DELAY TIME MODELLING AND SOFTWARE DEVELOPMENT

Xin CUI

Centre of Operational Research and Applied Statistics

School of Accounting, Economics and Management Science

University of Salford, Salford, UK

Submitted in Partial Fulfillment of the Requirements of the Degree of

Master of Philosophy, January 2002

Acknowledgement

The study contained in this report was carried out by the author in School of Accounting, Economics and Management Science at University of Salford.

I would like to thank my supervisor, Dr. Wenbin Wang, for his advice, encouragement and support during the research. Without his excellent supervision, constant motivation, elegant correction and feedback, it would be impossible for me to finish the research work in one year.

Appreciation also goes to Dr. Yunxian Jia and Dr. Xisheng Jia. The discussion with them has been very beneficial to my research.

Finally, a special word of thanks to my wife for her patience and understanding in the year.

Abstract

Delay time modelling (DTM) is the process to establish the mathematical model based on the delay time concept and then to use it for improving plant maintenance management. The delay time model can be divided into a single component model (component-tracking model) or a complex system model (pooled-components model). DTM has been proved to be a methodology readily embraced by engineers for modelling maintenance decisions. The application and research of delay time modelling has come to a stage where a semi-automated tool can be developed. In this thesis, the research on the software development of delay time modelling will be presented.

Firstly, delay time models for both a single component (or component-tracking model) and a complex system (or pooled-components model) are introduced. The key part is delay time parameter estimation, which will be presented in details using available subjective, objective or both.

Secondly, the development of the software package is presented. It includes project analysis, database design, and program design. In the project analysis phase, the delay time models are transformed to program models. All analysis of program models consists of three parts, such as input, processing and output. In the database design phase, some tables are created to store processing information, which is then used in subsequent mathematical modelling. Detailed programming work is given in the program design phase.

The major achievement of this research and an open discussion of future work conclude the thesis.

CONTENTS

Chapter 1 Introduction

1.1 DELAY TIME CONCEPT AND MODELLING

The time to failure of equipment is a function of its inspection and maintenance strategy. To model the inspection interaction, a concept called delay time may be utilised. The delay time concept regards the failure process as a two-stage process. The first stage is measured from new to when a fault could be first identified, if an inspection is carried out. The second stage is a further interval where the faulty component will subsequently lead to a failure or an unacceptable state if there is no maintenance intervention. The period of the time lapse from when a fault could first be noticed until the time of a failure or its repair can be delayed no longer because of unacceptable consequences is called the failure delay time, or delay time for short.

The use of the delay time concept in modelling plant maintenance decision making has been developed over the last two decades with considerable work on case applications and validating. It is noted however, that the applications were limited in cases where experienced delay time modellers have been involved.

Given the extensive application experience and delay time modelling development, it is the time to develop a semi-automated delay time modelling and demonstration tool to be widely accessible by industry without the heavy involvement of modellers.

1.2 SCOPE AND AIM

The aim of this research is to develop a software package on modelling plant maintenance using the delay time concept. It aims to cover major developments made in delay time modelling for both single component and complex system. Substantial work will be on software development, but a certain modelling work needs also to be explored. In particular, delay time model parameters estimation will be investigated using available subjective data, objective data or both.

1.3 OUTLINE OF THE THESIS

The thesis begins with a review of the previous Delay Time Modelling (DTM) and applications in Chapter 2.

In Chapters 3 and 4, aspects of DTM that are to be used in the software will be introduced. The delay time models can be divided into two categories. One is for a single component (or component-tracking model), and the other is for a complex system (or pooled-components model). In these chapters, decision models will be introduced first. The method of how to estimate those parameters involved in the decision model will be reported subsequently in detail.

Chapter 5 is devoted to introducing the software development. It includes project analysis, database design, program design, and the developing work. In the project analysis phase, the delay time models are transformed to program models. All analysis of program models consists of three aspects, input, processing and output. In the database design phase, the input and output information will be sorted and stored in different tables. Some tables are created to store processing information, which is used in mathematical modelling. More programming work will be listed in the program design phase. The internal process of every program are independent of each other, but connected through their output and input data. In the final section of this chapter, some special work, which should be considered in program development, will be introduced.

The major achievement of this research and an open discussion for future work conclude the thesis.

Chapter 2 Delay Time Modeling Review

2.1 INTRODUCTION OF DELAY TIME MODELLING (DTM)

The Delay Time concept and associated modelling work were first introduced by A.H.Christer (1973). The research has been developed for a number of years, and has provided a viable basis for the modelling of inspection process of production plant.

The delay time concept regards the failure process as a two-stage process. Firstly, the initial time u of a defect is the time point when the defect, which has developed within the system, can first be identified if an inspection is carried out at that time. Secondly, if there is no maintenance intervention, the faulty component will subsequently lead to a failure or an unacceptable state after some further interval h. The period of the time lapse from when a fault could first be noticed until the time of a failure or its repair can be delayed no longer because of unacceptable consequences, is called the failure delay time. The concept is illustrated in Figure 2-1.

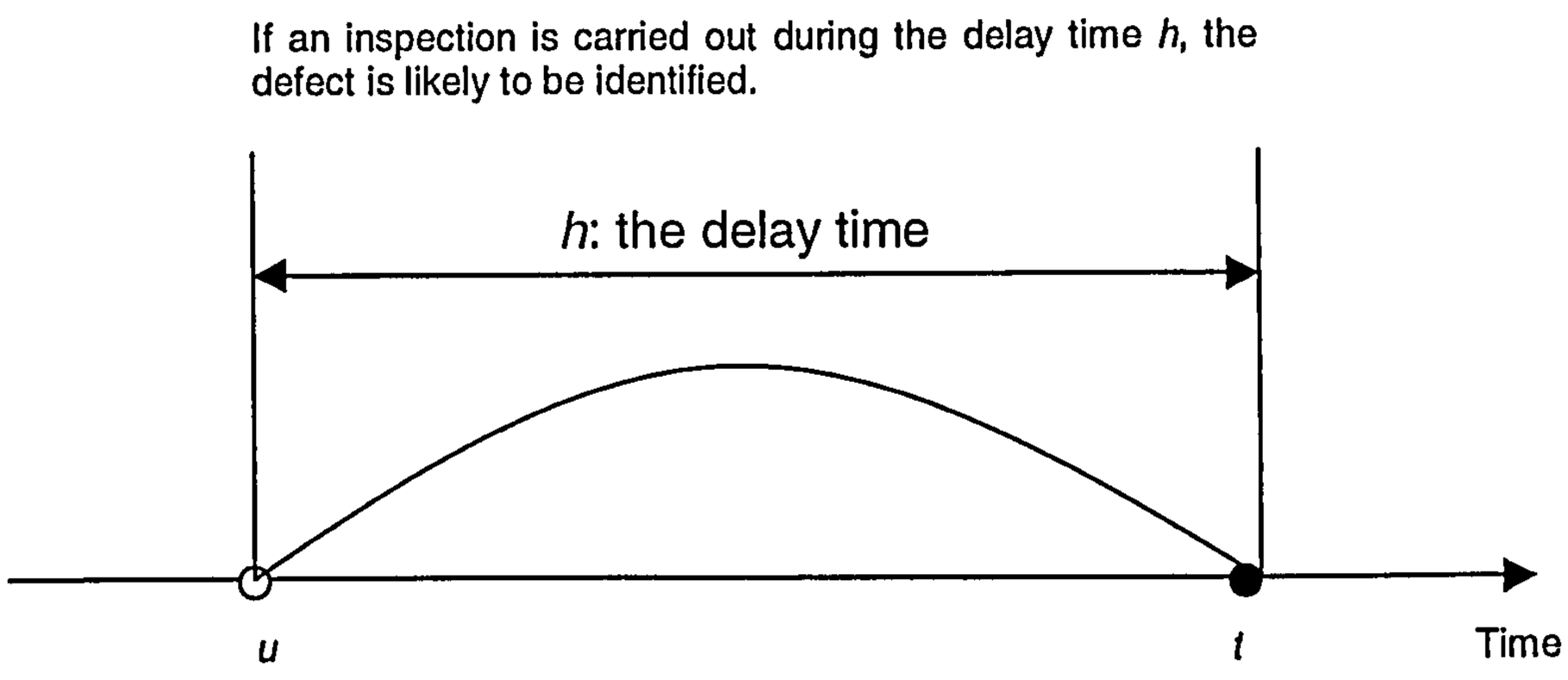


Figure 2-1 The delay time concept

The delay time models can be divided into a single component model (component-tracking model) and a complex system model (pooled-components model).

Preventive maintenance is assumed to consist primarily of an inspection resulting in the replacement or repair of identified faulty components.

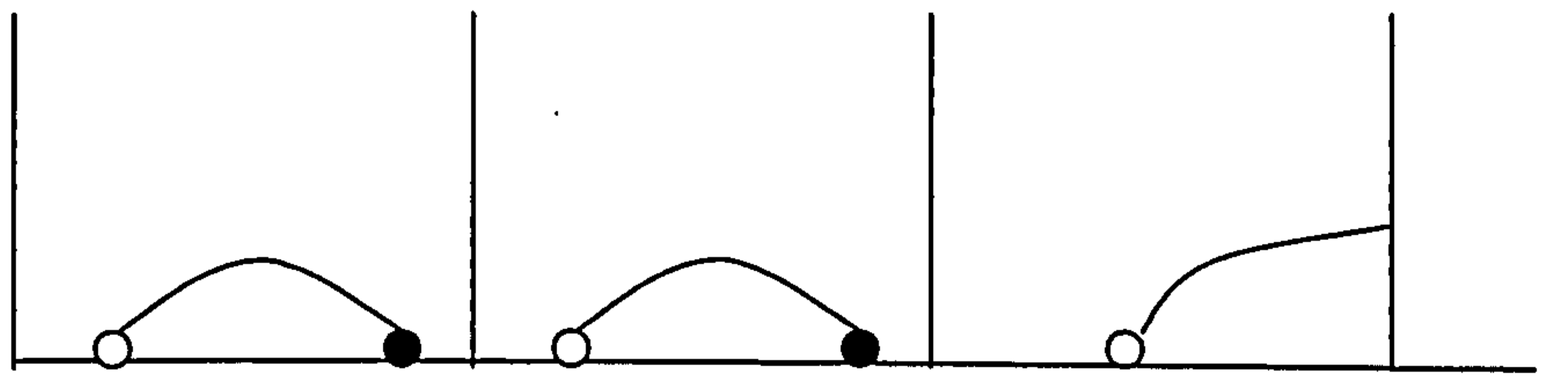


Figure 2-2 DTM applied in single component

Figure 2-2 shows the delay time concept applied to a single component. The circles represent the origination of faults, the dots represent failures, and the vertical lines are inspection points. At the fourth inspection point in Figure 2-2, a fault has been detected and a potential failure is eliminated if corrective action is taken at that inspection point.

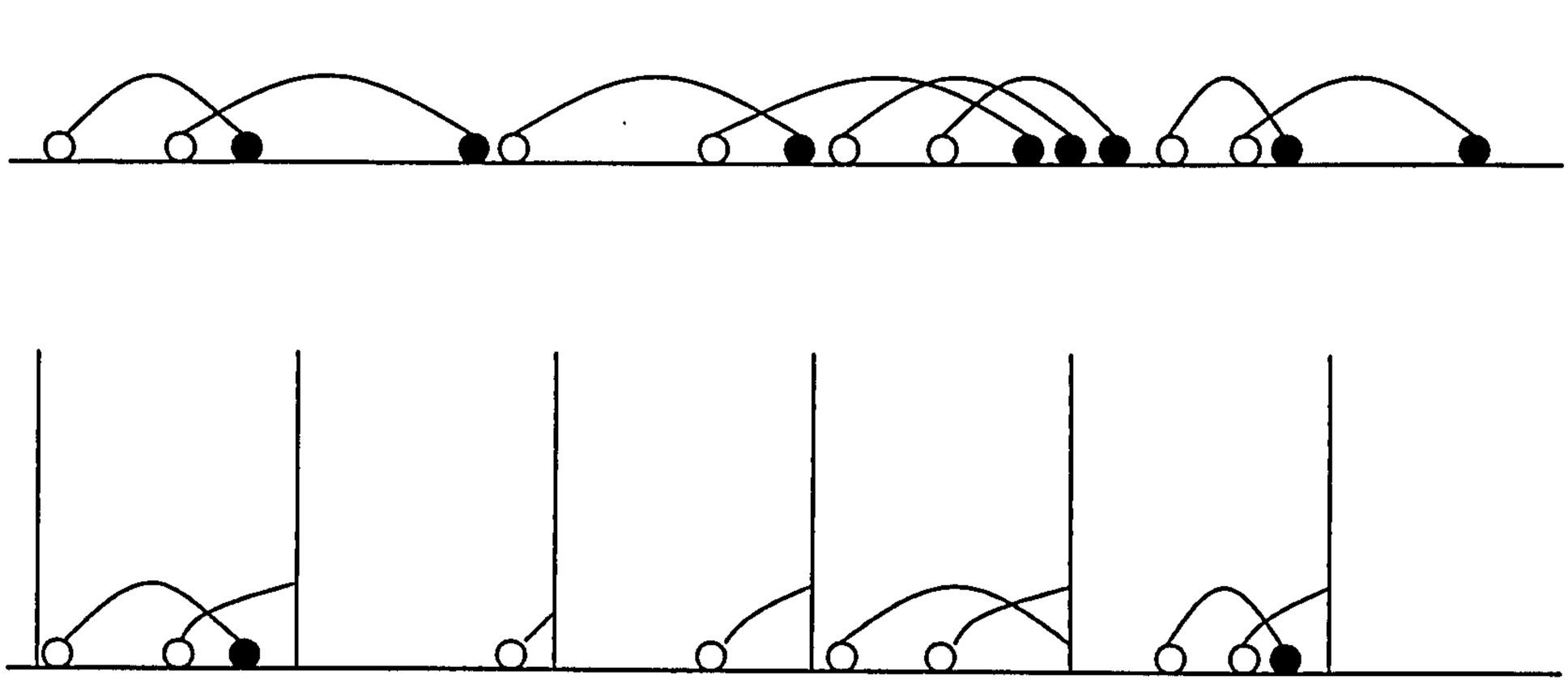


Figure 2-3 DTM applied in complex system

Figure 2-3 shows the delay time concept applied to a complex system with many components. In the figure, the second, fourth, sixth and eighth faults have been detected at inspection points and so have not caused failures, if corrective actions were taken at those inspection points.

The above figures demonstrate the fundamental concept of the Delay Time in modelling the inspection aspect of a maintenance policy. It is clear that if the optimal frequency of preventive maintenance (inspection) is chosen, the least maintenance cost and downtime of the system may be achieved.

2.2 THE DEVELOPMENT OF DTM

An early paper, Christer & Waller (1984a), assumed that the time of origin of a fault is uniformly distributed over time since the last inspection and is independent of the delay time h. Then the failure risk within one renewal cycle, the long run average cost rate and the down time models were developed as a function of the inspection interval T. A modified delay time model allows non-perfect inspection and arbitrary distribution of the initial fault time and delay time distributions, which make delay time models more practical, Christer & Waller (1984a). At the early stage of the development of the delay time model, Christer & Waller used subjective data to estimate the delay time. By asking engineers who maintained the machine the following questions when a fault was detected at a regular inspection, or when a failure occurred, in order to provide a subjective estimate of the delay time.

(i) How long ago could the fault have first been noticed by an inspection or by an operator (=HLA)?

(ii) If the repair was not carried out, how much long could this fault be delayed until it would have caused a failure (=HML)?

At an inspection, the subjective estimate of delay time

$$h = \text{HLA} + \text{HML} \tag{2-1}$$

In this way, by observing sufficient faults or failures, a distribution for $f(h)$ may be obtained.

In a case study, Christer & Waller (1984a) applied DTM at Pedigree Petfoods Limited. A delay time modelling analysis was used to derive an optimum-cost maintenance policy for the canning line, which was subsequently adopted by management. It is interesting to note that the distribution of h was observed to be

approximately exponential, but perhaps with a longer tail. In another case study, Christer & Waller (1984b), the DTM and failure analysis were applied to model the preventive maintenance of a vehicle fleet of tractor units operated by Hiram Walker Limited. The management also adopted and applied the recommended decrease in the frequency of maintenance interventions.

Christer (1987) developed a perfect inspection model of a single component. In the paper, component reliability as a function of the inspection interval was calculated using a recursive formula. Note that this assumes that the cycle of regularly spaced inspections commences at component renewal, and a modified formula would be need when the inspection cycle is independent of component renewal times. Cerone (1991) gave a calculation of an approximate reliability measure using a simplified method. Pellegrin (1991) proposed a graphical procedure to estimate the optimum interval between inspections based on DTM, which allows the various factors relevant to decision making to be emphasised.

In the paper of Christer & Redmond (1990), a further version of the complex system model was given. They note that using subjective data to estimate delay time may be biased, and suggested maximum likelihood estimation based on subjective data to overcome this. In contrast to the subjective method of estimating delay time distributions, Baker & Wang (1991,1993) proposed a method of using objective data collected from records kept by engineers maintaining several items of medical equipment. The model was extended to cope with multiple component system, and the maximum likelihood method was used to fit the model to data. This study showed that it was possible to estimate DTM parameters without the use of subjective data.

Christer & Wang (1995) developed a delay time model of a complex system. It is assumed that planned inspection takes place on regular time interval and opportunity inspection takes place on failure. Defects are assumed to be arising according to a non-homogeneous Poisson process. In this study, the only renewal point of the stochastic process is the system replacement time, therefore, unlike in the homogeneous case, the expected cost per unit time over one inspection interval can not be adopted as the long

term model objective. A paper focused on determining the non-equal inspection interval based on the paper by Christer & Wang (1995) was given in Wang and Christer (2001).

Christer & Lee (1997) developed a delay time model for the operational reliability of a ship over a mission under regular inspections. Wang & Christer (1997) proposed a safety inspection model for the expected consequence of inspections over a finite time horizon. A single dominant failure mode is modelled, which has considerable safety or risk consequences assumed either in cost terms or in terms of the probability of failure over the time horizon. The established model extends the earlier delay time models assuming an infinite time horizon.

Wang (1997) reported a model to estimate the parameters by utilising a set of expert judgements when there is no hard data of observed failure and maintenance actions available. Wang and Jia (2000) developed a Bayesian approach for delay time maintenance model parameter estimation using both subjective and objective data. In the model, when either subjective data or objective data is available, approaches to estimate the parameters of delay time models are available for each of the data type. It also can be used in the situation where one starts with subjective data first and then update the estimates when objective data becomes available.

2.3 THE NEED FOR DEVELOPING A PLANT MAINTENANCE OPTIMIZATION SOFTWARE PACKAGE BASED ON DTM

Although, DTM applications have been successful by any measure, they have entailed the concentrated input and experience of the Salford team of modellers. This level of expertise and knowledge using DTM will not generally be available within industry, which is an inhabitant to more general use. Our assertion is that this problem can be overcome if a semi-automated DTM tool can be developed. Observing a general robustness to most modelling assumption, has motivated us towards seeking to develop a semi-automated DTM tool based on models already developed and tested. This would enable competent engineers to themselves undertake primary data analysis, model building, model testing, and model based decision making. A vary simple demonstration

package of DTM has been developed at Salford in 1998, and been demonstrated in industry. The package is based on a simplified DTM of the complex system model using subjective input. It is a simplified tool developed for demonstration purpose only. However, demonstration within industry & workshops were well received and revealed an interest and a demand for an applicable tool in DTM. This research will be the first attempt to develop such a potentially applicable DTM tool.

Chapter 3 Single Component Delay Time Model

The Single Component Delay Time Model concerns the inspection maintenance of a repairable component which is assumed to have a single failure mode. At most one defect can exist at any given time. The time u of a defect firstly becoming visible is measured from the time of the last renewal, and is described by a pdf $g(u)$. The pdf of h, the delay time, is denoted by $f(h)$.

In this model the inspection is assumed to be perfect in that any fault present will be identified at the inspection time. This is an approximation because inspection may not be perfect in reality. This assumption can be relaxed with the expense of a more complicated model of one more parameters (Chister et al, 1995).

In the complex system delay time model, we classify failures and faults into areas where each area is a subsystem of the plant. In the case of a single component, there is only one area, and one type of fault. Failure need not be a sudden event, and could be a deterioration such that repair could no longer be delayed. Fault is a type of event that if no action is taken after it occurs, it will subsequently cause a failure.

3.1 DECISION MODEL OF A SINGLE COMPONENT

Assumptions and notation

- The system analysed here is a single component subject to perfect inspection.
- The fault arises at time u, and the pdf and cdf of u are know and denoted by $g(u)$ and $G(u)$, respectively.
- The pdf and cdf of delay time interval h are denoted by $f(h)$ and $F(h)$, respectively.
- The nth inspection is ended at time t_n, and we also assume a constant inspection interval, where $t_n - t_{n-1} = \ldots = t_1 - t_0 = T$

- If a fault is identified at the inspection, the item will be replaced, otherwise the item will continue to work until either a failure occurs or a fault is identified at an inspection. At either case, the component is renewed and the process resumes.
- $P_b(t_{i-1}, t_i)$ is the probability of a failure renewal occurring between (t_{i-1}, t_i).
- $P_m(t_i)$ is the probability of an inspection renewal at t_i.
- C_p, C_f, C_i : the average cost of a preventive maintenance, a failure repair action and the average cost of an inspection
- T_p, T_f, T_i : the average time duration of a preventive maintenance, a failure repair action and the average duration of an inspection
- $EC(T), ED(T)$ denotes the expected cost and down time over a life cycle of item, given T.
- $EL(T)$ denotes the expected life cycle of the item, given T.
- $C(T)$, $D(T)$ are the expected cost and down time per unit time over an infinite time horizon, given T.

Under the assumptions of perfect inspection, any defect present at an inspection will always be identified. If a fault occurs within an inspection interval, the component will, depending on its delay time length, either fail to function before the next inspection or be identified and replaced at the next inspection. Assuming a fault arises within $(u, u+du)$, $t_{i-1} < u < t_i$, then the renewal point of the life cycle is triggered by either a failure or a fault being found at an inspection, which are called failure renewal or inspection renewal respectively. The probability of a failure renewal before inspection time t_i is $g(u)\cdot du \cdot F(t_i - u)$, and the probability of an inspection renewal at time t_i is $g(u)\cdot du \cdot [1 - F(t_i - u)]$. See Figure 3-1.

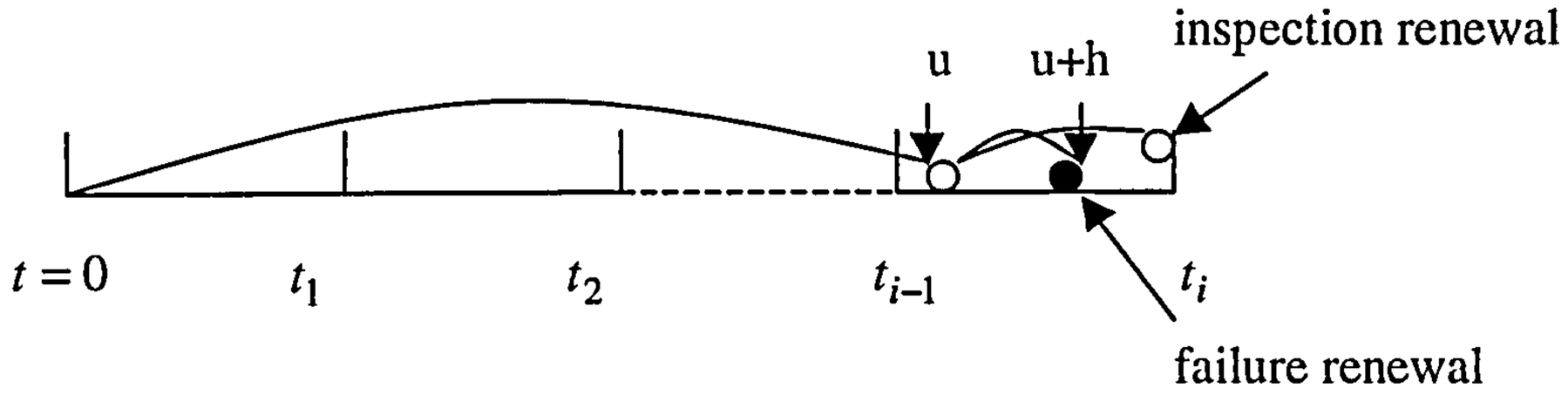


Figure 3-1 Single component inspection process

The probability of a failure renewal occurring between (t_{i-1}, t_i) and the probability of an inspection renewal at t_i are given respectively as

$$P_b(t_{i-1}, t_i) = \int_{t_{i-1}}^{t_i} g(u)F(t_i - u)du \tag{3-1}$$

and

$$P_m(t_i) = \int_{t_{i-1}}^{t_i} g(u)[1 - F(t_i - u)]du, \tag{3-2}$$

and the expected renewal cycle cost, expected renewal cycle down time and expected renewal cycle life length are

$$EC(T) = \sum_{i=1}^{\infty} \{[(i-1)C_i + C_f] \cdot P_b(t_{i-1}, t_i) + (iC_i + C_p) \cdot P_m(t_i)\}, \tag{3-3}$$

$$ED(T) = \sum_{i=1}^{\infty} \{[(i-1)T_i + T_f] \cdot P_b(t_{i-1}, t_i) + (iT_i + T_p) \cdot P_m(t_i)\}, \tag{3-4}$$

and

$$EL(T) = \sum_{i=1}^{\infty} \left[\int_{t_{i-1}}^{t_i} \int_{0}^{t_i - u} (u+h)g(u)f(h)dhdu + t_i \cdot P_m(t_i) \right]. \tag{3-5}$$

Then the long run mean cost and mean down time per unit time, $C(T)$ and $D(T)$, are

$$C(T)=\frac{\sum_{i=1}^{\infty}\left\{\left[(i-1)C_i+C_b\right]\cdot P_b(t_{i-1},t_i)+\left(iC_i+C_p\right)\cdot P_m(t_i)\right\}}{\sum_{i=1}^{\infty}\left[\int_{t_{i-1}}^{t_i}\int_{0}^{t_i-u}(u+h)g(u)f(h)dudh+t_i\cdot P_m(t_i)\right]} \tag{3-6}$$

and

$$D(T)=\frac{\sum_{i=1}^{\infty}\left\{\left[(i-1)T_i+T_b\right]\cdot P_b(t_{i-1},t_i)+\left(iT_i+T_p\right)\cdot P_m(t_i)\right\}}{\sum_{i=1}^{\infty}\left[\int_{t_{i-1}}^{t_i}\int_{0}^{t_i-u}(u+h)g(u)f(h)dudh+t_i\cdot P_m(t_i)\right]} \tag{3-7}$$

respectively.

The key task now is to determine the probability density functions for u and h, and to estimate their parameters, which will be introduced next.

3.2 PARAMETER ESTIMATION OF THE SINGLE COMPONENT DTM MODEL

In order to develop a mechanism which can be used to estimate DTM parameters using available subjective data, objective data or both, we propose to use the Bayesian approach introduced by Wang and Jia (2000). The initial estimates are made of using the empirical Bayesian method matching with a few summary subjective data collected from the experienced engineers. Then the updating mechanism is used in the process, if objective data becomes available, which requires a repeated evaluation of the likelihood function. Here we assume all the parameters with $g(u)$ and $f(h)$ are random variables following a prior distribution, say Gamma, and then if the hyper-parameters within such a prior distribution are known, the updated estimates of each parameter can be obtained using the Bayesian approach.

3.2.1 Estimation using subjective data only

When objective data is not sufficient or in quantity or quality for estimating purposes, we propose an approach using subjective data only.

Additional assumptions and notation

- The fault arising time u follows a Weibull distribution with a scale parameter α_u and a shape parameter β_u, i.e.,

$$g(u) = \beta_u \alpha_u{}^{\beta_u} u^{\beta_u - 1} \times e^{-(u\alpha_u)^{\beta_u}} \tag{3-8}$$

- The delay time h of a random defect is independent of its time origin and follows a Weibull distribution with a scale parameter α_h and a shape parameter β_h. i.e.

$$f(h) = \beta_h \alpha_h{}^{\beta_h} h^{\beta_h - 1} \times e^{-(h\alpha_h)^{\beta_h}} \tag{3-9}$$

- All parameters $\alpha_\bullet$, $\beta_\bullet$ are assumed to follow a Gamma distribution with hyper-parameters $\Phi_{\alpha_\bullet}$ and $\Phi_{\beta_\bullet}$ respectively, where $\Phi_{\alpha_\bullet} = (a_{\alpha_\bullet}, b_{\alpha_\bullet})$ and $\Phi_{\beta_\bullet} = (a_{\beta_\bullet}, b_{\beta_\bullet})$. Gamma is a well-known distribution chosen by statisticians as a prior distribution because of its computational advantages compared with others, such as Weibull or Lognormal.
- The pdf of a Gamma distribution, given Φ, is, for example if $\Phi = \Phi_\alpha$,

$$p(\alpha_\bullet | \Phi_{\alpha_\bullet}) = \frac{e^{-b_{\alpha_\bullet}\alpha_\bullet} b_{\alpha_\bullet} (b_{\alpha_\bullet}\alpha_\bullet)^{a_{\alpha_\bullet}-1}}{\Gamma(a_{\alpha_\bullet})} \tag{3-10}$$

- $\overline{\alpha_\bullet}$ and $\overline{\beta_\bullet}$ are the expected values of $\alpha_\bullet$ and $\beta_\bullet$, where $\alpha_\bullet$ and $\beta_\bullet$ are independent variables.
- E_{mnip} is the expected time to an initial point of the defect.
- E_{mndt} is the expected delay time .
- T_{mnip} is the mean time to an initial point, estimated by experts.
- T_{maip} is the mean maximum time to an initial point, estimated by experts.
- T_{mndt} is the mean delay time, estimated by experts.
- T_{madt} is the mean maximum time of the delay time, estimated by experts.

It is noted that the use of the Weibull distribution in DTM has proved appropriate in many previous case studies, Christer (1999).

Estimate parameters of the initial time distribution *g(u)*

The expected initial time, E_{mnip} is given by,

$$E_{mnip} = \int_0^\infty \int_0^\infty \left(\int_0^\infty u g(u|\alpha_u, \beta_u) du \right) p(\alpha_u|\Phi_{\alpha_u}) p(\beta_u|\Phi_{\beta_u}) d\alpha_u d\beta_u \qquad (3\text{-}11)$$

which can be approximated by T_{mnip}, that is

$$T_{mnip} \approx \int_0^\infty \int_0^\infty \left(\int_0^\infty u g(u|\alpha_u, \beta_u) du \right) p(\alpha_u|\Phi_{\alpha_u}) p(\beta_u|\Phi_{\beta_u}) d\alpha_u d\beta_u \qquad (3\text{-}12)$$

The parameters α_u, β_u are assumed to follow a Gamma distribution with the hyper-parameters Φ_{α_u} and Φ_{β_u} respectively, where $\Phi_u = (a_{\alpha_u}, b_{\alpha_u})$ and $\Phi_u = (a_{\beta_u}, b_{\beta_u})$, so the expected values of α_u and β_u are

$$\overline{\alpha_u} = \frac{b_{\alpha_u}}{a_{\alpha_u}} \qquad (3\text{-}13)$$

and

$$\overline{\beta_u} = \frac{b_{\beta_u}}{a_{\beta_u}}. \qquad (3\text{-}14)$$

Since the probability for u being less than or equal to the maximum time is almost one, we have, approximately,

$$P(x \le T_{maip}) = \int_0^\infty \int_0^\infty \left(\int_0^{T_{maip}} g(u|\alpha_u, \beta_u) du \right) p(\alpha_u|\Phi_{\alpha_u}) p(\beta_u|\Phi_{\beta_u}) d\alpha_u d\beta_u \approx 0.9999. \qquad (3\text{-}15)$$

After many calculations we know this equation is not very sensitive with the decimal digits of the probability given, when it is larger than 0.9999. The computing time will arise dramatically, when we increase the number of digital 9 after decimal point.

Now there are four parameters to be estimated in order to estimate $\overline{\alpha_u}$ and $\overline{\beta_u}$. If we want to estimate those four parameters, four independent equations involving a_{α_u}, b_{α_u}, a_{β_u} and b_{β_u} are needed for the estimation process. In order to reduce the complexity of the model, we fixed b_{α_u} and b_{β_u} to arbitrary values. This is done because we are only interested in $\overline{\alpha_u}$ and $\overline{\beta_u}$, and if b_{α_u} and b_{β_u} are fixed, then the values of $\overline{\alpha_u}$ and $\overline{\beta_u}$ will be uniquely determined by a_{α_u} and a_{β_u}.

After fixing the values of b_{α_u} and b_{β_u}, there are only two parameters to be estimated. By solving equations (3-12) and (3-15), we may be able to obtain the estimated values of parameters a_{α_u} and a_{β_u} of the Gamma distribution. Then we can obtain the expected values $\overline{\alpha_u}$ and $\overline{\beta_u}$ from equations (3-13) and (3-14). Alternatively, we can even further simplify the estimating process by using

$$P\left(x \le T_{maip}\right) = \int_0^{T_{maip}} g\left(u \middle| \overline{\alpha_u}, \overline{\beta_u}\right) du \approx 0.9999 \qquad (3\text{-}16)$$

directly instead of equation (3-15).

Example

Using the model introduced above, we set $T_{mnip} = 2$ years, $T_{maip} = 3$ years respectively to demonstrate the model. As a first step, we want to know if there are any benefits using equation (3-16) instead of equation (3-15). After comparing the calculation results of equations (3-12) & (3-15) and those of equations (3-12) & (3-16), it is obvious that the equations (3-12) & (3-16) produced better results using relatively less time. See Table 3-1. The absolute error is $\left|T_{mnip} - E_{mnip}\right| + \left|P\left(x \le T_{maip}\right) - 0.9999\right|$.

It can be seen from Table 3-1 that the absolute error of using equation (3-16) is always less than that of equation (3-15) as expected since equation (3-16) involves less integrals and hence is more accurate. The key point here is the computing time saved, which is the main reason why we recommend using equation (3-16).

Table 3-1 The results obtained from solving transcendental equations (3-12) & (3-15) and (3-12) & (3-16)

b_{α_u} and b_{β_u}	Equations (3-12) & (3-15)		Equations (3-12) & (3-16)	
	Absolute Error	Computing Time (mins)	Absolute Error	Computing Time (mins)
1	0.0913	more than 60	0.0001	about 10 minutes
2	0.0725	more than 60	0	about 10 minutes
3	0.0647	more than 60	0.0045	about 10 minutes
4	0.0634	more than 60	0.0111	about 10 minutes
5	0.0606	more than 60	0.0183	about 10 minutes
6	0.1548	more than 60	0.0256	about 10 minutes

In the second step, we want to obtain suitable fixed values of b_{α_u} and b_{β_u}, which may result in even better estimates of a_{α_u} and a_{β_u}. This is done by changing the values of b_{α_u} and b_{β_u} in equations (3-12) and (3-16), and then comparing the absolute error values. The one gives the smallest error should be the b_{α_u} and b_{β_u} values to use. See Table 3-2

Table 3-2 The equation values under different value of α_b and β_b in estimating parameters of $g(u)$

Value of α_b and β_b	Equation (3-12)		Equation (3-16)		Absolute error
	T_{mnip} (1)	E_{mnip} (2)	0.9999 (3)	$P(x = T_{maip})$ (4)	$\lvert(1)-(2)\rvert + \lvert(3)-(4)\rvert$
0.1	2	1.9999	0.9999	0	1
1	2	2	0.9999	1	1E-04
2	2	2	0.9999	0.9999	0

3	2	1.9999	0.9999	0.9955	0.0045
4	2	1.9998	0.9999	0.9890	0.0111
5	2	1.9996	0.9999	0.9820	0.0183
6	2	1.9983	0.9999	0.9760	0.0256
10	2	1.5612 E-55	0.9999	0	2.9999

It can be seen that the optimum fixed value of b_{α_u} and b_{β_u} is 2.

Estimate parameters of the delay time distribution *f(h)*

Using the same method in the model of estimating the parameters of the initial time distribution, we can build the model of parameter estimation of the delay time distribution. It is assumed that delay time h of a random defect is independent of its time origin and follows a Weibull distribution with a scale parameter α_h and a shape parameter β_h. These two parameters are also assumed to follow a two-parameter Gamma distribution. So the mean delay time can is given by

$$E_{mndt} = \int_0^\infty \int_0^\infty \left(\int_0^\infty h f(h|\alpha_h, \beta_h) dh \right) p(\alpha_h|\Phi_{\alpha_h}) p(\beta_h|\Phi_{\beta_h}) d\alpha_h d\beta_h, \qquad (3\text{-}17)$$

which can be approximated by T_{mndt}, that is

$$T_{mndt} \approx \int_0^\infty \int_0^\infty \left(\int_0^\infty h f(h|\alpha_h, \beta_h) dh \right) p(\alpha_h|\Phi_{\alpha_h}) p(\beta_h|\Phi_{\beta_h}) d\alpha_h d\beta_h. \qquad (3\text{-}18)$$

As before, the probability of $h \leq$ maximum delay time is almost one, that is

$$P(x \leq T_{madt}) = \int_0^\infty \int_0^\infty \left(\int_0^{T_{madt}} f(h|\alpha_h, \beta_h) dh \right) p(\alpha_h|\Phi_{\alpha_h}) p(\beta_h|\Phi_{\beta_h}) d\alpha_h d\beta_h \approx 0.9999, \qquad (3\text{-}19)$$

approximately.

The simplified form of equation (3-19) is

$$P(x \le T_{madt}) = \int_0^{T_{madt}} f\left(h \middle| \overline{\alpha_h}, \overline{\beta_h}\right) dh \approx 0.9999 \qquad (3\text{-}20)$$

Where, $\overline{\alpha_h} = \dfrac{b_{\alpha_h}}{a_{\alpha_h}}$ and $\overline{\beta_h} = \dfrac{b_{\beta_h}}{a_{\beta_h}}$.

Here b_{α_h} and b_{β_h} are also fixed with arbitrary values, then only a_{α_h} and a_{β_h} are left to be estimated. Solving equations (3-18) and (3-20), when subjective data of T_{mndt} and T_{madt} are available, the parameters of a_{α_h} and a_{β_h} can be obtained.

Example

In a case study, we obtain T_{mndt}= 2.5 years and T_{madt}= 4 years. Firstly the results derived from equations (3-18) & (3-19) and equations (3-18) & (3-20) are compared, in order to conclude the result of simplified equations (3-18) & (3-20). See Table 3-3. The absolute error is $\left|T_{mndt} - E_{mndt}\right| + \left|P(x \le T_{madt}) - 0.9999\right|$.

Table 3-3 The results obtained from transcendental equations (3-18) & (3-19) and (3-18) & (3-20)

b_{α_h} and b_{β_h}	Equations (3-18) & (3-19)		Equations (3-18) & (3-20)	
	Absolute Error	Computing Time (mins)	Absolute Error	Computing Time (mins)
1	0.0954	more than 40	0.0001	about 10 minutes
2	0.0796	more than 40	0	about 10 minutes
3	0.0916	more than 40	0.0070	about 10 minutes
4	0.0664	more than 40	0.0148	about 10 minutes
5	0.0638	more than 40	0.0225	about 10 minutes
6	0.0646	more than 40	0.0291	about 10 minutes

From Table 3-3, as expected, the absolute errors of the results of equations (3-18) & (3-20) are considerably less than that of equations (3-18) & (3-19), because less integrals are involved in equation (3-20). And the time duration of calculating equations (3-18) & (3-20) is substantially less than equations (3-18) & (3-19). It is the key point that equations (3-18) & (3-20) are recommend being used in this model.

Fixed values of b_{α_h} and b_{β_h} are also used here. The same as before, the values of equations (3-18) and (3-20) in Table 3-4 are compared via different values of b_{α_h} and b_{β_h}, in order to select the best values for b_{α_h} and b_{β_h}.

Table 3-4 The absolute errors under different values of b_{α_h} and b_{β_h} in estimating parameters of $f(h)$

Value of α_b and β_b	Equation (3-18)		Equation (3-20)		Absolute error
	T_{mndt} (1)	E_{mndt} (2)	0.9999 (3)	$P(x = T_{madt})$ (4)	$\lvert(1)-(2)\rvert+\lvert(3)-(4)\rvert$
0.1	2.5	2.5	0.9999	0	1
1	2.5	2.4999	0.9999	1	0.0001
2	2.5	2.5	0.9999	0.9999	0
3	2.5	2.4996	0.9999	0.9933	0.0070
4	2.5	2.4996	0.9999	0.9856	0.0148
5	2.5	2.4992	0.9999	0.9782	0.0225
6	2.5	2.4988	0.9999	0.9720	0.0291
10	2.5	1.0559	0.9999	0	2.4440

After comparing the absolute error results of equations of using (3-18) and (3-20) based on different values of b_{α_h} and b_{β_h}, we can fix b_{α_h} and b_{β_h} at 2, because at that fixed value the absolute error is the least.

3.2.2 Estimation using both subjective and objective data

As discussed before, it is possible to obtain either a subjective estimate or an objective estimate depending on what type of data is available. In practice, the common situation is that the objective data may not be sufficient at the developing stage of an application. To efficiently take advantage of the available information, it is possible to estimate the initial parameters from a set of subjective data, and then the estimates are updated once objective data becomes available by using the Bayesian approach.

Additional assumptions and notation

- N_f is the total number of failure renewals at times $(x_1, x_2, \ldots x_k, \ldots x_{N_f})$.
- N_s is the total number of inspection renewals at times $(t_1, t_2, \ldots t_j, \ldots t_{N_s})$.
- t is the end time of the observation which starts from the last renewal point. The component is still operable at the end of observation.
- t_j is the *j*th inspection renewal time which starts from the last renewal point.
- x_k is the *k*th failure renewal time which starts from the last renewal point.
- y_k is the last inspection time before the *k*th failure renewal which starts from the last renewal points.
- y_e is the last inspection time before the end of observation.
- $P_s(t_j)$ is the probability of the *j*th inspection renewal at time t_j.
- $p_f(x_k)$ is the p.d.f of the *k*th failure renewal at time x_k.
- $P_n(t)$ is the probability of no renewal events occurring at the end time t of the observation.

If the objective data is also available, the estimates can be updated. From the independent assumption between model parameters and the Bayes theorem, the joint posterior distribution $P(\alpha_u, \beta_u, \alpha_h, \beta_h | X_i)$ for α_u, β_u, α_h and β_h in light of available i observations is

$P(\alpha_u, \beta_u, \alpha_h, \beta_h | X_i) =$

$$\frac{\prod_{j=1}^{i} P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)p(\alpha_u|\Phi_{\alpha_u})p(\beta_u|\Phi_{\beta_u})p(\alpha_h|\Phi_{\alpha_h})p(\beta_h|\Phi_{\beta_h})}{\int_0^\infty\int_0^\infty\int_0^\infty\int_0^\infty\prod_{j=1}^{i} P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)p(\alpha_u|\Phi_{\alpha_u})p(\beta_u|\Phi_{\beta_u})p(\alpha_h|\Phi_{\alpha_h})p(\beta_h|\Phi_{\beta_h})d\alpha_u d\beta_u d\alpha_h d\beta_h} \quad (3\text{-}21)$$

where $X_i=\{x_1,x_2,...,x_i\}$ with x_j (j=1,2,...i) being the jth event observed, and $P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)$ is the probability of such an event, which could be a failure or a defect being identified at an inspection in this case. Since the denominator of equation (3-21) is a constant, we have

$$P(\alpha_u,\beta_u,\alpha_h,\beta_h|X_i)\propto\prod_{j=1}^{i} P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)p(\alpha_u|\Phi_{\alpha_u})p(\beta_u|\Phi_{\beta_u})p(\alpha_h|\Phi_{\alpha_h})p(\beta_h|\Phi_{\beta_h}) \quad (3\text{-}22)$$

Given that Φ_{α_u}, Φ_{β_u}, Φ_{α_h} and Φ_{β_h} are known, the updated point estimates for α_u, β_u, α_h and β_h can be obtained by maximising equation (3-22), which is equivalent to maximise its logarithm

$$\log\left(\prod_{j=1}^{i} P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)\right)+\log\left(p(\alpha_u|\Phi_{\alpha_u})\right)+\log\left(p(\beta_u|\Phi_{\beta_u})\right)+\log\left(p(\alpha_h|\Phi_{\alpha_h})\right)$$
$$+\log\left(p(\beta_h|\Phi_{\beta_h})\right)=\sum_{j=1}^{i}\log\left(P(x_j|\alpha_u,\beta_u,\alpha_h,\beta_h)\right)+\log\left(p(\alpha_u|\Phi_{\alpha_u})\right)+\log\left(p(\beta_u|\Phi_{\beta_u})\right)$$
$$+\log\left(p(\alpha_h|\Phi_{\alpha_h})\right)+\log\left(p(\beta_h|\Phi_{\beta_h})\right). \quad (3\text{-}23)$$

As i tends to be ∞, the last four terms on the right hand side of equation (3-23) are negligible, and therefore, the Bayesian approach reduces to the conventional maximum likelihood method. Maximising expression (3-23), the estimated parameters of DTM in a single component based on both subjective and objective data can be obtained. Now we look at the specific formulation of the likelihood function, namely, the first summation term on the right hand side of equation (3-23).

The likelihood of observing all renewal events over an observing period is

$$L=\left(\prod_{j=1}^{N_s}P_s(t_j)\bullet\prod_{k=1}^{N_f}p_f(x_k)\right)\bullet P_n(t)\ . \tag{3-24}$$

By maximising equation (3-24), in terms of the estimation parameters in the distribution, their values can be estimated. It is an easy way to take the logarithm of the likelihood function, so we have

$$\log L=\left(\sum_{j=1}^{N_s}\log[P_s(t_j)]+\sum_{k=1}^{N_f}\log[p_f(x_k)]\right)+\log[P_n(t)] \tag{3-25}$$

where,

$$P_s(t_j)=\int_{t_{j-1}}^{t_j}g(u)[1-F(t_j-u)]du \tag{3-26}$$

$$p_f(x_k)=\int_{y_k}^{x_k}g(u)f(x_k-u)du \tag{3-27}$$

$$P_n(t)=1-G(t)+\int_{y_e}^{t}g(u)(1-F(t-u))du \tag{3-28}$$

3.3 SUBJECTIVE DATA ACQUISITION

In the estimation model, we aim to seek as little as possible information from experts, but yet sufficient for parameter estimating. The method we used is based on the principle of moment matching. Basically, we ask a few summary statistics from the engineers (experts), such as the meantime to failure, the maximum time to failure, the mean time to an initial point of the defect, etc. Then we make them equal to the corresponding theoretical counterparts with unknown parameters. If we have established m such equations with m unknown parameters, we might be able to solve these equations in terms of these unknown parameters.

As introduced earlier, in the delay time model the initial time u and delay time h may follow a two-parameter Weibull distribution. Because we treat each parameter as a random variable, which follows, say, a two-parameter Gamma distribution, this effectively doubles the number of unknown parameters. Although the scale parameters of Gamma distribution was fixed to simplify the parameter estimating process, there are still four parameters left to be estimated, then we need at least four pieces of information from experts to establish four equations to solve the four unknown parameters. We propose to seek the following four pieces of information from experts,

- Mean time to an initial point (=T_{mnip})
- Mean maximum time to an initial point (=T_{maip})
- Mean delay time (=T_{mndt})
- Mean maximum delay time (=T_{madt})

If we can get those four subjective data from maintenance engineers, the four hyper-parameters may be estimated from the four equations established earlier. Since the delay time is not normally observable, and therefore, engineers may not have any experience in the delay time estimates. It is difficult to get a reasonably good answer to the questions, even though the concept can be explained to them. According to the delay time concept, we have

$$T_{mndt} = T_{mnfp} - T_{mnip} \tag{3-29}$$

and

$$T_{madt} = T_{mafp} - T_{miip} \tag{3-30}$$

where

- T_{miip} denotes the mean minimum time to an initial point.
- T_{mnfp} denotes the mean time to a failure point
- T_{mafp} denotes the mean maximum time to a failure point

Since the initial time can sometime be observed, we seek questions on the mean time to an initial point, the minimum time to an initial point, the mean time to a failure point, the maximum time to a failure point and the mean maximum time to an initial point in stead of the four mentioned earlier. When engineers answer the above questions, the subjective data can be obtained.

When those questions are presented to different engineers, they might give different answers. Here we use a weighted average method to combine those engineers' opinions. Obviously, the weight of engineers should be set according to their work experience.

In the decision model, we used several other parameters, such as the average cost and time duration of a preventive maintenance, failure repair action and the average cost of an inspection. Those information may be acquired directly form engineers or management of the plant or from existing data records. To complete the software design aiming to use subjective data in the first instance, the questionnaire acquiring those information is also co-operated in the package.

Chapter 4 Complex System Delay Time Model

The Complex System Delay Time Model concerns the inspection maintenance of a complex system. A complex system, or multi-component plant, is one where a large number of components and failure modes arise, and the correction of one defect or failure has nominal impact on the steady state upon the overall plant failure characteristics.

In the complex system delay time model, we first focus on modelling the maintenance engineering inspection decisions. Then models of estimating parameters, which are involved in the decision model, will be introduced.

4.1 DELAY TIME DECISION MODELS

Consider the following basic complex system maintenance modelling scenario where:

- An inspection takes place every T time units, cost C_i units and requires D_i time units.
- Inspections are perfect in that all defects present will be identified.
- The number of defects arising follows a HPP with a constant rate λ per unit time.
- The probability density function for delay time of faults $f(h)$ is independent of initial point u of a defect.
- Failure will be repaired immediately at an average cost C_f and downtime D_f.
- The plant has operated sufficiently long since new to be effectively in a steady state.
- Defects and failure only arise whilst plant is operating.

These assumptions characterize the simplest non-trivial inspection problem. Under these assumptions, for a defect with delay time h, the expected number of failures over $(0,T)$, $EN_f(T)$, is given by Christer and Wang (1995),

$$EN_f(T) = \lambda \int_0^T F(x)dx \tag{4-1}$$

Where $F(x) = \int_0^x f(h)dh$

When equation (4-1) is available, the expected unit time cost and downtime are given respectively as

$$C(T) = \left\{ \frac{C_i + C_f EN_f(T)}{T + D_I} \right\} \tag{4-2}$$

and

$$D(T) = \left\{ \frac{D_i + D_f EN_f(T)}{T + D_I} \right\} \tag{4-3}$$

In a perfect inspection situation, the expected number of faults identified at T, $EN_p(T)$, can be expressed as

$$EN_p(T) = \int_0^T \lambda[1 - F(T-u)]du \tag{4-4}$$

In this decision model, if we assume that the delay time follows a two-parameter distribution, namely, one is the shape parameter, the other is the scale parameter. Since the number of defects arising follows a HPP with a constant rate λ per unit time. Then there are three unknown parameters in the model.

Similar to the single component model, we treat each parameter as a random variable following a two-parameter Gamma distribution, and then we have six unknown hyper-parameters. Our task is to estimate these hyper-parameters first using available

subjective data and then update the estimates if objective becomes available, namely, the Bayesian approach used before. The other parameters required by the model, such as the average cost and down information for a failure and an inspection, may be obtained directly from the engineers.

4.2 PARAMETER ESTIMATION MODEL

4.2.1 Estimation using subjective data

The Bayesian approach is used in this parameter estimation model. As mentioned before, The benefit of using the Beyesian approach to estimate parameters is that the parameters can be estimated based on subjective data first, when objective data is not sufficient enough to do so. Then the estimates can be updated in the estimation process with available objective data. Of course, we can use objective directly, if it is sufficient and in good quality. This subjective parameter estimation model is developed under the assumptions and notation introduced earlier in addition to the following extra assumptions and notation.

Extra assumptions and notation

- The delay time h of a random defect is independent of its time origin and follows a Weibull distribution with shape parameter β and scale parameter α. i.e.

$$f(h) = \beta\alpha^{\beta} h^{\beta-1} \times e^{-(h\alpha)^{\beta}} \tag{4-5}$$

- The parameters λ, α, β are assumed to follow Gamma distribution with the hyper-parameters $a_{\lambda}, b_{\lambda}, a_{\alpha}, b_{\alpha}, a_{\beta}$ and b_{β} respectively, where $a_{\bullet}$ is the shape parameter and $b_{\bullet}$ is the scale parameter of the Gamma distribution.
- $\overline{\lambda}$, $\overline{\alpha}$ and $\overline{\beta}$ are the expected value of λ, α and β.
- Mean number of failures within $(0,T)$ $(=N_{afail})$
- Mean maximum number of failures within $(0,T)$ $(=N_{mfail})$
- Mean number of defects within $(0, T)$ $(=N_{ad})$

- Mean maximum number of defects within (0, T) ($=N_{md}$)

In the first section of this chapter, a decision model has been introduced, which includes several parameters to be estimated. Here we seek to estimate $a_\lambda, b_\lambda, a_\alpha, b_\alpha, a_\beta$ and b_β from available subjective information. According to the assumptions and notation, the expression of an average number of defects is given by

$$N_{ad} = \lambda \cdot T \tag{4-6}$$

because λ follows a Gamma distribution, (4-6) is equivalent to

$$N_{ad} = \overline{\lambda} \cdot T = \frac{b_\lambda}{a_\lambda} T \tag{4-7}$$

and the expected probability of the maximum number of defects occurred in (0,T) is given by.

$$P(x = N_{md} + 1) = \int_0^\infty \left[\frac{e^{-\lambda T} (\lambda T)^{N_{md}+1}}{(N_{md}+1)!} \right] f(\lambda | a_\lambda, b_\lambda) d\lambda \tag{4-8}$$

where,

$$f(\lambda | a_\lambda, b_\lambda) = \frac{e^{-(a_\lambda \lambda)} (a_\lambda \lambda)^{b_\lambda - 1} a_\lambda}{\Gamma(b_\lambda)} \tag{4-9}$$

Since $P(x = N_{md} + 1)$ should be extremely small, we set

$$P(x = N_{md} + 1) = \int_0^\infty \left[\frac{e^{-\lambda T} (\lambda T)^{N_{md}+1}}{(N_{md}+1)!} \right] f(\lambda | a_\lambda, b_\lambda) d\lambda \approx 0.0001 \tag{4-10}$$

Solving equations (4-7) and (4-10) simultaneously, the estimated values of a_λ and b_λ may be obtained.

The expected number of failures in (0,T) can be expressed as

$$EN_f(T) = \lambda \int_0^T F(h) dh \tag{4-11}$$

Because delay time h follows a Weibull distribution, then we have

$$EN_f(T) = \lambda \int_0^T \left(1 - e^{-(\alpha h)^\beta}\right) dh \tag{4-12}$$

Since the value of λ follows a Gamma distribution, if follows

$$\lambda \approx \overline{\lambda} = \frac{b_\lambda}{a_\lambda} \tag{4-13}$$

Now according to the assumptions and notion of the delay time, the expression of the mean number of failures can be derived as,

$$EN_f(T) = \int_0^\infty \int_0^\infty \overline{\lambda} \int_0^T \left[1 - e^{-(ah)^\beta}\right] dh \cdot f(\alpha | a_\alpha, b_\alpha) f(\beta | a_\beta, b_\beta) d\alpha d\beta \approx N_{afail} \tag{4-14}$$

and the probability of the maximum number of failures occurred in $(0,T)$ is given by,

$$P(x = N_{mfail} + 1) = \int_0^\infty \int_0^\infty \frac{e^{-EN_f(T)} [EN_f(T)]^{N_{mfail}+1}}{(N_{mfail} + 1)!} \cdot f(\alpha | a_\alpha, b_\alpha) f(\beta | a_\beta, b_\beta) d\alpha d\beta \tag{4-15}$$

which should be approaching zero, so we set $P(x = N_{mfail} + 1) \approx 0.0001$ as before.

Now, two expressions have been developed for estimating parameters of $a_\alpha, b_\alpha, a_\beta$ and b_β, namely expressions (4-14) and (4-15). We still need two more expressions to estimate the four parameters involved. As before, in order to reduce the number of hyper-parameters to be estimated, we fix b_α and b_β as constants. We can even further simplify expression (4-14) by using

$$\alpha \approx \overline{\alpha} = \frac{b_\alpha}{a_\alpha} \tag{4-16}$$

and,

$$\beta \approx \overline{\beta} = \frac{b_\beta}{a_\beta} \tag{4-17}$$

The expected number of failures over T is then

$$EN_f(T) \approx \overline{\lambda} \int_0^T \left[1 - e^{-(\overline{a}h)^{\overline{\beta}}}\right] dh \approx N_{afail} \tag{4-18}$$

Then the estimated values of a_α and a_β can be obtained by solving expressions (4-14) and (4-17).

Example

In this example, the subjective data are set arbitrarily, $N_{ad} = 5$, $N_{md} = 10$, $N_{afail} = 3$, $N_{mfail} = 5$, and the values of b_α and b_β are fixed. See Table 4-1. The absolute error is $\left|EN_f(T) - N_{afail}\right| - \left|P\left(x = N_{mfail} + 1\right) - 0.0001\right|$.

Table 4-1 The equation results obtained from solving transcendental equations (4-14) & (4-15) and equations (4-18) & (4-15)

b_α and b_β	Equations (4-14) & (4-15)		Equations (4-18) & (4-15)	
	Absolute Error	Computing Time (mins)	Absolute Error	Computing Time (mins)
1	0.05353	more than 30	0.04595	about 10 minutes
2	0.05283	more than 30	0.04368	About 10 minutes
3	0.05278	more than 30	0.04624	about 10 minutes
4	0.05276	more than 30	0.05029	about 10 minutes
5	0.05102	more than 30	0.04622	about 10 minutes

In the table above, the result indicate that the expressions (4-18) and (4-15) are better for parameter estimation, because the computing time of equation (4-18) and (4-15) is less.

In the model, b_α and b_β are fixed for reducing the complex of the estimating process. In Table 4-2 we compare subjective values with approximated values of equations (4-15) and (4-18) based on different values of b_α and b_β.

Table 4-2 The absolute errors under different values of b_α and b_β in estimating parameters of $f(h)$

Value of α_b and β_b	Equation (4-18)		Equation (4-15)		Absolute Error $\lvert(1)-(2)\rvert+\lvert(3)-(4)\rvert$
	N_{afail} (1)	$EN_f(T)$ (2)	0.0001 (3)	$P(x=N_{mfail}+1)$ (4)	
1	3	2.9992	0.0001	0.04524	0.04595
2	3	2.9987	0.0001	0.04248	0.04368
3	3	2.9972	0.0001	0.04354	0.04624
4	3	2.9937	0.0001	0.04409	0.05029
5	3	2.9984	0.0001	0.04472	0.04622
6	3	4.9993	0.0001	5.24E-21	2
7	3	4.9999	0.0001	5.62E-25	2

From the above table, compared with equations' absolute errors based on different values of b_α and b_β, it is obvious that when b_α and b_β are set to 2, the absolute error is the smallest.

4.2.2 Estimation using both subjective data and objective data

Here we consider the situation that the parameters are estimated initially from subjective data, then using both subjective data and objective data when objective data becomes available.

Additional assumptions and notation

- m_n : The number of faults identified at the nth inspection
- I_j^n : The jth failure time within the nth inspection interval
- k_n : The total number of failures in the nth inspection
- l : The number of inspections
- $P_r(\bullet)$ denotes cdf of $\bullet$
- $p(\bullet)$ denotes pdf of $\bullet$

- $\Phi_{\bullet}$ denotes the hyper-parameter set of λ, α and β

Figure 4-1 illustrates the failure times and the number of faults identified at inspections.

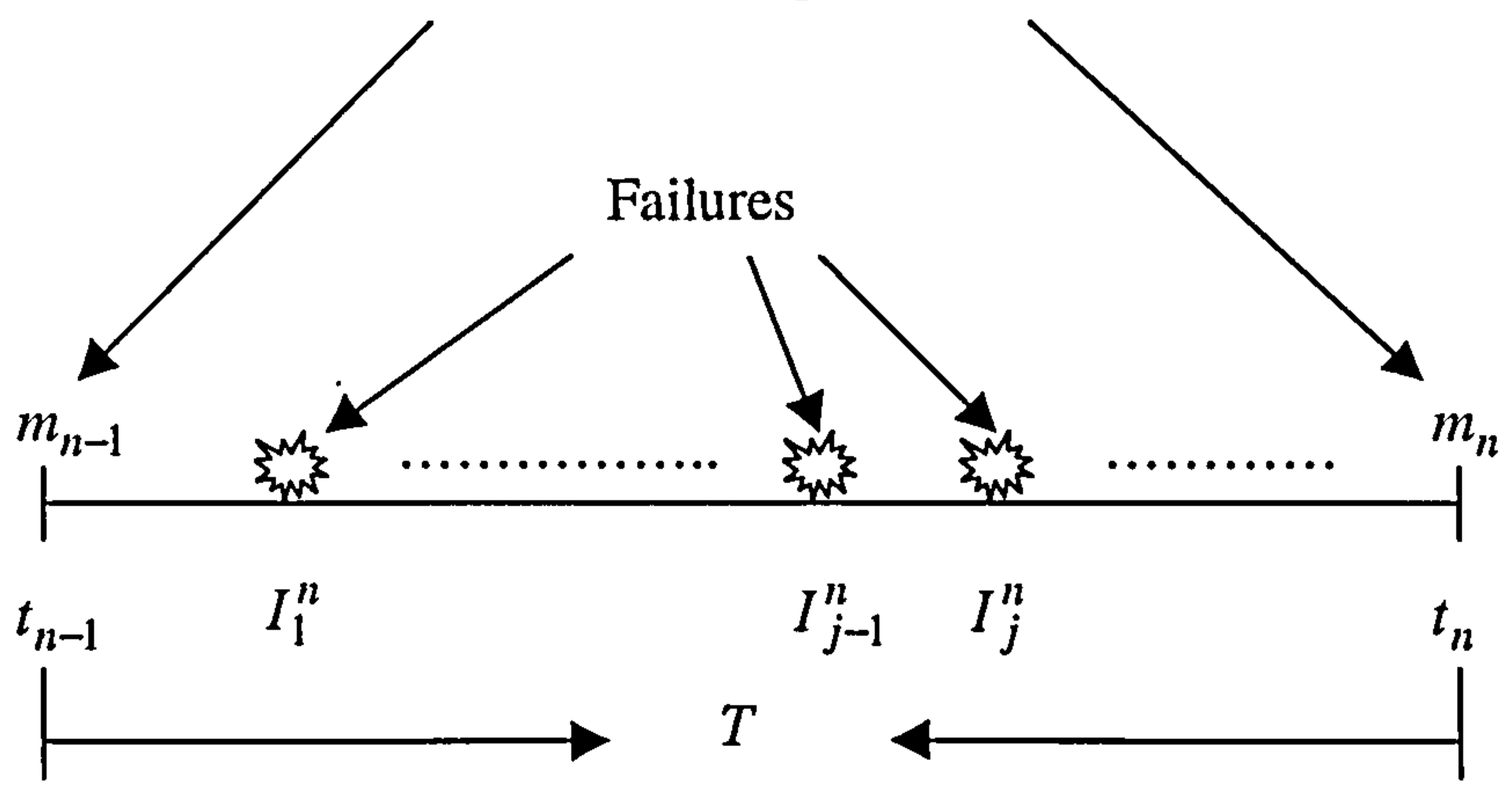


Figure 4-1 number of faults and failures over (t_{n-1}, t_n)

The parameter estimation model using subjective data has been introduced in previous sections. When objective data becomes available, these parameters associated with λ,α and β should be updated. Using the Bayesian theorem, the joint posterior distribution $P(\lambda,\alpha,\beta|X_i)$ for λ,α and β in light of available observations is

$$P(\lambda,\alpha,\beta|X_i)=\frac{\prod_{j=1}^{i}P(x_j|\lambda,\alpha,\beta)p(\lambda|\Phi_\lambda)p(\alpha|\Phi_\alpha)p(\beta|\Phi_\beta)}{\int_0^\infty\int_0^\infty\int_0^\infty\prod_{j=1}^{i}P(x_j|\lambda,\alpha,\beta)p(\lambda|\Phi_\lambda)p(\alpha|\Phi_\alpha)p(\beta|\Phi_\beta)d\lambda d\alpha d\beta} \quad (4\text{-}19)$$

where $X_i=\{x_1,x_2,...,x_i\}$, x_j (j=1,2,...i) is the jth event observed, and $P(x_j|\lambda,\alpha,\beta)$ is the probability of such an event given λ, α and β. This could be a failure or the number of defects identified at an inspection in this case. Since the denominator of equation (4-19) is a constant, we have

$$P(\lambda,\alpha,\beta|X_i) \propto \prod_{j=1}^{i} P(x_j|\lambda,\alpha,\beta)p(\lambda|\Phi_\lambda)p(\alpha|\Phi_\alpha)p(\beta|\Phi_\beta) \tag{4-20}$$

Given Φ_λ, Φ_α and Φ_β are known, the updated point estimates λ,α and β can be obtained by maximising equation (4-20), which is equivalent to maximising its logarithm

$$\begin{aligned} &\log\left(\prod_{j=1}^{i} P(x_j|\lambda,\alpha,\beta)\right)+\log(p(\lambda|\Phi_\lambda))+\log(p(\alpha|\Phi_\alpha))+\log(p(\beta|\Phi_\beta)) \\ &=\sum_{j=1}^{i}\log(P(x_j|\lambda,\alpha,\beta))+\log(p(\lambda|\Phi_\lambda))+\log(p(\alpha|\Phi_\alpha))+\log(p(\beta|\Phi_\beta)) \end{aligned} \tag{4-21}$$

If i tends to be ∞, the last four terms on the right hands side of equation (4-21) are negligible. Equation (4-21) will reduce to the maximum likelihood method.

The likelihood function for the observed events in this case is written as

$$L=\prod_{n=1}^{l}\left\{P_r(m_n \text{ faults at } t_n)\cdot\prod_{j=1}^{k_n}P_r\left(a \text{ failure in}\left(I_j^n, I_j^n+\Delta t\right)\right)\cdot\prod_{j=1}^{k_n+1}P_r\left(\text{no failure in}\left(I_{j-1}^n, I_j^n\right)\right)\right\} \tag{4-22}$$

where,

$$\prod_{j=1}^{k_n+1}P_r\left(\text{no failure in}\left(I_{j-1}^n, I_j^n\right)\right)=\prod_{j=1}^{k_n+1}e^{-EN_f\left(I_{j-1}^n, I_j^n\right)}=e^{\sum_{j=1}^{k_n+1}-EN_f\left(I_{j-1}^n, I_j^n\right)}=e^{-EN_f(t_n)} \tag{4-23}$$

$$\prod_{j=1}^{k_n}P_r\left(a \text{ failure in}\left(I_j^n, I_j^n+\Delta t\right)\right)=\prod_{j=1}^{k_n}\lambda\cdot\Delta t\cdot p\left(I_j^n\right) \tag{4-24}$$

$$P_r(m_n \text{ faults at } t_n)=\frac{\left[EN_p(t_n)\right]^{m_n}e^{-EN_p(t_n)}}{m_n!} \tag{4-25}$$

In perfect inspection, the expected number of faults identified at the nth inspection and the expected number of failures of the nth inspection interval are given as

$$EN_p(t_n) = \int_0^T \lambda[1 - F(T-u)]du \tag{4-26}$$

and

$$EN_f(t_n) = -\lambda \int_0^T F(u)du \tag{4-27}$$

Substituting equations (4-23), (4-24), (4-25), (4-26) and (4-27) into equation (4-22), and taking the logarithm of the likelihood function gives

$$\log(L) = \sum_{n=1}^{l} \left\{ \log\left[\frac{1}{m_n!} \cdot e^{-\lambda \int_0^T (1-F(T-u)du)} \cdot \left[\lambda \int_0^T (1 - F(T-u)du) \right]^{m_n} \right] + \log\left[e^{-\lambda \int_0^T F(u)du} \right] + \sum_{j=1}^{k_n} \log\left[\lambda p\left(I_j^n\right)\right] \right\} + CONSTANT\ . \tag{4-28}$$

When the failure number and the fault number identified at inspections are known over a specific period of observation, substituting the data into the log likelihood equation (4-28), the parameters α, β, λ can be obtained by maximising log$L$.

4.3 SUBJECTIVE DATA ACQUISITION

The same method used in the single component model, moment matching, may be used for complex system subjective parameter estimation. There are three parameters in the model, the defect arising rate of the HPP process, and the two parameters of the delay time distribution. Because these three parameters are treated as random variables following a two-parameter Gamma distribution, the number of parameters are doubled. If the scale parameters of two Gamma distributions for α and β are fixed, there are

four parameters remain to be estimated. We need at least four subjective data to equal to the corresponding counterparts with four unknown parameters.

In the parameter estimation model, the subjective data have been mentioned. They are

- Mean number of failures within (0, T) ($=N_{afail}$)
- Mean maximum number of failures within (0, T) ($=N_{mfail}$)
- Mean number of defects arising within (0, T) ($=N_{ad}$)
- Mean maximum number of defects arising within (0, T) ($=N_{md}$)

We also can obtain

$$N_{md} = N_{mfail} + N_{mfault} \tag{4-29}$$

and

$$N_{ad} = N_{afail} + N_{afault} \tag{4-30}$$

where

- N_{afault} denotes mean number of faults identified at T.
- N_{mfault} denotes mean maximum number of faults identified at T.

Other information could be required such as the mean downtime per week, which can also be used as part of the estimating process.

When these estimation questions are presented to different engineers, they might give different answers. Here we use a weight average to combine those engineers' opinions. Obviously, the work experience will determine the weight of engineers.

In the decision model, we used several other parameters, such as the average cost and time duration of inspection and failure repair. This information may be acquired directly from engineers or management of the plant, or from existing data records, To complete the software design for subjective data in the first instance, the

questionnaire utilise to acquire these information is incorporated within the software package.

Chapter 5 DTM Software Development

In previous chapters, the main delay time models have been introduced, which will be the basis for subsequent software development. In this chapter, the main steps of developing the software will be introduced, including system analysing, database creating, and program designing.

5.1 SYSTEM ANALYSIS AND DESIGN

Since there are fundamental differences between a single component model and a complex system model, the proposed software has been developed as almost two separate program packages. It is noted however, that the basic modelling steps of the two models are very similar. Those steps include subjective data acquisition, objective data acquisition, parameter estimation using both subjective data and objective data, and the decision model.

We first start with a single component delay time model. In order to develop the delay time model of a single component, it is necessary to analyse model variants carefully according to their input, output information and the processing procedure. In the subjective data acquisition phase, there are three types of input information, which are component information, expert information and expert opinion of the component. The component information is the ID data of the component, such as the identification code, name...etc. The expert information includes name, position, and the assessment weight given to each expert, which is used to combine the opinions of different experts. The experts' opinion is the subjective failure information on the component, which is obtained by an expert survey, and then used in subsequent parameter estimation and decision making models. The output information should be the combined subjective information from experts of the component. Figure 5-1 shows the idea of the system analysis of the model for a single component.

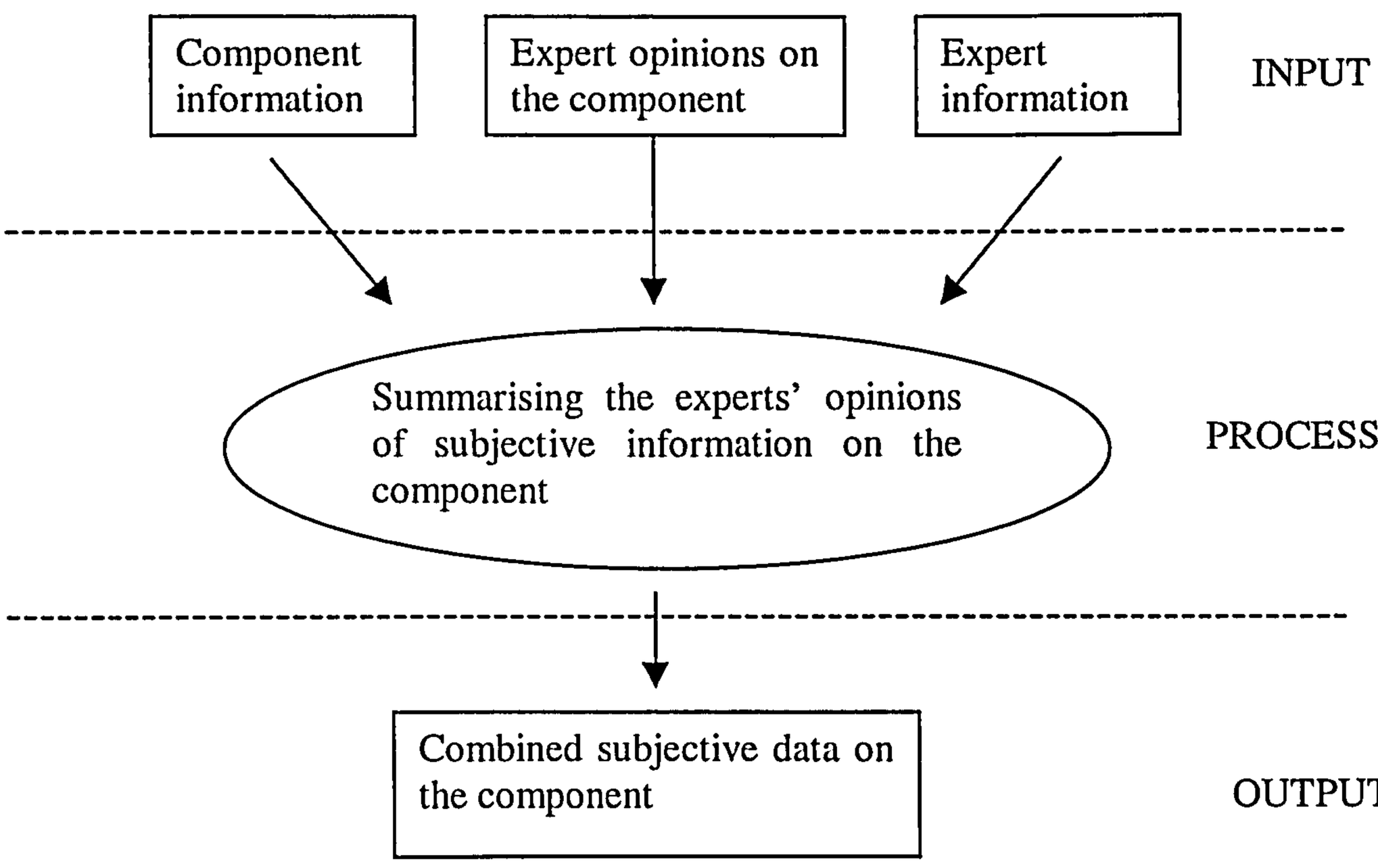


Figure 5-1 Subjective data acquisition model of the single component

As introduced in the previous chapters, the Bayesian approach is used for parameter estimation based on both subjective data and objective data. When either subjective data or objective data is available, Bayesian approach can also be applied to each of the data type. It is usually used in the situation where subjective data is first available and then the model updates the estimates when objective data becomes available. The input data is either subjective data or objective data on the component. Output data are the estimated parameters of the delay time model which are estimated by the Bayesian approach. See Figure 5-2.

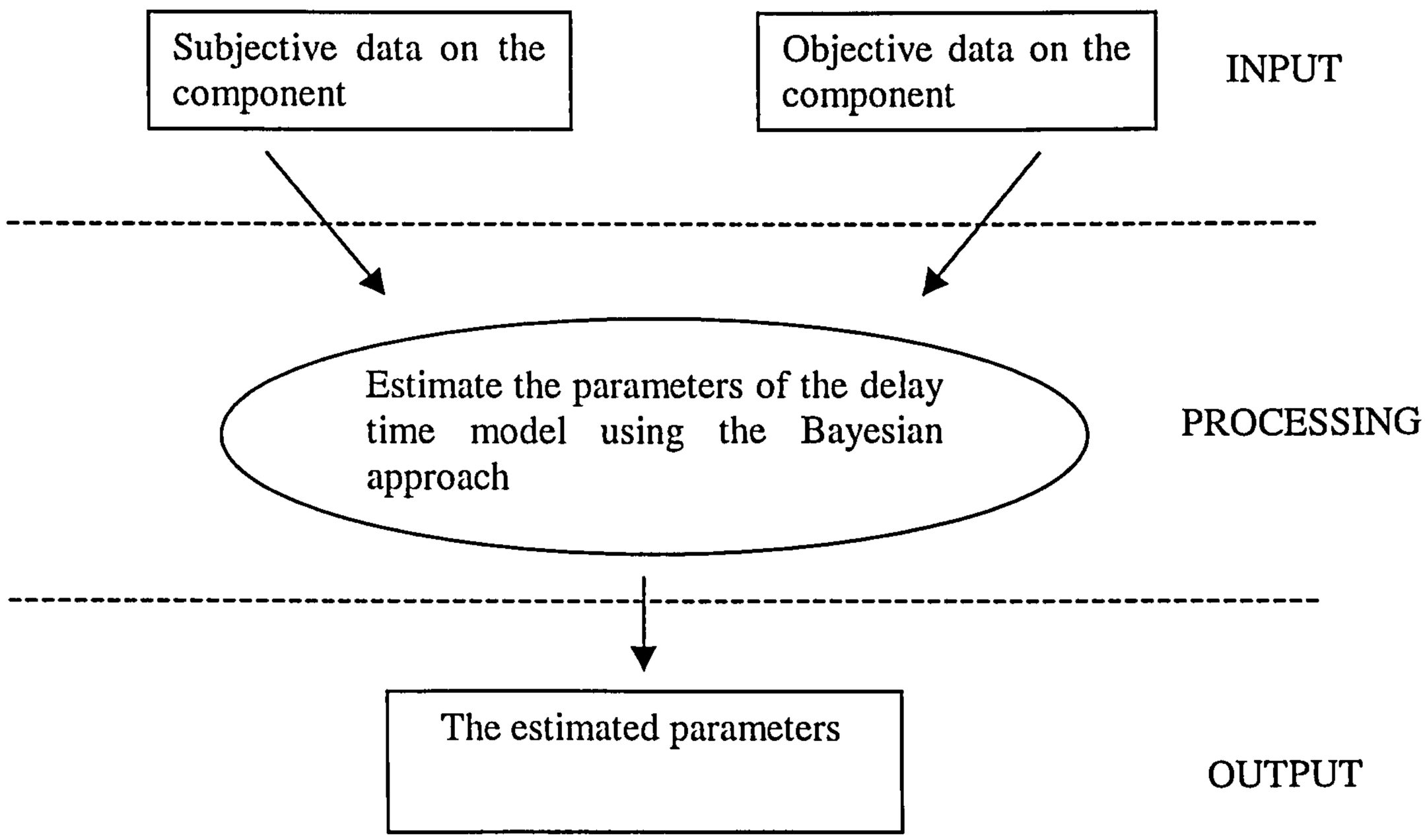


Figure 5-2 System analysis of parameter estimation model using both subjective data and objective ones

In subsequent decision making model, the input data are parameters of delay time models and some other subjective data obtained from engineers. The output is the expected downtime or cost per unit time of the component via inspection interval. Figure 5-3 shows the system analysis of the decision making model.

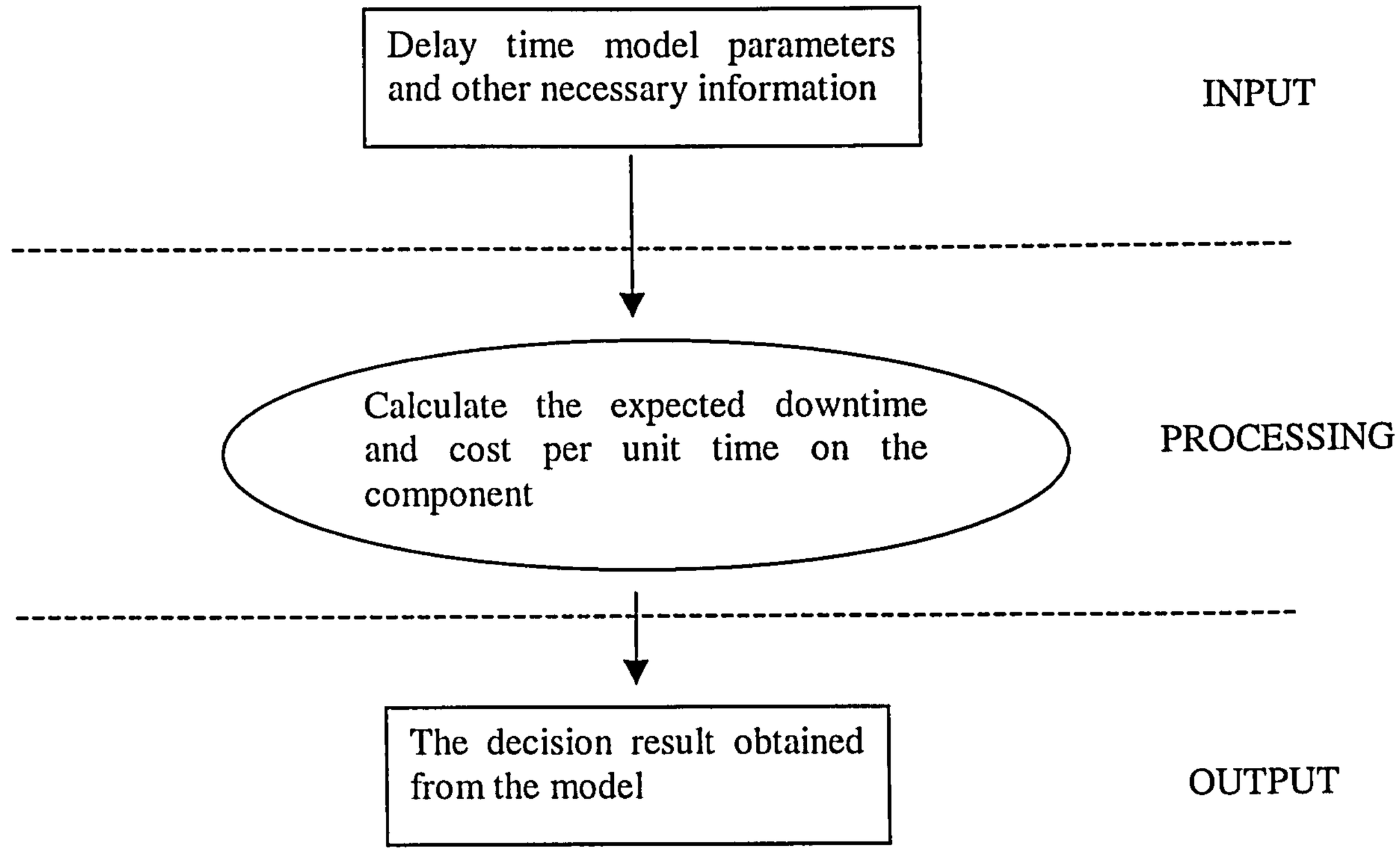


Figure 5-3 System analysis of the decision making model of a single component

From above input-process-output charts, the single component delay time model can be divided into several separated sub-models. As shown in Figures 5-1 – 5-3, the output information of one model is the input information of another, so the input and output information connect all models together.

The complex system model could also be divided into four sub-models, the same as the single component model. Although the mathematical process of the complex system model is different from that of a single component, the methodology of data processing is the same. In the three sub-models of the complex system, namely the subjective data acquisition model, estimation model and decision model, the input and output information are similar to that of the single component model. In Figures 5-4 – 5-6, the system analysis charts of the complex system delay time model are shown.

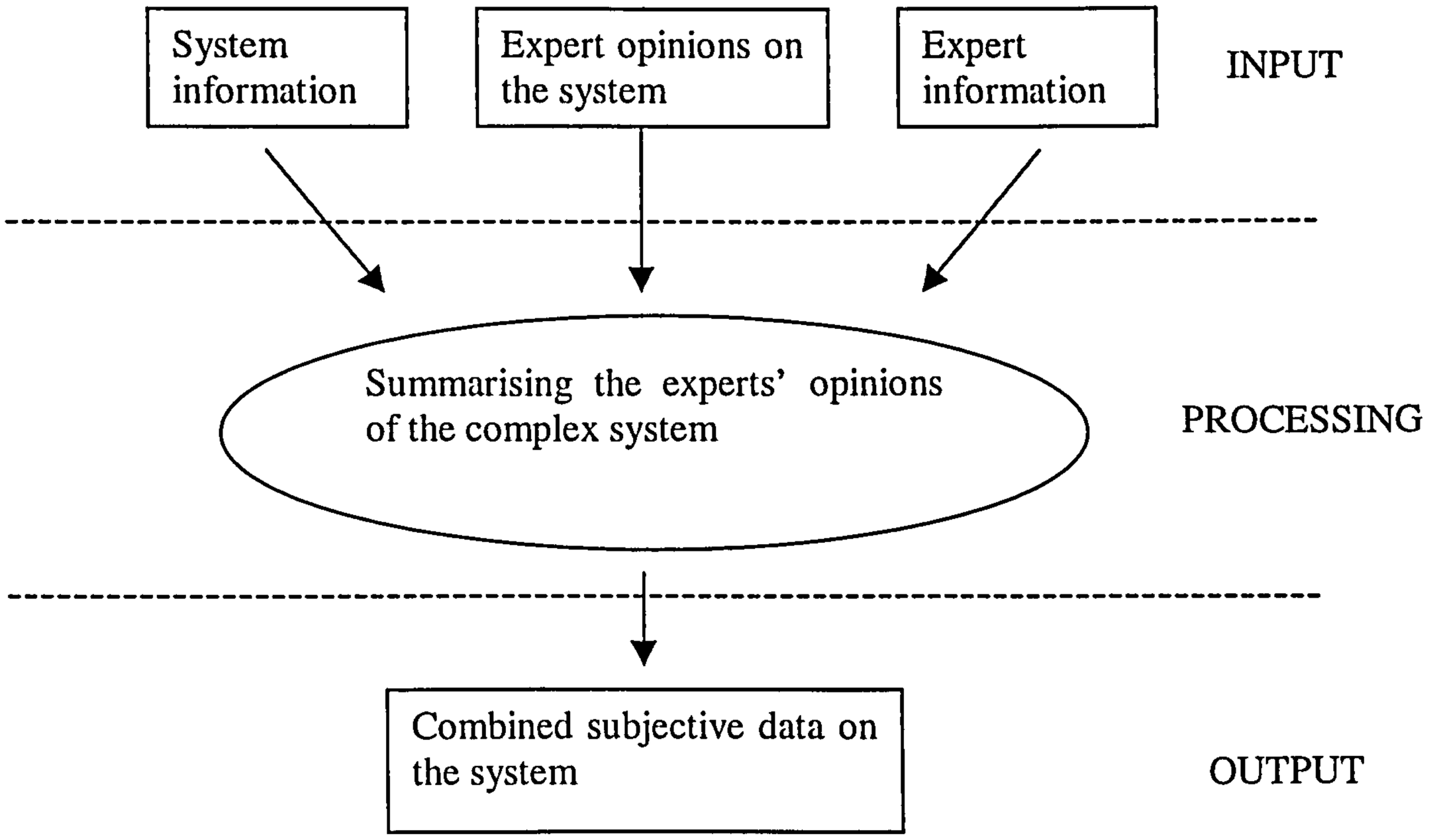


Figure 5-4 Subjective data acquisition model of the complex system

Note: In Figure 5-4, the system information is the information of the system itself, such as the identification code, name, failure areas of the system, ... etc. The expert information is the weight of each expert, which is used to combine the opinions of different engineers. The expert opinion is the subjective data of the system, which is used in parameter estimation and decision making models.

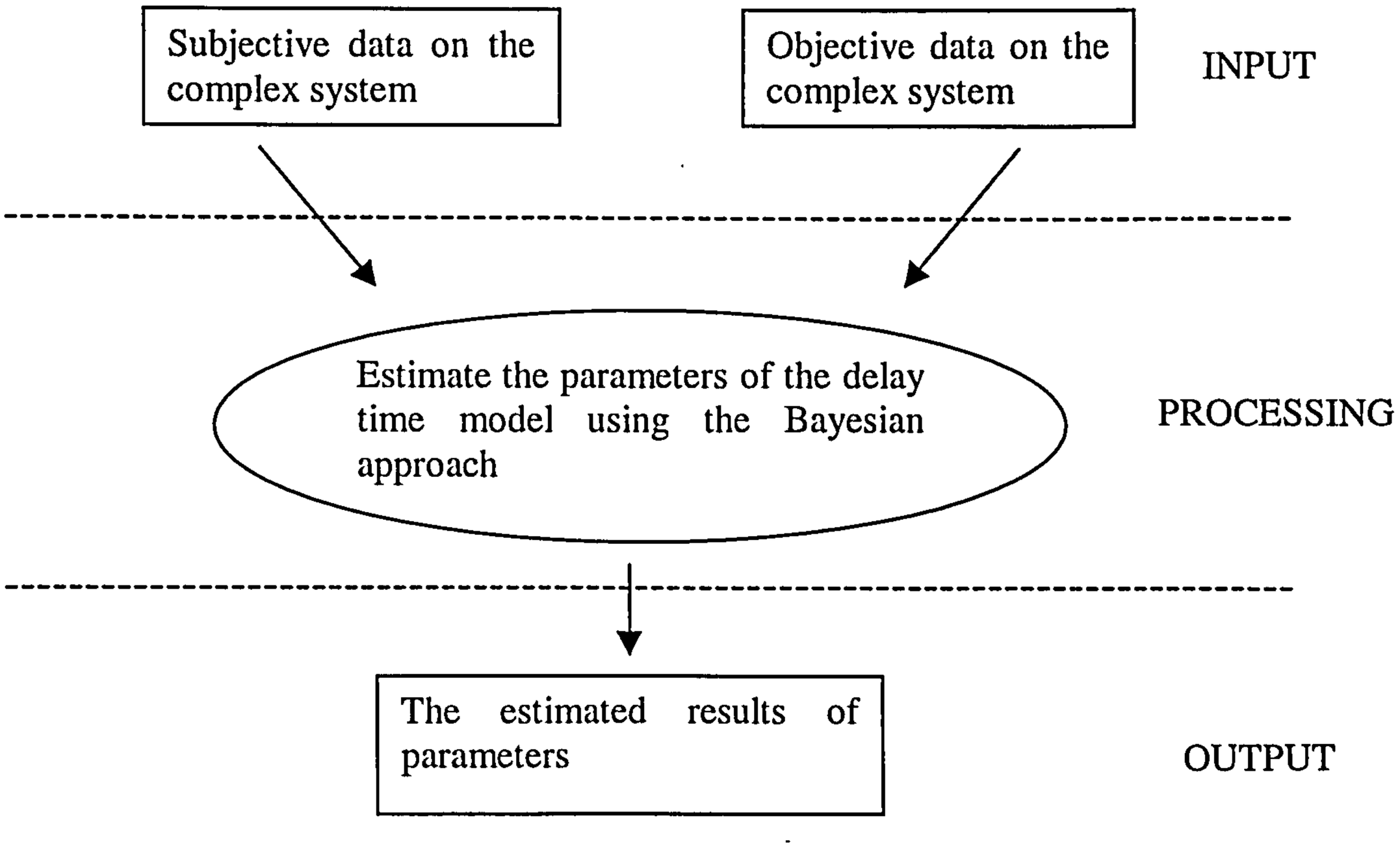


Figure 5-5 System analysis of parameter estimation model using both subjective data and objective data

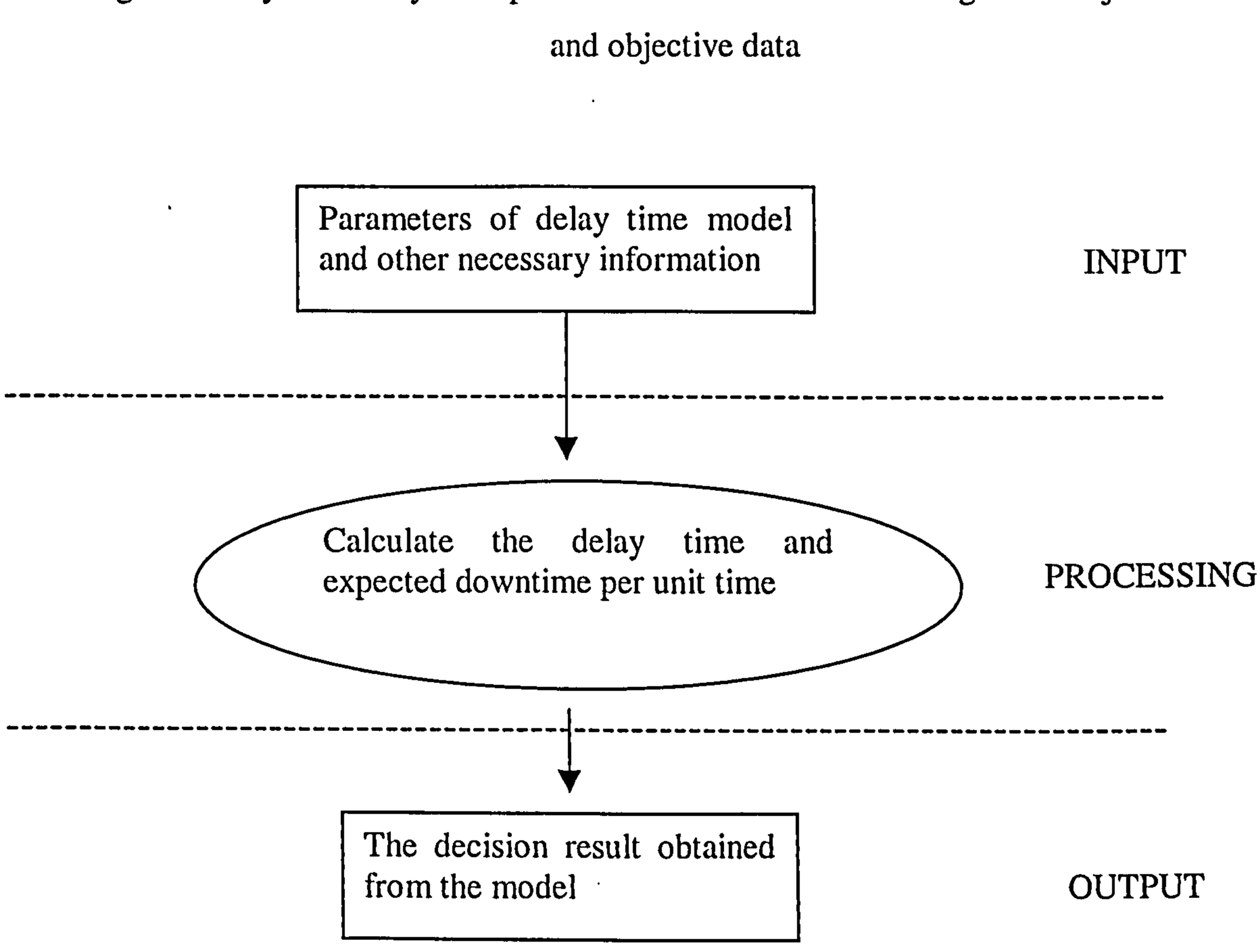


Figure 5-6 System analysis of decision making model of complex system

5.2 DATABASE DESIGN

The information needed by the delay time model must be stored in the system for subsequent processing. The system analysis, see Figure 5-7, illustrates the data flow of the delay time modelling application

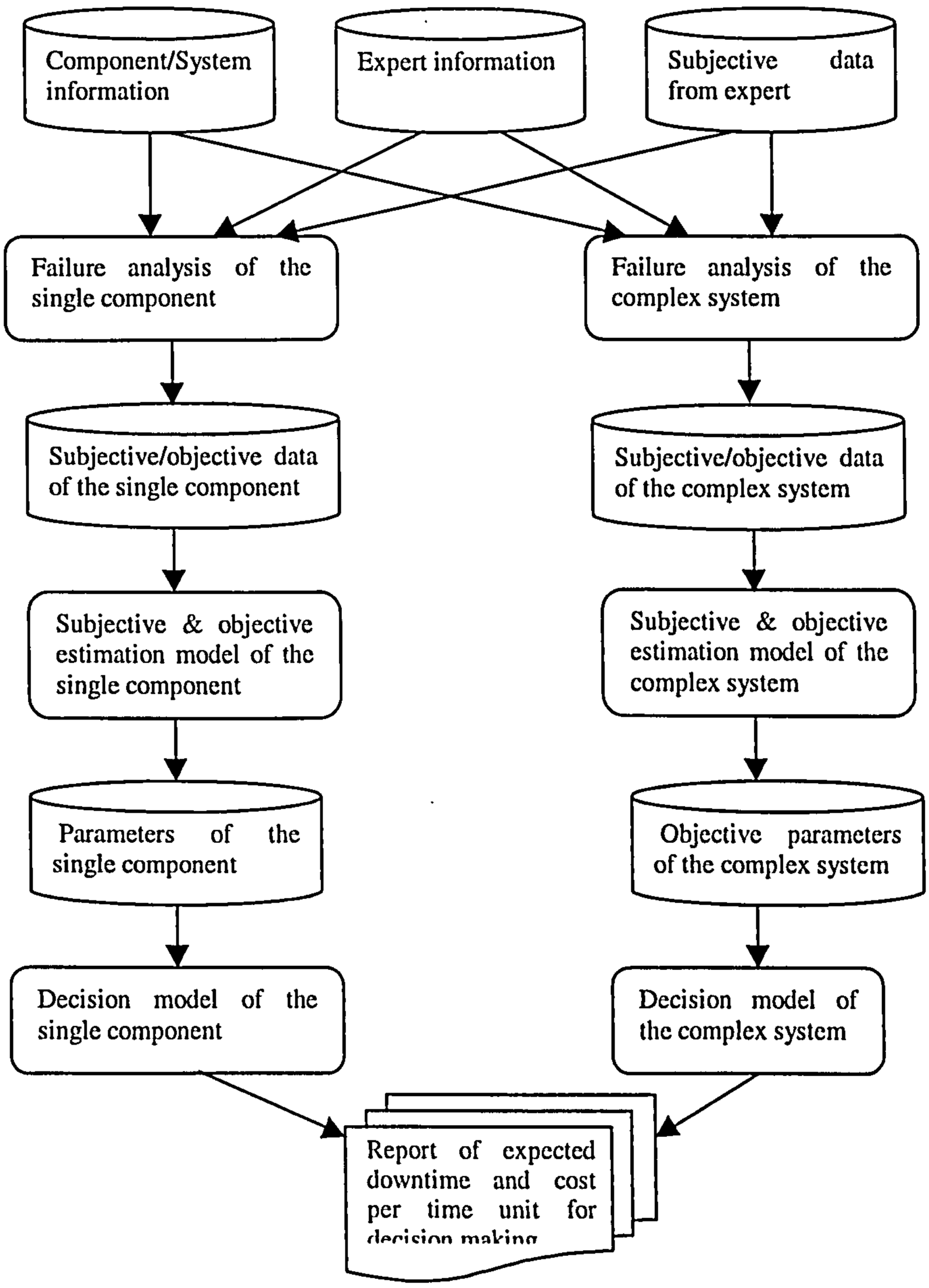


Figure 5-7 Data flow in the application

In Figure 5-7, the information unit will be designed as tables in the database. Every information unit is one column in the table, see the following description of the tables in the application.

- **Component/System information table**; for storing the technical information of the component or system, including component/system ID, name, and short name…
- **Expert information table**; a table of expert information of the experts including expert name, job title, and weight…
- **Subjective data information table**; a table including all possible subjective data on the component or system which will be used in the application, such as mean failure number per year, mean number of defect arising per year, …
- **Objective data information table**; a record of the daily maintenance information of the component or system. The columns include failure time, defect type, fault area, cause, cost, downtime duration…
- **Parameter information tables**; for storing the estimated results of the parameters of the delay time model. The columns are the parameters.

5.3 PROGRAM DESIGN

Based on the process of system analysis and database design, the application could consist of many programs, which implement the functions of the application and maintain the data flow of the application. Some of the programs are to process data input, output and to store the data into the corresponding database. Some of the programs are developed for manipulating the mathematical functions in the delay time model. The main program is shown in Figure 5-8.

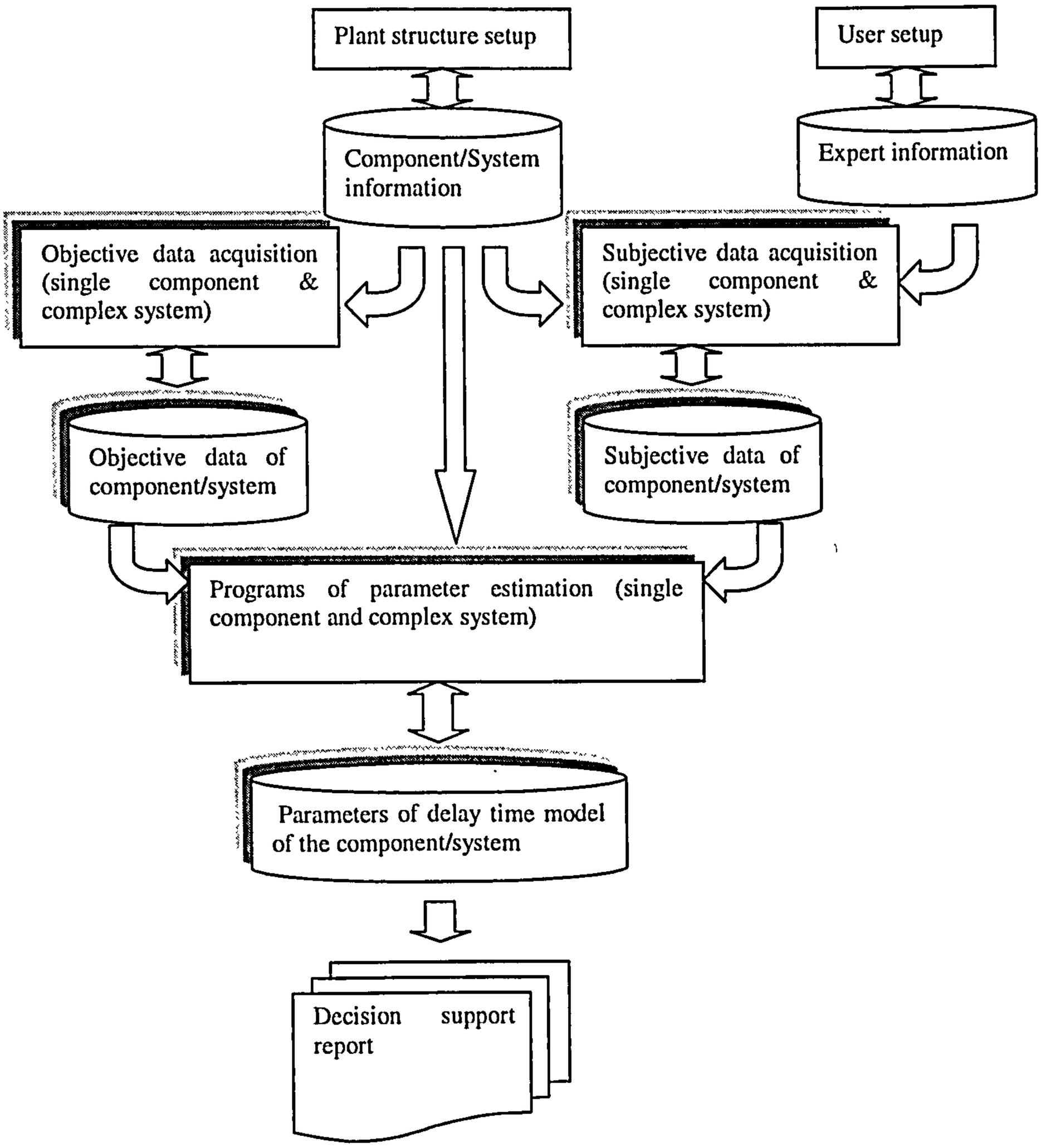


Figure 5-8 Programs and databases chart

5.3.1 Login program

It is the first program of the application. It ensures that only authorised person can access and use the software. The user, who wants to use the application, must type in the correct user name and password, which are pre-stored in the system.

Program Flow Chart

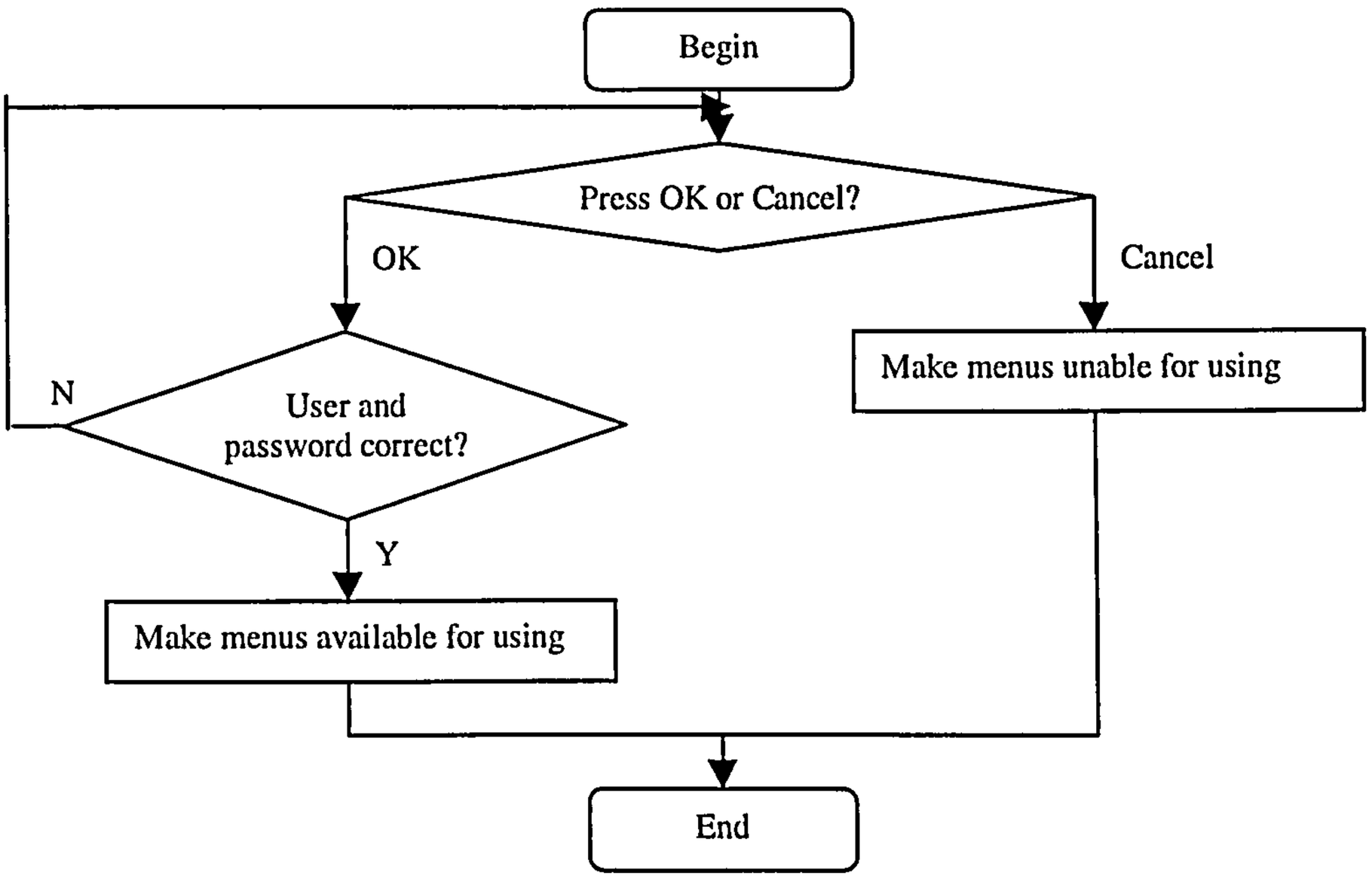


Figure 5-9 Program flow chart (login program)

Form Design

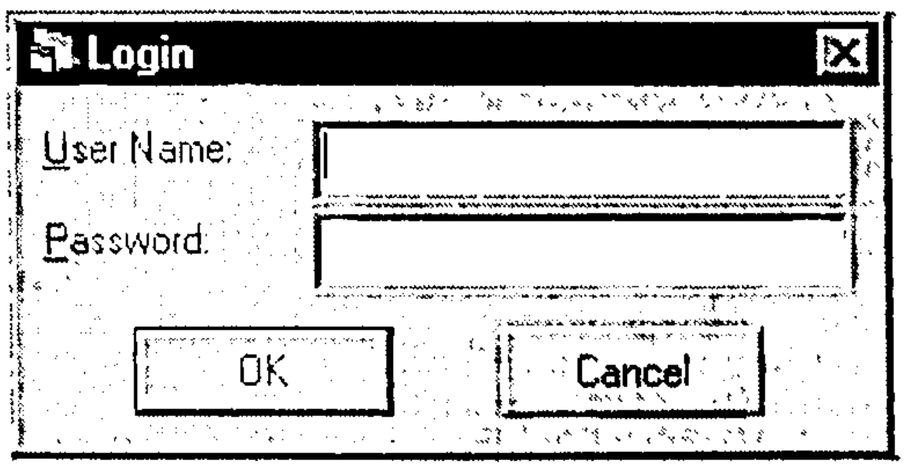


Figure 5-10 Form design of login program

Form Statement

1. On pressing the OK button, the program will check the user table for the input user and password. If all information are correct, all menus of the software will be enabled. When either of the information is incorrect, a message will be shown and ask the user to input again and all menus will be set to unavailable.
2. Upon pressing the Cancel button, all menus of the software will be unable for using. The form will be closed.

5.3.2 Plant structure setup program

This program is designed to process the plant structure information in a company. In a company there might be several workshops, which possibly have many production lines. There might be several machines in a product line. See figure 5-11. It shows a production line structure of a food company. There are two plants in the line, one is a canning plant, the other is a packing plant. Two machines are in canning plant, filler and seamer.

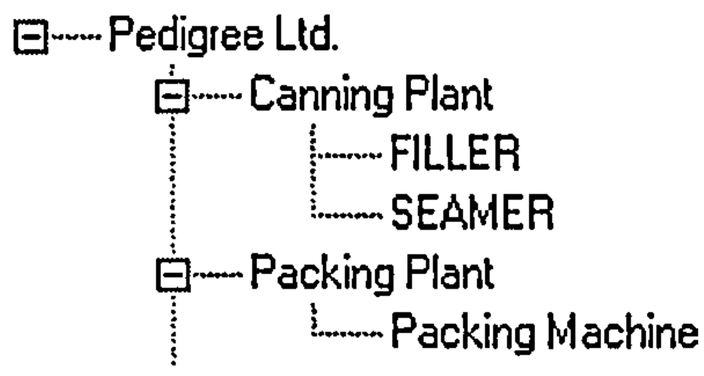


Figure 5-11 Plant structure of Pedigree Ltd.

The maintenance unit is in the bottom level of the structure. It could be a single component or a complex system. In figure 5-11, filler and seamer are the maintenance units.

Program Flow Chart

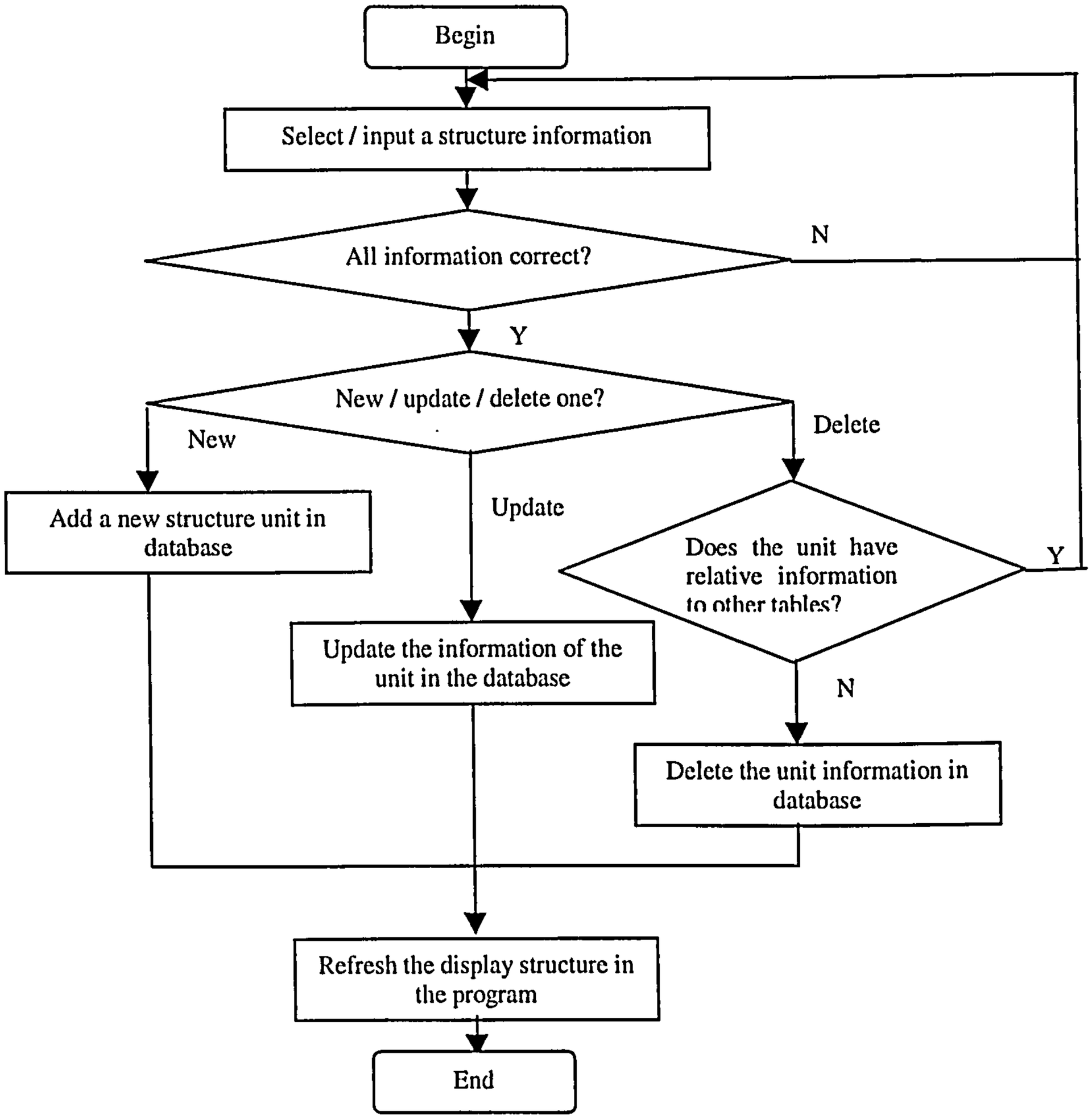


Figure 5-12 Program flow chart (Plant structure setup program)

Form Design

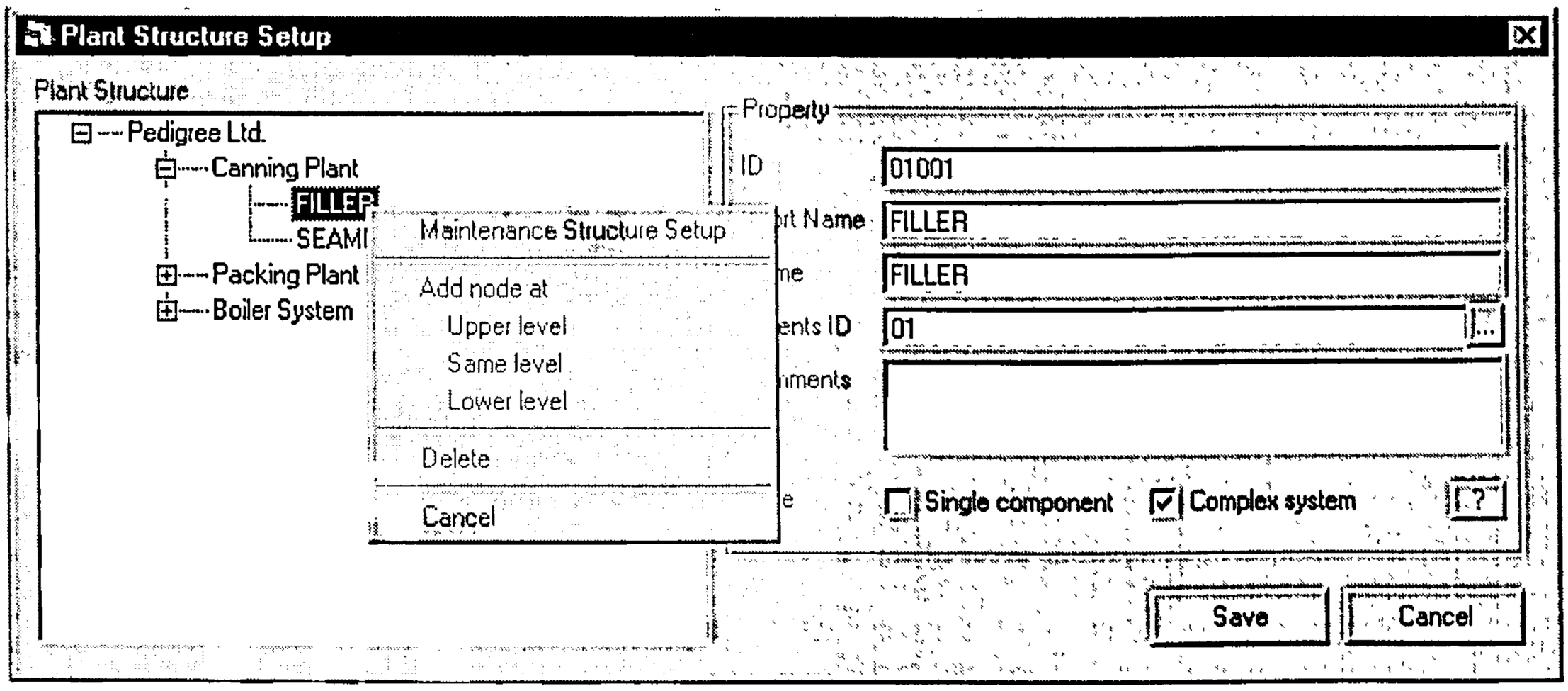


Figure 5-13 Form design of plant structure setup program

Form Statement

1. The left white frame displays the plant structure. Upon pressing the right button of the mouse on a node of the structure tree, a pop-up menu appears. When choosing the upper level menu item, the program will setup a new node, which is the upper level of the current node. Choosing the same level, the program will add a new node at the same level with the current node. Choosing the lower level, program will add a new node as a child of the current node. When choosing delete menu item, the program will check if there is some relative information with the current node in other tables. If more than one exists, the program will not delete the node unless the relative information has already been deleted.
2. The right area is the information area. When a user presses the mouse on the structure node, all information of the node will be displayed in this area. The user can modify all information except for the ID. Parent ID is the ID of the parent node of the current node. If the user changes this information, the current node will become the child of the node the user input.
3. The bottom level of the structure should be the maintenance unit. The user needs to tick the box of node style, either single component or complex system.

5.3.3 User setup program

This program is designed to manipulate personal information of the experts and users. It is possible for expert to be the user of the software who inputs daily maintenance records into the database. The personal information includes job title, name, user type... etc.

Program Flow Chart

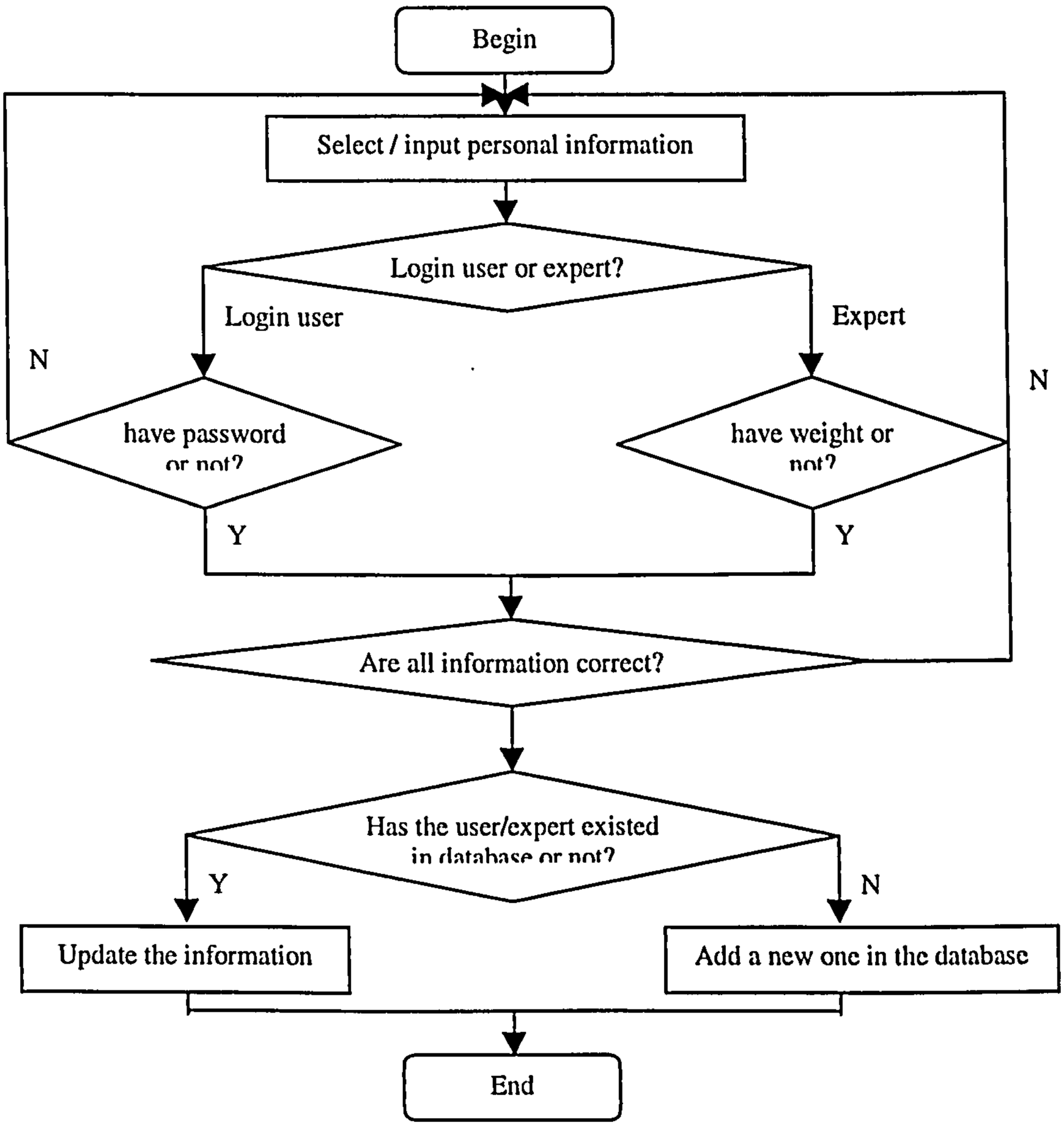


Figure 5-14 Program flow chart (User setup program)

Form Design

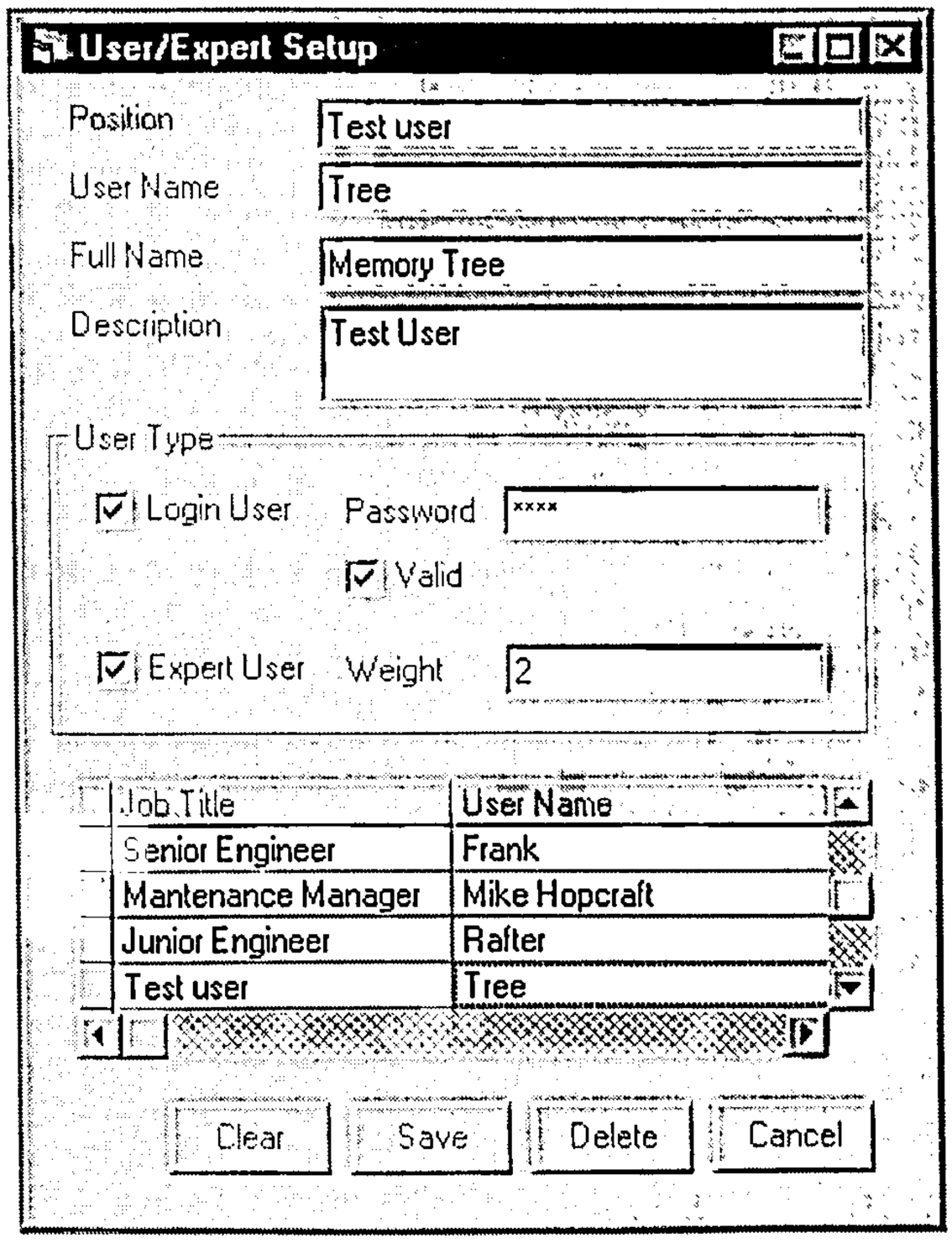


Figure 5-15 Form design of user setup program

Form Statement

1. Yellow fields are mandatory input fields.
2. The table at the bottom of the form can display all users stored in the database. Click on one row of the table, the information will be displayed on the top area. The user can do any change to the record, and then save it.
3. When click on the clear button, all input fields will be blank.
4. When click on the save button, the program will check whether all input fields are available? Then those pieces of information will be stored into the database. If a user has existed in the database, it will be updated. If not, a new user will be added into the database.
5. When click on the delete button, the selected user will be deleted from the database.
6. Click on the cancel button to exit from the program.

5.3.4 Subjective data input (single component)

In this program, user can input all subjective data of a single component into database, which will be used for parameter estimating and the decision model. Those data can be obtained from different engineers. The subjective data includes fault area information, the cause of the fault and prevention means of the faults.

Program Flow Chart

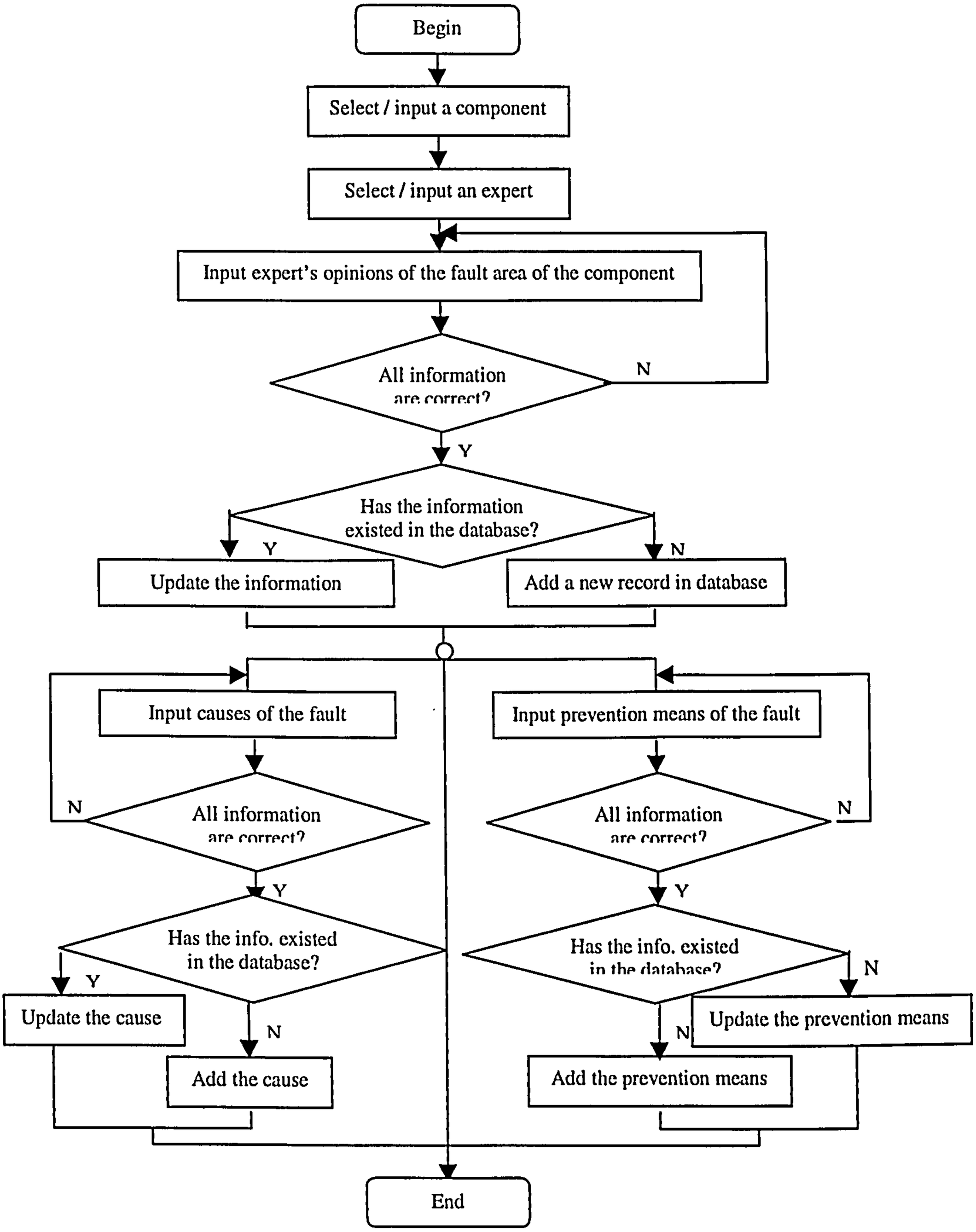


Figure 5-16 Program flow chart (Subjective data acquisition of a single component)

Form Design

Fault area form:

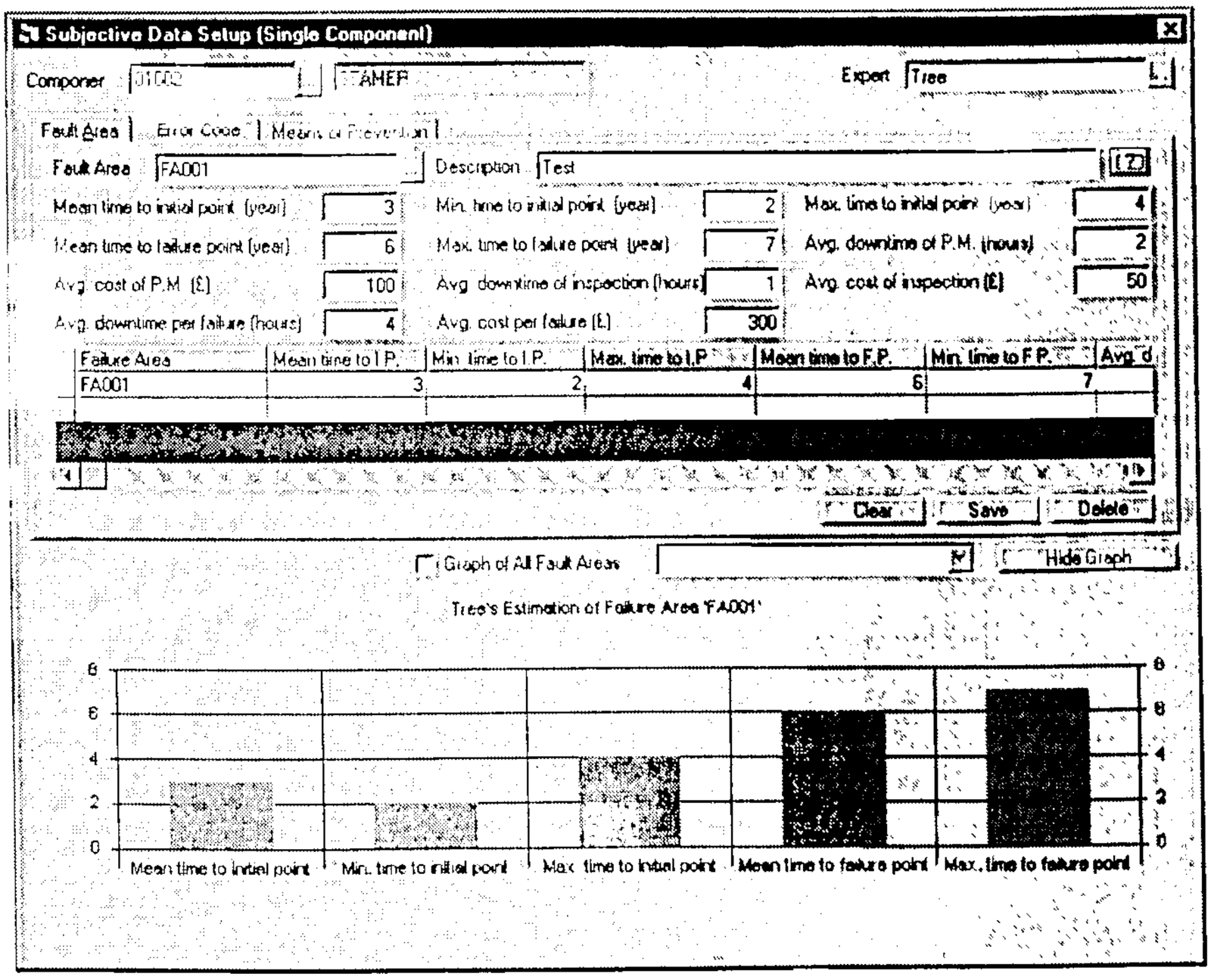


Figure 5-17 Form design of fault area information acquisition

Error code form:

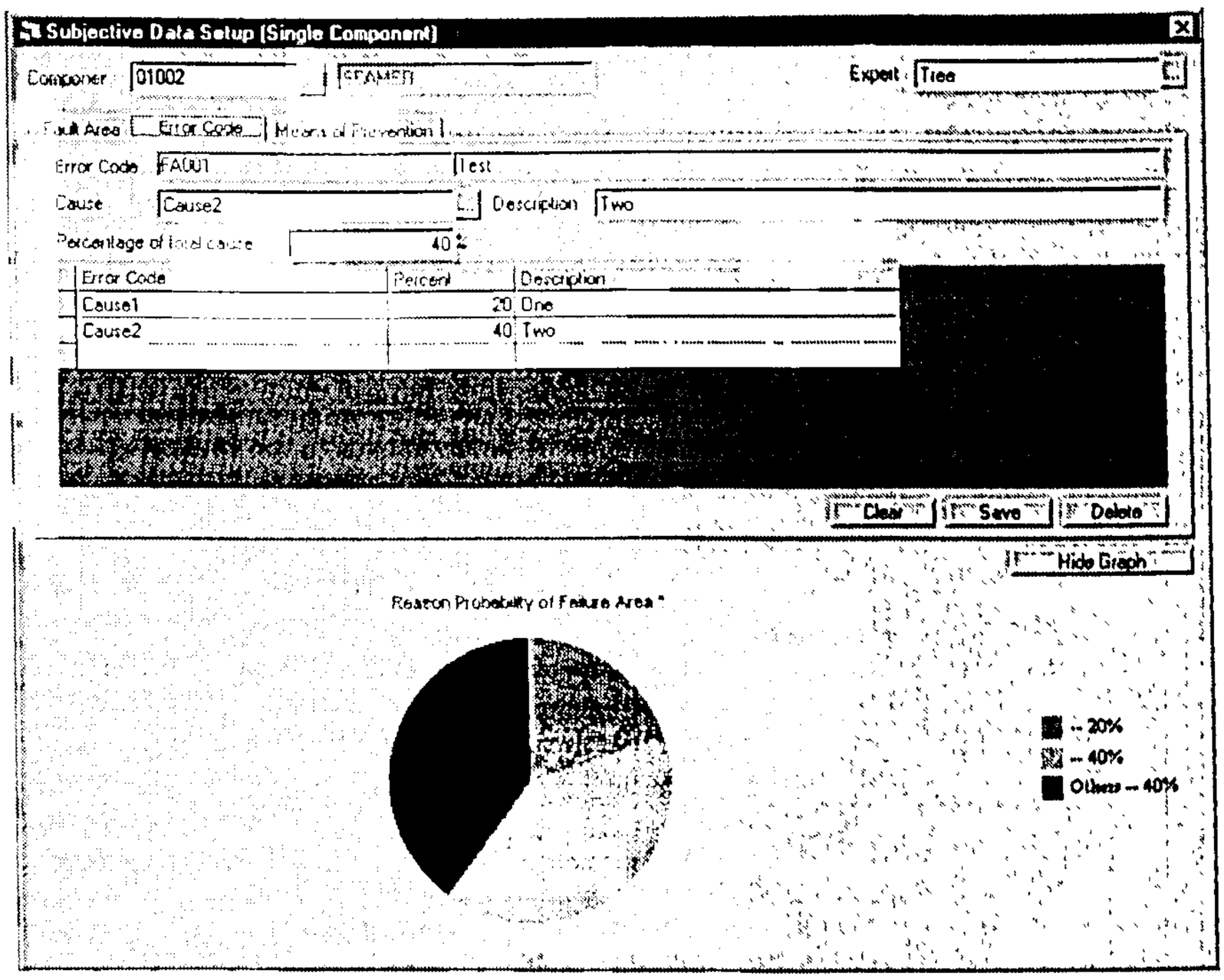


Figure 5-18 Form design of error code information acquisition

Prevention means of faults form:

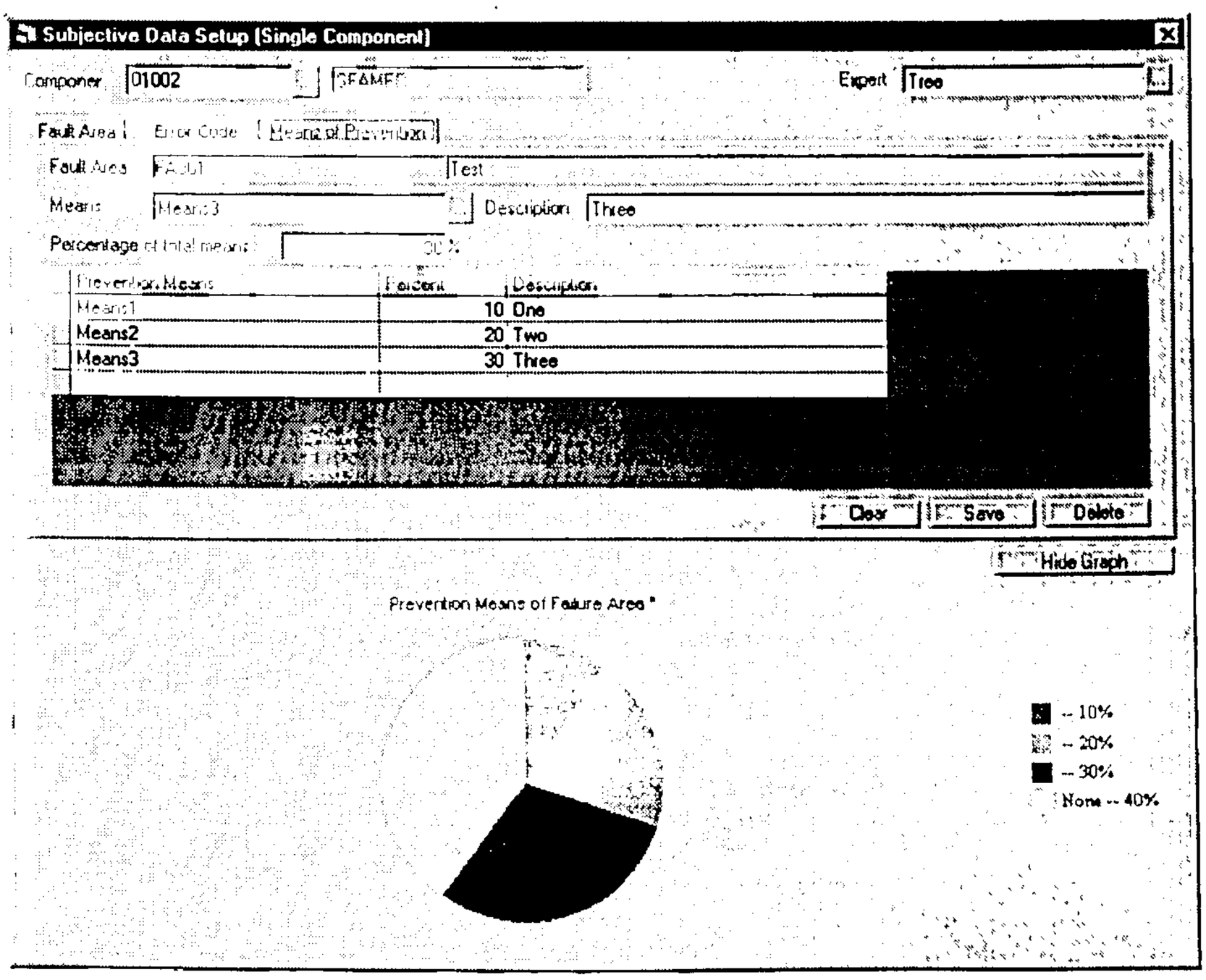


Figure 5-19 Form design of prevention means of fault acquisition

Form Statement

Fault area form:

1. Yellow fields are mandatory input fields.
2. When click on the clear button, the input fields of the fault area will be blank
3. When click on the save button, the program will check all input data first, and then save them into the database. When making any change on fault area information, the maximum data should not be less than the mean data and minimum data, such as max. time to initial point should be larger than mean time to initial point. The relationship between these three data should be

max. >= mean >= min. If there is conflicting information, the program will issue an error message.

4. The table will display the fault area information, which is from the database. Double click on the row of the table, the fault area information will be filled into areas above the table. User can also change the information and save it into the database.
5. The graph, which is at the bottom of the form, gives a direct display of the subjective data. It can help engineers to evaluate and check their opinions.

Error code form:

1. Yellow fields are mandatory input fields.
2. Before inputting the error code, a fault area must be selected and displayed in the grey fields.
3. A fault may be caused by many reasons. The summarised value of error average percent can't be larger than 100%.
4. The pie chart displays the percentage of each error.

Prevention means form:

1. Yellow fields are mandatory input fields.
2. Before input prevention means, a fault area must be selected and displayed in the grey fields.
3. A fault may be prevented from re-occurring by many means. The summarised value of all prevention means can't be larger than 100%.
4. The pie chart displays the percentage of each prevention means.

5.3.5 Subjective data input (complex system)

This program is similar to the subjective data input program of a single component model. There might be several fault areas in a complex system. The program can collect subjective data of the complex system from many engineers. A user can use this program to add and update the subjective date in the database.

Program Flow Chart

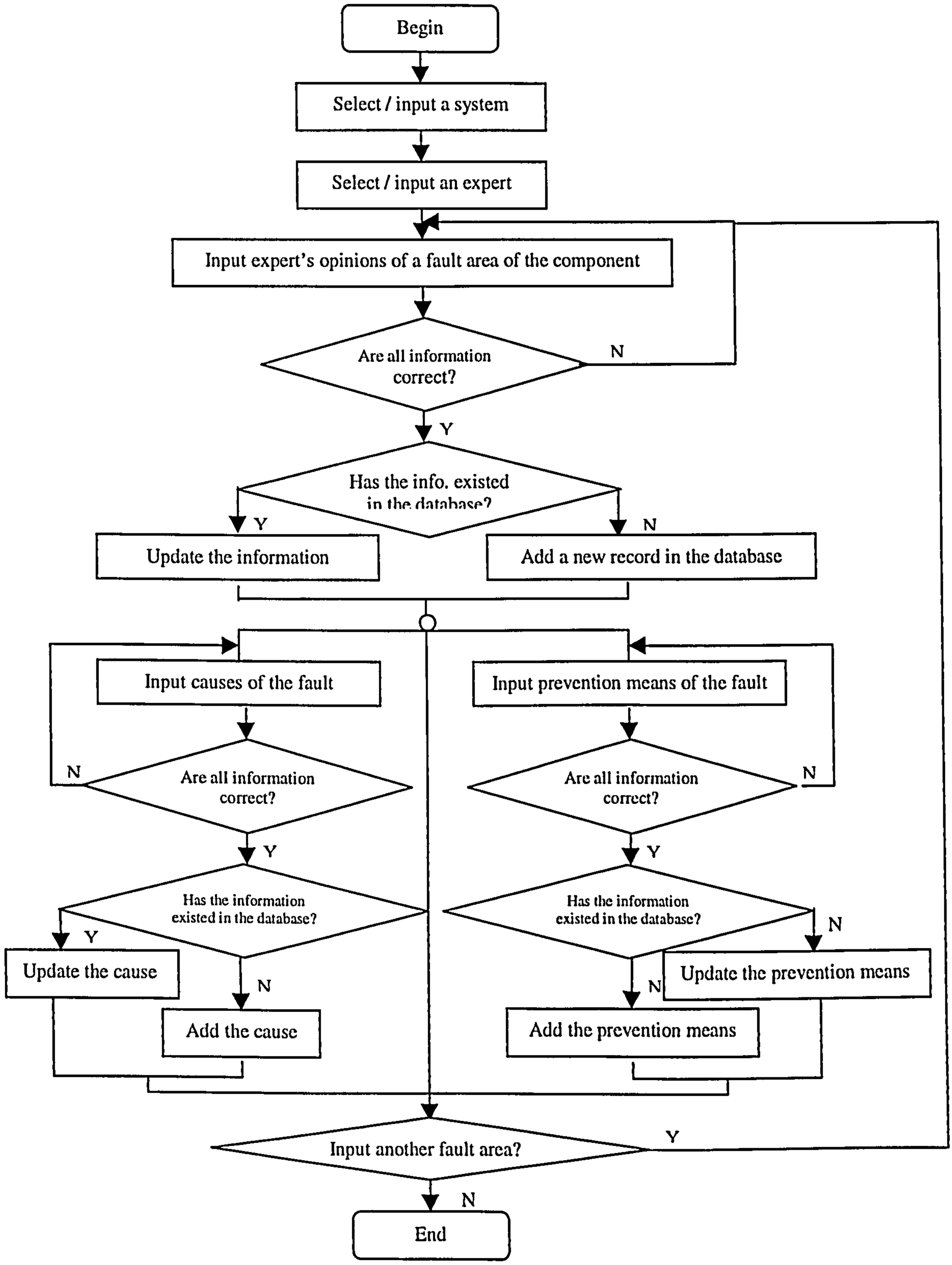


Figure 5-20 Program flow chart (Subjective data acquisition of a complex system)

Form Design

Fault area form:

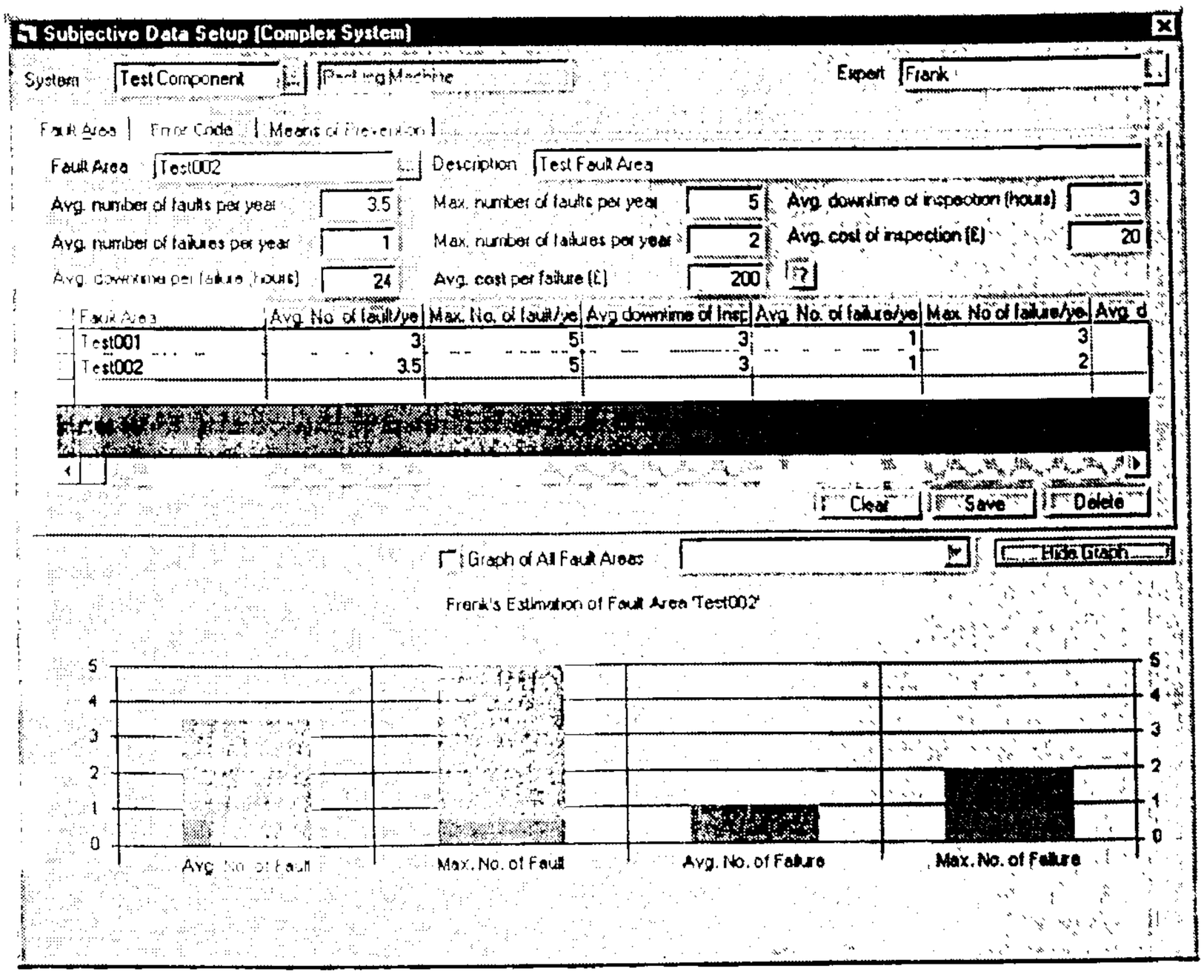


Figure 5-21 Form design of fault area information acquisition

Error code form:

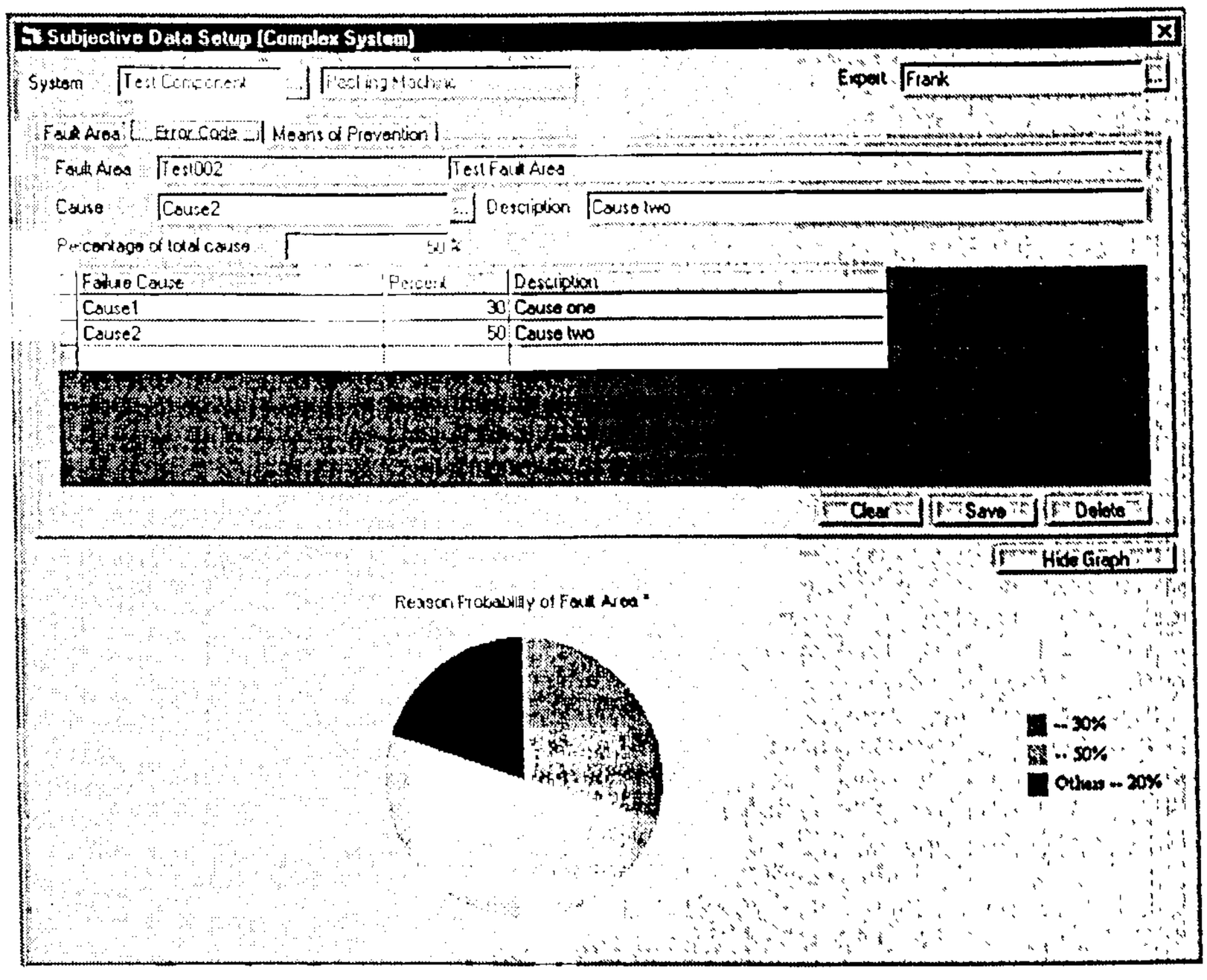


Figure 5-22 Form design of error code information acquisition

Prevention means of fault form:

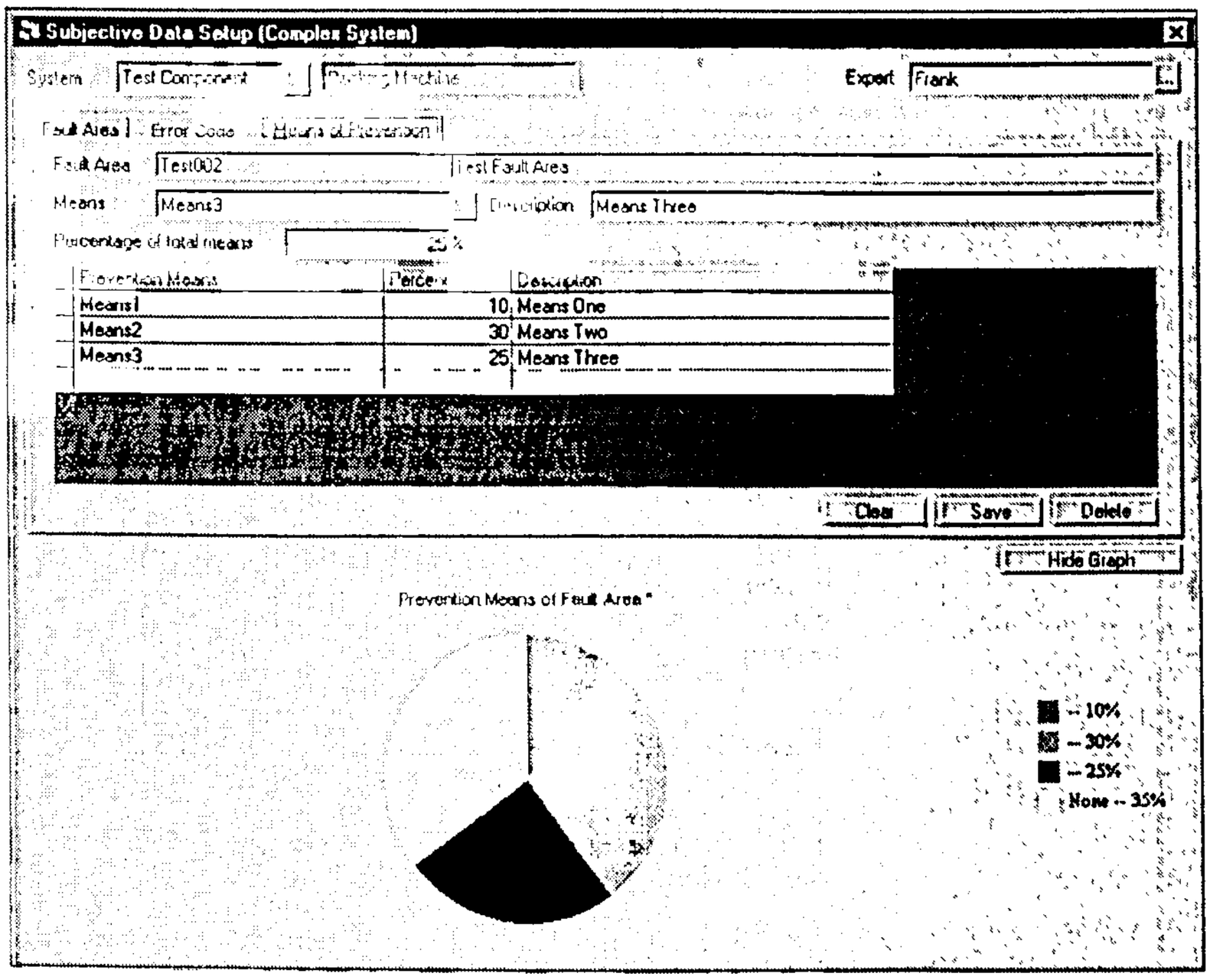


Figure 5-23 Form design of prevention means of fault acquisition

Form Statement

Fault area form:

1. Yellow fields are mandatory input fields.
2. There could be more than one fault areas in a complex system.
3. Upon clicking on the clear button, the input fields of the fault area will be blank
4. Upon clicking on the save button, the program will check all input data first, and then save them into the database. Before making any change on fault area information, the maximum data should not be less than the average data, such as Avg. number of faults per year should be less than Max number of faults per year. The relationship between them should be Max. ≥ Avg. If there is conflicting information, the program will issue an error message.

5. The table will display the information of the fault areas, which is from the database. By double clicking on the row of the table, the fault area information will be filled into the areas above the table. User can also change the information and save it into the database.
6. The graph, which is at the bottom of the form, gives a direct display of the subjective data. It can help engineers to evaluate and check their opinions.

Error code form:

1. Yellow fields are mandatory input fields.
2. Before inputting the error code, a fault area must be selected and displayed in the grey fields.
3. A fault may be caused by many reasons. The summarised value of error average percent can't be larger than 100%.
4. The pie chart displays the percentage of each error.

Prevention means form:

1. Yellow fields are mandatory input fields.
2. Before inputting prevention means, a fault area must be selected and displayed in the grey fields.
3. A fault may be prevented from re-occurring by many means. The summarised value of all prevention means can't be larger than 100%.
4. The pie chart displays the percentage of each prevention means

5.3.6 Objective data acquisition (for both single component and complex system)

This program is for inputting the daily maintenance record. It applies to a single component model or a complex system model. The main information includes the date of the maintenance events, fault type, cause...etc. The information will be stored into the database by using this program. A user can also use this program to maintain the information.

Program Flow Chart

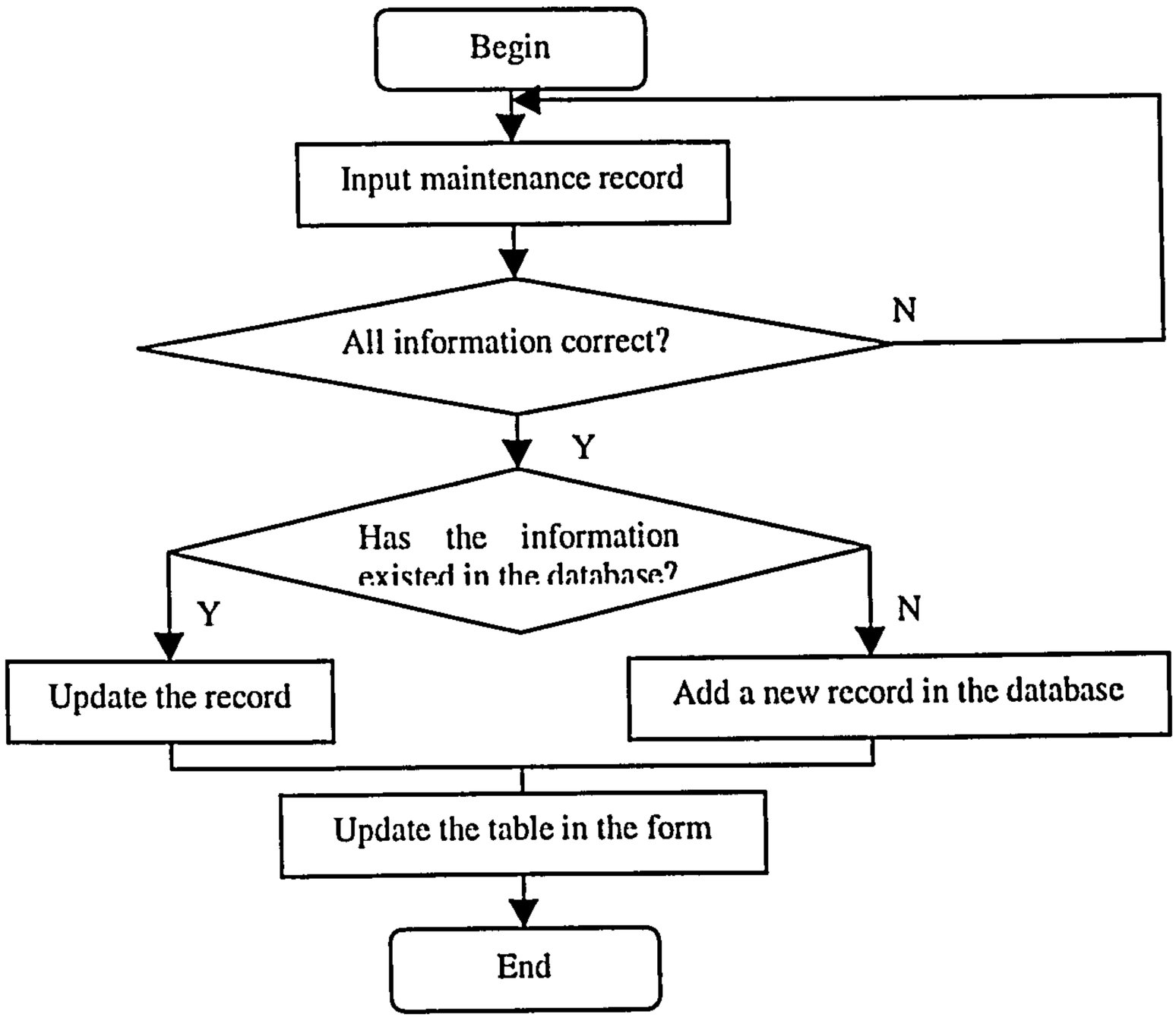
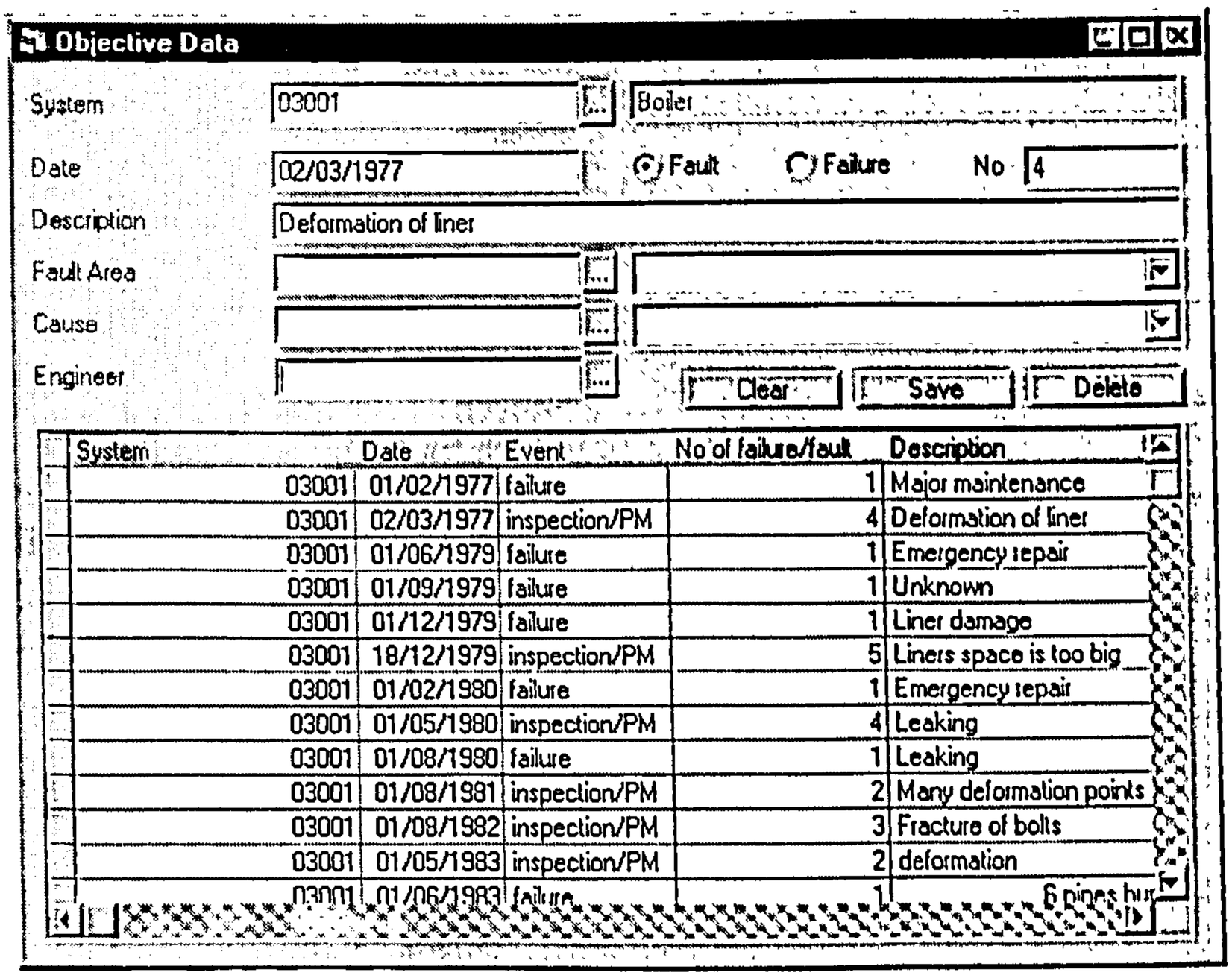


Figure 5-24 Program flow chart (Objective data acquisition)

Form Design



System	Date	Event	No of failure/fault	Description
03001	01/02/1977	failure	1	Major maintenance
03001	02/03/1977	inspection/PM	4	Deformation of liner
03001	01/06/1979	failure	1	Emergency repair
03001	01/09/1979	failure	1	Unknown
03001	01/12/1979	failure	1	Liner damage
03001	18/12/1979	inspection/PM	5	Liners space is too big
03001	01/02/1980	failure	1	Emergency repair
03001	01/05/1980	inspection/PM	4	Leaking
03001	01/08/1980	failure	1	Leaking
03001	01/08/1981	inspection/PM	2	Many deformation points
03001	01/08/1982	inspection/PM	3	Fracture of bolts
03001	01/05/1983	inspection/PM	2	deformation
03001	01/06/1983	failure	1	6 pipes bur

Figure 5-25 Form design of objective data input

Form Statement

1. Yellow fields are mandatory input fields.
2. The bottom table displays all maintenance records of the component or system. Click on a row of the table, the maintenance information will be displayed in the top area. A user can add or update the maintenance information in the database through this program.
3. Upon clicking on the clear button, the input fields will be blank.
4. Upon clicking on the save button, the program checks the input data firstly. If all data is correct or available, the information will be stored into the database when there is no same data in the database, or updated when the original data has existed in the database.
5. Upon clicking on the delete button, the information will be deleted from the database if there is a same data item in the database.
6. Fault area and cause can be referred from the database, which are collected in subjective data acquisition.

5.3.7 Subjective parameter estimation (single component)

This program estimates the parameters of the single component delay time model based on the subjective data collected using the subjective data acquisition program. NAG routines are used in this program for numerical evaluation. The mathematical model has been introduced in chapter three. Please see the single component subjective parameter estimation program model section in Appendix 1. When the parameters have been estimated, the decision support report can be generated in this program. The reports include a downtime report and a cost report.

Program Flow Chart

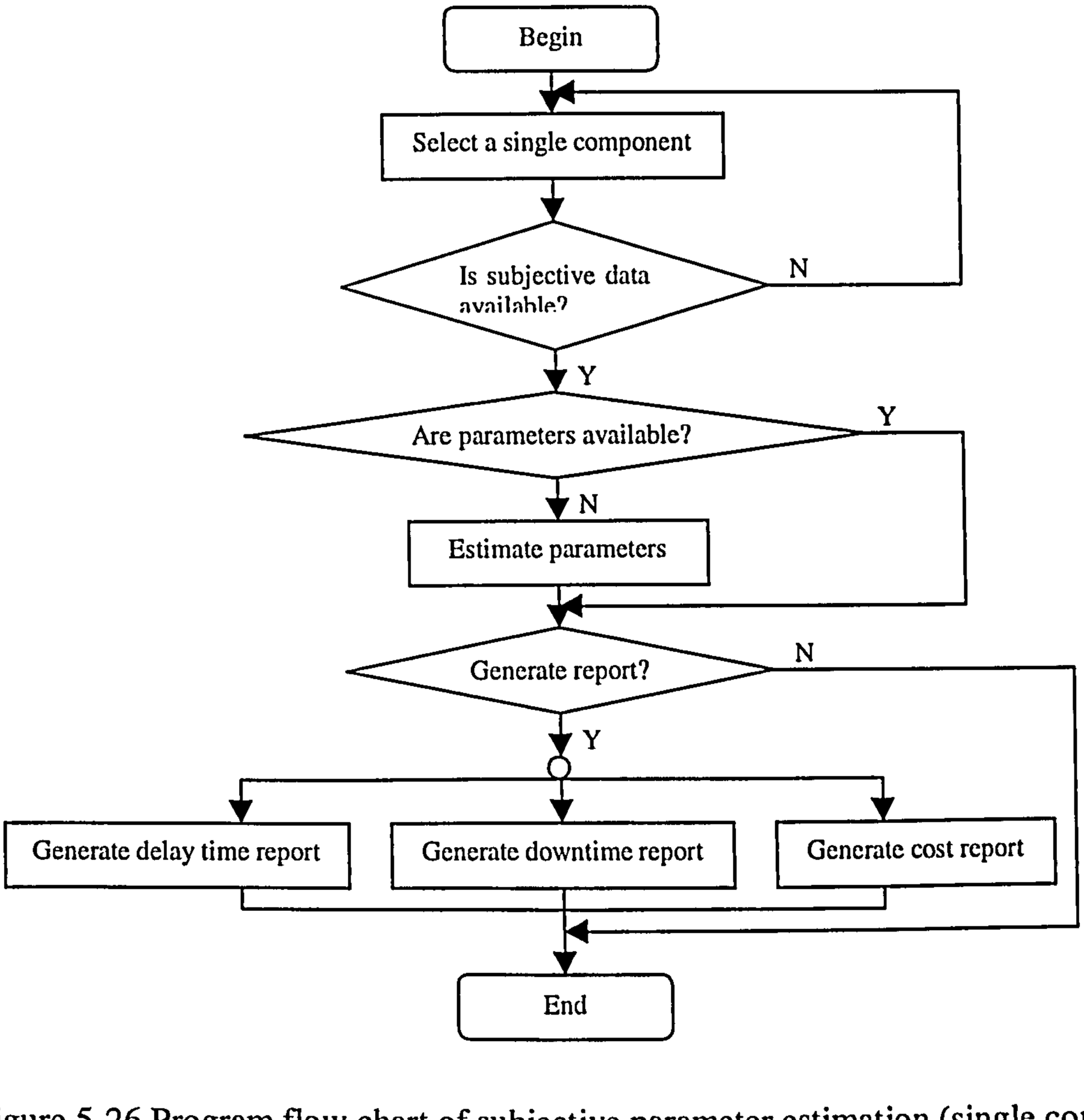


Figure 5-26 Program flow chart of subjective parameter estimation (single component)

Form Design

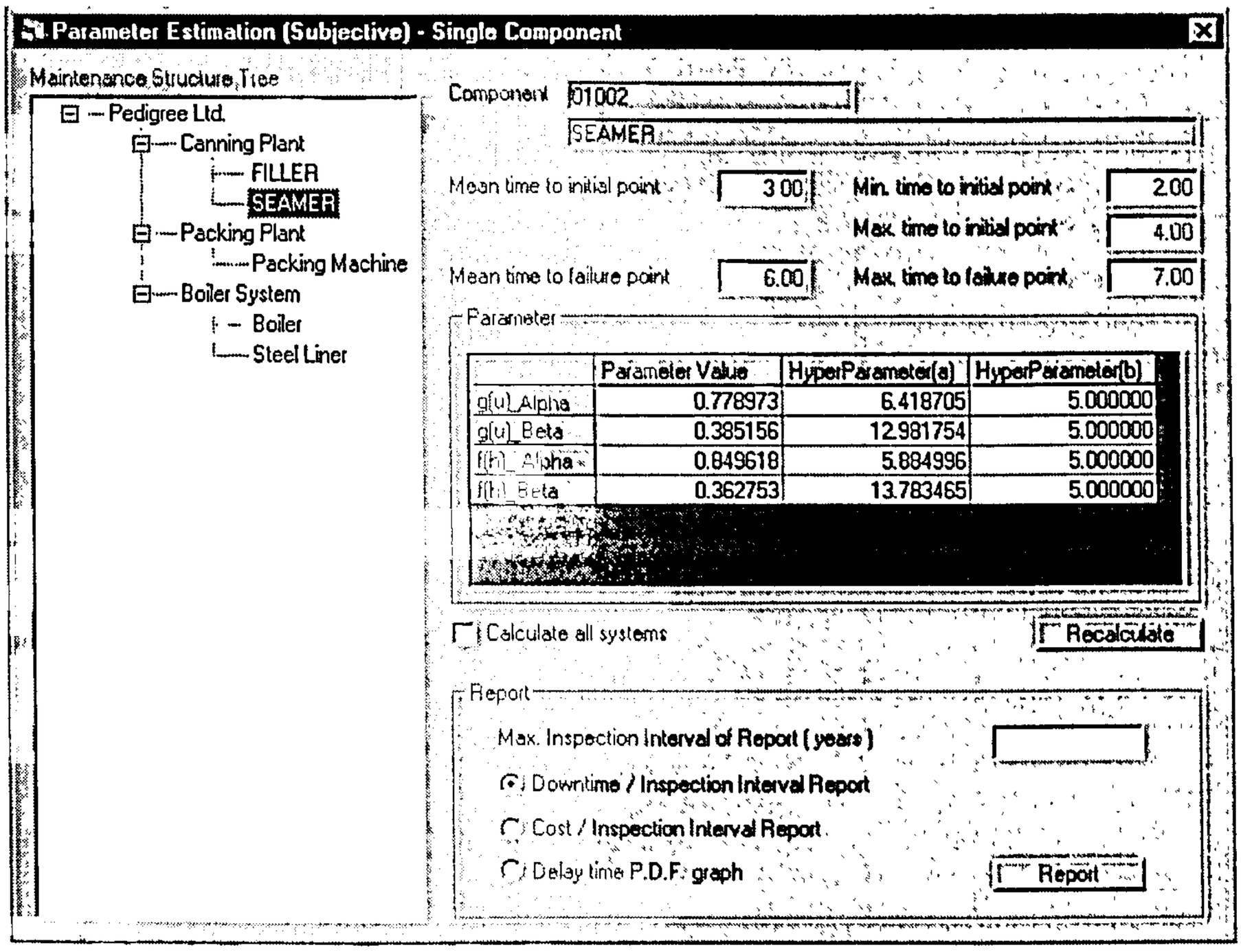


Figure 5-27 Form design of subjective parameter estimation (single component)

Form Statement

1. The left hand side frame is the plant structure display area. The chosen single component will be highlighted in blue.
2. When a single component is selected, the summarised value of subjective data will be displayed in the right hand side of the form, if subjective data of the component has been obtained from engineers.
3. If there are some values in the parameter table, it means that parameters have been estimated before. A user still can re-estimate the parameters by clicking the recalculate button.
4. When parameters are available, a user can generate a decision report by clicking on the report button. Before clicking on it, please input the max value of the x-axis in the yellow field.
5. If there are many single components which need to be processed, a user can check the 'calculate all system' box, then press the calculate button. Program

will calculate all single components automatically, if subjective data for them are available

5.3.8 Objective parameter estimation (single component)

When objective data is sufficient and in good quality, this program can be used to estimate the parameter from the set of objective data. Because the Bayesian approach was used in the parameter estimation model, when subjective data is also available, this program can estimate the parameters based on both subjective data and objective data. Please see the single component objective parameter estimation program model section in Appendix 1.

Program Flow Chart

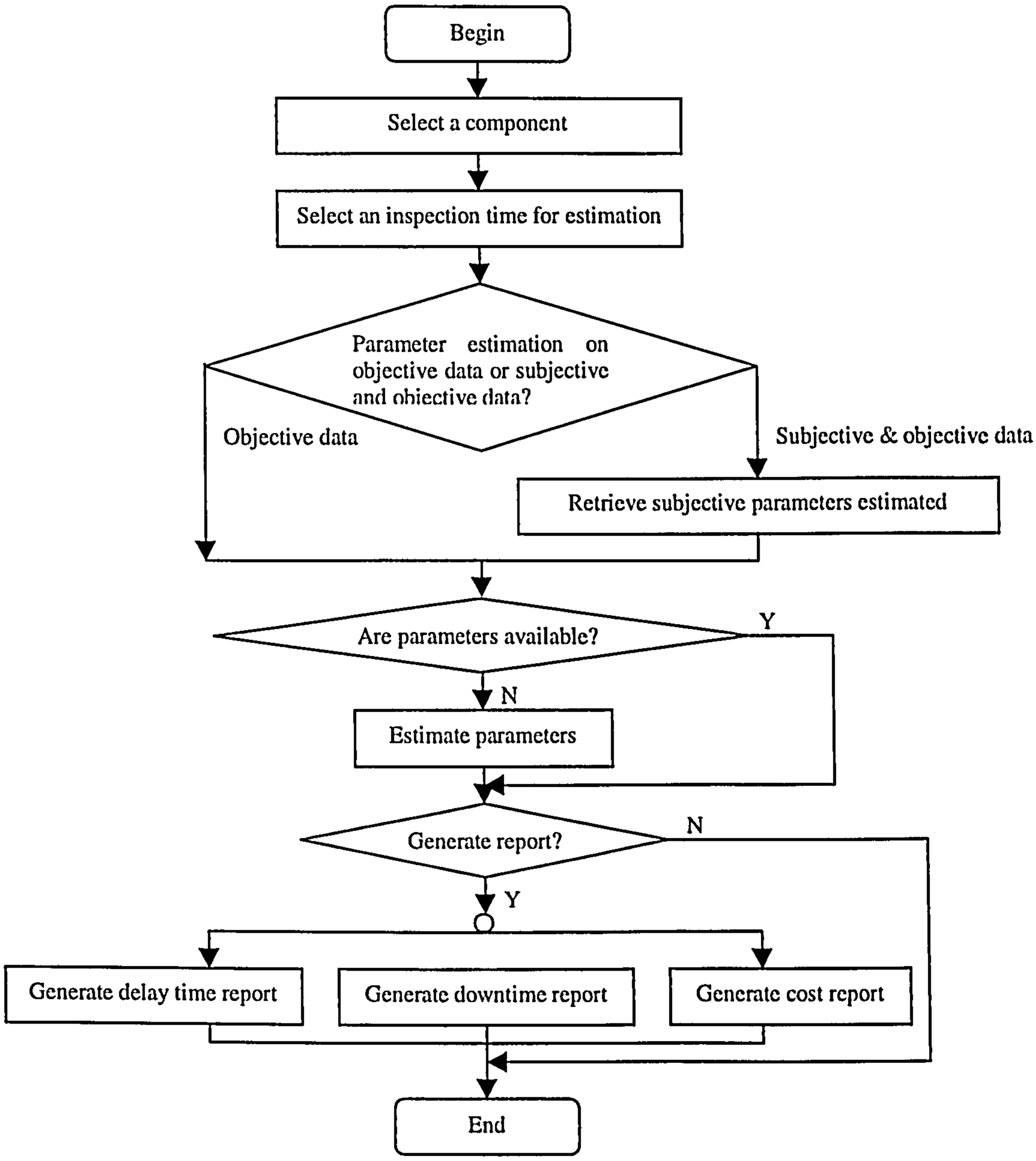


Figure 5-28 Program flow chart of objective parameter estimation (single component)

Form Design

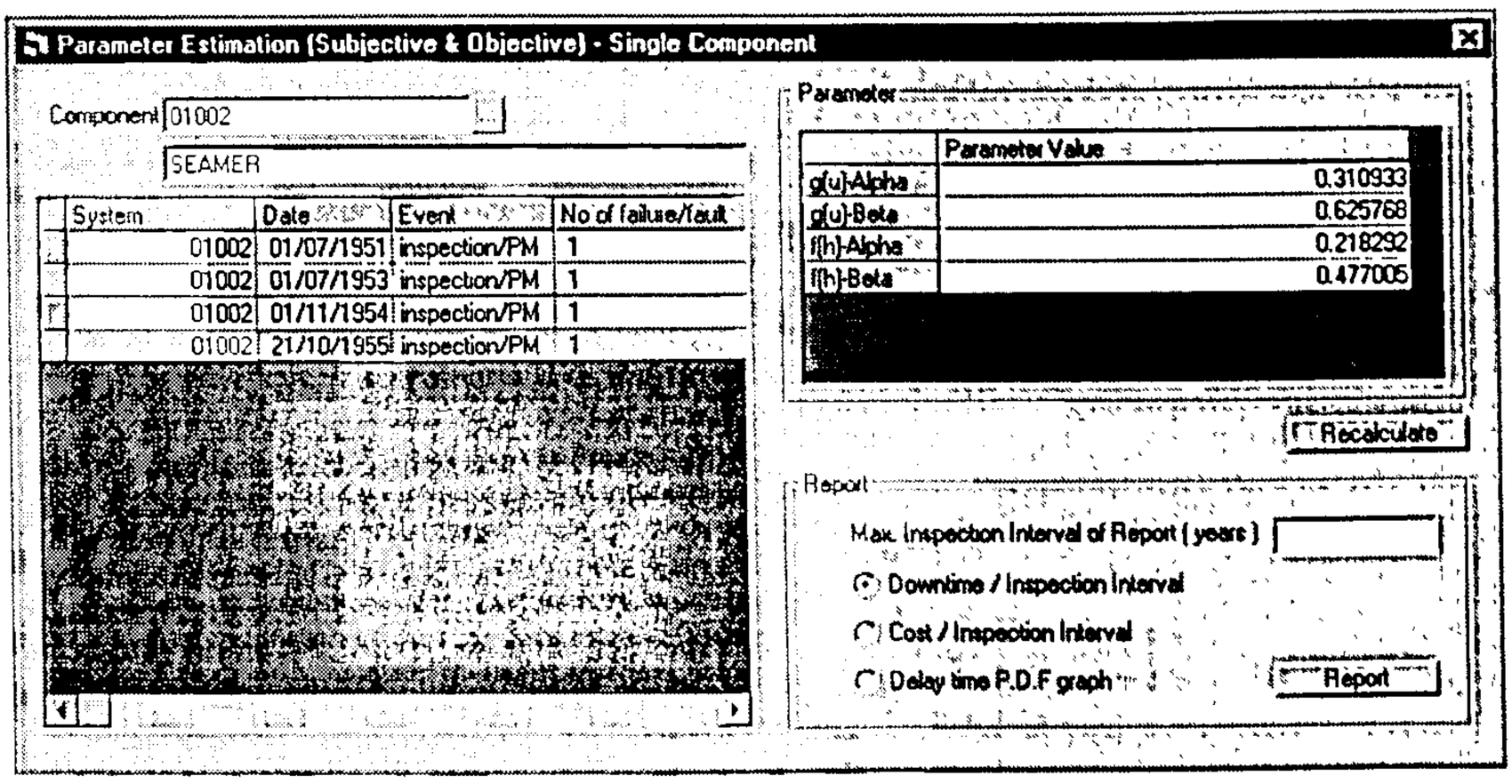


Figure 5-29 Form design of objective parameter estimation (single component)

Form Statement

1. The yellow field is mandatory input fields.
2. A user can input or select a single component information, then the maintenance record will be displayed in the left table.
3. When a user selects an inspection point, the row will be highlighted in yellow. If the parameters have been estimated before, those parameters will be displayed in the right table. The user can recalculate those parameters, if there are some changes in objective data or subjective data, by clicking the recalculate button.
4. When parameters are available, the user can generate a decision report by clicking on the report button.

5.3.9 Subjective and objective parameter estimation (complex component)

The programming method of these two programs is similar to that of the single component subjective parameter estimation program. The difference between these two programs is the mathematical model. The model has been introduced in chapter three. The program flow chart, form design and Form Statement of this program are the same to those of the single component subjective parameter estimation program, and will not

be repeated here. Please see the complex component subjective and objective parameter estimation program model section in Appendix 1.

5.4 THE TEST OF THE SOFTWARE USING A CASE STUDY

5.4.1 Introduction of the case

The original case study was carried out in a chemical fertiliser company in 1999. The company is one of the biggest chemical fertiliser companies in the north of China.

The data available is the maintenance log of No. 1 boiler, which has 20 years' objective data from 1976 to 1997. It includes details of inspection times, faults identified at each inspection and failures occurred between inspections.

5.4.2 Case study

Because the users of the plant are in China, it is not convenient to contact them, and therefore, the subjective data is not available at the time of study. We generated the required subjective data based on the objective data, which is used for testing the software.

Table 5-1 Subjective data of the boiler

For parameter estimation model	Mean number of faults per year	6
	Mean maximum number of faults per year	9
	Mean number of failures per year	1
	Mean maximum number of failures per year	4
For decision making model	Average downtime of inspection (hours)	50
	Average downtime per failure (hours)	75
	Average cost of inspection (£)	470
	Average cost of failure repair (£)	1000

Using above information, the estimated parameters are shown in table 5-2.

Table 5-2 Subjective parameters

λ (constant rate)	α (scale parameter of Weibull distribution)	β (shape parameter of Weibull distribution)
0.583333	0.008702	0.610735

We have considered in previous chapters the case where the umber of defects arising over a inspection interval follows a HPP with a constant rate λ per unit time, and the delay time h of a random defect is independent of its time origin and follows a Weibull distribution with shape parameter β and scale parameter α.

When the parameters have been estimated, the decision support report can be generated from the decision model. The decision support reports include a downtime report, a cost report and a delay time pdf. report.

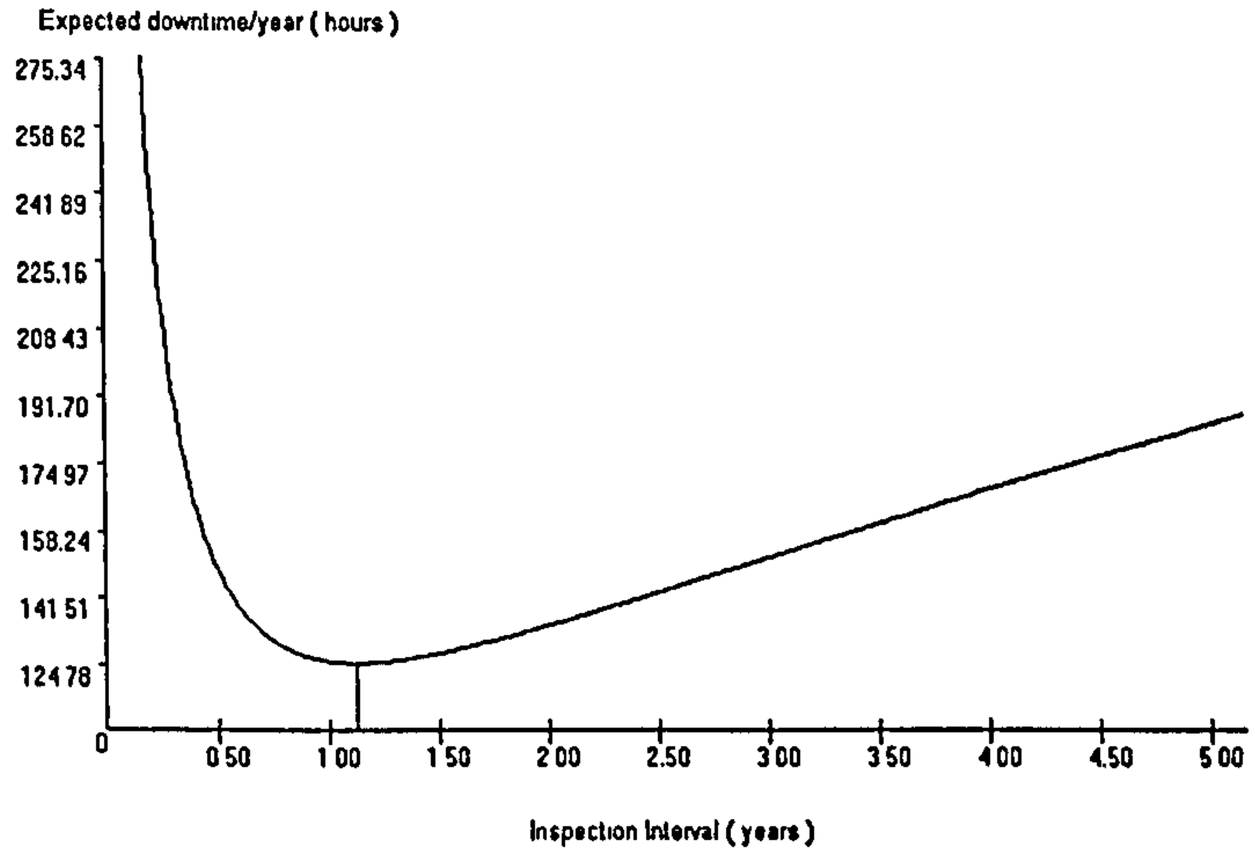


Figure 5-30 Downtime report based on subjective parameters

In the downtime report the optimum inspection interval is 1.13 years and the lowest downtime is 124.78 hours/year. The practical option is 1 year.

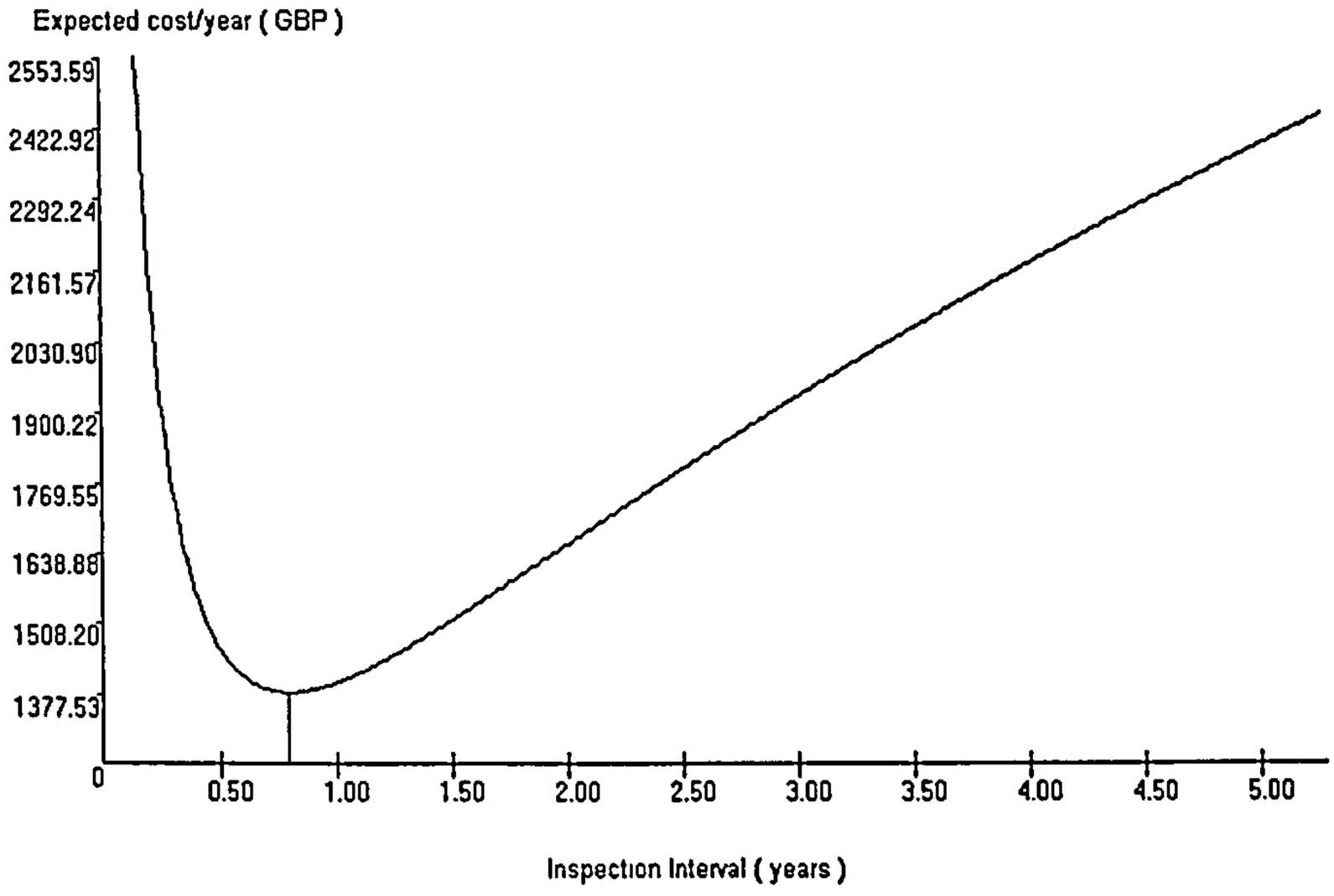


Figure 5-31 Cost report based on subjective parameters

In the cost report the optimum inspection interval is 0.8 year and the lowest cost is 1137.53 hours/year.

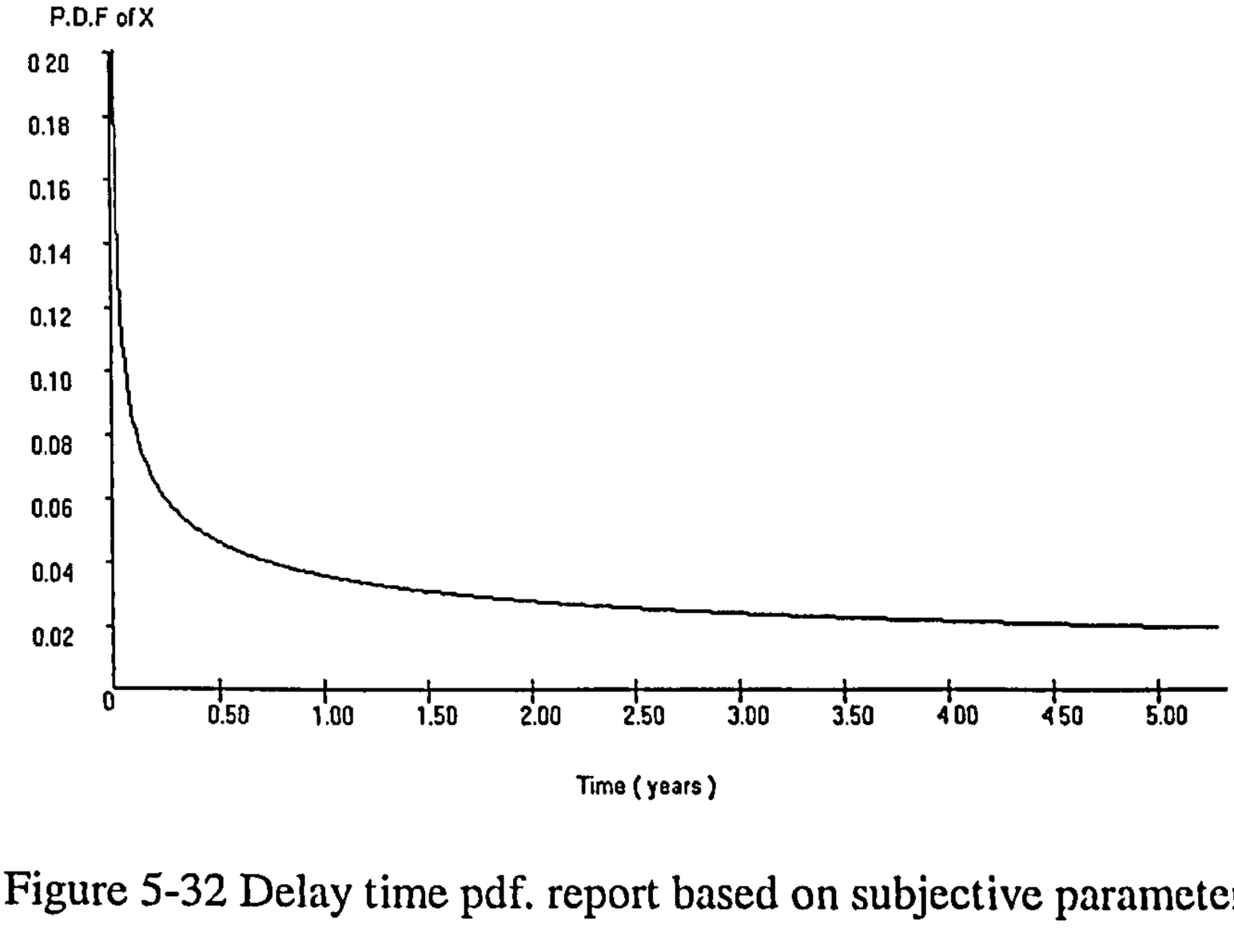


Figure 5-32 Delay time pdf. report based on subjective parameters

Table 5-3 lists a sample of objective data of the boiler.

Table 5-3 Objective data of the boiler

Date	No. of fault/failure	Event	Description
2/1/77	1	failure	Major maintenance
3/2/77	4	Inspection/PM	Deformation of liner
6/1/79	1	failure	Emergency repair
9/1/79	1	failure	Unknown
12/1/79	1	failure	Liner damage
12/18/79	5	Inspection/PM	Liners space is too big
2/1/80	1	failure	Emergency repair
5/1/80	4	Inspection/PM	Leaking
8/1/80	1	failure	Leaking
8/1/81	2	Inspection/PM	Many deformation points
8/1/82	3	Inspection/PM	Fracture of bolts
5/1/83	2	Inspection/PM	deformation
6/1/83	1	failure	6 pipes burst
4/1/84	1	Inspection/PM	Liner deformation
8/1/85	1	failure	Counter circulation
8/1/86	3	Inspection/PM	Fracture of pipe board
10/1/86	1	failure	A lot of damages
7/1/87	5	Inspection/PM	deformation
8/1/88	4	Inspection/PM	Fracture of locating board
8/1/89	2	Inspection/PM	Liner bottom deformation
2/1/90	1	failure	Fastening
4/1/90	1	failure	Fastening
8/1/90	3	Inspection/PM	Pipe core
7/1/91	2	Inspection/PM	Pipe core
10/1/91	1	failure	Injecting water
7/1/92	7	Inspection/PM	Distributor failed
7/1/93	3	Inspection/PM	Liner bottom
8/1/93	1	failure	Pipe burst
10/1/93	1	failure	2 pipes burst
4/1/94	3	Inspection/PM	Deformation
8/1/95	3	Inspection/PM	Liner deformation
4/1/97	1	failure	Pipe burst
5/1/97	6	Inspection/PM	Concrete structure

Using above objective data, the estimated parameters are shown in table 5-4.

Table 5-4 Objective parameters

λ (constant rate)	α (scale parameter of Weibull distribution)	β (shape parameter of Weibull distribution)
0.298107	0.0061197	0.458478

Using these three parameter in our decision model of the complex system, decision support report can be obtained, see Figure 5-33 to 5-35.

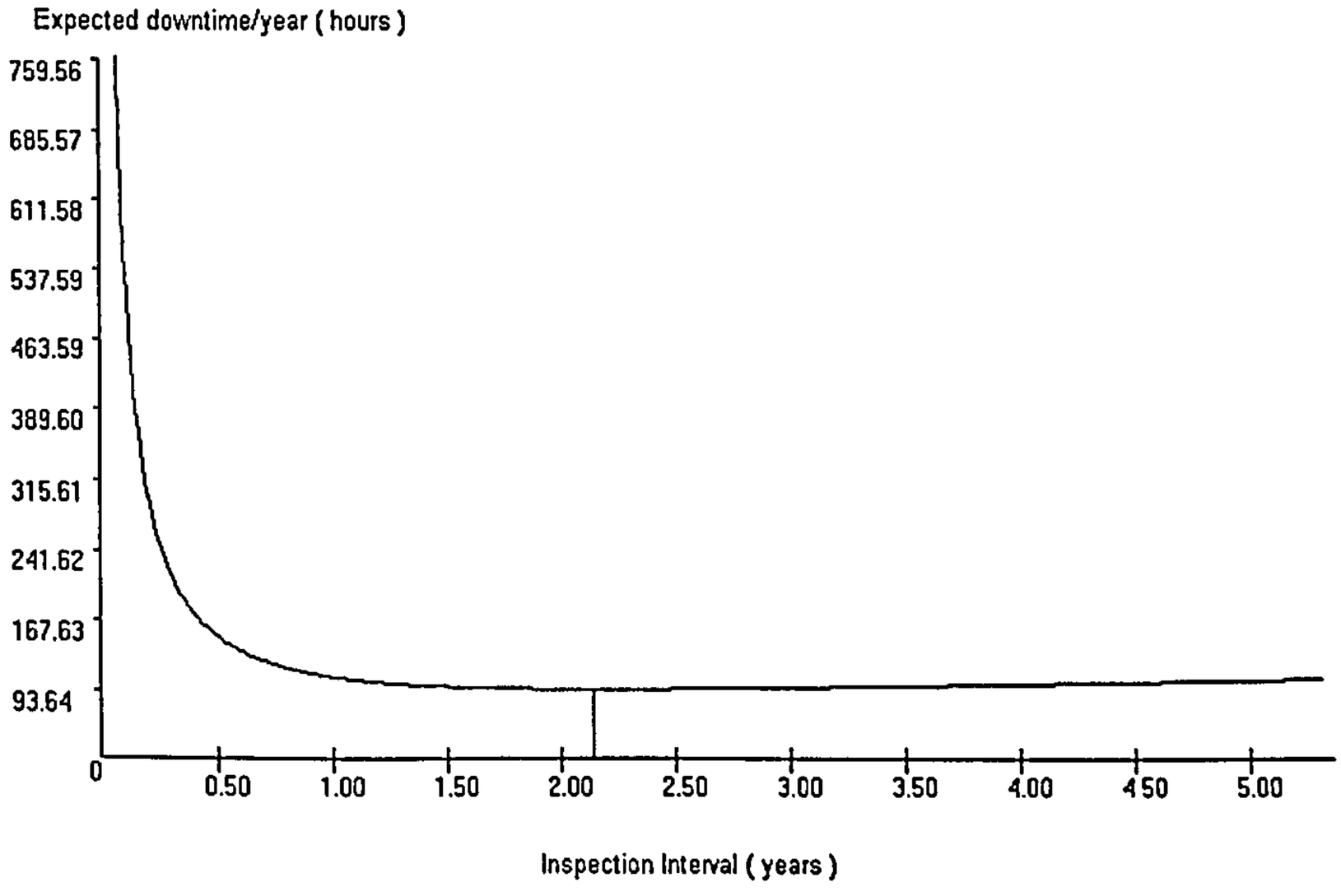


Figure 5-33 Downtime report based on objective parameters

In the downtime report the optimum inspection interval is 2.15 years and the lowest downtime is 93.64hours/year.

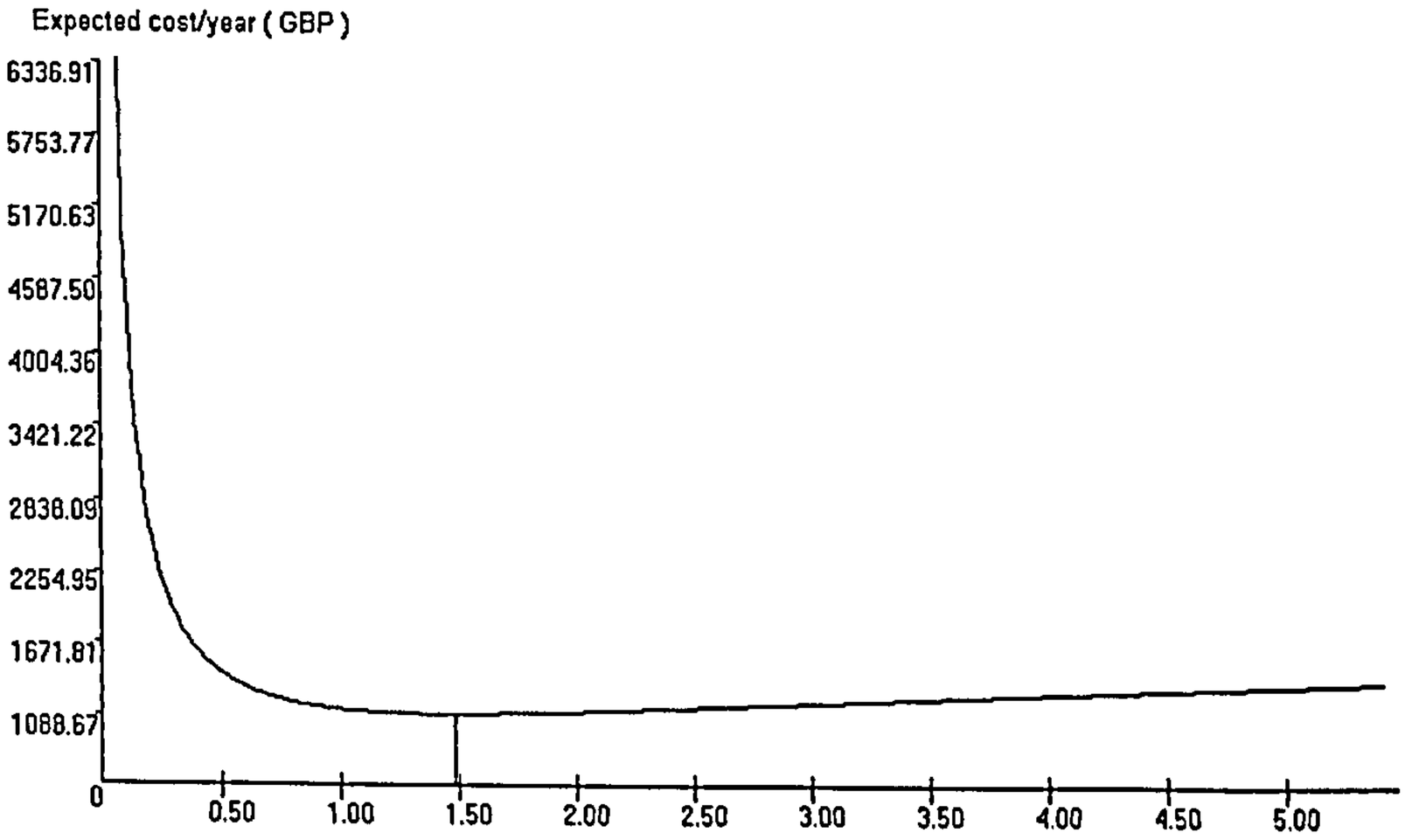


Figure 5-34 Cost report based on objective parameters

In the cost report the optimum inspection interval is 1.49 years and the lowest cost is 1088.67 GBP/year.

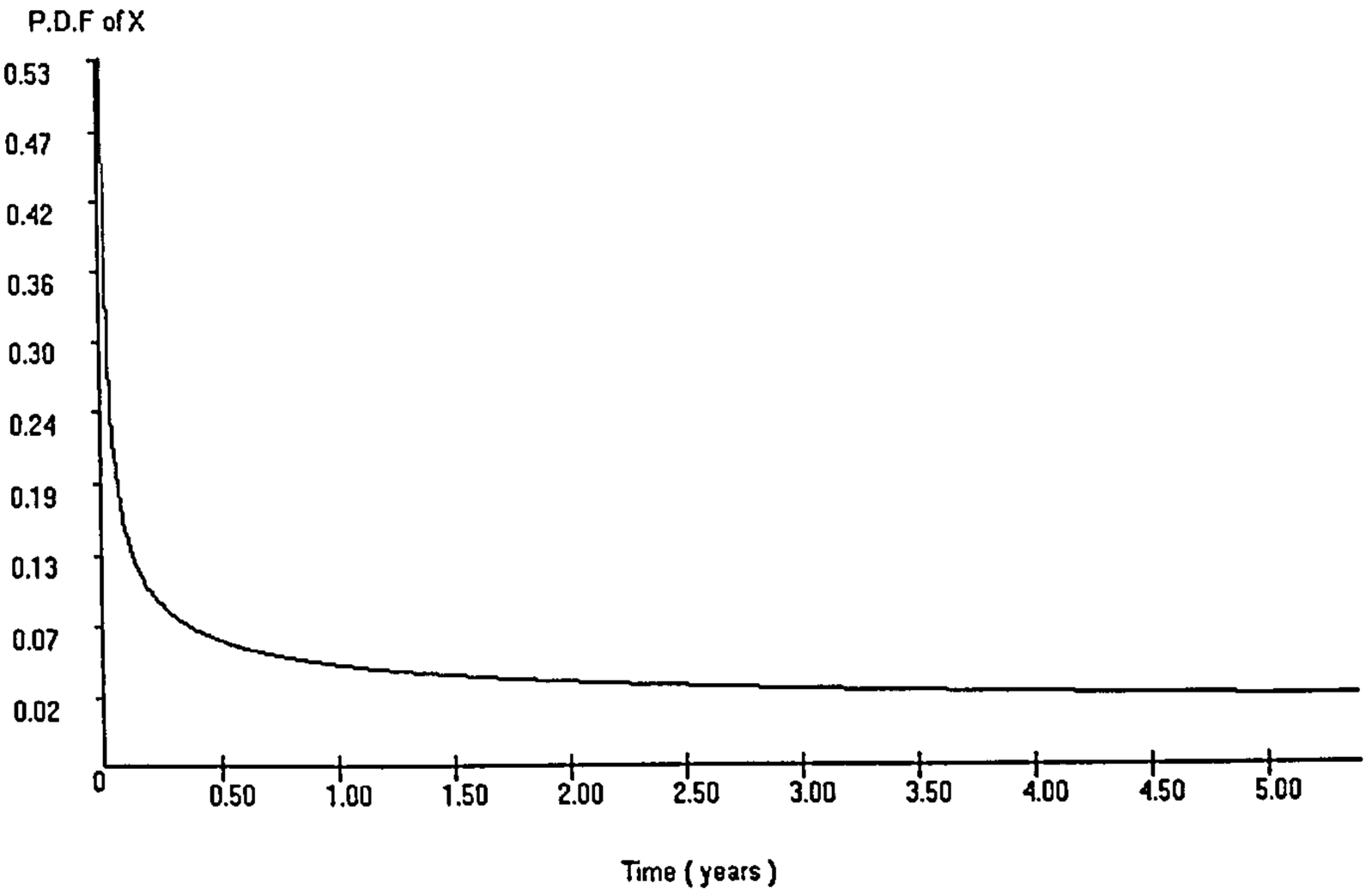


Figure 5-35 Delay time pdf. report on objective parameters

In the reports based on subjective data, we see that the optimal inspection interval is 1.13 years or 0.8 year depending on different criterion functions. After updating with the objective data, the optimum inspection interval becomes 2.15 years or 1.49 year, and the downtime is reduced from 124.78 hours/year to 93.64hours/year.

It is noted that the difference between the recommendations by using different criterion functions is caused by the difference between the downtime and cost ratios of inspection and failure. The ratio between the downtime of inspection and failure is higher than that between the cost of inspection and failure, which resulted in a shorter recommended inspection interval if using the downtime measure.

5.5 CONCLUSION

The delay time model has been proved to be a valid methodology for modelling inspection decision making. For maintenance decision-maker, the mathematics model is

too complex to be understood, so it is not easy to be applied by them. The work reported in this dissertation attempts to deliver a package to facilitate maintenance decision making.

All mathematics models used in this software have been introduced in chapters three and four. Some of the models were specially designed to make programming work easy. In the subjective data collection model, the information required were obtained by a set of questions, which are designed for easy use. The subjective data should be used in the parameter estimation model and the decision making model.

In the system analysis phase, the mathematical models are transformed into program models. The program models are not the final programs that need to be developed. They show how many main parts should be developed in the software. The input and output information will connect all program models together as a whole system.

Using the input and output information obtained from the system analysis phase, the table structure was designed in the database design phase. Not only was the database structure was obtained in this phase, the program structure also has an outline after data flow is available.

In the program design phase, all programs were analysed in details. The program flow chart gives the main function of the programming structure. The form design gives the graphical user interface of the program. The form statement tells the detailed specification in the programming phase.

After the software was completed, it has been tested using a set of real maintenance data and generated a set of decision support reports. The software can be used for decision support in plant maintenance management.

Chapter 6 A demo software development based on imperfect inspection case of the delay time model

As introduced earlier, the software was developed on perfect inspection case of the delay time model. When I finished the development of the software, It was suggested to do some research on software development of imperfect inspection case of the delay time model. In this chapter, a demo software will be introduced. It is still based on the delay time model, but focuses on imperfect inspection case.

The demo software is a complete application of the imperfect inspection case of the delay time model on a complex system. It is designed to implement the features of subjective parameter estimation and decision making in the imperfect inspection case. It will be served as a demo model only, and has not been designed using objective data.

6.1 DELAY TIME MODEL

In the model, only the subjective data is used for parameter estimation, so the Bayesian approach is not used here. When the parameters are estimated, the decision report could be made based on them.

Christer & Wang (1995) developed delay time model of a complex system in imperfect inspection case. In the imperfect inspection delay time model, there is one more parameter than that of the perfect case, which is the probability that the faults can be identified at an inspection. The delay time is assumed to follow a two-parameter Weibull distribution, so there are, in total, four parameters that need to be estimated in the model.

Assumptions and notation

- The plant analysed here is a complex system with imperfect inspection. At each inspection faults can be identified with a probability $r \leq 1$
- Defects arise according to a homogeneous Poisson process with rate λ
- Defects are assumed to arise independently of each other
- All identified defects are rectified by repairs or replacement during the inspection period
- The inspection and repairs do not influence the development of undetected defects in the system
- The delay time h of a random defect is independent of its time origin and follows Weibull distribution with scale parameter α and shape parameter β. See expression (3-9)
- $EN_f(T)$ denotes the expected number of failures in (0, T)
- $EN_p(T)$ denotes the expected number of faults identified at T
- $E(h)$ denotes the mean delay time
- N_{afault} is the subjective estimate of the mean number of faults identified at an inspection.
- N_{afail} is the subjective estimate of the mean number of failures identified in an inspection interval (0, T).
- T_d is the subjective estimate of the mean delay time

The expected failures in (0, T), is given, Christer (1999),

$$EN_f(T) = \lambda \sum_{n=1}^{\infty} r(1-r)^{n-1} \int_0^T F(nT-h)dh \qquad (6\text{-}1)$$

, the expected number of defects identified at T is

$$EN_p(T) = \lambda \sum_{n=1}^{\infty} r(1-r)^{n-1} \int_0^T [1-F(nT-h)]dh \qquad (6\text{-}2)$$

, and the mean delay time is ,

$$E(h) = \int_0^\infty hf(h)dh \tag{6-3}$$

If subjective data, N_{afault}, N_{afail} and T_d are available, we have

$$EN_f(T) = \lambda \sum_{n=1}^{\infty} r(1-r)^{n-1} \int_0^T F(nT-h)dh \approx N_{afault} \tag{6-4}$$

$$EN_p(T) = \lambda \sum_{n=1}^{\infty} r(1-r)^{n-1} \int_0^T [1-F(nT-h)]dh \approx N_{afault} + N_{afail} \tag{6-5}$$

$$T_d \approx E(h) = \int_0^\infty hf(h)dh \tag{6-6}$$

Now there are three equations with three unknown parameters λ, α and β, given r is known, so the parameters can be obtained by solving the equations (6-4), (6-5) and (6-6). r is normally obtained directly from the experts.

When all parameters have been estimated, we proceed to the decision model. The decision making model has been introduced in chapter four. It includes cost model and downtime model.

$$C(T) = \left\{ \frac{C_I + C_f EN_f(T)}{T + D_I} \right\} \tag{6-7}$$

and

$$D(T) = \left\{ \frac{D_I + D_f EN_f(T)}{T + D_I} \right\} \tag{6-8}$$

Where

- $C(T)$ is the expected cost per time unit
- $D(T)$ is the expected downtime per time unit
- C_I: denotes the mean cost per inspection, including the cost of inspection repairs.
- C_f: denotes the mean cost of a failure.

- D_I: denotes the mean inspection time, including the duration of inspection repires.
- D_f: denotes the mean down time of a failure.

6.2 DEMO SOFTWARE DEVELOPMENT

The demo software will be developed on the model introduced earlier. The first step is to analyze the delay time model and to transform it into program models. In the mathematical model, there are two parts, one is data collection, and the other is parameter estimation. In the decision making model, there are two parts, one is the cost model, and the other is the downtime model. In the software, two programs will be developed, one is for the cost model, and the other is for the downtime model. Data collection and parameter estimation will be combined in all programs.

Because the data collection process will be run each time the program is run, no database will be needed in the demo software.

6.2.1 Program analysis

Because the principle of programming the cost model and the downtime model is quite similar, the program analysis of these two programs will be introduced together.

Program Flow Chart

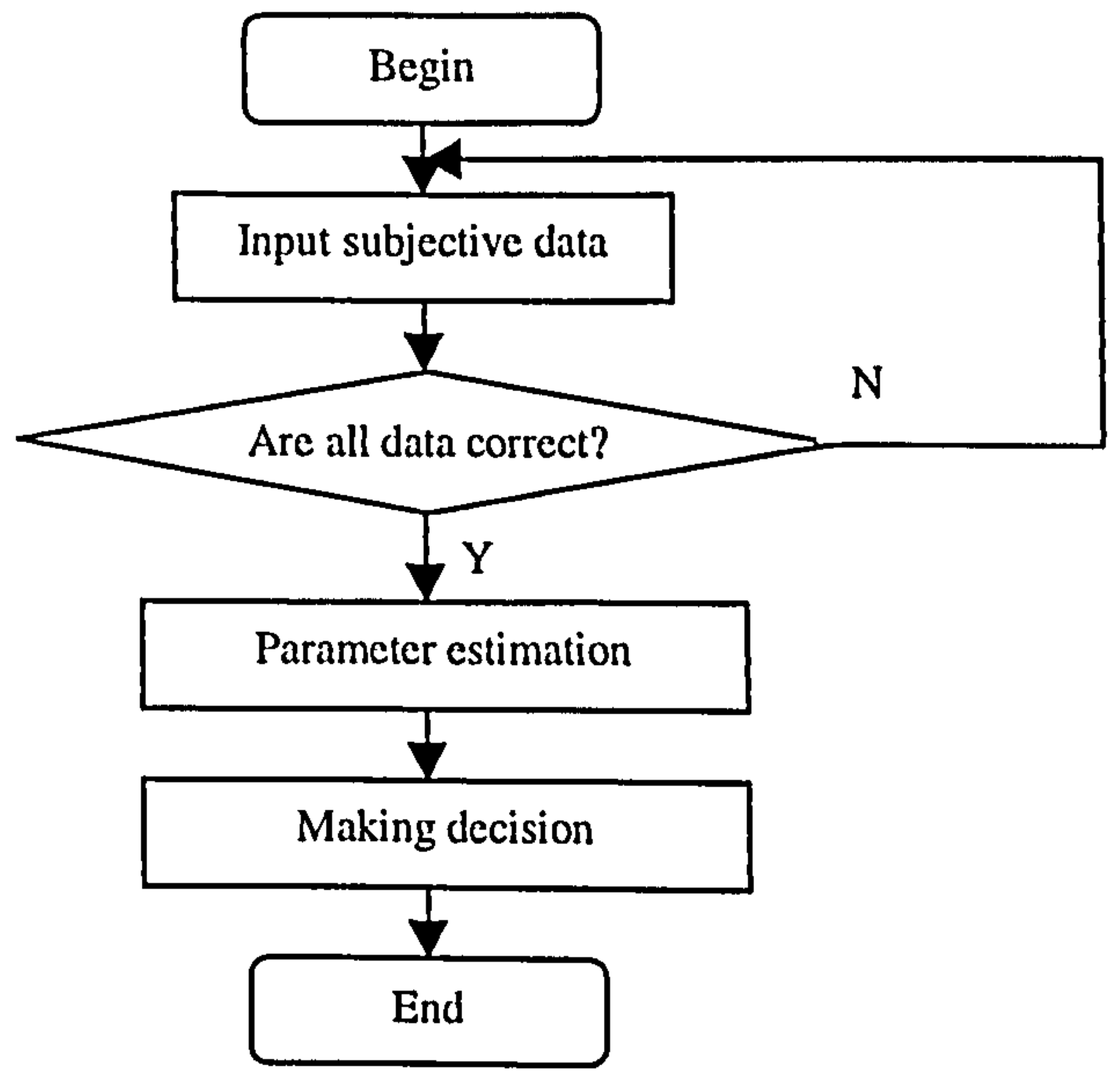


Figure 6-1 Program flow chart of decision making programs

Form design

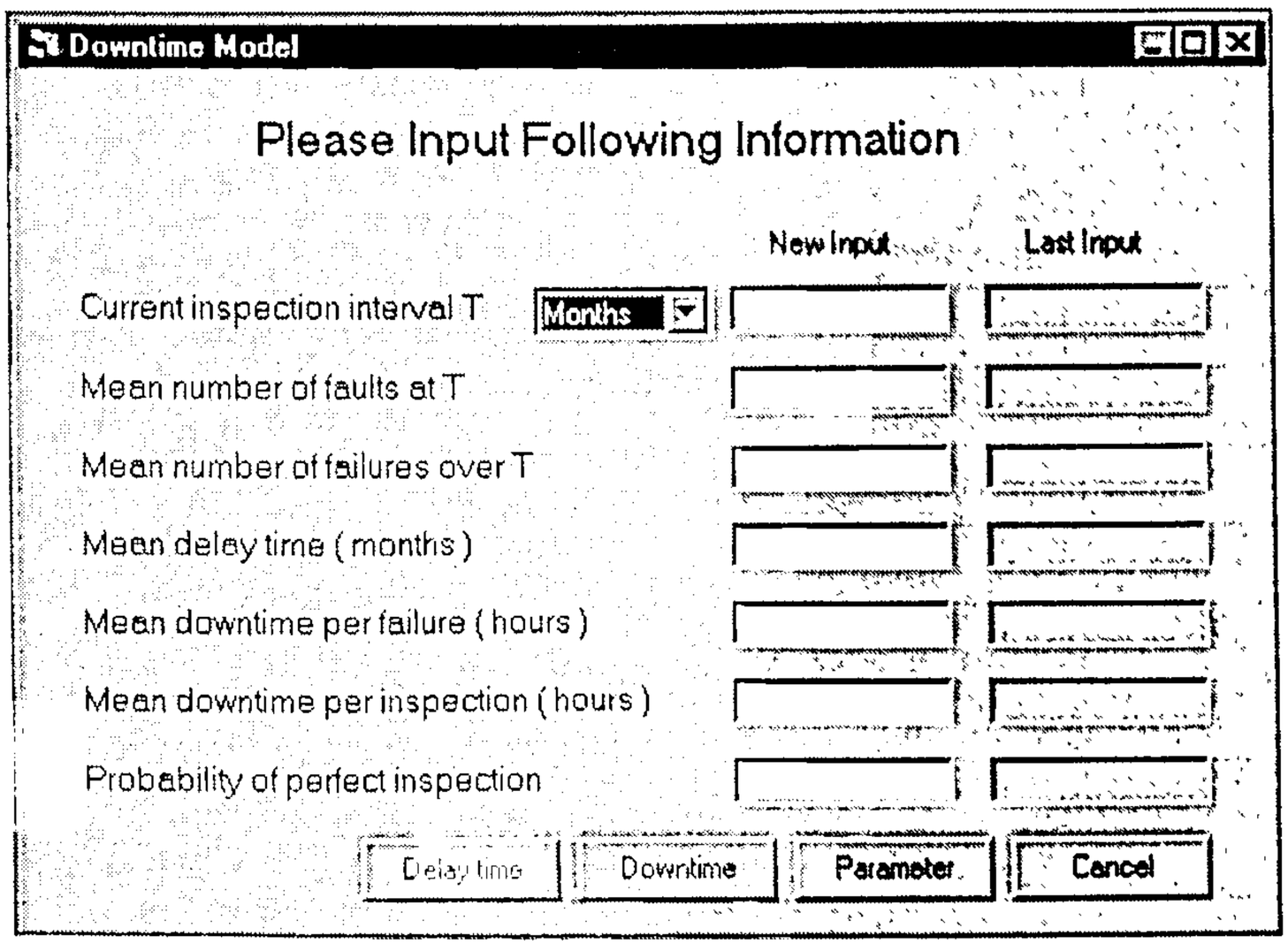


Figure 6-2 Form design of downtime model

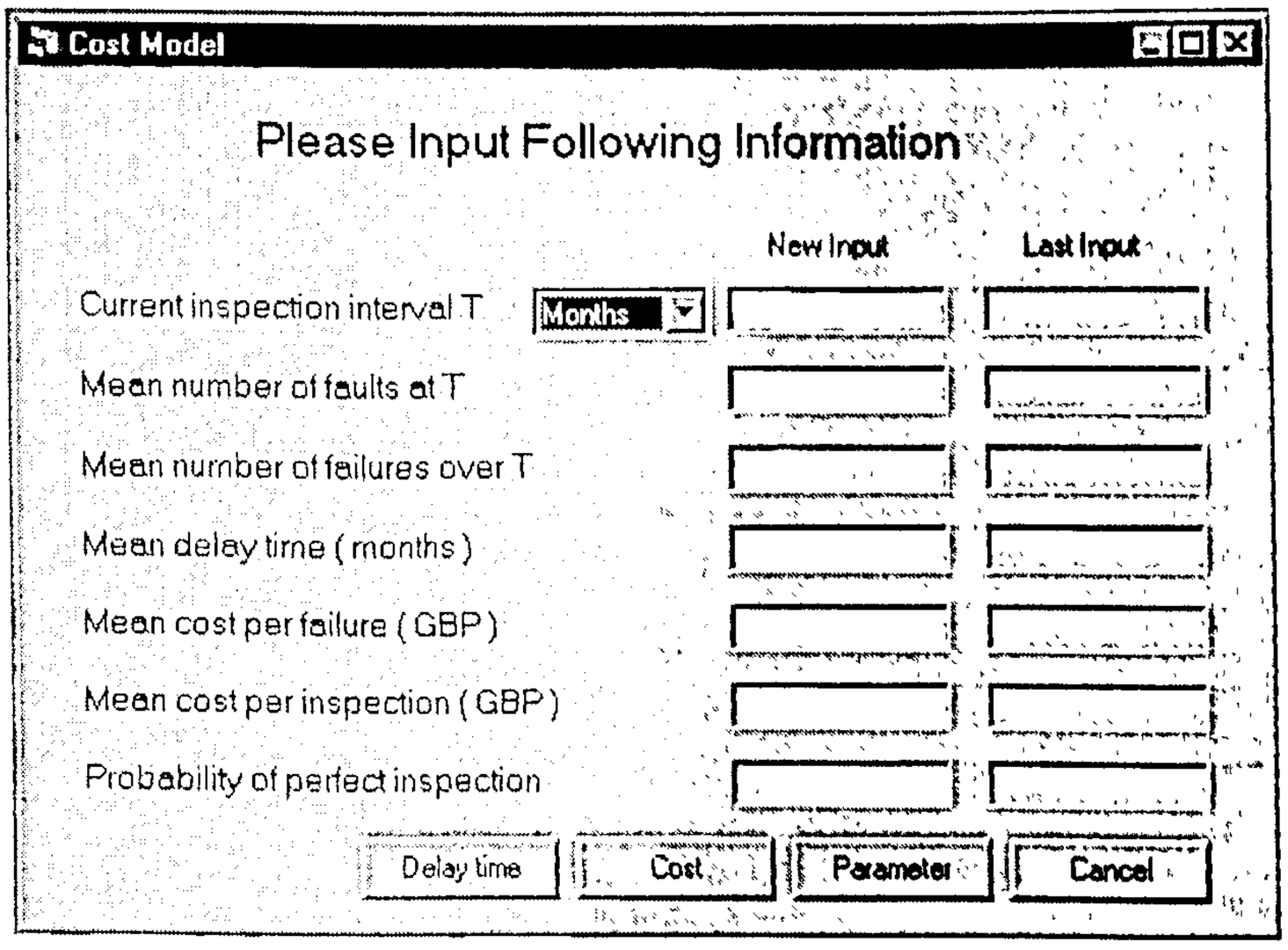


Figure 6-3 Form design of cost model

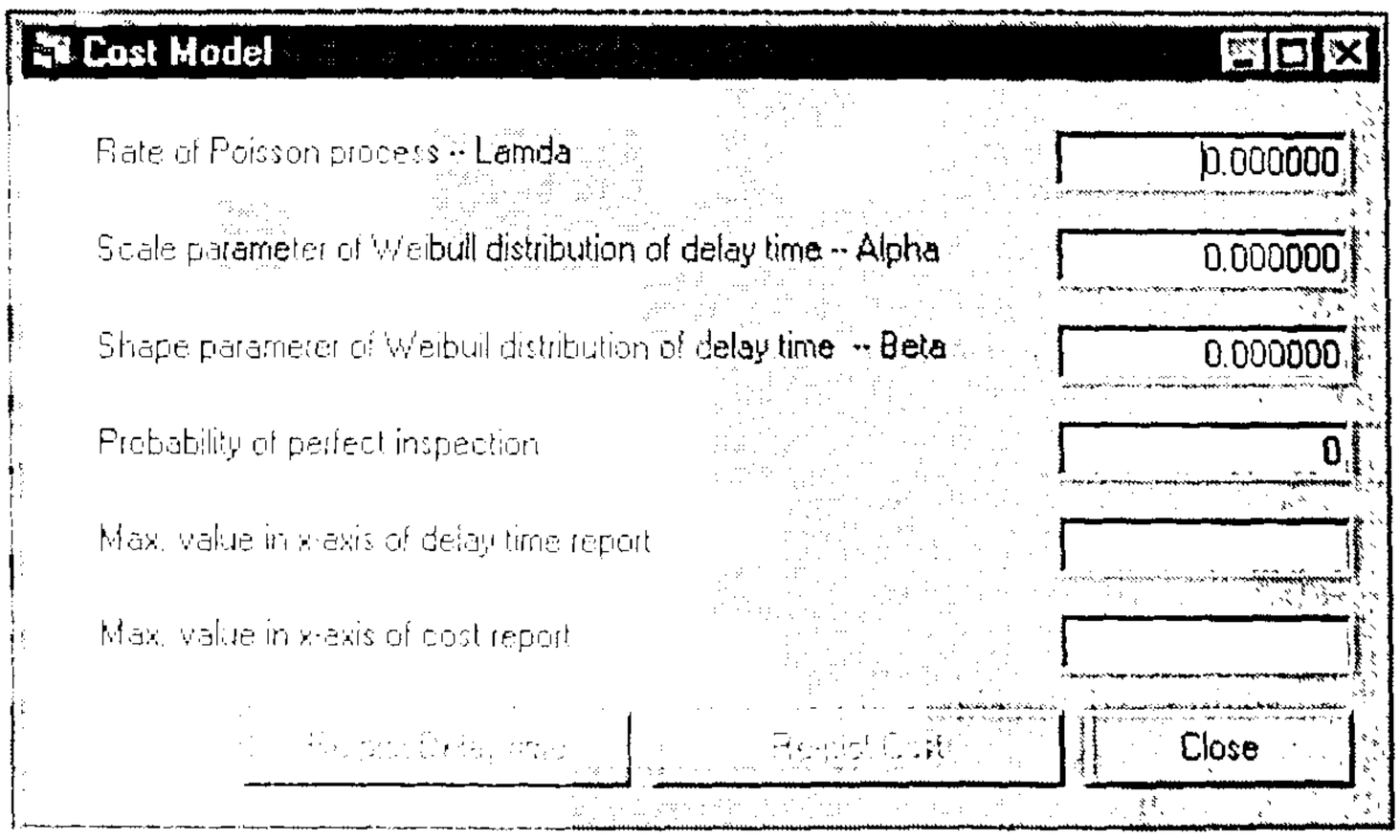


Figure 6-4 Form design of parameter program

Form Statement

1. Yellow fields are mandatory input fields.
2. The time unit of inspection interval can be selected by the user. It includes day, week, month and year. And the time unit of the mean delay time will be set as the same unit as that of the inspection interval.
3. Upon clicking on the downtime or cost button, the program will estimate parameters first, and then it will generate the decision report for the user.
4. Upon clicking on the delay time button, the program will generate the pdf report of the delay time.
5. Upon clicking on the parameter button, the parameter form will be displayed. The user can change the parameters and re-plot the reports.
6. Upon clicking on the cancel button, the program terminates.
7. The grey areas are the previous values of subjective data input.
8. If the output report is not well presented, because the unit of the x-axis is too large or small. The user can input a suitable number for the x-axis, and re-plot the report again.

6.3 CONCLUSION

The demo software will be used for demonstration only so that only a quick subjective data estimation procedure is programmed. In previous chapters, we introduced the Bayesian approach for parameter estimation, because it can estimate parameters based on subjective data first, and then update the estimates when objective data becomes available. In this model, we only use subjective data for parameter estimation, so the Bayesian approach is not used.

It will serve as an introduction to the delay time model. It provides a quick demo from data collection to report output, and hides the delay time model into a black box.

Chapter 7 Conclusion

The delay time model has been developed for more than twenty years. It is a model within the maintenance engineering context. More importantly, it can be used to build quantitative models of the inspection practice of plant, both existing and modified, which have proved in practice to be valid. Techniques exist to estimate delay time parameters given objective data, or subjective data, or both. The objective and subjective parameter estimation techniques were jointly tested, and gave consistent results.

Although the delay time model have been well developed, only academic staffs, who ever consider maintenance as an area of study, are interested in the development of it. The applications were limited in cases where experienced delay time modellers have to be involved. With the recent development and the take-up of IT, it is possible to develop a semi-automated delay time modelling and demonstration tool to be widely accessible by industry without the heavy involvement of modellers.

This thesis presents a software development based on the delay time model. In Chapter 3 and 4, the delay time model used in the software is introduced. The model has been modified to make it possible to be developed in this software. In the failure analysis phase, the subjective data was collected in a simple way, which is designed to obtain the data in a meeting with the experts. It uses a questionnaire with useful information implicitly embedded in the questions, which can be answered relatively easily. Because the model shall estimate the parameters given objective data, subjective data, or both, a Bayesian approach was used to meet the requirement. It can handle cases where subjective data may be required in the first place, and then update the estimate when objective data becomes available. If objective data is sufficient and in good quality, the approach will be the same as the conventional maximum likelihood method.

In Chapter 5, the software development is introduced in three main phases. The first step is the system analysis. In this phase, all mathematical models were transformed into program models, which were connected together by the input and output data. In the database design phase, the input data and output data were stored in the database designed. All program models can be obtained from the data flow of the program. In the program design phase, the system was divided into many different programs, which were designed for easing programming.

The software was developed on the perfect inspection delay time model. And another development of imperfect inspection delay time model was introduced in chapter six, and a demo software was developed accordingly. Those softwares were developed to be a semi-automated delay time modelling and a demonstration tool and to be used for decision support in plant maintenance management without the heavy involvement of modellers.

Appendix 1 Main source code of the delay time models

```
'Subjective delay time model of complex system
Option Explicit

Public GDB_MLND As Double      'Subjective data of Most Likely Number of Defaults
Public GDB_MND As Double       'Subjective data of Maximum Number of Defaults
Public GDB_MLNF As Double      'Subjective data of Most Likely Number of Failure
Public GDB_MNF As Double       'Subjective data of Maximum Number of Failure
Public GDB_MLH As Double       'The Possibility Number of Maximum Event be set as 0.001
Public GDB_INTERVAL As Double  'Inspection Interval
Public GDB_Sbj_Limda1 As Double      'For Limda_1
Public GDB_Sbj_Limda2 As Double      'For Limda_2
Public GDB_Sbj_Arfa1 As Double       'For Arfa_1
Public GDB_Sbj_Arfa2 As Double       'For Arfa_2
Public GDB_Sbj_Beta1 As Double       'For Beta_1
Public GDB_Sbj_Beta2 As Double       'For Beta_2
Public GDB_Sbj_Arfa As Double        'For Arfa Parameter
Public GDB_Sbj_Beta As Double        'For Beta parameter
Public GDB_Sbj_Limda As Double       'For Limda parameter
Public GDB_Sbj_Temp_Arfa1 As Double
Public GDB_Sbj_Temp_Beta1 As Double
Dim GDB_Temp_Least_Value As Double
Dim GDB_GAMA_Arfa As Double
Dim GDB_GAMA_Beta As Double
Dim GDB_Sbj_Arfa_UpBound As Double
Dim GDB_Sbj_Beta_UpBound As Double
Dim GDB_Sbj_Arfa_LowBound As Double
Dim GDB_Sbj_Beta_LowBound As Double

#If Win32 Then
   Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (_
   ByRef hpvDest As Any, ByVal hpvSource As Any, ByVal cbCopy As Long)
#Else
   Declare Sub CopyMemory Lib "KERNEL" Alias "hmemcpy" (_
   ByRef hpvDest As Any, ByVal hpvSource As Any, ByVal cbCopy As Long)
#End If



Declare Sub D01AJF Lib "NAGd01.DLL" (ByVal F As Any, A As Double, B As Double, _
      EPSABS As Double, EPSREL As Double, Result As Double, ABSERR As Double, W As Double, LW
As Long, IW As Long, _
      LIW As Long, LFAIL As Long)

Declare Function S14AAF Lib "NAGSX.DLL" (X As Double, IFail As Long) As Double

Declare Sub C05NBF Lib "NAGAC.DLL" (ByVal FCN As Any, N As Long, X As Double, _
      FVEC As Double, XTOL As Double, WA As Double, LWA As Long, IFail As Long)

Declare Sub D01AMF Lib "NAGd01.DLL" (ByVal F As Any, BOUND As Double, INF As Long, _
      EPSABS As Double, EPSREL As Double, Result As Double, ABSERR As Double, W As Double, LW
As Long, IW As Long, _
      LIW As Long, LFAIL As Long)
```

```
Declare Sub D01FCF Lib "NAGd01.DLL" (NDIM As Long, A As Double, B As Double, _
    MINPTS As Long, MAXPTS As Long, ByVal FUNCTN As Any, EPS As Double, ACC As Double, LENWRK As Long, WRKSTR As Double, _
    FINVAL As Double, IFail As Long)

Declare Sub E04JYF Lib "C:\Amytree\tree\Programs\DTM\dll\NAGE04.DLL" (N As Long, IBOUND As Long, ByVal FUNCT1 As Any, _
    BL As Double, BU As Double, X As Double, F As Double, IW As Long, LIW As Long, W As Double, LW As Long, IUSER As Long, _
    USER As Double, IFail As Long)

Declare Sub E04ABF Lib "C:\Amytree\tree\Programs\DTM\dll\NAGE04.DLL" (ByVal FUNCT As Any, E1 As Double, E2 As Double, A As Double, B As Double, MAXCAL As Long, X As Double, F As Double, IFail As Long)

Declare Function X02AJF Lib "NAGSX.DLL" () As Double

'Declare Sub E04KJF Lib "NAGE04.DLL" (N As Long, IBOUND As Long, ByVal FUNCT As Any, BL As Double, _
    BU As Double, X As Double, F As Double, G As Double, IW As Long, LIW As Long, W As Double, _
    LW As Long, IUSER As Long, USER As Double, IFAIL As Long)

' Main routine to get parameters. Limda, (Limda1,limda2); Arfa (Arfa1, Arfa2); Beta (Beta1,Beta2)

Public Sub Get_Parameter()
Dim f1 As Double
Dim f2 As Double

    Call Get_Limda
    GDB_Sbj_Limda = GDB_Sbj_Limda2 / GDB_Sbj_Limda1
    Call Adjust_AB_Bound
    Call Get_AB
    GDB_Sbj_Arfa1 = GDB_Sbj_Temp_Arfa1
    GDB_Sbj_Beta1 = GDB_Sbj_Temp_Beta1
    GDB_Sbj_Arfa = GDB_Sbj_Arfa2 / GDB_Sbj_Arfa1
    GDB_Sbj_Beta = GDB_Sbj_Beta2 / GDB_Sbj_Beta1

End Sub

'Get parameters. Limda1,limda2

Public Sub Get_Limda()
Const N As Long = 2
Dim FVEC(2) As Double
Dim XTOL As Double
Const LWA As Long = N * (3 * N + 13) / 2 + 10
Dim WA(LWA) As Double
Dim IFail As Long
Dim X(2) As Double
Dim L_Loop As Integer
Dim Temp_X1 As Double
Dim Temp_X2 As Double

    Temp_X1 = 0.00001
    Temp_X2 = 0.00001
    X(1) = Temp_X1
    X(2) = Temp_X2
AG: IFail = 1
    XTOL = 0.000001
    L_Loop = 0
    'XTOL = Sqr(X02AJF())
    Call C05NBF(AddressOf FCN_Limda, N, X(1), FVEC(1), XTOL, WA(1), LWA, IFail)
    If IFail = 2 And L_Loop < 5 Then
        L_Loop = L_Loop + 1
```

```
    GoTo AG
End If
If IFail = 3 Then
    Exit Sub
End If
If IFail <> 0 And L_Loop < 5 Then
    Temp_X1 = Temp_X1 * 10
    Temp_X2 = Temp_X2 * 10
    X(1) = Temp_X1
    X(2) = Temp_X2
    L_Loop = L_Loop + 1
    GoTo AG
    End If

End Sub


'Get parameters. Arfa1, Arfa2; Beta1, Beta2
Public Sub Get_AB()
Const N As Long = 2
Dim FVEC(N) As Double
Dim XTOL As Double
Const LWA As Long = N * (3 * N + 13) / 2
Dim WA(LWA) As Double
Dim IFail As Long
Dim X(N) As Double
Dim LI_Ifail As Long
Dim L_Loop As Integer
Dim Temp_X1 As Double
Dim Temp_X2 As Double
Dim LDB_f1 As Double
Dim LDB_f2 As Double

    Temp_X1 = 0.001
    Temp_X2 = 0.001
    X(1) = Temp_X1
    X(2) = Temp_X2
    GDB_Sbj_Arfa2 = 5
    GDB_Sbj_Beta2 = 5
    XTOL = 0.000001
    'XTOL = Sqr(X02AJF())

    LI_Ifail = 1
    GDB_GAMA_Arfa = S14AAF(GDB_Sbj_Arfa2, LI_Ifail)
    L_Loop = 0
    LI_Ifail = 1
    GDB_GAMA_Beta = S14AAF(GDB_Sbj_Beta2, LI_Ifail)
    GDB_Temp_Least_Value = 0
AG: IFail = 1
    Call C05NBF(AddressOf FCN_AB, N, X(1), FVEC(1), XTOL, WA(1), LWA, IFail)
    LDB_f1 = Multi_Qua1
    LDB_f2 = Multi_Qua2
    If (Abs(LDB_f1 - GDB_MLNF) + Abs(LDB_f2 - GDB_MLH)) < GDB_Temp_Least_Value Or
GDB_Temp_Least_Value = 0 Then
        GDB_Sbj_Temp_Arfa1 = GDB_Sbj_Arfa1
        GDB_Sbj_Temp_Beta1 = GDB_Sbj_Beta1
        GDB_Temp_Least_Value = Abs(LDB_f1 - GDB_MLNF) + Abs(LDB_f2 - GDB_MLH)
    End If

    If IFail = 2 And L_Loop < 5 Then
        L_Loop = L_Loop + 1
        GoTo AG
    End If
```

```
    If IFail = 3 Then
       Exit Sub
    End If
    If IFail <> 0 And L_Loop < 5 And (Abs(LDB_f1 - GDB_MLNF) > 0.01 Or Abs(LDB_f2 - GDB_MLH) >
0.01) Then  'Abs(Abs(GDB_MLNF) + Abs(GDB_MLH)) / 2 Then
'       Temp_X1 = Temp_X1 * 10
'       Temp_X2 = Temp_X2 * 10
'       X(1) = Temp_X1
'       X(2) = Temp_X2
       L_Loop = L_Loop + 1
       GoTo AG
    End If
    If IFail <> 0 Then
       GDB_Sbj_Arfa1 = GDB_Sbj_Temp_Arfa1
       GDB_Sbj_Beta1 = GDB_Sbj_Temp_Beta1
    End If

End Sub

Function FORMULA(X As Double) As Double     'For Limda Quadrature function
Dim i As Double
Dim j As Double
Dim LI_Ifail As Long

    i = (-1 * X * GDB_INTERVAL) + Log(X * GDB_INTERVAL) * (GDB_MND + 1)
    j = Log(X) * (GDB_Sbj_Limda2 - 1) + (-1 * GDB_Sbj_Limda1 * X)
    LI_Ifail = 1
    FORMULA = (Exp(i)) * Exp(-Log(Factorial(GDB_MND + 1)) + (j + Log(GDB_Sbj_Limda1 /
S14AAF(GDB_Sbj_Limda2, LI_Ifail))))

End Function

Function FORMULA_SubAB(X As Double) As Double     'For Sub Formula in Arfa & Beta Equation
    If GDB_Sbj_Beta * Log(GDB_Sbj_Arfa * X) > Log(100) Then
       FORMULA_SubAB = 1 - X * 0
    Else
       FORMULA_SubAB = 1 - Exp(-1 * (GDB_Sbj_Arfa * X) ^ GDB_Sbj_Beta)
    End If

End Function


Function FORMULA_AB2(NDIM As Long, ByVal Z As Long) As Double     'For Arfa,Beta Quadraturo
function 2
Dim i As Double
Dim j As Double
Dim K As Double
Dim Address As Long
Dim Result1 As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Dim B As Double
Dim LI_Ifail As Long
Dim AB_X(2) As Double

    For K = 1 To NDIM
       Address = Z + (K - 1) * 8
```

```
    Call CopyMemory(AB_X(K), Address, 8)
  Next K

  B = GDB_INTERVAL
  GDB_Sbj_Beta = AB_X(1)
  GDB_Sbj_Arfa = AB_X(2)
  IFail = 1
  Call D01AJF(AddressOf FORMULA_SubAB, A, B, EPSABS, EPSREL, Result1, ABSERR, W(1), LW,
IW(1), LIW, IFail)
  i = Exp(-1 * GDB_Sbj_Arfa1 * GDB_Sbj_Arfa) * GDB_Sbj_Arfa ^ (GDB_Sbj_Arfa2 - 1) * Exp(-1 *
GDB_Sbj_Beta1 * GDB_Sbj_Beta) * GDB_Sbj_Beta ^ (GDB_Sbj_Beta2 - 1)
  FORMULA_AB2 = (Exp(-1 * GDB_Sbj_Limda * Result1) * (Result1) ^ (GDB_MNF + 1)) * i
End Function

'Main function for the set of equations of limda1,limda2 used by Get_Limda

Sub FCN_Limda(N As Long, ByVal PTR_X As Long, ByVal FVEC As Long, IFLAG As Long)
Dim Address As Long
Dim LD_X(2) As Double
Dim i As Integer
Dim LD_FVEC(2) As Double
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Const B As Double = 400
Dim LDB_Temp_Result As Double

  For i = 1 To 2
    Address = PTR_X + 8 * (i - 1)
    Call CopyMemory(LD_X(i), ByVal Address, 8)
  Next i
  GDB_Sbj_Limda1 = Abs(LD_X(1))
  GDB_Sbj_Limda2 = Abs(LD_X(2))
  LD_FVEC(1) = GDB_Sbj_Limda2 / GDB_Sbj_Limda1 * GDB_INTERVAL - GDB_MLND
  IFail = 1
  LDB_Temp_Result = (GDB_Sbj_Limda2 - 1) * Log(GDB_Sbj_Limda1)
  Call D01AJF(AddressOf FORMULA, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW, IW(1), LIW,
IFail)
  If Result <= 0 Then
    Result = 0.00000000000001
  End If
  LD_FVEC(2) = LDB_Temp_Result + Log(Result) - Log(GDB_MLH)
  For i = 1 To 2
    Address = FVEC + 8 * (i - 1)
    Call CopyMemory(ByVal (Address), VarPtr(LD_FVEC(i)), 8)
  Next i

End Sub


'Main function for the set of equations of Arfa1,Arfa2,Beta1,Beta2 used by Get_AB

Sub FCN_AB(N As Long, ByVal PTR_X As Long, ByVal FVEC As Long, IFLAG As Long)
Dim Address As Long
Dim LD_X(2) As Double
Dim i As Integer
```

```
Dim LD_FVEC(2) As Double
Dim FINVAL1 As Double
Dim FINVAL2 As Double

    For i = 1 To 2
        Address = PTR_X + 8 * (i - 1)
        Call CopyMemory(LD_X(i), ByVal Address, 8)
    Next i
    GDB_Sbj_Arfa1 = Abs(LD_X(1))
    GDB_Sbj_Beta1 = Abs(LD_X(2))
    FINVAL1 = Multi_Qua1()
    LD_FVEC(1) = FINVAL1 - GDB_MLNF
    FINVAL2 = Multi_Qua2()
    LD_FVEC(2) = FINVAL2 - GDB_MLH
    For i = 1 To 2
        Address = FVEC + 8 * (i - 1)
        Call CopyMemory(ByVal (Address), VarPtr(LD_FVEC(i)), 8)
    Next i

End Sub


Function Multi_Qua1() As Double
Dim Address As Long
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Dim B As Double
Dim LI_Ifail As Long


    B = GDB_INTERVAL
    GDB_Sbj_Arfa = GDB_Sbj_Arfa2 / GDB_Sbj_Arfa1
    GDB_Sbj_Beta = GDB_Sbj_Beta2 / GDB_Sbj_Beta1
    IFail = 1
    Call D01AJF(AddressOf FORMULA_SubAB, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
    Multi_Qua1 = GDB_Sbj_Limda * Result

End Function

Function Multi_Qua2() As Double
Const NDIM As Long = 2
Dim A(NDIM) As Double
Dim B(NDIM) As Double
Dim MINPTS As Long
Const MAXPTS As Long = 2000 * NDIM
Dim EPS As Double
Dim ACC As Double
Const LENWRK As Long = (NDIM + 2) * (1 + MAXPTS / (2 ^ NDIM + 2 * NDIM * NDIM + 2 * NDIM + 1))
Dim WRKSTR(LENWRK) As Double
Dim FINVAL As Double
Dim IFail As Long
Dim Sub_Finval As Double

    ' Initial Parameters
    A(1) = GDB_Sbj_Beta_LowBound
```

```
   A(2) = GDB_Sbj_Arfa_LowBound
   B(1) = GDB_Sbj_Beta_UpBound
   B(2) = GDB_Sbj_Arfa_UpBound
   MINPTS = 0
   EPS = 0.0001
   IFail = 1
   Call D01FCF(NDIM, A(1), B(1), MINPTS, MAXPTS, AddressOf FORMULA_AB2, EPS, ACC, LENWRK,
WRKSTR(1), FINVAL, IFail)
      Sub_Finval = (GDB_Sbj_Limda ^ (GDB_MNF + 1) / Factorial(GDB_MNF + 1)) * GDB_Sbj_Arfa1 ^
GDB_Sbj_Arfa2 * GDB_Sbj_Beta1 ^ GDB_Sbj_Beta2 / (GDB_GAMA_Arfa * GDB_GAMA_Beta)
   Multi_Qua2 = FINVAL * Sub_Finval
End Function

Public Function Factorial(N As Long) As Double ' SHOULD LESS THAN 170 171!>1E308 will be overflow

If N = 1 Or N = 0 Then
   Factorial = 1
   Exit Function
End If

If N > 1 Then
   Factorial = N * Factorial(N - 1)
   Exit Function
End If

End Function


Private Sub Adjust_AB_Bound()
Dim LDB_Arfa_UpBound As Double
Dim LDB_Beta_UpBound As Double
Dim LDB_Arfa_LowBound As Double
Dim LDB_Beta_LowBound As Double
Dim LDB_Arfa As Double
Dim LDB_Beta As Double

   GDB_Sbj_Arfa_UpBound = 50
   GDB_Sbj_Beta_UpBound = 50
   GDB_Sbj_Arfa_LowBound = 0
   GDB_Sbj_Beta_LowBound = 0

End Sub


'Objective delay time model of complex system
Option Explicit

Public GDB_Obj_Arfa As Double
Public GDB_Obj_Beta As Double
Public GDB_Obj_Limda As Double
Public GDB_Obj_Interval() As Double
Public GDB_Obj_FailTime() As Long
Public GDB_Obj_Failures() As Long
Public GDB_Obj_Arfa1 As Double
Public GDB_Obj_Arfa2 As Double
Public GDB_Obj_Beta1 As Double
Public GDB_Obj_Beta2 As Double
Public GDB_Obj_Limda1 As Double
Public GDB_Obj_Limda2 As Double
Public GI_Obj_Loop As Integer
Public GDB_Min_FC As Double
Public GDB_Obj_Temp_Limda As Double
Public GDB_Obj_Temp_Arfa As Double
Public GDB_Obj_Temp_Beta As Double
```

```
Dim GDB_Temp_Least_Value As Double
Dim MDB_T As Double

Public Sub Get_Obj_LAB()
Const N As Long = 2
Const LIW As Long = N + 2
Const LW As Long = N * (N - 1) / 2 + 12 * N
Dim F As Double
Dim IBOUND As Long
Dim BL(N) As Double
Dim BU(N) As Double
Dim X(N) As Double
Dim IW(LIW) As Long
Dim W(LW) As Double
Dim IUSER(2) As Long
Dim USER(2) As Double
Dim IFail As Long
Dim LI_Ifail As Long
Dim Temp_X1 As Double
Dim Temp_X2 As Double
Dim L_Loop As Integer
Dim A, B, T, EPS, f1 As Double
Dim MAXCAL As Long
Dim X1 As Double

    EPS = 0.0000000001
    T = 0.0000000001
    A = 0.00000001
    B = 10

    IFail = 1
    MAXCAL = 29

    Call E04ABF(AddressOf Obj_FUNCT_Limda, EPS, T, A, B, MAXCAL, X1, f1, IFail)
    GDB_Obj_Limda = X1

    IBOUND = 0

    X(1) = 1
    X(2) = 1
    Temp_X1 = X(1)
    Temp_X2 = X(2)

    BL(1) = 1E-100
    BL(2) = 1E-100
    BU(1) = 1000
    BU(2) = 1000

Agn:  IFail = 1

Call E04JYF(N, IBOUND, AddressOf Obj_FUNCT_Obj, BL(1), BU(1), X(1), F, IW(1), LIW, W(1), LW,
IUSER(1), USER(1), IFail)

End Sub


Function Obj_FUNCT_Limda(XC As Double, FC As Double) As Double  'For Limda Quadrature function 1
Dim K As Double
Dim Address As Long
Dim LI_Fail As Integer
Dim i As Integer
Dim LDB_FC As Double
'
    LDB_FC = 0
```

```
   For K = 1 To GI_Obj_Loop
      i = 1
      LI_Fail = 0
      Do While GDB_Obj_FailTime(K, i) > -1
         LI_Fail = LI_Fail + 1
         i = i + 1
      Loop
      LDB_FC = LDB_FC + ((LI_Fail + GDB_Obj_Failures(K)) * Log(XC) - XC * (GDB_Obj_Interval(K) /
30.5))
   Next K
   If GI_SO_Flag = 0 Then
      FC = -1 * LDB_FC
   Else
      FC = -1 * (LDB_FC + Get_Obj_Gama(GDB_Obj_Limda, GDB_Obj_Limda1, GDB_Obj_Limda2))
   End If

End Function

Function Obj_FUNCT_Obj(N As Long, ByVal XC As Long, FC As Double, IUSER As Long, USER As
Long) As Double  'For Arfa,Beta Quadrature function 1
Dim K As Double
Dim Address As Long
Dim LD_X(3) As Double
Dim LDB_Qua As Double
Dim LDB_Qua_FS As Double
Dim LDB_NF As Double    ' No failure in [I(j-1),I(j)]
Dim LDB_AF As Double    ' A failure in [I(j),I(j)+t]
Dim LDB_FS As Double    ' N failures in [T(i-1),T(i)]
Dim i As Long
Dim LDB_Failtime As Double
Dim LDB_FC As Double
Dim LDB_Temp_AF As Double
Dim LDB_Int_Beg As Double   'Interval Beginning time
Dim LDB_Int_End As Double   'Interval End time

   For K = 1 To N
      Address = XC + (K - 1) * 8
      Call CopyMemory(LD_X(K), Address, 8)
   Next K

   GDB_Obj_Arfa = Abs(LD_X(1))
   GDB_Obj_Beta = Abs(LD_X(2))

   If GDB_Obj_Beta = 0 Then
      GDB_Obj_Beta = 1E-305
   End If
   If GDB_Obj_Arfa = 0 Then
      GDB_Obj_Arfa = 1E-305
   End If

   LDB_FC = 0
For K = 1 To GI_Obj_Loop
   If GDB_Obj_Failures(K) = 0 Then
      GoTo Agn
   End If
   LDB_Qua_FS = Get_Obj_FS(0, GDB_Obj_Interval(K) / 30.5)
   If LDB_Qua_FS = 0 Then
      LDB_FS = (GDB_Obj_Failures(K)) * Log(1E-305)
   Else
      LDB_FS = (GDB_Obj_Failures(K)) * Log(LDB_Qua_FS)
   End If
   LDB_AF = 0
   i = 1
   Do While GDB_Obj_FailTime(K, i) > -1
```

```
LDB_Failtime = (GDB_Obj_FailTime(K, i) / 30.5)

If GDB_Obj_Beta * Log(GDB_Obj_Arfa * LDB_Failtime) > Log(1000000) Then
   LDB_Temp_AF = 0
Else
   If 1 - Exp(-1 * ((GDB_Obj_Arfa * LDB_Failtime) ^ GDB_Obj_Beta)) = 0 Then
      LDB_Temp_AF = 100 * Log(1E-305)
   Else
      LDB_Temp_AF = Log(1 - Exp(-1 * (GDB_Obj_Arfa * LDB_Failtime) ^ GDB_Obj_Beta))
   End If
End If
LDB_AF = LDB_AF + LDB_Temp_AF
        i = i + 1
      Loop
      LDB_FC = LDB_FC + (LDB_FS + LDB_AF)
Agn:  Next K
   If GI_SO_Flag = 0 Then
      FC = -1 * LDB_FC
   Else
      FC = -1 * (LDB_FC + Get_Obj_Gama(GDB_Obj_Arfa, GDB_Obj_Arfa1, GDB_Obj_Arfa2) +
Get_Obj_Gama(GDB_Obj_Beta, GDB_Obj_Beta1, GDB_Obj_Beta2))
   End If
   If FC < GDB_Min_FC Then
      GDB_Min_FC = FC
   End If
End Function

Function Get_Obj_FS(ByVal PDB_Int_Beg As Double, PDB_Int_End) As Double
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.0000000001
Dim A As Double
Dim B As Double

   IFail = 1
   A = PDB_Int_Beg
   B = PDB_Int_End
   Call D01AJF(AddressOf Obj_FORMULA_FS, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
   Get_Obj_FS = Result

End Function


Public Function Obj_FORMULA_FS(X As Double) As Double     'For Sub Formula in Arfa & Beta Equation
   If GDB_Obj_Beta * Log(GDB_Obj_Arfa * X) > Log(1000000) Then
      Obj_FORMULA_FS = X * 0
   Else
      Obj_FORMULA_FS = Exp(-1 * ((GDB_Obj_Arfa * X) ^ GDB_Obj_Beta))
   End If

End Function

Function Get_Obj_Gama(LDB_Var As Double, LDB_Arfa As Double, LDB_Beta As Double) As Double
Dim IFail As Long
   If LDB_Var = 0 Then
      LDB_Var = 1E-100
   End If
```

```
    IFail = 1
    Get_Obj_Gama = -1 * (LDB_Arfa * LDB_Var) + LDB_Beta * Log(LDB_Arfa) + (LDB_Beta - 1) *
Log(LDB_Var) - Log(S14AAF(LDB_Beta, IFail))

End Function


'Objective delay time model of single component
Option Explicit

Public GDB_SObj_GArfa As Double
Public GDB_SObj_GArfa1 As Double
Public GDB_SObj_GArfa2 As Double
Public GDB_SObj_GBeta As Double
Public GDB_SObj_GBeta1 As Double
Public GDB_SObj_GBeta2 As Double
Public GDB_SObj_FArfa As Double
Public GDB_SObj_FArfa1 As Double
Public GDB_SObj_FArfa2 As Double
Public GDB_SObj_FBeta As Double
Public GDB_SObj_FBeta1 As Double
Public GDB_SObj_FBeta2 As Double
Dim GDB_SObj_Temp_GArfa As Double
Dim GDB_SObj_Temp_GBeta As Double
Dim GDB_SObj_Temp_FArfa As Double
Dim GDB_SObj_Temp_FBeta As Double
Dim GDB_Min_FC As Double
Dim MDB_TE As Double
Dim GDB_Temp_Least_Value As Double

Public GI_SObj_IR As Integer
Public GI_SObj_FR As Integer
Public GDB_IR_Begin() As Double
Public GDB_IR_End() As Double
Public GDB_FR_Begin() As Double
Public GDB_FR_End() As Double


Public Sub Get_SObj_LAB()
Const N As Long = 4
Const LIW As Long = N + 2
Const LW As Long = N * (N - 1) / 2 + 12 * N
Dim F As Double
Dim IBOUND As Long
Dim BL(N) As Double
Dim BU(N) As Double
Dim X(N) As Double
Dim IW(LIW) As Long
Dim W(LW) As Double
Dim IUSER(2) As Long
Dim USER(2) As Double
Dim IFail As Long
Dim LI_Ifail As Long
Dim Temp_X1 As Double
Dim Temp_X2 As Double
Dim Temp_X3 As Double
Dim Temp_X4 As Double
Dim L_Loop As Integer


    IBOUND = 0

    X(1) = 0.00001
```

```
X(2) = 0.00001
X(3) = 0.00001
X(4) = 0.00001
Temp_X1 = X(1)
Temp_X2 = X(2)
Temp_X3 = X(3)
Temp_X4 = X(4)


BL(1) = 0.0000000000001
BL(2) = 0.0000000000001
BL(3) = 0.0000000000001
BL(4) = 0.0000000000001
BU(1) = 10000
BU(2) = 10000
BU(3) = 10000
BU(4) = 10000

IFail = 1
GDB_Min_FC = 1000
Agn:  Call E04JYF(N, IBOUND, AddressOf Obj_FUNCT, BL(1), BU(1), X(1), F, IW(1), LIW, W(1), LW,
IUSER(1), USER(1), IFail)

If GDB_Min_FC < GDB_Temp_Least_Value Or GDB_Temp_Least_Value = 0 Then
   GDB_SObj_Temp_GArfa = GDB_SObj_GArfa
   GDB_SObj_Temp_GBeta = GDB_SObj_GBeta
   GDB_SObj_Temp_FArfa = GDB_SObj_FArfa
   GDB_SObj_Temp_FBeta = GDB_SObj_FBeta
   GDB_Temp_Least_Value = GDB_Min_FC
End If
If IFail = 2 And L_Loop < 5 Then
   L_Loop = L_Loop + 1
   GoTo Agn
End If
If IFail = 3 Then
   Exit Sub
End If

If IFail <> 0 And L_Loop < 5 Then
   L_Loop = L_Loop + 1
   GoTo Agn
End If

If IFail <> 0 Then
   GDB_SObj_Temp_GArfa = GDB_SObj_GArfa
   GDB_SObj_Temp_GBeta = GDB_SObj_GBeta
   GDB_SObj_Temp_FArfa = GDB_SObj_FArfa
   GDB_SObj_Temp_FBeta = GDB_SObj_FBeta
End If

End Sub


Function Obj_FUNCT(N As Long, ByVal XC As Long, FC As Double, IUSER As Long, USER As Long) As
Double  'For Arfa,Beta Quadrature function 1
Dim K As Double
Dim Address As Long
Dim LD_X(4) As Double
Dim LDB_Qua As Double
Dim LDB_IR As Double    ' Inspection Renewal
Dim LDB_FR As Double    ' Failure Renewal
Dim LDB_EO As Double    ' End of Observation
Dim i As Long
Dim LDB_Failtime As Double
```

```
Dim LDB_FC As Double
Dim LDB_Temp_AF As Double
Dim LDB_IR_Begin As Double
Dim LDB_IR_End As Double
Dim LDB_FR_Begin As Double
Dim LDB_FR_End As Double


    For K = 1 To N
       Address = XC + (K - 1) * 8
       Call CopyMemory(LD_X(K), Address, 8)
    Next K

    GDB_SObj_GArfa = Abs(LD_X(1))
    GDB_SObj_GBeta = Abs(LD_X(2))
    GDB_SObj_FArfa = Abs(LD_X(3))
    GDB_SObj_FBeta = Abs(LD_X(4))
    LDB_FC = 0

    LDB_IR = 0
    If GI_SObj_IR > 0 Then
       For K = 1 To GI_SObj_IR
          LDB_IR_Begin = GDB_IR_Begin(K) / 365
          LDB_IR_End = GDB_IR_End(K) / 365
          LDB_IR = LDB_IR + Get_IR(LDB_IR_Begin, LDB_IR_End)
       Next K
    End If

    LDB_FR = 0
    If GI_SObj_FR > 0 Then
       For K = 1 To GI_SObj_FR
          LDB_FR_Begin = GDB_FR_Begin(K) / 365
          LDB_FR_End = GDB_FR_End(K) / 365
          LDB_FR = LDB_FR + Get_FR(LDB_FR_Begin, LDB_FR_End)
       Next K
    End If
    If GI_SO_Flag = 0 Then
       FC = -1 * (LDB_FR + LDB_IR)
    Else
       FC = -1 * (LDB_FR + LDB_IR + Get_SObj_Gama(GDB_SObj_GArfa, GDB_SObj_GArfa1,
GDB_SObj_GArfa2) + Get_SObj_Gama(GDB_SObj_GBeta, GDB_SObj_GBeta1, GDB_SObj_GBeta2) +
_
       Get_SObj_Gama(GDB_SObj_FArfa, GDB_SObj_FArfa1, GDB_SObj_FArfa2) +
Get_SObj_Gama(GDB_SObj_FBeta, GDB_SObj_FBeta1, GDB_SObj_FBeta2))
    End If
    If Abs(FC) > Abs(GDB_Min_FC) Then
       GDB_Min_FC = FC
    End If

End Function

Function Get_SObj_Gama(LDB_Var As Double, LDB_Arfa As Double, LDB_Beta As Double) As Double
Dim IFail As Long

    IFail = 1
    Get_SObj_Gama = -1 * (LDB_Arfa * LDB_Var) + LDB_Beta * Log(LDB_Arfa) + (LDB_Beta - 1) *
Log(LDB_Var) - Log(S14AAF(LDB_Beta, IFail))

End Function

Function Get_IR(PDB_TB As Double, PDB_TE As Double) As Double
Dim Result As Double
Dim ABSERR As Double
```

```
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Dim A As Double
Dim B As Double

   IFail = 1
   A = PDB_TB
   B = PDB_TE
   MDB_TE = B
   Call D01AJF(AddressOf SObj_IR_FORMULA, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
   Get_IR = Result + Log(GDB_SObj_GBeta) + GDB_SObj_GBeta * Log(GDB_SObj_GArfa)

End Function


Function SObj_IR_FORMULA(X As Double) As Double     'For Sub Formula in Arfa & Beta Equation
SObj_IR_FORMULA = (GDB_SObj_GBeta - 1) * Log(X) - (X * GDB_SObj_GArfa) ^ GDB_SObj_GBeta -
(GDB_SObj_FArfa * MDB_TE - GDB_SObj_FArfa * X) ^ GDB_SObj_FBeta

End Function

Function Get_FR(PDB_TB As Double, PDB_TE As Double) As Double
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Dim A As Double
Dim B As Double

   IFail = 1
   A = PDB_TB
   B = PDB_TE
   MDB_TE = B
   Call D01AJF(AddressOf SObj_FR_FORMULA, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
   Get_FR = Result + Log(GDB_SObj_GBeta) + GDB_SObj_GBeta * Log(GDB_SObj_GArfa) +
Log(GDB_SObj_FBeta) + GDB_SObj_FBeta * Log(GDB_SObj_FArfa)

End Function

Function SObj_FR_FORMULA(X As Double) As Double     'For Sub Formula in Arfa & Beta Equation
SObj_FR_FORMULA = (GDB_SObj_GBeta - 1) * Log(X) - (X * GDB_SObj_GArfa) ^ GDB_SObj_GBeta *
(GDB_SObj_FBeta - 1) * Log(MDB_TE - X) - ((MDB_TE - X) * GDB_SObj_FArfa) ^ GDB_SObj_FBeta

End Function


'Subjective delay time model of single component
Option Explicit

Public GDB_MNTIP As Double      'Subjective data of Mean Time to Initial Point
Public GDB_MITIP As Double      'Subjective data of Minimum Time to Initial Point
```

```
Public GDB_MXTIP As Double      'Subjective data of Maximum Time to Initial Point
Public GDB_MNDT As Double       'Subjective data of Mean Delay Time
Public GDB_MXDT As Double       'Subjective data of Maximum Delay Time
Public GDB_MIDT As Double       'Subjective data of Minimum Delay Time

Public GDB_Sin_GARFA1 As Double    'For Hyper Parameter Arfa1 of g(u)
Public GDB_Sin_GARFA2 As Double    'For Hyper Parameter Arfa2 of g(u)
Public GDB_Sin_FARFA1 As Double    'For Hyper Parameter Beta1 of g(u)
Public GDB_Sin_FARFA2 As Double    'For Hyper Parameter Beta2 of g(u)
Public GDB_Sin_GBETA1 As Double    'For Hyper Parameter Arfa1 of f(h)
Public GDB_Sin_GBETA2 As Double    'For Hyper Parameter Arfa2 of f(h)
Public GDB_Sin_FBETA1 As Double    'For Hyper Parameter Beta1 of f(h)
Public GDB_Sin_FBETA2 As Double    'For Hyper Parameter Beta2 of f(h)
Public GDB_Sin_GArfa As Double     'For Parameter Arfa of g(u)
Public GDB_Sin_GBeta As Double     'For Parameter Beta of g(u)
Public GDB_Sin_FArfa As Double     'For Parameter Arfa of f(h)
Public GDB_Sin_FBeta As Double     'For Parameter Arfa of f(h)
Dim GDB_Sin_GGArfa As Double
Dim GDB_Sin_GGBeta As Double
Dim GDB_Sin_GFArfa As Double
Dim GDB_Sin_GFBeta As Double
Dim GDB_Sin_Temp_Arfa1 As Double
Dim GDB_Sin_Temp_Beta1 As Double
Dim GDB_Temp_Least_Value As Double
Dim GDB_Temp_Least_ValueF As Double
Dim GDB_Sin_Arfa_UpBound As Double
Dim GDB_Sin_Beta_UpBound As Double
Dim GDB_Sin_Arfa_LowBound As Double
Dim GDB_Sin_Beta_LowBound As Double

' Main routine to get parameters of single component. Arfa (Arfa1, Arfa2) of g(u); Beta (Beta1,Beta2) of
g(u); Arfa (Arfa1, Arfa2) of f(h); Beta (Beta1,Beta2) of f(h)

Public Sub Get_Sin_Parameters()
Call Adjust_AB_Bound
   GDB_MLH = 0.9999
   Call Get_GAB
   Call Get_FAB
   GDB_Sin_GArfa = GDB_Sin_GARFA2 / GDB_Sin_GARFA1
   GDB_Sin_GBeta = GDB_Sin_GBETA2 / GDB_Sin_GBETA1
   GDB_Sin_FArfa = GDB_Sin_FARFA2 / GDB_Sin_FARFA1
   GDB_Sin_FBeta = GDB_Sin_FBETA2 / GDB_Sin_FBETA1


End Sub


Public Sub Get_GAB()
Const N As Long = 2
Dim FVEC(N) As Double
Dim XTOL As Double
Const LWA As Long = N * (3 * N + 13) / 2
Dim WA(LWA) As Double
Dim IFail As Long
Dim X(N) As Double
Dim LI_Ifail As Long
Dim L_Loop As Integer
Dim Temp_X1 As Double
Dim Temp_X2 As Double
Dim LDB_f1 As Double
Dim LDB_f2 As Double
```

```
Temp_X1 = 0.001
Temp_X2 = 0.001
X(1) = Temp_X1
X(2) = Temp_X2
GDB_Sin_GARFA2 = 2
GDB_Sin_GBETA2 = 2
XTOL = 0.0001
'XTOL = Sqr(X02AJF())

LI_Ifail = 1
GDB_Sin_GGArfa = S14AAF(GDB_Sin_GARFA2, LI_Ifail)
L_Loop = 0
   LI_Ifail = 1
   GDB_Sin_GGBeta = S14AAF(GDB_Sin_GBETA2, LI_Ifail)
   GDB_Temp_Least_Value = 0
AG: IFail = 1
   Call C05NBF(AddressOf FCN_GAB, N, X(1), FVEC(1), XTOL, WA(1), LWA, IFail)

   LDB_f1 = Multi_GQua1
   LDB_f2 = Multi_GQua2
   If (Abs(LDB_f1 - GDB_MNTIP) + Abs(LDB_f2 - GDB_MLH)) < GDB_Temp_Least_Value Or
GDB_Temp_Least_Value = 0 Then
      GDB_Sin_Temp_Arfa1 = GDB_Sin_GARFA1
      GDB_Sin_Temp_Beta1 = GDB_Sin_GBETA1
      GDB_Temp_Least_Value = Abs(LDB_f1 - GDB_MNTIP) + Abs(LDB_f2 - GDB_MLH)
   End If

   If IFail = 2 And L_Loop < 3 Then
      L_Loop = L_Loop + 1
      GoTo AG
   End If
   If IFail = 3 Then
      Exit Sub
   End If
   If IFail <> 0 And L_Loop < 3 And (Abs(LDB_f1 - GDB_MNTIP) > 0.001 Or Abs(LDB_f2 - GDB_MLH) >
0.01) Then  'Abs(Abs(GDB_MLNF) + Abs(GDB_MLH)) / 2 Then
L_Loop = L_Loop + 1
      GoTo AG
   End If
   If IFail <> 0 Then
      GDB_Sin_GARFA1 = GDB_Sin_Temp_Arfa1
      GDB_Sin_GBETA1 = GDB_Sin_Temp_Beta1
   End If

End Sub
' Get Parameter Arfa

'Main function for the set of equations of Arfa1,Arfa2,Beta1,Beta2 used by Get_AB

Sub FCN_GAB(N As Long, ByVal PTR_X As Long, ByVal FVEC As Long, IFLAG As Long)
Dim Address As Long
Dim LD_X(2) As Double
Dim i As Integer
Dim LD_FVEC(2) As Double
Dim FINVAL1 As Double
Dim FINVAL2 As Double

   For i = 1 To 2
      Address = PTR_X + 8 * (i - 1)
      Call CopyMemory(LD_X(i), ByVal Address, 8)
   Next i
   GDB_Sin_GARFA1 = Abs(LD_X(1))
```

```
   GDB_Sin_GBETA1 = Abs(LD_X(2))
   FINVAL1 = Multi_GQua1()
   LD_FVEC(1) = FINVAL1 - GDB_MNTIP
   FINVAL2 = Multi_GQua2()
   LD_FVEC(2) = FINVAL2 - GDB_MLH
   For i = 1 To 2
      Address = FVEC + 8 * (i - 1)
      Call CopyMemory(ByVal (Address), VarPtr(LD_FVEC(i)), 8)
   Next i

End Sub

'Main function for the set of equations of Arfa1,Arfa2,Beta1,Beta2 used by Get_AB


Function Multi_GQua2() As Double

Dim Address As Long
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Dim B As Double
Dim LI_Ifail As Long

   B = GDB_MXTIP
   GDB_Sin_GArfa = GDB_Sin_GARFA2 / GDB_Sin_GARFA1
   GDB_Sin_GBeta = GDB_Sin_GBETA2 / GDB_Sin_GBETA1
   IFail = 1
   Call D01AJF(AddressOf FORMULA_GSubAB1, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
   Multi_GQua2 = GDB_Sin_GBeta * Result


End Function


Function FORMULA_GSubAB1(X As Double) As Double     'For Sub Formula in Arfa & Beta Equation

   If GDB_Sin_GBeta * Log(X * GDB_Sin_GArfa) > Log(10000) Then
      FORMULA_GSubAB1 = 0
   Else
      FORMULA_GSubAB1 = (X * GDB_Sin_GArfa) ^ GDB_Sin_GBeta * Exp(-1 * (X * GDB_Sin_GArfa) ^
GDB_Sin_GBeta) / X
   End If

End Function


Function Multi_GQua1() As Double

Const NDIM As Long = 2
Dim A(NDIM) As Double
Dim B(NDIM) As Double
Dim MINPTS As Long
Const MAXPTS As Long = 2000 * NDIM
```

```
Dim EPS As Double
Dim ACC As Double
Const LENWRK As Long = (NDIM + 2) * (1 + MAXPTS / (2 ^ NDIM + 2 * NDIM * NDIM + 2 * NDIM + 1))
Dim WRKSTR(LENWRK) As Double
Dim FINVAL As Double
Dim IFail As Long
Dim Sub_Finval As Double

    ' Initial Parameters
    A(1) = GDB_Sin_Beta_LowBound
    A(2) = GDB_Sin_Arfa_LowBound
    B(1) = GDB_Sin_Beta_UpBound
    B(2) = GDB_Sin_Arfa_UpBound
    MINPTS = 0
    EPS = 0.0001
    IFail = 1
    Call D01FCF(NDIM, A(1), B(1), MINPTS, MAXPTS, AddressOf FORMULA_GAB2, EPS, ACC,
LENWRK, WRKSTR(1), FINVAL, IFail)
    Multi_GQua1 = FINVAL / GDB_Sin_GGArfa / GDB_Sin_GGBeta * GDB_Sin_GARFA1 ^
GDB_Sin_GARFA2 * GDB_Sin_GBETA1 ^ GDB_Sin_GBETA2

End Function



Function FORMULA_GAB2(NDIM As Long, ByVal Z As Long) As Double   'For Arfa,Beta Quadrature
function 2
Dim K As Double
Dim Address As Long
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Dim B As Double
Dim LI_Ifail As Long
Dim AB_X(2) As Double

    For K = 1 To NDIM
        Address = Z + (K - 1) * 8
        Call CopyMemory(AB_X(K), Address, 8)
    Next K

    B = 500
    GDB_Sin_GBeta = AB_X(1)
    GDB_Sin_GArfa = AB_X(2)
    IFail = 1
    Call D01AJF(AddressOf FORMULA_GSubAB2, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
    FORMULA_GAB2 = Result * Exp((GDB_Sin_GARFA2 - 1) * Log(GDB_Sin_GArfa) +
(GDB_Sin_GBETA2) * Log(GDB_Sin_GBeta) - (GDB_Sin_GARFA1 * GDB_Sin_GArfa) -
(GDB_Sin_GBETA1 * GDB_Sin_GBeta))

End Function



Function FORMULA_GSubAB2(X As Double) As Double    'For Sub Formula in Arfa & Beta Equation

    If (GDB_Sin_GBeta - 1) * Log(X * GDB_Sin_GArfa) > Log(10000) Then
```

```
      FORMULA_GSubAB2 = 0
   Else
      FORMULA_GSubAB2 = (X * GDB_Sin_GArfa) ^ (GDB_Sin_GBeta) * Exp(-1 * (X * GDB_Sin_GArfa)
^ GDB_Sin_GBeta)
   End If

End Function


Public Function Factorial(N As Long) As Double ' SHOULD LESS THAN 170 171!>1E308 will be overflow

If N = 1 Or N = 0 Then
   Factorial = 1
   Exit Function
End If

If N > 1 Then
   Factorial = N * Factorial(N - 1)
   Exit Function
End If

End Function


Private Sub Adjust_AB_Bound()

   GDB_Sin_Arfa_UpBound = 50
   GDB_Sin_Beta_UpBound = 50
   GDB_Sin_Arfa_LowBound = 0
   GDB_Sin_Beta_LowBound = 0

End Sub


Public Sub Get_FAB()

Const N As Long = 2
Dim FVEC(N) As Double
Dim XTOL As Double
Const LWA As Long = N * (3 * N + 13) / 2
Dim WA(LWA) As Double
Dim IFail As Long
Dim X(N) As Double
Dim LI_Ifail As Long
Dim L_Loop As Integer
Dim Temp_X1 As Double
Dim Temp_X2 As Double
Dim LDB_f1 As Double
Dim LDB_f2 As Double

   Temp_X1 = 0.001
   Temp_X2 = 0.001
   X(1) = Temp_X1
   X(2) = Temp_X2
   GDB_Sin_FARFA2 = 2
   GDB_Sin_FBETA2 = 2
   XTOL = 0.0001
   'XTOL = Sqr(X02AJF())

   LI_Ifail = 1
   GDB_Sin_GFArfa = S14AAF(GDB_Sin_FARFA2, LI_Ifail)
```

```
    L_Loop = 0
    LI_Ifail = 1
    GDB_Sin_GFBeta = S14AAF(GDB_Sin_FBETA2, LI_Ifail)
    GDB_Temp_Least_ValueF = 0
AG: IFail = 1
    Call C05NBF(AddressOf FCN_FAB, N, X(1), FVEC(1), XTOL, WA(1), LWA, IFail)


    LDB_f1 = Multi_FQua1
    LDB_f2 = Multi_FQua2
    If (Abs(LDB_f1 - GDB_MNDT) + Abs(LDB_f2 - GDB_MLH)) < GDB_Temp_Least_ValueF Or
GDB_Temp_Least_ValueF = 0 Then
      GDB_Sin_Temp_Arfa1 = GDB_Sin_FARFA1
      GDB_Sin_Temp_Beta1 = GDB_Sin_FBETA1
      GDB_Temp_Least_ValueF = Abs(LDB_f1 - GDB_MNDT) + Abs(LDB_f2 - GDB_MLH)
    End If

    If IFail = 2 And L_Loop < 3 Then
      L_Loop = L_Loop + 1
      GoTo AG
    End If
    If IFail = 3 Then
      Exit Sub
    End If
    If IFail <> 0 And L_Loop < 3 And (Abs(LDB_f1 - GDB_MNDT) > 0.001 Or Abs(LDB_f2 - GDB_MLH) >
0.01) Then  'Abs(Abs(GDB_MLNF) + Abs(GDB_MLH)) / 2 Then
      Temp_X1 = Temp_X1 * 10
      Temp_X2 = Temp_X2 * 10
      X(1) = Temp_X1
      X(2) = Temp_X2
      L_Loop = L_Loop + 1
      GoTo AG
    End If
    If IFail <> 0 Then
      GDB_Sin_FARFA1 = GDB_Sin_Temp_Arfa1
      GDB_Sin_FBETA1 = GDB_Sin_Temp_Beta1
    End If

End Sub
' Get Parameter Arfa

'Main function for the set of equations of Arfa1,Arfa2,Beta1,Beta2 used by Get_AB

Sub FCN_FAB(N As Long, ByVal PTR_X As Long, ByVal FVEC As Long, IFLAG As Long)

Dim Address As Long
Dim LD_X(2) As Double
Dim i As Integer
Dim LD_FVEC(2) As Double
Dim FINVAL1 As Double
Dim FINVAL2 As Double

For i = 1 To 2
   Address = PTR_X + 8 * (i - 1)
   Call CopyMemory(LD_X(i), ByVal Address, 8)
Next i
GDB_Sin_FARFA1 = Abs(LD_X(1))
GDB_Sin_FBETA1 = Abs(LD_X(2))
FINVAL1 = Multi_FQua1()
LD_FVEC(1) = FINVAL1 - GDB_MNDT
FINVAL2 = Multi_FQua2()
LD_FVEC(2) = FINVAL2 - GDB_MLH
For i = 1 To 2
   Address = FVEC + 8 * (i - 1)
```

```
    Call CopyMemory(ByVal (Address), VarPtr(LD_FVEC(i)), 8)
  Next i

End Sub


'Main function for the set of equations of Arfa1,Arfa2,Beta1,Beta2 used by Get_AB

Function Multi_FQua2() As Double

Dim Address As Long
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Const A As Double = 0
Dim B As Double
Dim LI_Ifail As Long

B = GDB_MIDT
   GDB_Sin_FArfa = GDB_Sin_FARFA2 / GDB_Sin_FARFA1
   GDB_Sin_FBeta = GDB_Sin_FBETA2 / GDB_Sin_FBETA1
   IFail = 1
   Call D01AJF(AddressOf FORMULA_FSubAB1, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
   Multi_FQua2 = GDB_Sin_FBeta * Result
'   Multi_FQua1 = 1 - GDB_Sin_FBeta * GDB_Sin_FArfa * Result

End Function


Function FORMULA_FSubAB1(X As Double) As Double    'For Sub Formula in Arfa & Beta Equation

   If GDB_Sin_FBeta * Log(X * GDB_Sin_FArfa) > Log(10000) Then
      FORMULA_FSubAB1 = 0
   Else
      FORMULA_FSubAB1 = (X * GDB_Sin_FArfa) ^ GDB_Sin_FBeta * Exp(-1 * (X * GDB_Sin_FArfa) ^
GDB_Sin_FBeta) / X
   End If

End Function
Function Multi_FQua1() As Double

Const NDIM As Long = 2
Dim A(NDIM) As Double
Dim B(NDIM) As Double
Dim MINPTS As Long
Const MAXPTS As Long = 2000 * NDIM
Dim EPS As Double
Dim ACC As Double
Const LENWRK As Long = (NDIM + 2) * (1 + MAXPTS / (2 ^ NDIM + 2 * NDIM * NDIM + 2 * NDIM + 1))
Dim WRKSTR(LENWRK) As Double
Dim FINVAL As Double
Dim IFail As Long
Dim Sub_Finval As Double

   ' Initial Parameters
   A(1) = GDB_Sin_Beta_LowBound
   A(2) = GDB_Sin_Arfa_LowBound
```

```
    B(1) = GDB_Sin_Beta_UpBound
    B(2) = GDB_Sin_Arfa_UpBound
    MINPTS = 0
    EPS = 0.0001
    IFail = 1
    Call D01FCF(NDIM, A(1), B(1), MINPTS, MAXPTS, AddressOf FORMULA_FAB2, EPS, ACC,
LENWRK, WRKSTR(1), FINVAL, IFail)
'    Multi_FQua2 = 1 - FINVAL / GDB_Sin_GFArfa / GDB_Sin_GFBeta
    Multi_FQua1 = FINVAL / GDB_Sin_GFArfa / GDB_Sin_GFBeta * GDB_Sin_FARFA1 ^
GDB_Sin_FARFA2 * GDB_Sin_FBETA1 ^ GDB_Sin_FBETA2

End Function

Function FORMULA_FAB2(NDIM As Long, ByVal Z As Long) As Double   'For Arfa,Beta Quadrature
function 2
Dim K As Double
Dim Address As Long
Dim Result As Double
Dim ABSERR As Double
Const LW As Long = 2000
Dim W(LW) As Double
Const LIW As Long = LW / 4
Dim IW(LIW) As Long
Dim IFail As Long
Const EPSABS As Double = 0
Const EPSREL As Double = 0.001
Dim A As Double
Dim B As Double
Dim LI_Ifail As Long
Dim AB_X(2) As Double

    For K = 1 To NDIM
        Address = Z + (K - 1) * 8
        Call CopyMemory(AB_X(K), Address, 8)
    Next K

    A = 0
    B = 500
    GDB_Sin_FBeta = AB_X(1)
    GDB_Sin_FArfa = AB_X(2)
    IFail = 1
    Call D01AJF(AddressOf FORMULA_FSubAB2, A, B, EPSABS, EPSREL, Result, ABSERR, W(1), LW,
IW(1), LIW, IFail)
FORMULA_FAB2 = Result * Exp((GDB_Sin_FARFA2 - 1) * Log(GDB_Sin_FArfa) + (GDB_Sin_FBETA2) *
Log(GDB_Sin_FBeta) - (GDB_Sin_FARFA1 * GDB_Sin_FArfa) - (GDB_Sin_FBETA1 * GDB_Sin_FBeta))
End Function


Function FORMULA_FSubAB2(X As Double) As Double    'For Sub Formula in Arfa & Beta Equation
    If (GDB_Sin_FBeta - 1) * Log(X * GDB_Sin_FArfa) > Log(10000) Then
        FORMULA_FSubAB2 = 0
    Else
        FORMULA_FSubAB2 = (X * GDB_Sin_FArfa) ^ (GDB_Sin_FBeta) * Exp(-1 * (X * GDB_Sin_FArfa) ^
GDB_Sin_FBeta)
    End If

End Function
```

Reference

1. Baker, R.D. and Wang, W., "Estimating the delay-time distribution of faults in repairable machinery from failure data", *IMA Journal of Mathematics Applied in Business and Industry* 3 (1991) 259-281

2. Baker, R.D. and Wang, W., "Developing and testing the delay-time model", Salfrod University Mathematics Dept. Technical Report MCS-92-09, 1992, *and Journal of the Operational Research Society* 44 (1993) 361-374

3. Cerone, P., "On a simplified delay time model of reliability of equipment subject to inspection monitoring", *Journal of the Operational Research Society* 42 (1991) 505-511

4. Christer, A.H., "Operational Research applied to industrial maintenance and replacement" in: Eglese and Rand (eds.), *Developments in Operational Research*, Pergamon Press, Oxford, 1984, 31-58

5. Christer, A.H., "Delay-time model of reliability of equipment subject to inspection monitoring", *Journal of the Operational Research Society* 38 (1987) 329-334

6. Christer, A.H., and Redmond, D.F., "A recent mathematical development in maintenance theory", *IMA Journal of Mathematics Applied in Business and Industry* 2 (1990) 97-108

7. Christer, A.H., and Redmond, D.F., "Revising models of maintenance and inspection", *International Journal of Production Economics* 24 (1992) 227-234

8. Christer, A.H., and Waller, W.M., "Delay time models of industrial maintenance problems" *Journal of the Operational Research Society* 35 (1984) 401-406

9. Christer, A.H. and Waller, W.M., "An operational research approach to planned maintenance: Modelling PM for a vehicle fleet", *Journal of the Operational Research Society* 35 (1984a) 967-984

10. Christer, A.H. and Waller, W.M., "Reducing production downtime using delay-time analysis", *Journal of the Operational Research Society* 35 (1984b) 499-512

11. Christer, A.H. and Wang W. and Baker R.D., (1995), "Modelling maintenance practice of production plant using the delay-time concept", *IMA Journal of Mathematics Applied in Business & Industry* 6, 67-83

12. Christer, A.H. and W. Wang "A delay time based maintenance model of multi-component system", *IMA Journal of Mathematics Applied in Business & Industry* 6 (1995) 205-222

13. Christer, A.H. "Developments in delay time analysis for modelling plant maintenance", *Journal of the Operational Research Society* 50 (1999) 1120-1137

14. Pellegrin, C., "A graphical procedure for an on-condition maintenance policy: Imperfect-inspection model and interpretation", *IMA Journal of Mathematics Applied in Business and Industry* 3 (1991) 177-191

15. Tijms, H.C. "Stochastic models - an algorithmic approach", *John Wiley and Sons*, New York, 1994

16. W. Wang, "Subjective estimation of the delay time distribution in maintenance modelling" , *European Journal of Operational Research* 99 (1997) 516-529

17. W. Wang and X. Jia "A Bayesian approach in delay time maintenance model parameters estimation using both subjective data and objective data", (2000) *Unpublished.*