to a degree. The raw simulation results of configurations FCFF, CCFF and NCFF are provided in Appendix II-4, which shows the inspection/test performance phase by phase. Comparing the performance of these three configurations, it can be found that both CCFF and NCFF correct more errors than FCFF in UT, FVT and SVT phases. Also both CCFF and NCFF have more remaining errors than FCFF. However, the expected effect, that FCFF has Full inspection in the Functional Specification and thus relieves inspection pressure of HLD inspection, is not apparent based on these simulation results.

2. The rest of process candidates on the top 15 list can be divided into two classes: CFFF, CCFF, NFFF, FCCF, CFCF, NCFF and FFCF that have Remaining Error Mean below 4.8, and Life Cycle Duration Mean below 2000; NCCF, CNCC, CCCF, NCCC and NFCF that have Remaining Error Mean above 4.8, and Life Cycle Duration Mean above 2000. We have known from section 2.3 that there is a very strong correlation between Remaining Error Mean and Life Cycle Duration Mean (correlation coefficient = 0.9 from Table 2-2). So the above division is not surprising.

   According to Table 3-1, most process candidates have zero weight(s) on at least one dimension of process performance (only FFCF and CFFF have non-zero weights on all three performance measures). This phenomenon can also be observed if we look at Appendix II-1 for other process configurations. However giving zero weight on a particular dimension in trading-off the three performance measures is highly unacceptable in software project management practices. This necessitates a more advanced DEA model — weight-restricted DEA model developed in last section, namely Model 3-2.

   In order to better prioritize the many process configurations on and close to efficiency frontier, besides improving the DEA model, it is also desired to use judgements based on our knowledge on software process practices. For example, a process configuration NFFF might not be accepted even the simulation results shows it is among the good processes. Because in a software process, the critical errors injected during the earlier phases impact on the overall quality of the product more seriously and cost more effort to correct, compared to the critical errors injected during later phases. Inspection for the earliest development phase — Functional Specification is obviously important. Therefore, process configurations such as NFFF, NCCC, NFCF, NCFF and NCCF are unlikely to be accepted even if they have high efficiencies.

Here we identify a limitation of the current simulation model, which is the uniform treatment of the property of errors injected during development phases. In the simulation models we used in this study, the errors are treated as uniform, which means an error injected in FS phase costs the same amount of effort for the software engineers to correct it along the development life cycle. However this is not true in the real world. The errors injected during FS and HLD are more likely to cause larger problems than the ones injected during later phases, and in turn they cost more rework effort. This wasn't captured by the current simulation model yet and this model will be improved in our follow-up study.

3. The AS_IS process has efficiency score as 0.9667 and ranks 38th among the pool of processes, which suggests that if we successfully implemented the best proposal based on our analysis and our assumptions are essentially approaching the reality, the process efficiency would be significantly improved.

   Also, even this first round analysis has already demonstrated the power of DEA being able to benchmark such a large pool of process proposals in a handy way. In section 3.2.2., we will further improve the analysis by using a more advanced model.

4. In Analysis I, the mean efficiency of total 82 configurations is 0.95719 and the standard deviation of efficiency is 0.03267. The following figure shows the distribution of the efficiency scores from this round analysis. It can be noted that the mode occurs at Efficiency = 0.98x.

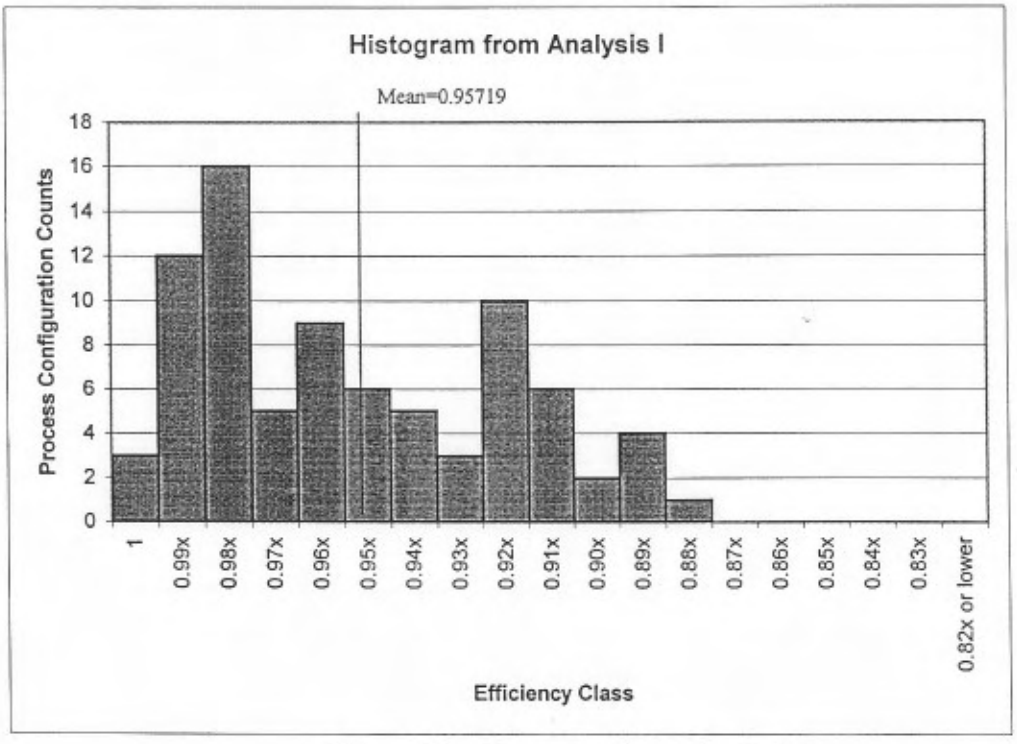

Figure 3-1 Histogram from Analysis I

3.2.2 *Analysis II — Weight-Restricted DEA*

Model 3-2 incorporates weight restrictions. The way Model 3-2 poses weight restrictions is called Assurance Region (Allen, Athanassopoulos et al. 1997), as introduced in section 3.1. The implication of introducing weight restrictions is to eliminate the unbounded flexibility of DEA model on choosing a most favorable weight scheme for each process candidate. Such unbounded flexibility, demonstrated in the DEA analysis I in section 3.2.1, allows each process candidate to have zero weight to one or two performance measures among Remaining Error, Life Cycle Effort, and Life Cycle Duration, which is unrealistic.

With the weight restrictions developed in section 3.1, it is expected that when assessing a process candidate, the flexibility of DEA model in choosing weights for three dimensions of performance measures is restricted to a degree that a ratio of each pair of weights(for one process candidate, three ratios for three pair of weights, respectively) has lower and upper boundaries. These lower and upper boundaries, developed based on a cautious elicitation procedure, reflect the management judgements in trading-off the three dimensions of process performance and represent the best (with context) decision making policy scheme regarding to prioritizing such a large pool of process proposals.

As suggested in section 3.1, these pairs of lower and upper boundaries represents the ranges of conversion equality. In studying Model 3-2, we find that from the perspective of unit scheme, the posed weight restrictions favor Remaining Error most, and mostly favor Life Cycle Duration next though there is a small range within which the Life Cycle Effort has higher conversion "value" (higher conversion value means that, e.g. if 1 unit Duration "equals" more than 1 unit Effort, then Duration has higher conversion value). We can note that Remaining Error has dominant (highest) conversion value over the other two measures: Life Cycle Effort and Life Cycle Duration. Mostly, Life Cycle Duration has higher conversion value than Life Cycle Effort except for a small range. When we divide the range inequality $0.5 =< v_3/v_2 =< 6.5$ to two inequalities: $0.5 =< v_3/v_2 =< 1$, and $1 =< v_3/v_2 =< 6.5$, the preceding argument becomes apparent.

Given the above observations, it is expected that in comparison with Analysis I, the current analysis (Analysis II) would favor Remaining Error most and Life Cycle Effort least. As a result, those process candidates that have relatively superior Remaining Error and Life Cycle Duration would probably achieve higher ranks, at least stay on their original ranks in Analysis I, and those process candidates that have relatively inferior Remaining Error and Life Cycle Duration would probably go down, at most stay on their original ranks in Analysis I.

From linear programming perspective, imposing more restrictions (constraints) to a linear program won't increase the value of objective function in a maximization problem (here DEA is a maximization problem). So it is expected that the individual efficiency scores of 82 process candidates wouldn't increase in Analysis II. Another inference is that in comparison with the results from Analysis I, the efficient frontier in Analysis II will contract, which approaches our objective — to identify the best process candidate.

Appendix II-2 demonstrates the results from this analysis.

Table 3-2 presents the efficiency scores, ranks, and chosen weights of those top 15 process configurations with highest efficiency scores and AS_IS process, based on Appendix II-2.

Table 3-2 Selection of the Results from Analysis II: The Ranks and Efficiency Scores of The Top 15 Process Configurations and AS_IS Process

RANK	CONFIG	RE MEAN	LEFF MEAN	DUR MEAN	Weight - RE $\times 10^{-3}$	Weight - LEFF $\times 10^{-4}$	Weight - DUR $\times 10^{-5}$	Efficiency Score
1	**FFFF**	3.3	7615.9	1776.9	2.334	1.167	5.834	1.0000
2	**CFFF**	3.6	7610.0	1829.0	2.327	1.163	5.816	0.9970
3	**FCFF**	3.7	7604.1	1860.3	2.323	1.162	5.808	0.9956
4	**CCFF**	4.0	7601.4	1897.2	2.318	1.159	5.794	0.9931
5	**NFFF**	4.1	7607.1	1904.0	2.315	1.157	5.786	0.9918
6	**FFCF**	4.1	7634.6	1881.1	2.310	1.155	5.776	0.9900
7	**FCCF**	4.5	7603.2	1955.4	2.307	1.153	5.766	0.9884
8	**CFCF**	4.4	7625.1	1938.9	2.303	1.152	5.759	0.9871
9	**NCFF**	4.6	7622.7	1965.5	2.300	1.150	5.749	0.9854
10	**CCCF**	4.9	7607.8	2006.3	2.296	1.148	5.741	0.9841
11	**CCCC**	6.6	7560.3	2055.7	2.294	1.147	5.734	0.9828
12	**NFCF**	5.0	7614.6	2013.2	2.293	1.147	5.733	0.9827
13	**FFFC**	4.9	7733.9	1843.0	2.285	1.142	5.712	0.9791
14	**FNFF**	4.8	7683.0	1972.2	2.282	1.141	5.704	0.9778
15	**NCCF**	5.4	7629.9	2069.8	2.280	1.140	5.699	0.9769
44	**AS_IS**	10.0	7820.9	2426.9	2.166	1.083	5.415	0.9281

Where: the legend is the same for all column headings as shown for Table 3-1.

We found the following based on this analysis:

1. FFFF is the only process candidate staying on the efficient frontier. The other two process candidates on the efficient frontier in Analysis I but not efficient in current analysis (Analysis II) are FCFF and CCCC. FCFF ranks 3rd and has efficiency as 0.9956. CCCC ranks 11th and has efficiency as 0.9826.

   These results are not surprising based on our prediction at earlier part of this sub-section. First of all FFFF has dominant performance on Remaining Error and Life Cycle Duration, but non-superior Life Cycle Effort among the top process candidates (in Analysis I). As suggested before, Analysis II — weight-restricted DEA will favor most on Remaining Error and second on Life Cycle Duration. So it is not surprising that FFFF runs ahead further in Analysis II.

   CCCC excels in Analysis I because it has extremely good performance on Life Cycle Effort. However, now in Analysis II its advantage on Life Cycle Effort has been restricted to a degree that its efficiency and ranking suffer. The implication is that under the weight scheme elicited in this study for the particular software project, even CCCC saves development effort it is not overall efficient in comparison with FFFF.

   FCFF is not on the efficient frontier any longer, but its efficiency is very close to 1. It can be noted that FCFF has zero weight on Life Cycle Duration in Analysis I but this tactic has been prohibited in current analysis.

   The reason why FCFF and CCCC get lower efficiency scores can be explained by Table 3-3 which is derived from Table 3-1 and Table 3-2, and Table 3-4 which is derived from Table 3-

3. Table 3-3 presents the product of each performance measure and its corresponding weight for process configurations CCCC, FCFF and FFFF, in two analyses. Table 3-4 presents the proportion of the above product (performance measure * weight) over the sum of all the products of input and corresponding weight. The DEA input variables are Remaining Error(RE), Life Cycle Effort(LEFF), or Life Cycle Duration(DUR). Here we let x_1, x_2 and x_3 be the three performance measures RE, LEFF, and DUR, respectively, and v_1, v_2, and v_3 are the weights for RE, LEFF, and DUR, respectively.

Here is an example for computing Table 3-3. The product $x_1 \cdot v_1$, namely RE × RE_Weight, for FFFF in Analysis I in Table 3-3 is equal to $3.3 \times (3.825 \times 10^{-3}) = 0.0126$, 3.3 is the Remaining Error Mean and 3.825×10^{-3} is its corresponding weight, both from Table 3-1.

An example for computing Table 3-4 is the following. The top right data cell in Table 3-4, $\frac{x_3 \cdot v_3}{\sum_{i=1}^{3}(x_i \cdot v_i)}$, is the proportion of DUR × DUR_Weight over the sum of RE × RE_Weight, EFF × EFF_Weight, and DUR × DUR_Weight for CCCC. The ratio is $0.1179/(0.0152+0.8670+0.1179) = 0.12$.

Table 3-3. The "Performance Measure × Weight" Data for CCCC, FCFF and FFFF in Two Analyses

CONFIG	Analysis I			Analysis II		
	$x_1 \cdot v_1$	$x_2 \cdot v_2$	$x_3 \cdot v_3$	$x_1 \cdot v_1$	$x_2 \cdot v_2$	$x_3 \cdot v_3$
CCCC	0.0000	0.9486	0.0514	0.0152	0.8670	0.1179
FCFF	0.0072	0.9927	0.0000	0.0085	0.8834	0.1081
FFFF	0.0126	0.9874	0.0000	0.0076	0.8886	0.1037

Where:

CONFIG	is the abbreviation of "Configuration".
Analysis I	is DEA Analysis I presented in sub-section 3.2.1.
Analysis II	is DEA Analysis II presented in this sub-section.
$x_1 \cdot v_1$	is the product of RE and the DEA weight for RE.
$x_2 \cdot v_2$	is the product of LEFF and the DEA weight for LEFF.
$x_3 \cdot v_3$	is the product of DUR and the DEA weight for DUR.

Table 3-4. The Ratio "(Input × Weight)/Sum of All (Input × Weight)" Data for CCCC, FCFF and FFFF in Two Analyses

CONFIG	Analysis I			Analysis II		
	$\frac{x_1 \cdot v_1}{\sum_{i=1}^{3}(x_i \cdot v_i)}$	$\frac{x_2 \cdot v_2}{\sum_{i=1}^{3}(x_i \cdot v_i)}$	$\frac{x_3 \cdot v_3}{\sum_{i=1}^{3}(x_i \cdot v_i)}$	$\frac{x_1 \cdot v_1}{\sum_{i=1}^{3}(x_i \cdot v_i)}$	$\frac{x_2 \cdot v_2}{\sum_{i=1}^{3}(x_i \cdot v_i)}$	$\frac{x_3 \cdot v_3}{\sum_{i=1}^{3}(x_i \cdot v_i)}$
CCCC	0.0000	0.9486	0.0514	0.0152	0.8670	0.1179
FCFF	0.0072	0.9927	0.0000	0.0085	0.8834	0.1081
FFFF	0.0126	0.9874	0.0000	0.0076	0.8886	0.1037

Where:

$\frac{x_1 \cdot v_1}{\sum_{i=1}^{3}(x_i \cdot v_i)}$ is the proportion: RE × RE_Weight over the sum of all such products which is (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight).

$\frac{x_2 \cdot v_2}{\sum_{i=1}^{3}(x_i \cdot v_i)}$ is the proportion: LEFF × LEFF_Weight over the sum of all such products which is (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight).

$\frac{x_3 \cdot v_3}{\sum_{i=1}^{3}(x_i \cdot v_i)}$ is the proportion: DUR × DUR_Weight over the sum of all such products which is (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight).

From Table 3-4, comparing the ratios in Analysis I and II, it is clear that in Analysis II CCCC is forced to allocate bigger portion of the total inputs to RE and DUR, which are its non-superior performance measures. The same scenario happens on FCFF. In Analysis II, no zero weighting is allowed.

2. CFFF, FCFF, CCFF, NFFF, FFCF, FCCF, CFCF, and NCFF are the sub-set of candidates that are closest to the efficient frontier. In section 3.2.1 we noted that there is a sub-set of candidates among the top 15 list that have Remaining Error Mean below 4.8, and Life Cycle Duration Mean below 2000. Here we found the two sub-sets mentioned above are essentially the same sub-set. The only exception is FCFF in the former sub-set, which is efficient process candidate in Analysis I. From the following Table 3-5, it is clear that all the rankings of this sub-set of candidates have been improved in Analysis II, except FCCF which stay in rank 7th.

Table 3-5. The Ranking Comparison Between Two Analyses for CFFF, CCFF, NFFF, FFCF, FCCF, CFCF, and NCFF

Rank in Analysis	CFFF	CCFF	NFFF	FFCF	FCCF	CFCF	NCFF
I	4	5	6	13	7	11	12
II	2	4	5	6	7	8	9

In contrast, for the comparison sub-set comprised of NCCF, CNCC, CCCF, NCCC and NFCF that have Remaining Error Mean above 4.8, and Life Cycle Duration Mean above 2000, all their rankings are worsen, based on the following Table 3-6. It can be noted that CNCC and NCCC even leave the top 15 list. Their vacancies are filled by FFFC and FNFF, which have relatively better performance on Remaining Error and Life Cycle Duration.

Table 3-6. The Ranking Comparison Between Two Analyses for NCCF, CNCC, CCCF, NCCC, and NFCF

Rank in Analysis	NCCF	CNCC	CCCF	NCCC	NFCF
I	14	15	8	9	10
II	15	30	10	18	12

The above comparison supports our prediction earlier at this sub-section stating that those

process candidates that have relatively superior Remaining Error and Life Cycle Duration would probably achieve higher ranks, at least stay on their original ranks in Analysis I; and those process candidates that have relatively inferior Remaining Error and Life Cycle Duration would probably go down, at most stay on their original ranks in Analysis I. When we further looking at the rest of the whole set of process configurations from Appendix II-1 and II-2, the same kind of stories happen without exception.

3. CFFF becomes the process candidate that is closest to efficient frontier. But given the same consideration in section 3.2.1, that the inspection for the earliest development phase — Functional Specification is very important, CFFF become much less attractive than the efficient candidate FFFF.

   A closer examination at the efficient candidate FFFF and other "near efficient" candidates such as CFFF, FCFF, CCFF and NFFF, finds that on only the performance measure "Life Cycle Effort Mean", FFFF is worse than other candidates by a difference, which is between 5.9 to 14.5 person hours. The differences between FFFF and these candidates on "Remaining Error Mean" and "Life Cycle Duration Mean" are 0.3 ~ 0.8, and 52.1 ~ 127.1 hours, respectively.

   When we refer to the utility function described in section 3.1, we know the differences of 5.9 ~ 14.5 person hours effort and 0.3 ~ 0.8 remaining errors will fall into the first piece utility function, and 52.1 ~ 127.1 hours duration difference will fall into the second piece utility function. If we convert the above differences into util based on the above observation, 5.9 ~ 14.5 person hours effort equals to 11.8 ~ 29 utils, 0.3 ~ 0.8 remaining errors equals to 12 ~ 32 utils, and 52.1 ~ 127.1 hours duration equals to 338.65 ~ 826.15 utils. It is clear that the disadvantage that FFFF has on Life Cycle Effort is overwhelmed by its advantages on both Remaining Error and Life Cycle Duration. Even we use the first piece utility function to assess the util equivalent of 52.1 ~ 127.1 hours duration difference, the results will be 52.1 ~ 127.1 utils, the preceding statement holds. In Chapter 4, we will analyze all process candidates by a simplified PTA method which applies directly the aforementioned utility function. There we will draw the above conclusion in a more rigorous way.

   It's also interesting to follow the above logic to compare FFFF and CCCC. The differences between these two candidates on three performance measures are (CCCC - FFFF): 3.3 remaining error, -55.6 person hours, and 278.8 hours. Again we know FFFF's advantage on Remaining Error and Life Cycle Duration overrides its disadvantage on Life Cycle Effort in benchmarking with CCCC.

   Now that the interesting competitors FCFF and CCCC in Analysis I become obviously less attractive than FFFF, we are confident to say FFFF is the best process candidate among the total 82 process proposals. And it also demonstrates an apparent improvement upon the AS_IS process.

4. In Analysis II, the mean efficiency of total 82 configurations is 0.93001 and the standard deviation of efficiency is 0.046563. Compared to the mean efficiency 0.95719 and the standard deviation of efficiency 0.03267 in Analysis I, Analysis II has a more spread efficiency distribution and the mean shift lower, as shown in the following Figure 3-2. This suggests the introduction of weight restriction has an overall "pulling down" effect on the whole set of process configurations. This is natural because as the management expresses preferences on the weights scheme, fewer configurations meet expectations. It can be noted that the mode occurs at Efficiency = 0.97x, compared to 0.98x in Analysis I.

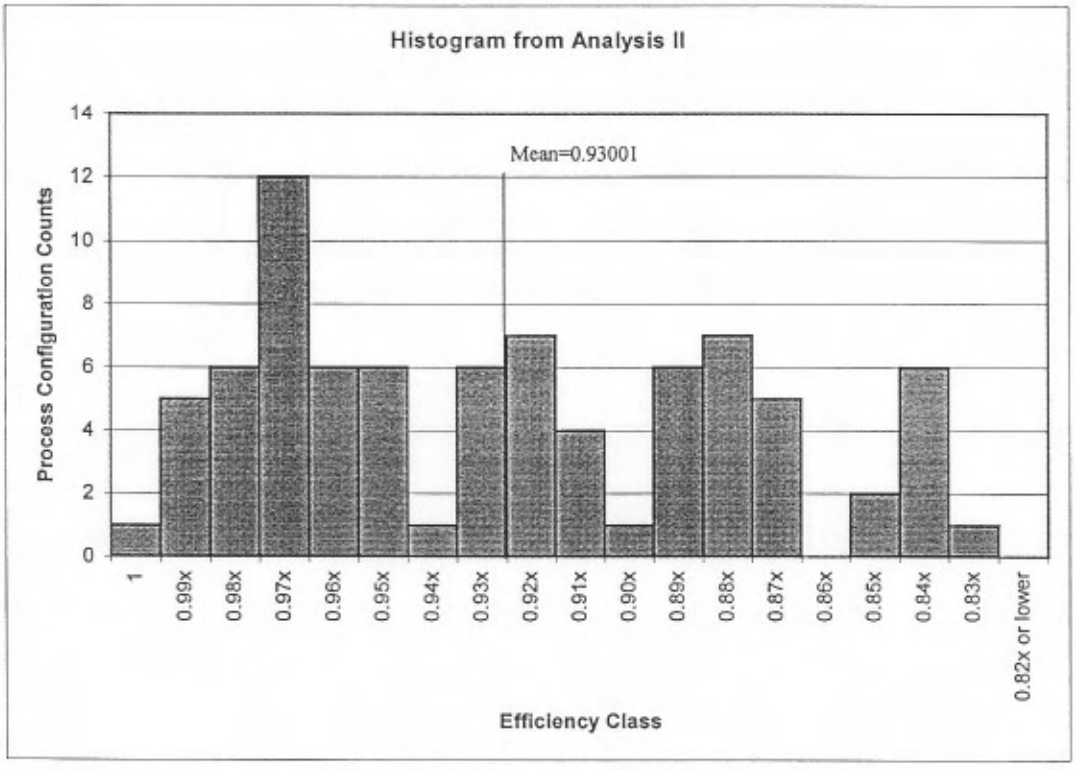

Figure 3-2 Histogram from Analysis II

3.2.3 Analysis III —DEA Incorporated with Standard Deviations

In section 3.2.1 we used a DEA model without weight restrictions to analyze the total 82 process candidates and identify CCCC, FCFF and FFFF being efficient process configurations. The inherent limitation of such a DEA model is that the weights chosen for each performance measure in assessing a DMU (here a DMU is a process configuration) might be unrealistic. As a result, the analysis might be biased due to unacceptable weights selection. In order to remove the possible bias via incorporating management judgements on the weights selection during DEA analysis, we used a weight-restricted DEA model to analyze the total 82 process candidates in section 3.2.2. Then we identify only FFFF being efficient.

Compared to statistical analysis, DEA is an extreme point technique (Anderson 1998). A regular DEA model like Model 3-1 or 3-2 does not deal with the spread of data. In this study, the DEA inputs are the outputs of a discrete simulation model, which are stochastic. During Analysis I and II, we used the mean values of performance measures for each process candidate as the DEA inputs to assess the process efficiency, which accords to a normal DEA approach. However, assessing the process candidates only based on the means of performance measures without considering the spread of data may not be sufficient.

Chance Constrained DEA techniques have been developed to address the problem of dealing with the stochastic type data (Sengupta 1987; Cooper, Huang et al. 1993; Olesen and Petersen 1995). However, these methods still have some difficulties in real world applications and the Chance Constrained approaches have been primarily theoretical explorations insofar.

In this study, we propose a new technique to handle the variation within each individual performance measure of each DMU in a DEA analysis. The idea is that, besides treating the mean value of each performance measure of a DMU as the data for DEA analysis, we also include the standard deviation of each performance measure of each DMU as the model input data. Here by "model input data" we mean the raw data available for DEA analysis, which could be the data values of input variables and/or output variables in a DEA model.

In this case, we include not only the mean values of Remaining Error, Life Cycle Effort, and Life Cycle Duration for each process candidate into the DEA model, but also the standard deviation of each performance measure for each process candidate, namely Standard Deviation(St.D.) of Remaining Error, Standard Deviation(St.D.) of Life Cycle Effort, and Standard Deviation(St.D.) of Life Cycle Duration. The implication is that, a process candidate has to perform overall well on both its mean value measures and its standard deviation measures in order to be efficient. As a result, if a process candidate evaluated as efficient in Analysis II has actually inferior standard deviation(s) of its performance measure(s), it will be "punished" in the analysis following. Or, if a process candidate has superior standard deviation(s) of its performance measure(s), its efficiency score may be improved in the analysis following.

Model 3-3 is shown below. Based on Model 3-2, the Model 3-3 further includes the standard deviations of the input variables in Model 3-2 as input variables. We let x_i (i =1, …, 6) be Remaining Error, Life Cycle Effort, Life Cycle Duration, St.D. of Remaining Error, St.D. of Life Cycle Effort, and St.D. of Life Cycle Duration, respectively. And v_i is the corresponding weight for each x_i.

MODEL 3-3

Maximize $u_1 y_{1j_0}$

Subject to $\sum_{i=1}^{6} v_i x_{ij_0} = 1,$

$$u_1 y_{1j} - \sum_{i=1}^{6} v_i x_{ij} \le 0, \qquad j = 1, .., 82,$$

$$20v_2 \le v_1 \le 48v_2,$$
$$0.5v_2 \le v_3 \le 6.5v_2,$$
$$3.69v_3 \le v_1 \le 40v_3,$$
$$0.1v_1 \le v_4 \le 0.2v_1,$$
$$0.1v_2 \le v_5 \le 0.2v_2,$$
$$0.1v_3 \le v_6 \le 0.2v_3,$$
$$-v_i \le -\varepsilon, \qquad i = 1, .., 6$$
$$-u_1 \le -\varepsilon.$$

(ε is a non-Archimedian infinitesimal.)

The derivation of the restrictions which consist of v_1 & v_2, v_3 & v_2, and v_1 & v_3 are the same as that in the development of Model 3-2. In deriving the restrictions about v_4 & v_1, v_5 & v_2, and v_6 & v_3, we made an arbitrary assumption that the standard deviation of a variable has to have weight only as much as 10% to 20% of the weight of that variable. This pair of percentage boundaries are subject to the judgements of a project manager (or an analyst) on how much s/he is willing to let the standard deviation contribute to the analysis.

Appendix II-3 demonstrates the results from this model. Table 3-7 presents the efficiency scores, ranks, and chosen weights of those top 15 process configurations with highest efficiency scores and AS_IS process, based on Appendix II-3.

Table 3-7 Selection of The Results from Analysis III: The Ranks and Efficiency Scores of The Top 15 Process Configurations and AS_IS Process

RANK	CONFIG	Weight - RE $\times 10^{-3}$	Weight - LEFF $\times 10^{-4}$	Weight - DUR $\times 10^{-5}$	Weight - STD RE $\times 10^{-4}$	Weight - STD LEFF $\times 10^{-5}$	Weight - STD DUR $\times 10^{-6}$	Efficiency Score
1	FFFF	2.326	1.163	5.815	2.326	1.163	5.815	1.0000
2	CFFF	2.318	1.159	5.795	2.318	1.159	5.795	0.9967
3	FCFF	2.314	1.157	5.784	4.627	1.157	5.784	0.9952
4	CCFF	2.309	1.154	5.772	2.309	1.154	5.772	0.9926
5	NFFF	2.306	1.153	5.764	2.306	1.153	5.764	0.9913
6	FFCF	2.301	1.151	5.753	2.301	1.151	5.753	0.9895
7	FCCF	2.297	1.149	5.743	2.297	1.149	5.743	0.9877
8	CFCF	2.294	1.147	5.736	2.294	1.147	5.736	0.9864
9	NCFF	2.290	1.145	5.725	2.290	1.145	5.725	0.9845
10	CCCF	2.286	1.143	5.715	2.286	1.143	5.715	0.9829
11	NFCF	2.283	1.142	5.709	2.283	1.142	5.709	0.9818
12	CCCC	2.282	1.141	5.704	2.282	1.141	5.704	0.9809
13	FFFC	2.276	1.138	5.689	2.276	1.138	5.689	0.9785
14	FNFF	2.273	1.136	5.682	2.273	1.136	5.682	0.9771
15	NCCF	2.269	1.135	5.673	2.269	1.135	5.673	0.9756
44	AS_IS	2.154	1.077	5.385	2.154	1.077	5.385	0.9260

Where:

RANK	is the efficiency ranking of each configuration in this analysis.
CONFIG	is the abbreviation of "Configuration".
Weight -RE	is the weight for RE measure resulting from DEA analysis.
Weight -LEFF	is the weight for LEFF measure resulting from DEA analysis.
Weight - DUR	is the weight for DUR measure resulting from DEA analysis.
Weight -STD RE	is the weight for RE standard deviation measure resulting from DEA analysis.
Weight -STD LEFF	is the weight for LEFF standard deviation measure resulting from DEA analysis.
Weight -STD DUR	is the weight for DUR standard deviation measure resulting from DEA analysis.
Efficiency Score	is efficiency score obtained from DEA analysis.

Based on Table 3-7 and Appendix II-3, we observe the following:

1. The efficiency scores of all process candidates go down slightly (except for FFFF which stays 1).The sub-set of the top 15 process configurations remain the same as the one obtained in Analysis II. And the rankings of most of them remain the same, too. Only CCCC and NFCF trade their rankings in Analysis II and III (current analysis is III). In Analysis II CCCC ranks 11^{th} and NFCF 12^{th} but in Analysis III CCCC 12^{th} and NFCF 11^{th}. This implies NFCF might have better (lower) Standard Deviation over Mean ratio(s), when we compute the ratios

according to Table 2-2, we found NFCF is better (less) than CCCC on two out of three STD/Mean ratios, as shown in the following table. This shows, to some degree, the Model 3-3 has the capability to identify the degree of the spread of the performance measures for a process configuration and take them into account in efficiency evaluation, which provides useful information for the user to make better decision on choosing the right candidate(s).

Table 3-8. The "Standard Deviation / Mean" Ratios of CCCC and NFCF

CONFIG	STD/Mean Ratio		
	RE	LEFF	DUR
CCCC	0.424	0.032	0.159
NFCF	0.460	0.027	0.129

Where:

STD/Mean Ratio	is the ratio: Standard Deviation/Mean.
RE	is the Remaining Error measure.
LEFF	is the Life Cycle Effort measure.
DUR	is the Life Cycle Duration measure.

The reason why the rankings of most of the process configurations remain the same can be explained by calculating the correlation coefficients among the different dimensions of performance measure, as shown in the following table. This table is computed based on the simulation results in Appendix I-1.

Table 3-9. The Correlation Between Mean and Standard Deviation for each Performance Measure

	RE & RE_STD	LEFF & LEFF_STD	DUR & DUR_STD
Correlation Coefficient	0.993	0.897	0.973

Where:

STD/Mean Ratio	is the ratio: Standard Deviation/Mean.
RE&RE_STD	is the correlation coefficient between the RE and RE standard deviation measures.
LEFF& LEFF_STD	is the correlation coefficient between LEFF and LEFF standard deviation measures.
DUR& DUR_STD	is the correlation coefficient between DUR and DUR standard deviation

There are three input variables in the DEA model in Analysis II, which are RE (Remaining Error), LEFF (Life Cycle Effort) and DUR (Life Cycle Duration). In the DEA model in Analysis III, besides the above three input variables, there are three additional input variables which are the RE_STD (standard deviation of Remaining Error), LEFF_STD (standard deviation of Life Cycle Effort), and DUR_STD (standard deviation of Life Cycle Duration).

However, there are very strong correlations between RE and RE_STD (correlation = 0.993), LEFF and LEFF_STD (correlation = 0.897), and, DUR and DUR_STD (correlation = 0.973). In effect, during Analysis III, the input variables RE and RE_STD act essentially as a single variable. The same thing happens on DUR and DUR_STD. The correlation coefficient between LEFF and LEFF_STD is 0.897, which also indicates a strong correlation. Thus

LEFF and LEFF_STD are highly unified in Analysis III, too. In short, even though we include the standard deviations of RE, LEFF, and DUR in the Model 3-3, the effect of such inclusion is not apparent in this case. However, it doesn't mean that this technique is useless. Only in this study for this particular set of data, this technique doesn't demonstrate its full strength. In the preceding example we have already shown that this technique has the capability to identify the degree of the spread of the performance measures for a DMU and take them into account in efficiency evaluation. It is expected that in other applications, if the correlations between mean and standard deviation for the performance measures are not so high as in this study, this technique may demonstrate its strength.

2. In Analysis III, the mean efficiency of total 82 configurations is 0.9281 and the standard deviation of efficiency is 0.04740. The efficiency histogram of Analysis III is shown in Figure 3-3. Compared to Analysis I the mean efficiency 0.9572 and the standard deviation of efficiency 0.03267, and Analysis II the mean efficiency 0.9300 and the standard deviation of efficiency 0.04656, the results in Analysis III show a more spread efficiency distribution and the mean shift further lower. A box plot of the comparison among the three analyses about the efficiency distribution is shown in Figure 3-4. Also it can be noted that the mode of the results of Analysis III occurs at Efficiency = 0.97x.

   Table 3-10 shows the Spearman rank correlations between the efficiency ranking results of analyses I, II and III. The extremely strong Spearman rank correlation between II and III (0.999) explained why most of the process configurations have the same rankings in Analysis II and III. There are also strong Spearman rank correlations between I and II, as well as I and III.

Table 3-10. The Spearman Rank Correlations between the Efficiency Results of DEA Analysis I, II and III

Analysis	I	II	III
I	1	0.977	0.976
II		1	0.999
III			1

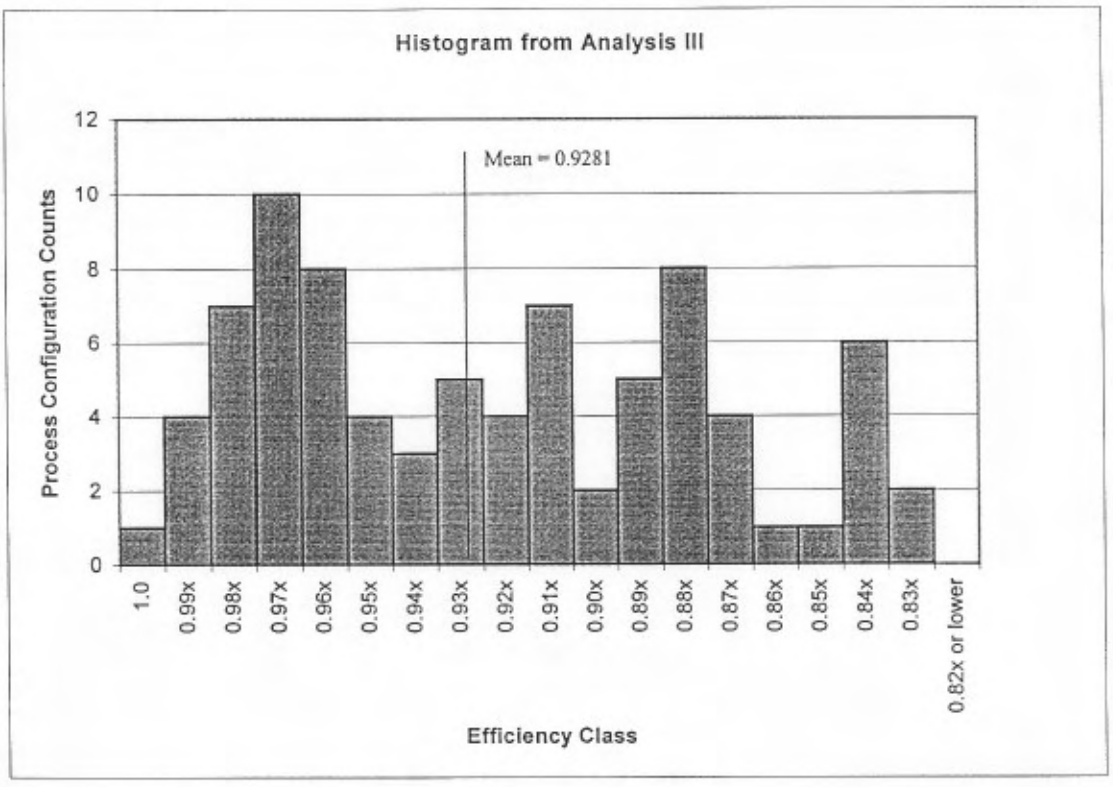

Figure 3-3. Histogram from Analysis III

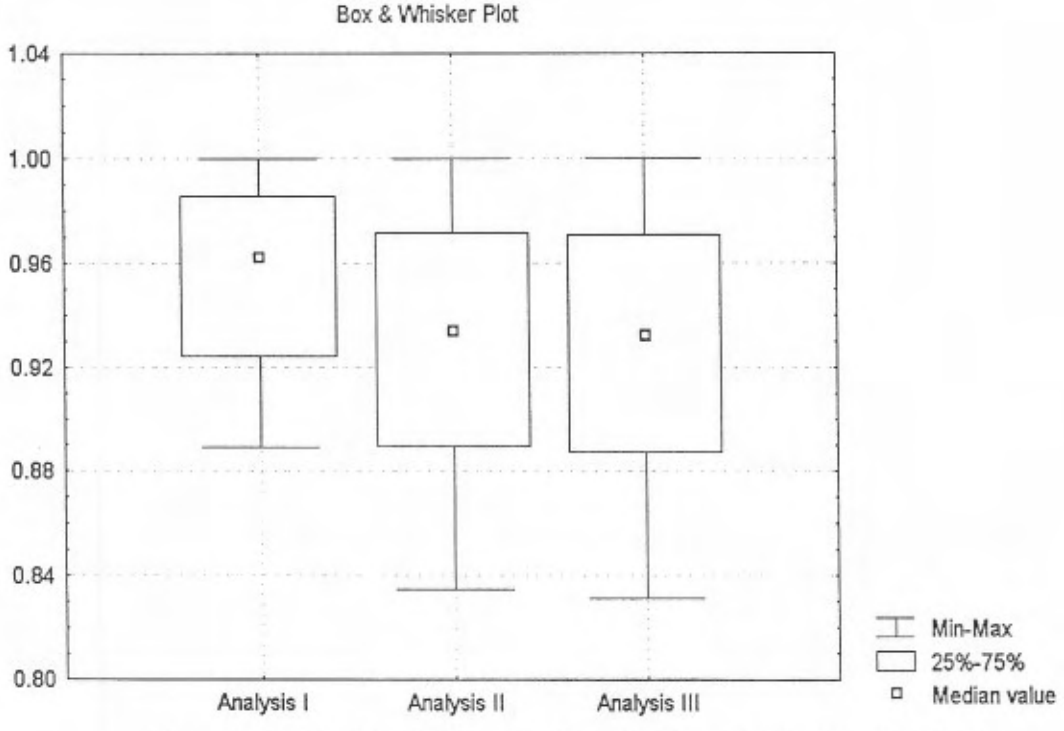

Figure 3-4. The Box Plot of the Results from Analysis I, II and III

3. The DEA model (Model 3-1) used in Analysis I have three input variables, namely Remaining Error Mean, Life Cycle Effort Mean, and Life Cycle Duration Mean.. No weight restrictions in this Model. The Model 3-2 used in Analysis II has the same three input variables as in Model 3-1 but also has weight restrictions on the weights for these three variables. The Model 3-3 used in Analysis III add the respective standard deviations of the three performance measures and has weight restrictions on the weights for these six variables.

   In Analysis III, the mean efficiency of total 82 configurations is 0.9281, which is lower than 0.9572 the mean efficiency in Analysis I, and 0.9300 the mean efficiency in Analysis II. We have already explained the fact that the mean efficiency in Analysis II is lower than the mean efficiency in Analysis I, is caused by posing the weight restrictions on the aforementioned three input variables. In section 3.2.2 we show that in Analysis II not only the efficiency frontier contracts to only FFFF (compared to CCCC, FCFF and FFFF in Analysis I) but also the overall efficiency scores are lowered. Also we explained in section 3.2.2, why the efficiency scores of CCCC and FCFF in Analysis II are lowered than in Analysis I, as particular examples of this efficiency down trend from Analysis I to II.

   However, the cause of the efficiency down trend from Analysis II to Analysis III is not apparent. By incorporating additional variables(the respective standard deviations of the three performance measures) in Analysis III, we expect that the overall efficiency scores are more likely increased from the efficiencies in Analysis II.

   In order to explain the efficiency down trend from Analysis II to Analysis III, we conduct another "intermediate" analysis (Analysis II+) which include the same set of variables as in Analysis III but with weight restrictions only on RE, LEFF and DUR (not on St.D. of RE, St.D. of LEFF, and St.D. of DUR). The weight restrictions on RE, LEFF and DUR are the same as those in Analysis II and III. The descriptions of the four models are shown in the following Table 3-11. The results from DEA Analysis II+ are provided in Appendix II-2+.

Table 3-11. The Descriptions of the DEA Models Used in DEA Analysis I, II, II+, and III

Analysis	I	II	II+	III
Input Variables	RE, LEFF, DUR	RE, LEFF, DUR	RE, LEFF, DUR, RE_STD, LEFF_STD, DUR_STD	RE, LEFF, DUR, RE_STD, LEFF_STD, DUR_STD
Weights	v_1, v_2, v_3	v_1, v_2, v_3	$v_1, v_2, v_3, v_4, v_5, v_6$	$v_1, v_2, v_3, v_4, v_5, v_6$
Restrictions On	N/A	v_1, v_2, v_3	v_1, v_2, v_3	$v_1, v_2, v_3, v_4, v_5, v_6$
Mean Efficiency	0.9572	0.9300	0.9301	0.9281

   Based on Table 3-11, it is clear that when Analysis II+ includes new variables RE_STD, LEFF_STD and DUR_STD, the mean efficiency slightly increases to 0.9301, from the mean efficiency as 0.9300 in Analysis II. This is expected. However, when weights restrictions are posed on these newly added variables in Analysis III, the mean efficiency is lowered to 0.9281, which is understandable because it is a natural effect of posing weight restrictions.

   A comparison between Appendix II+ and III finds that, the efficiencies of all configurations except FFFF decrease from Analysis II+ to III. FFFF stays on 1 simply because FFFF dominates all 6 input variables over all other process configurations.

3.3 A Proposed Procedure for Eliciting Weights for DEA Modeling

Evaluating whether a proposed software process is better than the AS-IS process with respect to three performance dimensions — quality, effort, and schedule, is a multi-criteria decision problem in nature. A widely-used method to handle such a multi-criteria decision problem is to establish a utility function, which allows the decision maker to calculate a composite index *utility* for each alternative with multi-dimensional outcomes. Then by comparing the magnitude of such index values obtained by the respective alternatives, the decision maker is able to benchmark these alternatives. The reader is referred to (Stigler 1968) for an overview of the development of utility theory and (Cleland 1988) for a description of how to apply the utility function in a multi-criteria decision problem.

When we consider a three criteria case, a utility function can take the form as:

$$V_j = f(X_{1,j}, X_{2,j}, X_{3,j}), \qquad (3\text{-}2)$$

where V_j is the utility value of alternative j. $X_{1,j}$ represents the outcome of alternative j on criterion 1, $X_{2,j}$ represents the outcome of alternative j on criterion 2, and $X_{3,j}$ represents the outcome of alternative j on criterion 3. In order to use the above equation (3-2), the weights for $X_{1,j}$, $X_{2,j}$, and $X_{3,j}$ need to be provided by the decision maker and equation (3-2) need take either an additive form or a multiplicative form. If an additive form is used, equation (3-2) becomes the following:

$$V_j = (X_{1,j} \times W_1) + (X_{2,j} \times W_2) + (X_{1,j} \times W_3), \qquad (3\text{-}3)$$

where W_1, W_2, and W_3 are the weights for $X_{1,j}$, $X_{2,j}$, and $X_{3,j}$, respectively. The reader is referred to (Siden 1979) for the derivation and application of the above utility function.

When we let $X_{1,j}$ be the incremental change in the number of Remaining Errors between a proposed process and the AS-IS process, $X_{2,j}$ the incremental change in the hours of staff effort, and $X_{3,j}$ the incremental change in the hours of project duration, the above equation (3-3) represents the form of the utility function shown in section 3.1.3, developed by Raffo (Raffo 1996).

In (Raffo 1996), Raffo developed the aforementioned piece-wise linear utility function. Since each piece of the utility function determines a unique set of weights for the performance measures, it is readily to obtain the lower and upper boundaries of the weight for each performance measure by viewing these pieces as a whole. And the range of the weight for each performance measure is needed for DEA model with weight restrictions in this study.

Raffo's approach in deriving the weights is to interview with software project managers, let them trade off two performance measures at a time, and finally integrate their preferences on weighting three performance measures into three pieces of a unified utility function.

Based on the approach used in (Raffo 1996), we develop a procedure to elicit weight restrictions for DEA analysis on a software project. The procedure is described below.

Assume RE designates the number of remaining errors, EFF the number of person hours of staff effort, and DUR the number of hours of project duration, ΔRE the incremental change in the number of Remaining Errors between a proposed process and the AS-IS process, ΔEFF the

incremental change in the hours of staff effort, and ΔDUR the incremental change in the hours of project duration.

Step 1: Starting to identify critical ranges. This task should be carried on through steps 1, 2, 3, 4 and 5;
Step 2: Working with project manager to compare ΔRE and ΔEFF by following a pair-wise comparison process, come up with the relationship equations for each critical range;
Step 3: Working with project manager to compare ΔRE and ΔDUR by following a pair-wise comparison process, come up with the relationship equations for each critical range;
Step 4: Working with project manager to compare ΔEFF and ΔDUR by following a pair-wise comparison process, come up with the relationship equations for each critical range;
Step 5: Reconciling the disagreement among the three range schemes developed during step 2, 3, and 4. Finalizing the partitioning of critical ranges to a unique scheme;
Step 6: Based on the relationship equations developed in steps 2, 3, and 4, determining the weight restrictions on ΔRE, ΔEFF, and ΔDUR;
Step 7: Using the weights developed for ΔRE, ΔEFF, and ΔDUR to approximate the weights for RE, EFF, and DUR.

We will use step 2 — comparing RE and EFF as an example to show how the relationship equations are derived.

Step 2-1. The analyst will ask the project manager the first question: how many staff hours would you be willing to spend in order to eliminate one delivered error to the customer, if there would be no change in schedule? In responding to this question, the manager may answer:

for ΔRE, ΔEFF in range 0=<ΔRE=<dre1, and 0=<ΔEFF=<deff1, $a_{1,1}*\Delta RE = b_{1,1}*\Delta EFF$;

for ΔRE, ΔEFF in range dre1<ΔRE=<dre2, and/or
deff1<ΔEFF=<deff2 but excluding either ΔRE > dre2, or ΔEFF > deff2, $a_{1,2}*\Delta RE = b_{1,2}*\Delta EFF$;

for ΔRE, ΔEFF in range dre2<ΔRE, and/or deff2<ΔEFF $a_{1,3}*\Delta RE = b_{1,3}*\Delta EFF$,

where ΔRE designates the change in the number of error, ΔEFF designates the change in hours of staff effort, dre1 and dre2 are critical values which divide the range of the number of remaining errors, deff1 and deff2 are critical values which divide the range of the change in staff effort, and $a_{1,1}$, $a_{1,2}$, $a_{1,3}$, $b_{1,1}$, $b_{1,2}$, and $b_{1,3}$ are the coefficients regarding to the judgements made about the relative values of remaining error and staff effort. The first subscript of $a_{1,1}$, $a_{1,2}$, $a_{1,3}$, $b_{1,1}$, $b_{1,2}$, and $b_{1,3}$ indicates that this is the 1st round inquiry on trading off between ΔRE and ΔEFF, and the second subscript indicates within which critical range one of the above equations is drawn.

Here, it should be noted that even though it is possible to use only one coefficient to represent the relationship between, such as $\Delta RE = h_{1,1}*\Delta EFF$, we choose to use two coefficients to keep all coefficients as integers. For instance, in a case when a manager say he is willing to spend additional 3.5 staff hours in order to eliminate one delivered error to the customer, we have $a_{1,1}$ =

2 and $b_{1,1} = 7$. In a case when a manager say he is willing to spend additional 15 staff hours in order to eliminate one delivered error to the customer, we have $a_{1,1} = 1$ and $b_{1,1} = 15$.

According to (Raffo 1996), it was found from the interviews that the weight managers put on a given dimension of performance depended upon the magnitude of the change in that particular measure. For example, managers were not concerned about expending additional staff effort by less than 20 person hours (equal to 0.5 person week) in order to remove one error. Reducing errors(by 1) might be deemed more important than expending this amount of staff effort. However, managers do have different answers in responding to how many person hours of staff effort is equivalent to removing one error given that schedule is not changed, when larger number of errors or magnitude of staff hours are concerned. Raffo (Raffo 1996) found a tendency that the higher range of errors is reached, the more additional staff effort is a manager willing to expend, which is understandable because when large number of errors are involved, the quality of the software to be released becomes more critical than saving staff effort.

Step 2-2. To check the consistency of the respondent during the above elicitation, the analyst will ask the project manager the second question: how many extra errors would you be willing to ship in order to reduce the amount of habitual overtime people work by 2, 5, 10 hours per week? Based on what the manager says, the analyst may develop the following equations:

for ΔRE, ΔEFF in range $0 =< \Delta RE =< dre1$, and $0 =< \Delta EFF =< dre1$, $\quad a_{2,1} * \Delta RE = b_{2,1} * \Delta EFF$;

for ΔRE, ΔEFF in range $dre1 < \Delta RE =< dre2$, and/or $deff1 < \Delta Y =< deff2$

(but excluding either $\Delta RE > dre2$, or $\Delta EFF > deff2$), $\quad a_{2,2} * \Delta RE = b_{2,2} * \Delta EFF$;

for ΔRE, ΔEFF in range $\Delta RE > dre2$, and/or $\Delta EFF > dre2$, $\quad a_{2,3} * \Delta RE = b_{2,3} * \Delta EFF$,

where ΔRE designates the change in the number of error, ΔEFF designates the change in hours of staff effort, dre1 and dre2 are critical values which divide the range of the number of remaining errors, deff1 and deff2 are critical values which divide the range of the staff effort, and $a_{2,1}$, $a_{2,2}$, $a_{2,3}$, $b_{2,1}$, $b_{2,2}$, and $b_{2,3}$ are the coefficients regarding to the judgements made about the relative values of remaining error and staff effort. The first subscript of $a_{2,1}$, $a_{2,2}$, $a_{2,3}$, $b_{2,1}$, $b_{2,2}$, and $b_{2,3}$ indicates that this is the 2nd round inquiry on trading off between ΔRE and ΔEFF, and the second subscript indicates within which critical range one of the above equations is drawn.

Then the third question would be: how would you reconcile the inconsistency

between $a_{1,1} * \Delta RE = b_{1,1} * \Delta EFF$ and $a_{2,1} * \Delta RE = b_{2,1} * \Delta EFF$, for range $0 =< \Delta RE =< dre1$, and $0 =< \Delta EFF =< deff1$? and

between $a_{1,2} * \Delta RE = b_{1,2} * \Delta EFF$ and $a_{2,2} * \Delta RE = b_{2,2} * \Delta EFF$ for range $dre1 < \Delta RE =< dre2$, and/or $deff1 < \Delta EFF =< deff2$ (excluding either $\Delta RE > dre2$, or $\Delta EFF > deff2$)? and

between $a_{1,3}$*ΔRE = $b_{1,3}$*ΔEFF and

$a_{2,3}$*ΔRE = $b_{2,3}$*ΔEFF for range ΔRE > dre2, and/or ΔEFF > deff2?

It is recommended that when the above range schemes developed during steps 2-1 and 2-2 are identical. If not, the analyst should question the underlying rationale and reconcile the discrepancy.

By resolving the inconsistencies, finally the analyst and the manager will be able to come up with:

A1*ΔRE = B1*ΔEFF for range 0=<ΔRE=<dre1, and 0=<ΔEFF=<deff1;

A2*ΔRE = B2*ΔEFF for range dre1<ΔRE=<dre2, and/or deff1<ΔEFF=<deff2, but excluding either ΔRE > dre2, or ΔEFF > deff2;

A3*ΔRE = B3*ΔEFF for range ΔRE> dre2, and/or ΔEFF >deff2.

A1 and B1, A2 and B2, and A3 and B3 are coefficients resulting from resolving the inconsistencies.

Step 3 will yield:

C1*ΔRE = D1*ΔDUR for range 0=<ΔRE=<dre1', and 0=<ΔDUR=<ddur1,

C2*ΔRE = D2*ΔDUR for range dre1'< ΔRE=<dre2', and/or ddur1<ΔDUR=<ddur2, but excluding either ΔRE > dre2', or ΔDUR > ddur2;

C3*ΔRE = D3*ΔDUR for range ΔRE> dre2', and/or ΔDUR>ddur2,

where ΔDUR is the change in the project duration. Similar to the above, dre1', dre2', ddur1, and ddur2 are critical values for ranges. C1 and D1, C2 and D2, and C3 and D3 are coefficients resulting from resolving the inconsistencies.

Step 4 will yield:

E1*ΔEFF = F1*ΔDUR for range 0=<ΔEFF=<deff1', and 0=<ΔDUR=<ddur1',

E2*ΔEFF = F2*ΔDUR for range deff1'<ΔEFF=<deff2', and/or ddur1'<ΔDUR=<ddur2', but excluding either ΔEFF > deff2', or ΔDUR > ddur2';

E3*ΔEFF = F3*ΔDUR for range ΔEFF > deff2', and/or ΔDUR > ddur2'.

Similar to the above, deff1', deff2', ddur1', and ddur2' are critical values for ranges.E1 and F1, E2 and F2, and E3 and F3 are coefficients resulting from resolving the inconsistencies.

At Step 5, the differences between dre1 and dre1', dre2 and dre2', deff1 and deff1', deff2 and deff2', ddur1 and ddur1', and ddur2 and ddur2' need to be reconciled. The goal for this step is to

develop the relationships under the following unique critical range scheme (now assuming that, finally the critical ranges of ΔRE is divided by dre1 and dre2, the critical ranges of ΔEFF is divided by deff1 and deff2, and the critical ranges of ΔDUR is divided by ddur1 and ddur2):

i)for range 0=<ΔRE=<dre1, 0=<ΔEFF=<deff1, and 0=<ΔDUR=<ddur1, we have:
A1*ΔRE = B1*ΔEFF, C1*ΔRE = D1*ΔDUR, and E1*ΔEFF = F1*ΔDUR;

ii)for range dre1<ΔRE=<dre2, deff1<ΔEFF=<deff2 and/or ddur1<ΔDUR=<ddur2, but excluding either ΔRE > dre2, or ΔEFF > deff2, or ΔDUR>ddur2, we have:
A2*ΔRE = B2*ΔEFF, C2*ΔRE = D2*ΔDUR, and E2*ΔEFF = F2*ΔDUR;

iii)for range ΔRE > dre2, and/or ΔEFF > deff2, and/or ΔDUR>ddur2, we have:
A3*ΔRE = B3*ΔEFF, C3*ΔRE = D3*ΔDUR, and E3*ΔEFF = F3*ΔDUR.

Then, by solving the relationship equations within each range, we have the following:

i)for range 0=<ΔRE=<dre1, 0=<ΔEFF=<deff1, and 0=<ΔDUR=<ddur1, we have:

ΔRE/ΔEFF= B1/A1 or (E1*D1)/(C1*F1),
ΔRE/ΔDUR = D1/C1 or (F1*B1)/(A1*E1),
ΔEFF/ΔDUR = F1/E1 or (F1*B1)/(A1*E1).

ii)for range dre1<ΔRE=<dre2, deff1<ΔEFF=<deff2 and/or ddur1<ΔDUR=<ddur2, but excluding either ΔRE > dre2, or ΔEFF > deff2, or ΔDUR>ddur2, we have:

ΔRE/ΔEFF= B2/A2 or (E2*D2)/(C2*F2),
ΔRE/ΔDUR = D2/C2 or (F2*B2)/(A2*E2),
ΔEFF/ΔDUR = F2/E2 or (F2*B2)/(A2*E2).

iii)for range dre2<ΔRE, and/or deff2<ΔEFF, and/or ΔDUR>ddur2, we have:

ΔRE/ΔEFF= B3/A3 or (E3*D3)/(C3*F3),
ΔRE/ΔDUR = D3/C3 or (F3*B3)/(A3*E3),
ΔEFF/ΔDUR = F3/E3 or (F3*B3)/(A3*E3).

Studying the above relationship equations across the three ranges, one can easily determine the upper and lower limits for the ratios ΔRE/ΔEFF, ΔRE/ΔDUR and ΔEFF/ΔDUR. Finally we have the following weight restrictions:

p1=< ΔRE/ΔEFF =< p2, and
q1=< ΔRE/ΔDUR =< q2, and
r1=< ΔEFF/ΔDUR =< r2.

The above three inequalities consist of the set of weight restrictions that we are looking for. Since utility function is not needed in a DEA analysis, we can ignore the derivation of a piecewise utility function that is done in Raffo (Raffo 1996).

After interviewing the project manager, the analyst can also interview other project key staff by showing them the relationship equations developed and discussing them, so that the above results can be validated across the project management.

There are a few issues associated the above procedures. As Raffo (Raffo 1996) noted, some important issues are quoted below:

1. Most managers think about trading-off only two performance measures at a time rather than three. So the above derivation procedure is easy to handle for both the respondent (project manager) and the analyst.

2. The managers may feel bound by certain constraints. For example, the number of staff on the project was fixed and the amount of overtime that the staff could be expected to work was limited. Consequently, if a project found itself needing to exceed these staff constraints or quality constraints late in the process, the release date has to be extended. This observation supports the piecewise approach of the above derivations.

   In this study, by allowing DEA to have some flexibility in choosing weights within a range that allow the uncertainty among the different critical ranges, we can achieve a kind of balance view of the assessed process configuration.

3. When making a decision which requires tradeoffs among performance measures, managers typically focused on how the decision would impact on the current phase of development rather than taking a view of the entire life cycle process or project. This implies that the above results from interview may vary across the time frame.

4. Managers felt tradeoff decisions were context sensitive. For example, managers would like to know the type of errors and its possible impact on the customer before s/he trade-off the number of errors with other performance measures.

It should be noted that the division of three ranges as described in this sub-section is not mandatory. In other applications more or less ranges may be appropriate. Finally as mentioned earlier, based on the above procedure, the weights restrictions derived are actually for ΔRE, ΔEFF and ΔDUR not directly for RE, EFF, and DUR. In DEA analysis we use results from this derivation, which is indeed an approximation.

Chapter 4 Comparing DEA to Utility Analysis & DEA Model Sensitivity

In Chapter 3, we identify three efficient process candidates (CCCC, FFFF and FCFF) via a DEA model without weight restrictions and then refine the efficiency frontier to only the FFFF process configuration via a weight-restricted DEA model. The applied weights are derived based on a piecewise utility function that were developed in a previous study (Raffo 1996). The above utility function represents the management judgements on the "prices" of the three performance measures: Remaining Error, Life Cycle Effort and Life Cycle Duration. Thus the weights derived upon it carry the information about the management view on trading-off these three performance measures for the particular software project under research.

Raffo (Raffo 1996) studied a problem on assessing a proposed process (TO_BE process) against an existing one (AS_IS process), via Process Trade-off Analysis (PTA) Method. The PTA Method is also used as an overarching method for this study. In the analysis phase of PTAM, Raffo (Raffo 1996) used utility function and financial measures such as return on investment to justify the selection of one process change. For the analysis phase of PTAM in this study, we used DEA instead of directly using utility function to assess this large scale process prioritizing problem and we found it will be interesting to compare the results from DEA analysis to the analysis directly using utility function, which we called ***Utility Analysis*** in this Chapter. Section 4.1 presents the results from Utility Analysis on the same problem and compare them to the result in Chapter 3.

Another interesting issue remaining in the DEA analyses in Chapter 3 is the model sensitivity to the selection of the model input variables. The three input variables applied in those analyses are Remaining Error, Life Cycle Effort and Life Cycle Duration. All these three measures are among the output variables of the process simulation model. Among the outputs of the process simulation model, there is another variable called Total Effort, which is Life Cycle Effort plus the Field Service Effort. The Field Service Effort is incurred by correcting the defects remaining in the product after it is released to the public.

In the DEA analyses in Chapter 3, we choose Life Cycle Effort as the variable being included in DEA analysis instead of Total Effort. The reason is that by choosing Remaining Error, Life Cycle Effort and Life Cycle Duration, the effort, duration and quality (designated by Remaining Error) measures are taken at the same point when the product development is done and ready to release. This will give all three measures a sound parallelism which makes the inclusion of these three measures as a reasonable input frame for DEA analysis.

For example, DCUT Effort is another process simulation output variable, which means the staff effort used required to execute the Functional Specification through Unit Test operations. DUCT Effort does not include Functional Test Effort, System Test Effort and Field Service Effort. If we select Remaining Error, DCUT Effort, and Life Cycle Effort as the input variables of DEA analysis, the resulting input frame may seem not a meaningful one.

However, one may argue that Total Effort could be a interesting variable for DEA analysis. Total Effort is Life Cycle Effort plus Field Service Effort. The time point when the last field service would be delivered to the customers might not be concerned by the software manager because it's nearly unpredictable due to so much uncertainty after the release, which signals sort of "Total Duration" of the whole project. But the amount of the potential Field Service Support

due to having staff to correct the remaining errors reported at the field site may be quite concerned by the software project manager. Therefore, including Total Effort instead of Life Cycle Effort may be a choice for project manager. In Section 4.2, we present the results of a DEA analysis by treating Total Effort as the effort measure of the process proposal instead of Life Cycle Effort.

4.1 Comparing DEA and Utility Analysis On The Same Prioritizing Problem

Utility Analysis or directly applying utility function, in essence, is to apply fixed-weights scheme to calculate the utility score of each process candidate, as shown by Equation 3-3 described in section 3.3. Then all the process candidates are benchmarked with respect to their utility scores. For all the process candidates evaluated via a same utility function, there is only one unique weight scheme for the performance measures, which means each performance measure has identical weight disregarding which process candidate is under evaluation.

DEA analysis is to obtain the efficiency scores of the process candidates. A DEA model without weight restrictions has unbounded flexibility in choosing weights for the performance measures of a process candidate being assessed. Such unbounded flexibility allows DEA to maximize the efficiency score of an assessed process candidate. In assessing different candidates, such DEA model may end up choosing different weight schemes, which is the key difference between DEA and Utility analyses.

In a DEA model with weight restrictions, the aforementioned flexibility in choosing weights for the performance measures of a process candidate being assessed is bounded by a set of weight restrictions. It's interesting to compare the results from Utility Analysis and DEA analysis with weight restrictions done in Chapter 3.

To apply the utility function (Eq. 3-1), we need to transform the simulation results for all 81 configurations except AS_IS shown in Appendix I into incremental changes based on AS_IS, which means, we subtract Remaining Error, Life Cycle Effort and Life Cycle Duration of all other configurations from corresponding performance measures of AS_IS to obtain each performance measure "delta", as shown in Appendix III-1. For AS_IS itself, RE DELTA = 0, LEFF DELTA = 0 and DUR DELTA = 0. Then we compute utility for each process configuration using the above utility function as shown in Appendix III-1.

The following Table 4-1 presents a selection of the utility results based on Appendix III-1.

Table 4-1 Selection of The Results from Utility Analysis: The Ranks and Utility of The Top 15 Process Configurations and AS_IS Process

Rank	CONFIG	RE	LEFF	DUR	RE DELTA	LEFF DELTA	DUR DELTA	Utility
1	FFFF	3.3	7615.9	1776.9	6.7	205.0	650.1	1826.8
2	CFFF	3.6	7610.0	1829.0	6.4	210.9	598.0	1714.1
3	FCFF	3.7	7604.1	1860.3	6.3	216.8	566.6	1652.5
4	CCFF	4.0	7601.4	1897.2	5.9	219.5	529.7	1564.1
5	FFCF	4.1	7634.6	1881.1	5.9	186.3	545.8	1560.2
6	NFFF	4.1	7607.1	1904.0	5.9	213.9	522.9	1542.9
7	FFFC	4.9	7733.9	1843.0	5.1	87.0	583.9	1498.7
8	CFCF	4.4	7625.1	1938.9	5.5	195.8	488.1	1436.9
9	FCCF	4.5	7603.2	1955.4	5.5	217.7	471.5	1422.8
10	NCFF	4.6	7622.7	1965.5	5.3	198.2	461.4	1376.4
11	FCFC	5.5	7715.5	1908.9	4.5	105.4	518.1	1355.6
12	CFFC	5.3	7727.3	1908.1	4.6	93.6	518.9	1353.2
13	CCCF	4.9	7607.8	2006.3	5.0	213.1	420.6	1296.2
14	FNFF	4.8	7683.0	1972.2	5.2	137.9	454.7	1295.0
15	NFCF	5.0	7614.6	2013.2	5.0	206.4	413.7	1271.9
50	AS_IS	10.0	7820.9	2426.9	0.0	0.0	0.0	0.0

Where:

RANK	is the efficiency ranking of each configuration in this analysis.
CONFIG	is the abbreviation of "Configuration".
RE	is the Remaining Error measure.
TEFF	is the Total Effort measure, which includes implementation costs.
DUR	is the Life Cycle Duration Measure.
RE DELTA	is AS-IS's RE measure minus the configuration's RE measure.
LEFF DELTA	is AS-IS's LEFF measure minus the configuration's LEFF measure.
DUR DELTA	is AS-IS's DUR measure minus the configuration's DUR measure.
Utility	the "util" value computed from utility function.

Comparing Analysis II — DEA analysis with weight restrictions — and Utility Analysis, we have the following findings:

1. There is considerable similarity between the results from these two analyses. When we compare Table 3-2 with Table 4-1, we found that among the top 15 process configurations, 13 out of them appear in both two tables. Moreover, it can be noted that the ranking of a process configuration in one analysis is similar to its ranking in another analysis.

   We compute the Spearman rank coefficient for the rankings of all 82 configurations from Utility Analysis and the rankings of the same series of configurations from DEA Analysis II, which is 0.977. This result suggests an very strong correlation and explains why the rankings of all process configurations come from two analyses are similar.

2. Both analyses show that FFFF is the best process candidate. Other top candidates agreed by both analyses include CFFF, FCFF and CCFF. The AS_IS process ranks 44th in DEA Analysis II and 50th in PTA analysis, which are comparable.

3. CCCC ranks 11th in DEA Analysis II and 19th in Utility Analysis. From the DEA analyses of Chapter 3 we know that CCCC achieves higher ranking because CCCC has super performance on Life Cycle Effort. In DEA model without weight restrictions (Analysis I), DEA allows CCCC to take fully advantage of its super performance on Life Cycle Effort so it gets efficiency = 1. Later in the weight-restricted DEA model (Analysis II) CCCC is restricted from freely taking advantage of its super performance on Life Cycle Effort then its ranking is lowered down. We found that in Analysis II CCCC is forced to have non-zero weight on Remaining Error and higher weight on Life Cycle Duration, which are its relatively disadvantageous performance measure, and then as a result CCCC has lower ranking in Analysis II. Now in Utility Analysis CCCC is further lowered down in ranking, we have reason to doubt in Utility Analysis CCCC was forced again to have even higher weighting on Remaining Error and Life Cycle Duration, which are its relatively disadvantageous performance measures.

   We follow the logic described below to prove that our doubt is true. The derivation of utility for each process configuration is to apply a fixed-weights scheme on assessing each candidate. On its opposite extreme, DEA without weight restrictions is to apply a flexible-weights scheme on assessing each candidate. And weight-restricted DEA is to apply a partially-flexible-weight scheme on assessing each candidate.

   In any of the above DEA analyses, following through a maximization process specified by the DEA model applied, a DMU (here is each process configuration) will finally be assigned a weight scheme in its most favorable light given that all the restrictions applied in the DEA model were conformed. Here we called this resulting weight scheme as the executive weight scheme for a DMU in the DEA analysis.

   Now in a Utility Analysis, a process candidate is forced to be using one of the three fixed weight schemes (determined via three pieces of the utility function, respectively) on computing its utility. Each piece of the utility function determines the utility scores of a sub-set of 82 process configurations. Using which piece of the utility function for which process configuration is dependent on which range a process configuration falls into.

   We can convert the fixed-weight scheme applied in the Utility Analysis into a DEA format and compare it to the executive weighting scheme applied in DEA Analysis II then we will find whether it is true that CCCC be forced to have higher weighting on Remaining Error under Utility Analysis than under DEA Analysis II. The following table 4-2 shows this analysis.

Table 4-2 Comparison of the Variable Weighting of Three Performance Measures for Process Configuration CCCC Under DEA Analysis I, II and Utility Analysis

		Executive Weights	Performance	Proportion
DEA I	**RE**	0.000	6.6	0%
	LEFF	0.000125	7560.3	94.8%
	DUR	0.0000250	2055.7	5.2%
DEA II	**RE**	0.00229	6.6	1.5%
	LEFF	0.000115	7560.3	86.7%
	DUR	0.0000573	2055.7	11.8%
Utility Analysis	**RE**	0.0413	3.3	13.6%
	LEFF	0.000861	260.7	22.5%
	DUR	0.00172	371.3	63.9%

Where:

Executive Weights — are i) the weights for RE, LEFF, and DUR measures resulting from DEA analyses; or ii) the formalized weights which are converted from the weights applied in Utility analysis such that (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight) =1.

Performance — are the measures such as RE, LEFF, DUR, RE DELTA, LEFF DELTA, or DUR DELTA. See also Table 4-1 legend.

Proportion — Proportions such as (RE × RE_Weight) / (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight), (LEFF × LEFF_Weight) / (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight), or (DUR × DUR_Weight) / (RE × RE_Weight + LEFF × LEFF_Weight + DUR × DUR_Weight).

In the above table, the "Performance" are either performance measures used in DEA analyses or the performance improvements used in Utility Analysis. The proportion is calculated by dividing the product of a performance measure (or improvements) and its corresponding weight by the sum of the all products of "performance measure (or improvement) × executive weight". The executive weights in DEA and the performance measures can be referred to Appendix II-1 and II-2. The performance improvements can be referred to Appendix III-1.

The Executive Weights for CCCC in Utility Analysis are computed based on the following Table 4-3. The "Executive Weights" for CCCC are transformed from explicit weights shown in the third piece of the utility function (Equation 3-1). This transformation is for conveniently comparing the weights used in Utility Analysis to the executive weights used in DEA Analysis I and II. Here is an example for how we compute the Executive Weight in Utility Analysis based on Table 4-3. The "total inputs" is 48 × 3.3 + 1 × 260.7 + 2 × 371.3 = 1161.7. The Executive Weight for RE is 48/1161.7 = 0.0413.

Table 4-3 The Computation of "Executive Weights" Used for Configuration CCCC in Utility Analysis

	RE	LEFF	DUR
Explicit Weights Applied	48	1	2
Performance Improvements	3.3	260.7	371.3
Executive Weights	0.0413	0.000859	0.00172

From Table 4-2, the proportion on RE increase from 0 under DEA I, to 1.5% under DEA II, and to 13.6% under Utility Analysis. The proportion on DUR increases from 5.2% under DEA I, to 11.8% under DEA II, and to 63.9% under Utility Analysis. In contrast, the proportion on LEFF decreases from 94.8% under DEA I, to 86.7% under DEA II, and to 22.5% under Utility Analysis.

It is clear that from DEA Analysis I, II to Utility Analysis, the weighting(proportion) for CCCC on Remaining Error and Life Cycle Duration continually increase while the weighting(proportion) on Life Cycle Effort continually decreases. From the above table 4-2 it can also be verified that this sum of all products of "performance measure × executive weight" is 1 for both DEA and Utility Analysis. And for both DEA and Utility Analysis, the sum of three weightings (proportion) adds up to 1.

Following the same logic as described above, we will be able to explain the other disagreements between Utility Analysis and weight-restricted DEA analysis (DEA II).

To some degree, one would agree that DEA gives a more "fair" benchmarking for the process candidates, by allowing reasonably limited flexibility on choosing individual weight scheme for the performance measures of each process candidate. For the particular data set in this study, both DEA and Utility Analysis identify the same process candidate FFFF being the best proposal though.

It should be noted that the way we conduct Utility Analysis in this sub-section is simplified from its original version in (Raffo 1996). In (Raffo 1996), only two processes are compared with each other and extensive statistical tests are done for the simulation replications of each process. In this sub-section, we simply use utility function and the mean values of process performance measures to compute utility. However, this treatment is illustrative and enough to allow us to discuss the above issues.

Finally as we mentioned in section 3.1, in the DEA analysis of this study, we have used the weights derived for performance improvements ΔRE, ΔEFF, and ΔDUR to approximate the weights for the absolute number of Remaining Errors, person hours of Life Cycle Effort, and hours of Life Cycle Duration. Given the similarity of the results from DEA analysis and the Utility Analysis, it appears that this approximation doesn't impact our results very much in this study. However, as suggested before, the effect of such approximation is left to be future research.

4.2 Sensitivity of DEA Model To Input Variable Selection

In this section we use the Total Effort, instead of Life Cycle Effort which is used in previous analyses in Chapter 3, to be one of the DEA model input variables and conduct DEA analysis without weight restrictions (Model 3-1 used in Analysis I, section 3.2.1 of Chapter 3). Then the DEA input variables set is Remaining Error Mean, Total Effort Mean, and Life Cycle Duration Mean. It should be noted that, as described in section 2.3, both the Total Effort and the Life Cycle Effort used in all quantitative analyses in this study include implementation costs. The results are shown in Appendix III-2. The following Table 4-4 presents a selection of the results based on Appendix III-2.

Table 4-4 Selection of the Results from DEA Sensitivity Analysis: The Ranks and Efficiency Scores of The Top 15 Process Configurations and AS_IS Process

RANK	CONFIG	RE	TEFF	DUR	Weight - RE $\times 10^{-3}$	Weight - TEFF $\times 10^{-4}$	Weight - DUR $\times 10^{-4}$	Efficiency Score
1	**FFFF**	3.3	8104.9	1776.9	0	1.23	0	1.0000
2	**CFFF**	3.6	8144.0	1829.0	0	1.23	0	0.9952
3	**FCFF**	3.7	8153.1	1860.3	0	1.23	0	0.9941
4	**CCFF**	4.0	8204.4	1897.2	0	1.22	0	0.9879
5	**NFFF**	4.1	8216.1	1904.0	0	1.22	0	0.9865
6	**FFCF**	4.1	8246.6	1881.1	0	1.21	0	0.9828
7	**FCCF**	4.5	8278.2	1955.4	0	1.21	0	0.9791
8	**CFCF**	4.4	8291.1	1938.9	0	1.21	0	0.9775
9	**NCFF**	4.6	8318.7	1965.5	0	1.20	0	0.9743
10	**CCCF**	4.9	8345.8	2006.3	0	1.20	0	0.9711
11	**NFCF**	5.0	8364.6	2013.2	0	1.20	0	0.9690
12	**FNFF**	4.8	8403.0	1972.2	0	1.19	0	0.9645
13	**FFFC**	4.9	8465.9	1843.0	0	0.00	5.43	0.9641
14	**NCCF**	5.4	8445.9	2069.8	0	1.18	0	0.9596
15	**CNFF**	5.2	8466.6	2030.2	0	1.18	0	0.9573
50	**AS_IS**	10.0	9314.9	2426.9	0	1.07	0	0.8701

Where:

RANK is the efficiency ranking of each configuration in this analysis.
CONFIG is the abbreviation of "Configuration".
RE is the mean value of Remaining Error for 50 simulation replications.
TEFF is the mean value of Total Effort for 50 simulation replications and include implementation costs.
DUR is the mean value of Life Cycle Duration for 50 simulation replications.
Weight -RE is the weight for RE resulting from DEA analysis.
Weight -TEFF is the weight for TEFF resulting from DEA analysis.
Weight - DUR is the weight for DUR resulting from DEA analysis.
Efficiency Score is efficiency score obtained from DEA analysis.

The following figure 4-1 shows the efficiency distribution of the process configurations in this analysis.

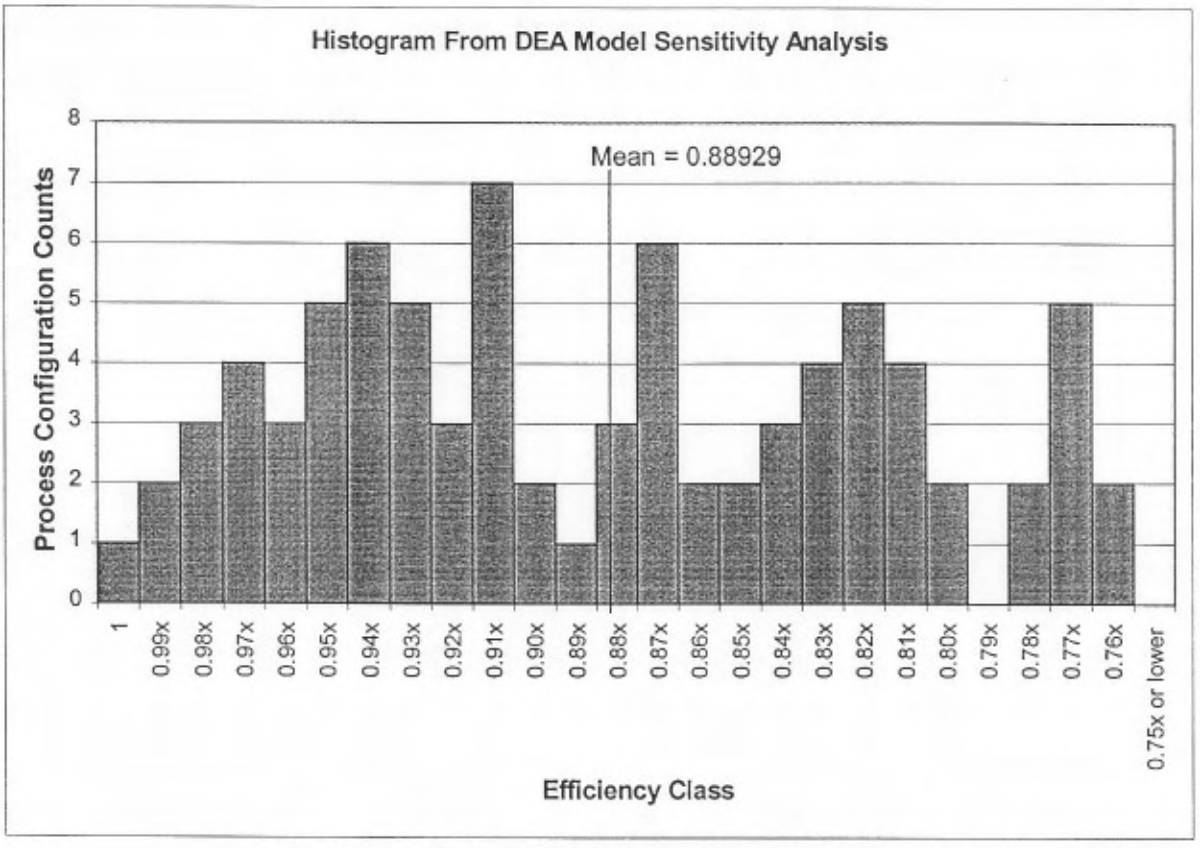

Figure 4-1 Histogram from DEA Sensitivity Analysis

The mean efficiency of the total 82 configurations is 0.88929 and the standard deviation of efficiency is 0.06678. The mode occurs at Efficiency = 0.91x. Compared to DEA Analysis I, II and III in Chapter 3, the efficiency distribution from current analysis has lower mean and wider spread.

From this analysis, FFFF is the only efficient process configuration. A close examination on Appendix III-2 finds the reason is that FFFF dominates all three dimensions of performance measures over all other process configurations. Therefore this result will be robust to any DEA models on this data set. FFFF will again be the only efficient candidate even if we include the standard deviations of these three variables into DEA analysis as we did in section 3.2.3 Analysis III, because FFFF dominates all three dimensions of performance measures and the corresponding standard deviations over all other process configurations. The data for standard deviations can be referred to Appendix I-1.

The implication of this result is that conducting full inspection on all the development phases will eventually save potentially large effort on Field Service Effort due to correcting defects reported at the field site. A process configuration CCCC does have good result on Life Cycle Effort measure, as shown in Analysis I of Chapter 3. But rushing the inspections, as suggested in CCCC, still has negative impact over a longer term. Therefore CCCC does not have advantage on Total Effort. This result may provide good insight for software project management.

Chapter 5 Conclusion

In this chapter, we conclude this study by summarizing the research results, limitations, contributions and future research.

5.1 Results Summary, Limitations and Contributions of This Study

Based on our simulation experiments in Chapter 2, DEA analyses in Chapter 3, and sensitivity analysis in Chapter 4, we summarize the results of this study as following.

In DEA analysis without weight restrictions, process configuration proposals FFFF, CCCC and FCFF are identified as efficient among the total 82 process candidates including the AS-IS process. Among them, FFFF excels by its outstanding performance on all three performance measures, CCCC is Life-Cycle-Effort-saving champion, and FCFF excels by demonstrating a "best mix" of three performance measures.

It is an interesting result that FCFF is efficient. Because it appears to help on addressing a management question like "Can we compress the inspection effort on High Level Design given that we have already done an excellent job(inspection) on its preceding phase, namely Functional Specification?". However, the expected effect, that FCFF has Full inspection in the Functional Specification and thus relieves inspection pressure of HLD inspection, is not apparent based on a comparison of the detailed simulation results for FCFF, NCFF and CCFF.

FFFF outshines all other process candidates during DEA analyses incorporated weights restriction and/or input variable standard deviations, as well as sensitivity analysis. CCCC, even provides excellent result on development effort, but rated as inefficient after a thorough analysis when comparing to FFFF. The validity that FCFF is inefficient still deserves more exploration when other data sets are under research. It could be efficient under a different weight scheme.

The above result, reflects the management perceptions in trading-off three performance measures at a time in this study, which are captured by the weight restrictions derived from an interactive process with the manager. Meanwhile, the benchmarking results from DEA weight-restricted analyses and the results from the simplified Utility analysis agree with each other to a considerable degree.

Moreover, this result supports Fagan's (Fagan 1986) claim that the uses of inspection "in each instance…were found to be cost effective, i.e. it saves more than it cost", in a rigorous form. A process decision maker should be fully aware that the probable negative consequences of doing unstructured or compressed inspection, or skipping inspection.

Sensitivity analysis in this study shows that when Total Effort instead of Life Cycle Effort is taken as an input variable of DEA, the dominant position of FFFF is strengthened. When Life Cycle Effort, Life Cycle Duration and Remaining Error are treated as DEA input variables, CCCC does the best on Life Cycle Effort and three configurations CCCC, FCFF and FFFF are all rated as efficient in DEA Analysis I without weight restrictions. But when instead, Total Effort, Life Cycle Duration and Remaining Error are treated as DEA input variables, FFFF dominates these three performance measures over all other process configurations. This further supports Fagan's view (Fagan 1986) that the inspections when applied appropriately, can provides considerable reduction in corrective maintenance.

Still we found that there are some important limitations of this study. Firstly, there is limitation on simulation model. The simulation model used in this study cannot distinguish between different types of error detected in a given phase. For example, a design error(which means error injected during design phase) found during the coding phase in practice will likely cost more than a coding error. But in current simulation model, they are treated as costing the same. In short, all errors cost the same to repair at a given phase. Although this is a reasonable approximation and was a significant step forward at the time when the model is developed, it does cause a limitation for this study.

Secondly, the DEA analysis currently applied in this study shows that it is a handy tool in identifying the efficient process candidate(s), which meets the need of PTAM for its analysis phase. In this study, we convince ourselves the result by scrutiny and incorporating our judgements. However, DEA allows a more clear assessment about the distance among the efficient vs. inefficient process candidates, i.e. two-phase DEA approach, which is not pursued in this study though.

This study presents an update of PTAM by demonstrating an integral, viable approach on large size process proposal prioritization problem. Potentially it can be used by a company with such needs on prioritizing process configurations.

We propose a procedure to elicit weights for DEA model by using a pair-wise comparison approach. By incorporating variables' standard deviations of each input measurement into DEA model, the DEA model is able to provide more information than using only mean values as inputs in assessing each candidate, which may have significant implication in practice.

5.2 Future Research

In this study, we used the data mostly available in a previous research and extended the power of PTAM on a large size process proposal benchmarking problem. The approach presented in this study can potentially find other real world applications and these applications will in turn help sharpen the approach.

Simulation model needs enrichment by incorporating the capability to take account the errors of different properties. Therefore, the different rework costs for errors with different properties will be reasonably taken into account.

The DEA model incorporating input variable (could be output variable, too) standard deviations shows its capability in identifying the degree of the spread of the performance measures and then taking them account into efficiency evaluation, which provides more useful information than using only mean values. The full potential of this technique deserves further exploration.

As to the weight elicitation procedure proposed in this study, future research should be done to show the effect of approximating the weights for RE, EFF and DUR with the weights directly derived for ΔRE, ΔEFF and ΔDUR.

Acknowledgement

This capstone project is fully supported by Dr. Anderson and Dr. Raffo. I would first express my hearted thanks to the two mentors. I would say thanks to my most respectful mentor, Dr. Kocaoglu, who gave me full support during my graduate study and nurtured me in the discipline of decision science, which helped me to work on an important chapter of this report. Thanks to Don, Mikki, Anne, Liono and Pega who gave me great office support during my study at EMP. I would like also to thank Peter Ghavami who helped me to initiate my research work on this topic. Thanks to the other great EMP classmates who make me feel so warm and happy at EMP. Finally, I would thank my wife, Xiaohong Yang. Without her love and support, this project could not be completed.

References

Abdel-Hamid, T. and S. Madnick (1991). Software Project Dynamics: An Integrated Approach, Prentice-Hall.

Allen, R., A. Athanassopoulos, et al. (1997). "Weights restrictions and value judgements in data envelopment analysis: Evolution, development and future directions." Annals of Operations Research 73: 13-34.

Anderson, T. R. (1998). Productivity Measurement Using Data Envelopment Analysis. Portland, Oregon, Portland State University: 33-35.

Anderson, T. R. and G. P. Sharp (1997). "A new measure of baseball batters using DEA." Annals of Operations Research 73: 141-155.

Banker, R. D., S. M. Datar, et al. (1991). "A model to evaluate variables impacting the productivity of software maintenance projects." Management Science 37(1).

Banker, R. D. and C. F. Kemerer (1989). "Scale economies in new software development." IEEE Transactions on Software Engineering 15(10): 1199-205.

Banker, R. D. and R. C. Morey (1989). Incorporating value judgements in efficiency analysis. Research in Governmental and Nonprofit Accounting, JAI Press. 5: 125-163.

Beasley, J. E. (1990). "Comparing university departments." Omega 18(2): 171-83.

Boehm (1981). "An Experiment in Small Scale Application SOftware Engineering." IEEE Transactions on Software Engineering SE-7, No.5(September 1981).

Chang, Y.-L., T. Sueyoshi, et al. (1996). "Ranking dispatching rules by data envelopment analysis in a job shop environment." IIE Transactions 28(8): 631-642.

Charnes, A., W. W. Cooper, et al. (1990). "Polyhedral cone-ratio DEA models with an illustrative application to large commercial banking." Journal of Econometrics.

Charnes, A., W. W. Cooper, et al. (1978). "Measuring the efficiency of decision making units." European Journal of Operational Research 2(6): 429-44.

Christie, A. M. (1998). Simulation in Support of Process Improvement. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Cleland, D. and Kocaoglu, D (1988). Decision Making Methodologies. Engineering Management. New York, McGraw-Hill: 161-165.

Cook, W. D., M. Kress, et al. (1992). "Prioritization models for frontier decision making units in DEA." European Journal of Operational Research 59(2): 319-23.

Cooper, W. W., Z. Huang, et al. (1993). Stochastic models in data envelopment analysis. ORSA/TIMS Convention, Phoenix, Fall 1993.

Dieters, W. and V. Gruhn (1991). "Software Process Model Analysis based on FUNSOFT Nets." Mathematical Modeling and Simulation 8(May 1991).

Dyson, R. G. and E. Thanassoulis (1988). "Reducing weight flexibility in data envelopment analysis." Journal of the Operational Research Society 39(6): 563-76.

Elam, J. (1991). "Evaluating the efficiency of IS organizations using Data Envelopment Analysis." Proc. of International Function Point Users Group 1990 Fall Conference, October 1991.

Fagan, M. E. (1976). "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal 15(no. 3, 1976).

Fagan, M. E. (1986). "Advances in Software Inspections." IEEE Transactions on Software Engineering SE-12(no. 7, July 1986).

Farrell, M. J. (1957). "The measurement of productive efficiency." Journal of the Royal Statistical Society 120(3): 253-281.

Golany, B. (1988). "A note on including ordinal relations among multipliers in data envelopment analysis." Management Science 34(8): 1029-33.

Kanji, G. K. (1993). 100 Statistical Tests. London, SAGE Publications Ltd: 33.

Kellner, M. (1991). Software Process Modeling Support for Management Planning and Control, Redondo Beach, California, October 21-22, 1991, IEEE Computer Society Press, Los Alamitos.

Kellner, M. I. (1988). Position Paper: Software Process Modeling at SEI. Software Maintenance-1988, Phoenix, Arizona, IEEE Computer Society Press.

Kocaoglu, D., Martin, R., and Raffo, D. (1998). Moving Toward a Unified Continuous and Discrete Event Simulation Model for Software Development. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Lehman, M. (1991). "Software Engineering , The Software Process and Their Support." IEEE Software Engineering Journal 6(n.5, September 1991).

Madachy, R. (1994). A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment. Dept. of Industrial and Systems Engineering. Los Angeles, California, University of South California.

Olesen, O. B. and N. C. Petersen (1995). "Chance constrained efficiency evaluation." Management Science 41(3): 442-57.

Orme, C. and P. Smith (1996). "The potential for endogeneity bias in data envelopment analysis." Journal of the Operational Research Society 47(1): 73-83.

Paradi, J. C., D. N. Reese, et al. (1997). "Applications of DEA to measure the efficiency of software production at two large Canadian banks." Annals of Operations Research 73: 91-115.

Paulk, M. C., Curtis, W., Chrissis, M.B., and Weber, C.V. (1993). Capability Maturity Model for Software, Version 1.1. Pittsburgh, P.A., Software Engineering Institute, Carnegie Mellon University.

Pedraja-Chaparro, F., J. Salinas-Jimenez, et al. (1997). "On the Role of Weight Restrictions in Data Envelopment Analysis." Journal of Productivity Analysis 8(2): 215-230.

Pfahl, D. (1998). Integration of System Dynamics Modeling with Descriptive Process Modeling and Goal-Oriented Measurement. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Powell, A., and Mander, Keith (1998). Strategies for Lifecycle Concurrency and Iteration - A System Dynamics Approach. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Putnam, L. (1980). Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers. New York, The Institution of Electrical Engineers, Inc.

Raffo, D., Kellner, M. (1997). "Evaluating the Performance of Process Alternatives Quatitatively Using Software Processes Models." Working paper, Portland State of University, 1997.

Raffo, D., and Vandeville, Joseph V. (1998). Software Process Simulation to Achieve Higher CMM Levels. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Raffo, D. M. (1996). Modeling Software Processes Quantitatively and Assessing the Impact of Potential Process Changes on Process Performance. Graduate School of Industrial Administration. Pittsburgh, Pennsylvania, Carnegie Mellon University.

Russell, G. W. (1991). "Experience with Inspection in Ultralarge-Scale Developments." IEEE Software Vol. 8(No. 1, Jan. 1991): 25-31.

Scacchi, W. (1998). Experience with Software Process Simulation and Modeling. ProSim'98, Silver Falls, Oregon, June 22-24, 1998.

Seiford, L. M. (1997). "A bibliography for Data Envelopment Analysis (1978-1996)." Annals of Operations Research 73: 393-438.

Sengupta, J. K. (1987). "Data envelopment analysis for efficiency measurement in the stochastic case." Comput. & Oper. Res. (GB) 14(2): 117-29.

Shafer, S. M. and J. W. Bradford (1995). "Efficiency measurement of alternative machine component grouping solutions via data envelopment analysis." IEEE Transactions on Engineering Management 42(2): 159-165.

Shang, J. and T. Sueyoshi (1995). "A unified framework for the selection of a flexible manufacturing system." European Journal of Operational Research 85(2): 297-315.

Siden, J. and Worrell, A., (1979). Analyses Involving Multiple Objectives. Unpriced Values -- "Decision without Market Prices". New York, John Wiley & Sons: 142-154.

Smith, P. (1997). "Model misspecification in Data Envelopment Analysis." Annals of Operations Research 73: 233-252.

Stigler, G. (1968). The Development of Utility Theory. Utility Theory: A Book of Readings. A. Page. New York, John Wiley & Sons: 55-119.

Talluri, S. and J. Sarkis (1997). "Extensions in efficiency measurement of alternate machine component grouping solutions via data envelopment analysis." IEEE Transactions on Engineering Management 44(4): 299-304.

Thanassoulis, E., A. Boussofiane, et al. (1995). "Exploring output quality targets in the provision of perinatal care in England using data envelopment analysis." European Journal of Operational Research 80(3): 588-607.

Thompson, R. G., E. J. Brinkmann, et al. (1997). "DEA/AR profit ratios and sensitivity of 100 large U.S. banks." European Journal of Operational Research 98(2): 213-229.

Thompson, R. G., P. S. Dharmapala, et al. (1994). "DEA ARs and CRs applied to worldwide major oil companies." Journal of Productivity Analysis 5(2): 181-203.

Thompson, R. G., L. N. Langemeier, et al. (1990). "The role of multiplier bounds in efficiency analysis with application to Kansas farming." Journal of Econometrics 46: 93-108.

Thompson, R. G., F. D. Singleton, Jr., et al. (1986). "Comparative site evaluations for locating a high-energy physics lab in Texas." Interfaces 16(6): 35-49.

Tvedt, J. (1995). A Systems Dynamics Model of the Software Inspection Process. Tempe, AZ, Computer Science and Engineering Department, Arizona State University.

Tvedt, J. and J. Collofello (1995). "Evaluating the Effectiveness of Process Improvements on Software Develpoment Cycle Time Via Systems Dynamics Modeling." Proceedings of Compass.

Weller, E. F. (1993). "Lessons from Three Years of Inspection Data." IEEE Software Vol. 10(No. 5, Sept. 1993): 38-45.

Wong, Y.-H. B. and J. E. Beasley (1990). "Restricting weight flexibility in data envelopment analysis." Journal of the Operational Research Society 41(9): 829-35.

List of Appendices