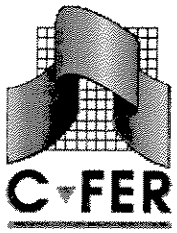
221AR

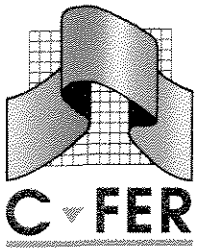
# *PIRAMID*

## Program User's Guide

## Version 10




C-FER Technologies Inc.
200 Karl Clark Road
Edmonton, Alberta
Canada T6N 1H2



200
Karl Clark Road
Edmonton, Alberta
Canada T6N 1H2

Tel: (403) 450-3300
Fax: (403) 450-3700

713
One Palliser Square
Box 1646 Stalon M
Calgary, Alberta
Canada T2P 2L7

Tel: (403) 290-0588
Fax: (403) 290-0596

# *PIRAMID*

# Program User's Guide

## Version 10

**Confidential to**

**C-FER's Pipeline Program Participants**

**Copyright © 1995-98**
**C-FER Technologies Inc.**

**November 1998**

# Table of Contents

# Appendix C

## PIRAMID Inspection Version 1.0 - Model Input Definitions

### 1.0 Scope

The *PIRAMID* **Inspection** modules, **Onshore Inspection** for land-based pipelines and **Offshore Inspection** for subsea systems, optimize maintenance activities aimed at locating and eliminating time-dependent defects including corrosion, manufacturing cracks, SCC, dent/gouge features, and ground movement-induced deformations. Both modules calculate the probabilities of failure associated with a given pipeline for the status quo and evaluates changes in this probability that would result from implementing a user-defined maintenance strategy (*i.e.*, inspection and repair). They then carry out an analysis of the failure consequences and combines the results with the failure probabilities to calculate the risk levels associated with different candidate maintenance strategies. The results can then be used to compare different options and identify the optimal choice. Choices considered include inspection methods and tools, repair criteria and time between inspections.

This appendix describes the probability analysis part of the **Onshore Inspection** and **Offshore Inspection** influence diagrams. The consequence part of the diagrams is as described in Appendix A for onshore pipelines and Appendix B for offshore pipelines.

## 2.0 User Customization Options

The probability analysis portion of the influence diagrams used in ***PIRAMID* Inspection** can be customized to accommodate user-specific inspection methods, detection probability functions, excavation criteria, repair criteria, pipe failure models, parameter correlations, and parameter defaults. This can be achieved by programming external DLL's that include the necessary models and linking them to the program through a sequence of input dialogs (see Section 5.0 for a description of the formats of these functions). The linked code is then referred to as a specific *configuration* of ***PIRAMID* Inspection**.

A *configuration* is characterized by a set of inspection methods and a set of deterministic failure-cause-specific functions that will be used in subsequent simulations. It is noted that a configuration can address only one failure cause, however many configurations can be created for a given failure cause. For example, a corrosion-specific configuration may include high-resolution and low-resolution as possible inspection methods and may incorporate a particular corrosion failure model (B31G say). A different corrosion configuration may add coating damage surveys to the inspection methods and use a different failure model.

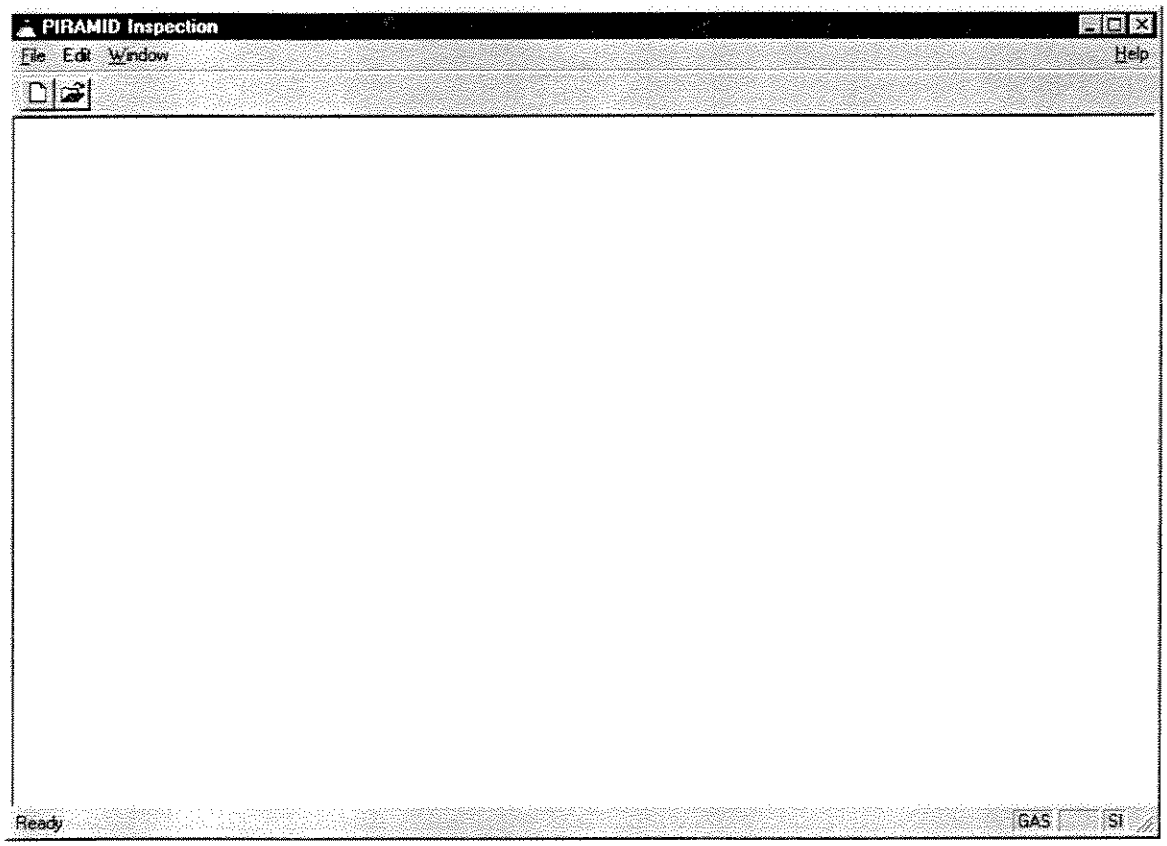
Once a configuration is defined, it can be saved and used to carry out as many *simulations* as required. Each simulation represents a particular set of choices for a particular pipeline segment. In a given simulation, the inspection methods included in the underlying configuration will be available for the user to choose from and the deterministic models built into the configuration will be used in the calculation. The diagram will conform to the requirements of the simulation by:

- creating new nodes representing the random parameters required by the user-defined functions; and
- expanding the list of line attributes to include ones that are required by the user-defined functions.

C-FER has defined default configurations for each time-dependent failure cause within the scope of the project. These include corrosion, manufacturing cracks, SCC, dent-gouge features and ground movements. Users can open these configurations and utilize them in various simulations, or define their own configurations.

## 3.0 Defining or Modifying a Configuration

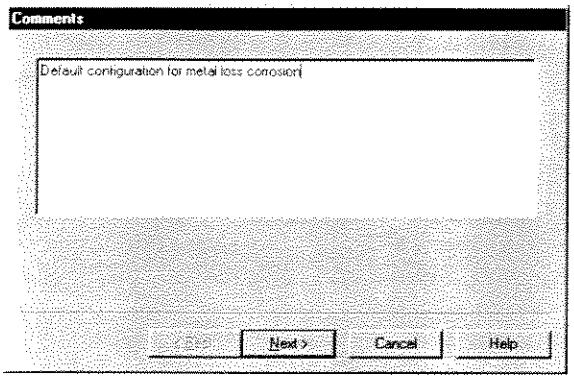
From the program main window, the *File⇨New* can be used to define a new configuration for any single time-dependent failure cause. Also the *File⇨Open⇨Configuration* command can be used to modify an existing configuration that has been previously saved. Configuration files have a file name extension of ".gpi" (**G**eneralized **PIRAMID** **I**nspection) (*e.g.*, MyConfiguration.gpi).




**Main Window**

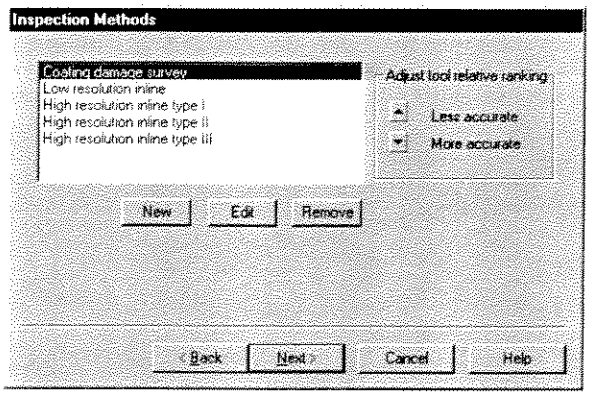
The first step in creating a new configuration is to provide a brief description. This description can be viewed from the simulation main window *Help⇨ About this simulation* menu option.

### Configuration Comments Dialog Box




The next step is to define the inspection methods that will be made available in simulations based on the configuration being defined. Inspection methods can be added, modified, or removed. The methods should be defined in order of increasing accuracy (least accurate method at the top of the list and least accurate method at the bottom).

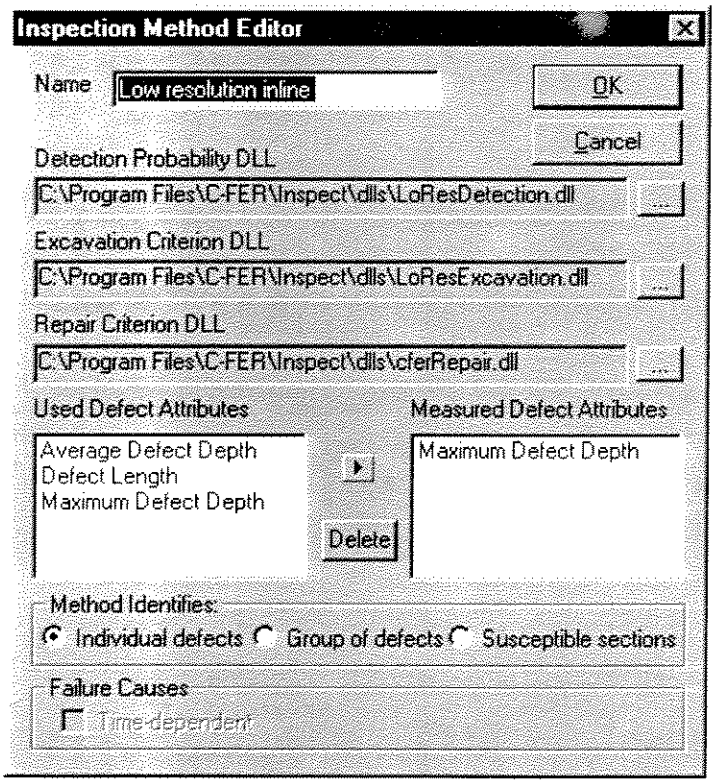
### Inspection Methods Dialog Box




Creating a new inspection method requires a collection of modules that characterize the associated Detection Probability, Excavation Criterion, and Repair Criterion that will be

used in connection with the method. The contents and formats of these functions are described in Section 5.0. The appropriate modules can be selected by clicking the button on the right hand side of the box containing the path to the function and using a standard Windows browser to select the function. It is noted that once the path specifying the location of the function within the user directories for a given configuration is defined, the function must exist at the exact same location when the program is run using that configuration.

### Inspection Method Edit or Dialog Box

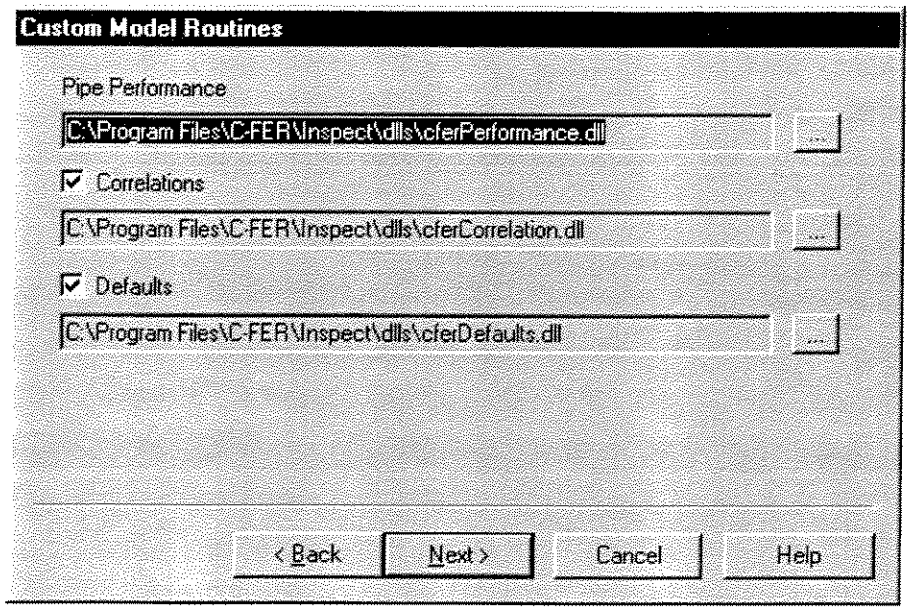



The inspection method is also characterized by the defect attributes it measures. The program will scan the above-mentioned functions to identify the defect attributes used, and will present these attributes to the user in the Used Defect Attributes box. The user can then specify the attributes measured by the current tool by highlighting them in the Used Defect Attributes box and clicking the arrow button to copy them into the Measured Defect Attributes box.

In addition radio buttons are available to specify the manner in which the inspection method identifies defects. The possible choices include single defects (*e.g.*, inline inspection) groups of defects (*e.g.*, coating damage surveys for corrosion) or susceptible sections (*e.g.*, targeted excavation methods).

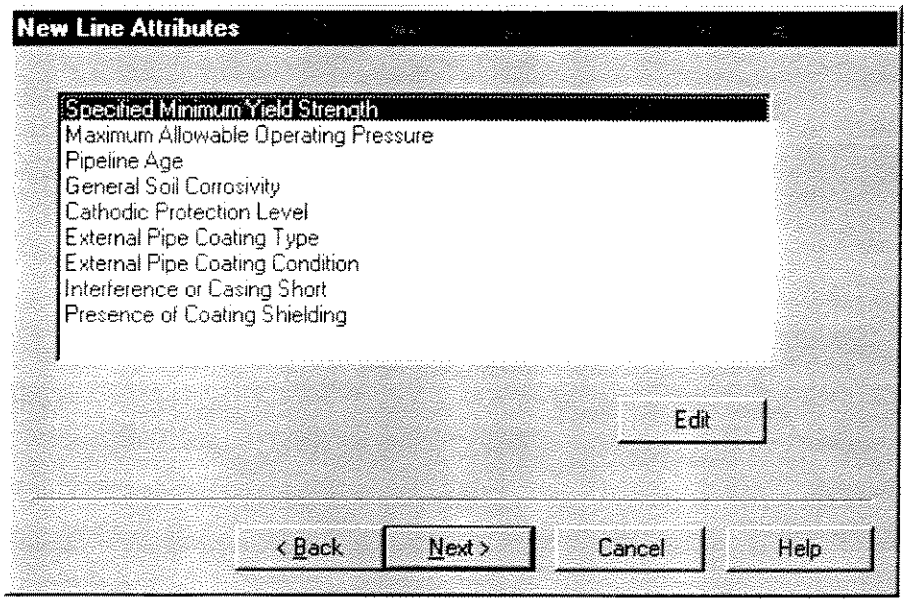
Once the inspection methods are defined, some additional user routines are required. The first is the Pipe Performance routine which calculates the time to failure and failure mode for a given defect. The Correlations and Defaults modules are optional and do not need to be defined. The Correlation module provides a means of relating two defect attributes so that one may be derived from the other. The Defaults module defines line-attribute-dependent default values for any number of influence diagram parameters. These values will be retrievable during a simulation, by clicking the *Defaults* button for the corresponding node. The Pipe Performance, Correlation, and Default functions can be selected by clicking the button on the right hand side of the box containing the path to the function. The contents and formats of these functions are described in Section 5.0. Again, it is important to note that once the path specifying the location of the function within the user directories for a given configuration is defined, the function must exist at the exact same location when the program is run using that configuration.

**Custom Model Routines Dialog Box**



If the selected modules require new line attributes that are not already known to the program, these attributes will be identified by the program and listed. The user must provide the program with some information regarding the type, units and description of each of these line attributes for use when the program requests input of the attribute value.

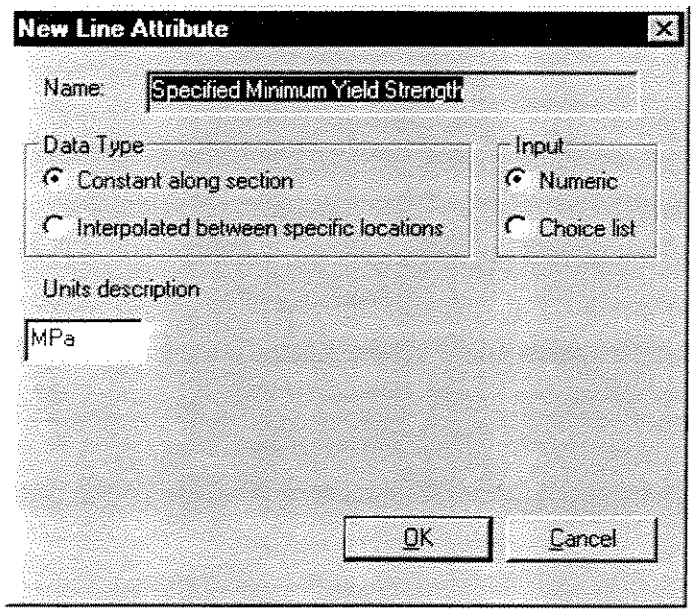
### New Line Attribute Dialog Box




The required attribute information can be defined by highlighting the attribute in the New Line Attributes dialog box and clicking the *Edit* button (or by double clicking the attribute).

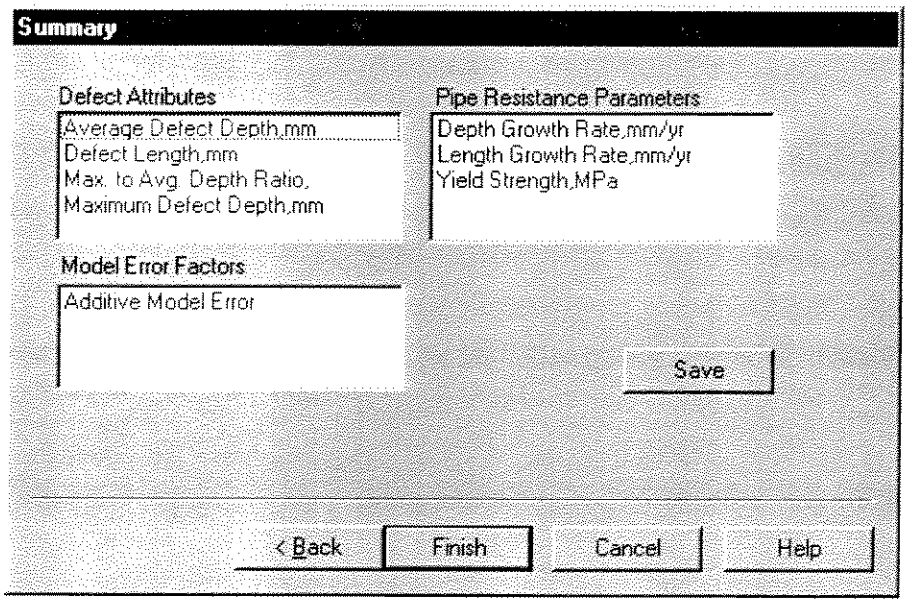
This action causes the New Line Attribute edit box to appear which contains radio buttons to specify whether the attribute is defined as a numeric value or as a choice from a list of character strings. For numeric parameters, the data can either be constant along a pipeline section (*e.g.*, diameter) or interpolated between the values given at specific locations (*e.g.*, pressure). The units associated with a numeric parameter should be defined. Line attributes described by a choice list are always constant along a section. The required input for this type is a set of text strings defining the possible values of the parameter (*e.g.*, Epoxy, Tape, or Asphalt for the attributes representing coating type).

### New Line Attribute Edit or Dialog Box



The last step of the configuration definition process is to view a summary of the parameters used. These parameters are identified by scanning the user defined functions and are classified in three categories: Defect Attributes, Pipe Resistance Parameters, and Model Error Factors. It this information is accurate, the configuration can be accepted by clicking the *Finish* button. The configuration may be saved at this stage (before clicking the *Finish* button). It can also be saved later using the *File⇨ Save Configuration* menu option). Configuration files are saved with a file name extension of ".gpi" (*e.g.*, MyConfiguration.gpi).

### Configuration Summary Dialog Box

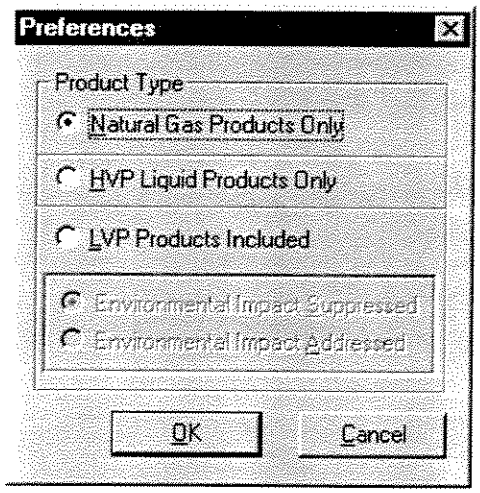


## 4.0 Running A Simulation

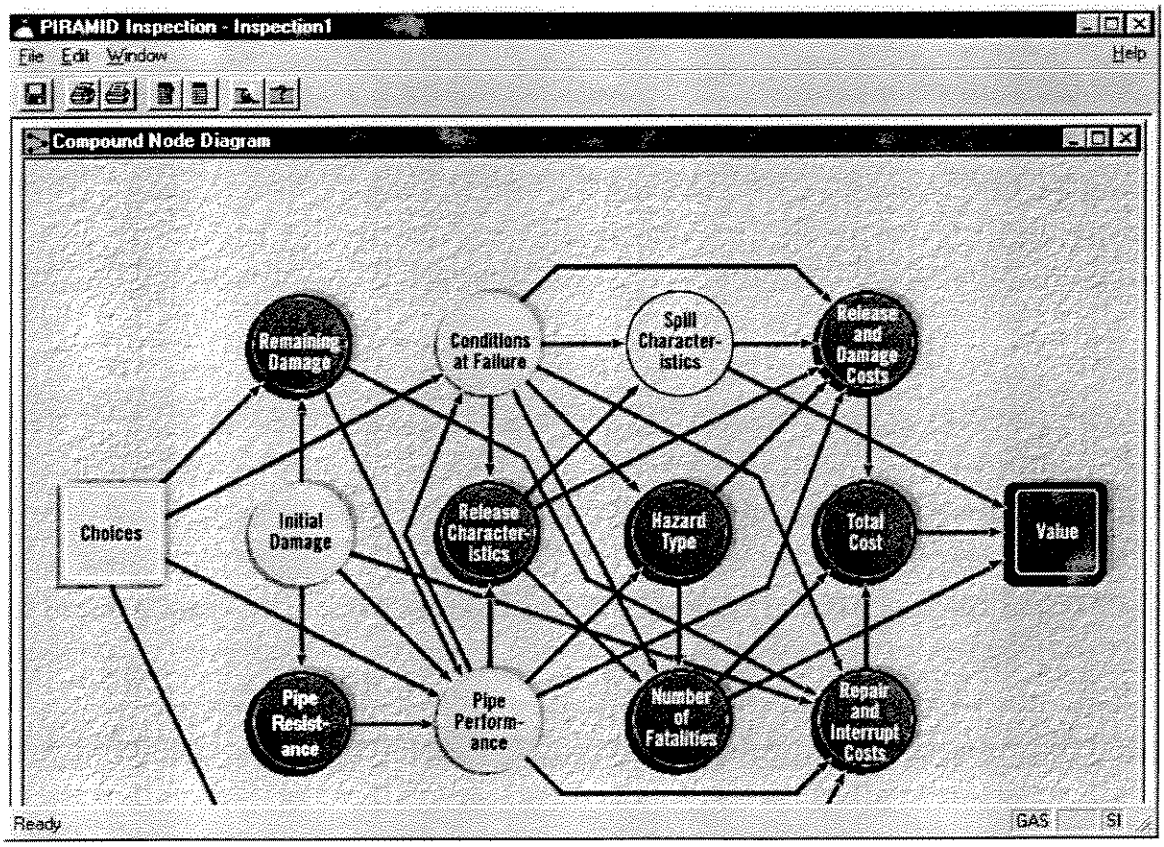
An existing simulation file can be accessed by using the *File⇨Open⇨Simulation* from the main program window or clicking on the *File Open* icon on the toolbar. Simulation files have a file name extension of ".pin" (*e.g.*, SegmentABC.pin). To start a new simulation based on an existing configuration the user must choose *File⇨New⇨Simulation*, or the *File New* icon on the tool bar. The program will then prompt for the configuration file on which the simulation is to be based and display the influence diagram for the simulation.

The first step in the simulation is to define the product type and analysis preference, with regard to environmental impact, through the Preferences dialog box. If liquids are selected the user must also specify whether or not environmental impact is considered.

**Preferences Dialog Box**



Once this is completed, the simulation main window will appear on the screen with the compound node influence diagram. The simulation main window has the same three basic components as the main program window – a menu bar, a tool bar, and a status bar. A calculation bar will appear within the status bar to display the calculation of the influence diagram during calculation.

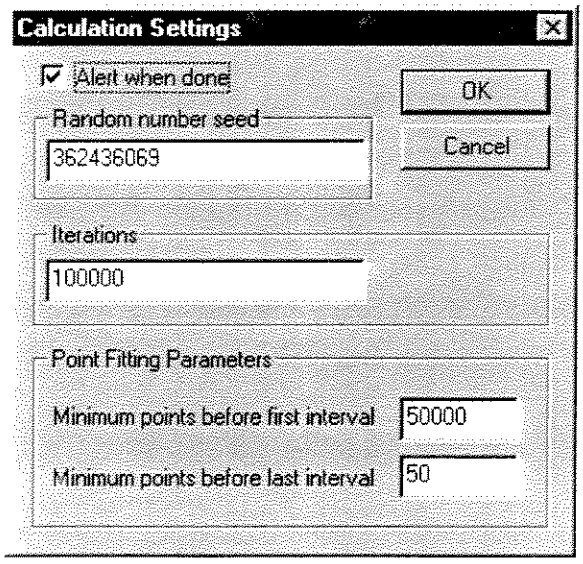



**Main Simulation Window**

The main Edit menu for PIRAMID Inspection has a Calculation Settings option (also accessible through the tool bar) which allows the user to modify the statistical parameters used in the simulation. These parameters include the seed used in random number generation (any positive integer can be used) and the number of iterations used in the simulation. It also includes two parameters that control the smoothing option for failure probability versus time plots (at the Performance at Defect node). Smoothing is based on a best fit probability distribution through the simulated points between the minimum and maximum values on the time scale. If the probability of failure corresponding to the minimum value on the time scale is based on a number of simulations that exceeds the "Minimum points before first interval", the simulated points are used directly without a

fit. On the other hand, if smoothing is required, the portion of the curve between the minimum and maximum values on the time scale will be used in the fit, provided that the number of simulations used in calculating the probability at the maximum value on the time scale is greater than the "Minimum points before last interval". If this condition is not met, the maximum value on the time scale will be increased (for fitting purposes only) until it is. Adjustments to these parameters may improve the fit for some problems. It is noted, however, that changing the Fitting Parameters does not result in an automatic re-calculation and, therefore, the user must force a re-calculation by invalidating a selected node and clicking the calculate button.

### Calculation Settings Dialog Box



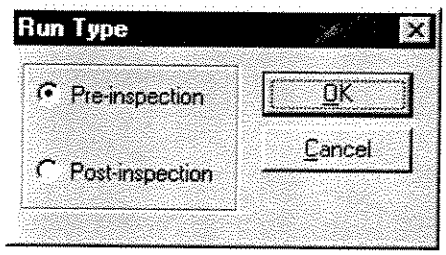
## 4.1 Choices Node Group

### 4.1.1 Inspection Method Node

The parameter of this node is represented by a discrete set of inspection methods that are considered in the run.

#### 4.1.1.1 Run Type

**Run Type Dialog Box**




The run type parameter is required upon first entry to the Inspection Method node. It identifies the type of inspection decision being made which has implications regarding choices that will be made available to the user, as well as the interpretation of the inputs for the Initial Defect nodes (see Sections 4.1.1.2 and 4.1.1.3). The possible run types are as follows:

**Pre-inspection.** A pre-inspection run assumes that no inspections have been carried out as part of the current maintenance event. In a pre-inspection run the program assumes that the user is deciding on whether or not an inspection is necessary and what inspection method to use. The program assumes that inputs defining defect density and/or coating density and all defect attributes represent estimates of the actual corrosion defect population in the pipeline.

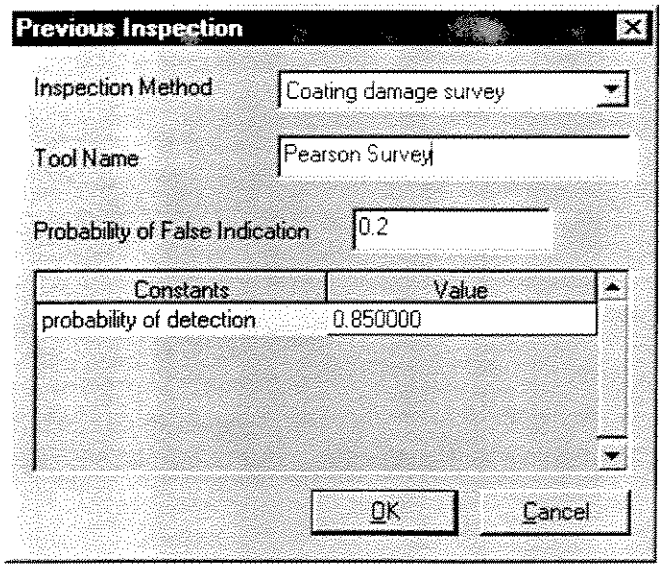
**Post-inspection.** A post-inspection run assumes that at least one inspection has been carried out as part of the current maintenance event. In a post-inspection run the program assumes that the user is deciding whether to carry out repairs based on the results of the most recent inspection results or inspect the line again using a more accurate method. The program assumes that inputs defining corrosion attributes (*i.e.*, defects per indication,

defect density, and defect attributes ) that are measurable by the inspection method used represent the values obtained from the inspection results.

### 4.1.1.2 Inspection Method Attributes

For a post-inspection run, the attributes of the inspection method that was used to collect the inspection information must be defined in order for the program to properly interpret the inspection results. The inspection method attributes dialog box allows the user to enter this information

**Inspection Method Dialog Box**



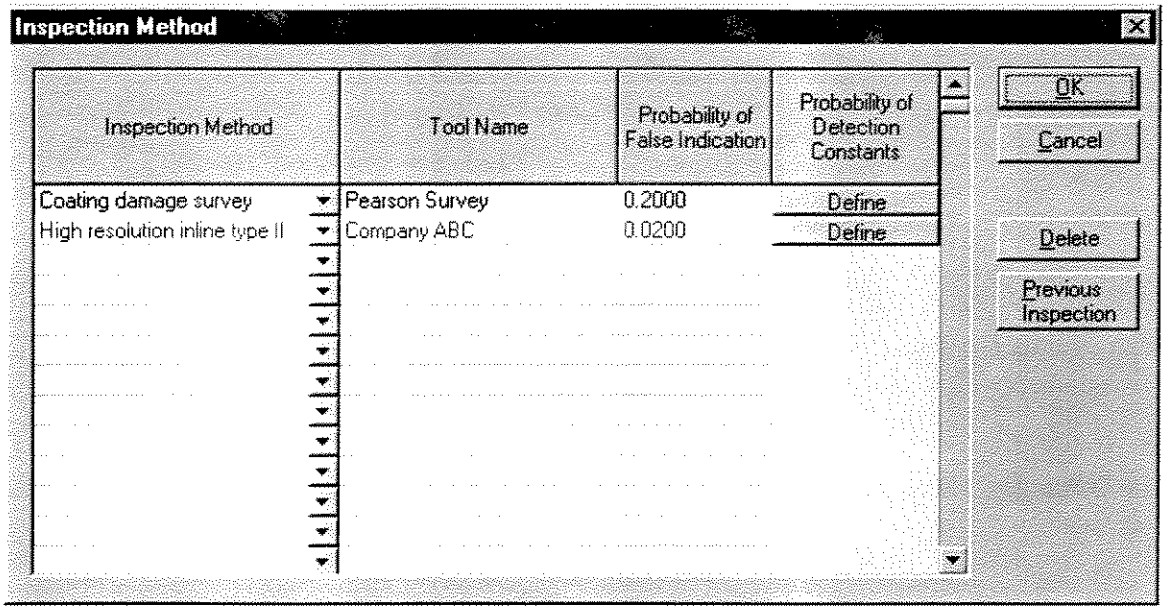

The parameters in the dialog box are defined as follows:

- **Inspection Method.** The inspection method previously used.
- **Tool Name.** This is a label that identifies inspection tool used.
- **Probability of False Indication.** Probability that an indication provided by the tool does not correspond to an actual defect.
- **Constants.** Any constants required by the detection module for the chosen tool type.

### 4.1.1.3 Inspection Method Choices

The inspection method choices include the different inspection tools that are being considered, as well as the status quo option (None) for a Pre-inspection run and the Repair option for a Post-inspection run (see also Section 3.2, *PIRAMID* Technical Reference Manual No. 6.x).

**Inspection Method Choices Dialog Box**




The dialog box contains inputs that define the name of each inspection method, the name of the inspection tool, the probability of false indication, and any required constants for the detection module.

- **Inspection Method.** The inspection methods available are those which were defined in the configuration used.

- **Tool Name.** This is a label that identifies inspection tools within a given inspection method category. For example, if three tools are being considered under H/R In-line inspection method category, the tool names may be Vendor 1, Vendor 2 and Vendor 3.

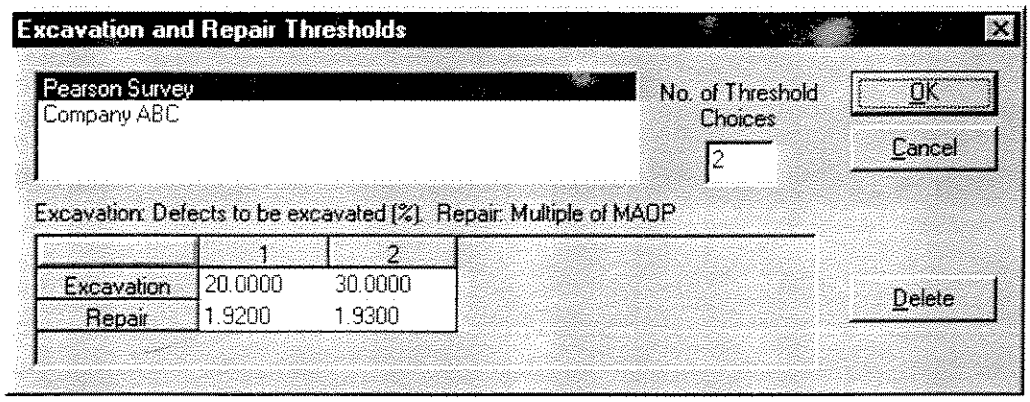
- **Probability of False Indication.** Probability that an indication provided by the tool does not correspond to an actual defect.

- **Probability of Detection Constants.** Any constants required by the detection module for the chosen tool type.

### 4.1.2 Excavation and Repair Thresholds Node

The node parameter is a specific pair of excavation and repair thresholds, defined for each tool. These thresholds are compared to the values returned by the excavation and repair criteria functions to determine whether a particular defect is excavated or repaired (see also Section 3.3 in *PIRAMID* Technical Reference Manual 6.x). The definitions of the threshold value is derived from the corresponding criterion (a user defined function) in the current configuration. For the example shown below, both the excavation and repair thresholds are defined as multiples of the maximum allowable operating pressure.

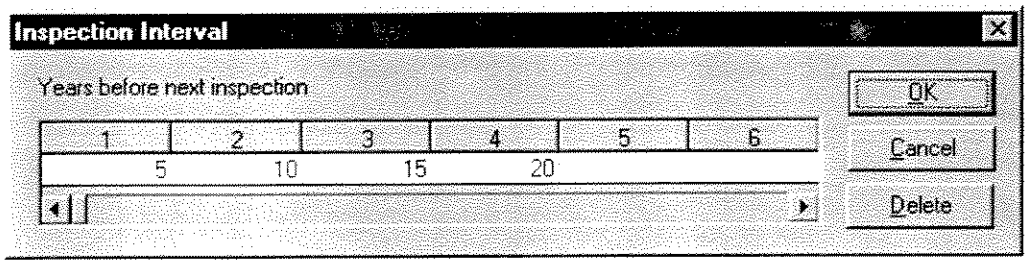
**Excavation and Repair Criteria Dialog Box**



### 4.1.3 Inspection Interval Node

The parameter of this node is the time interval (in years) between the present inspection and the next inspection (see also Section 3.4, *PIRAMID* Technical Reference Manual No. 6.x).

#### Inspection Interval Dialog Box



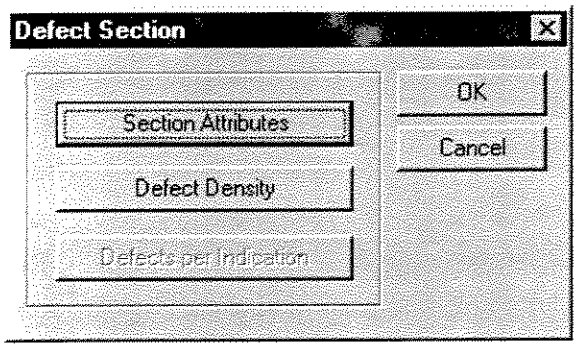

The user can specify the inspection intervals to be considered by the program by typing them in the appropriate cells. The inspection intervals need not be equally spaced, although they must be specified in increasing order.

## 4.2 Initial Damage Node Group

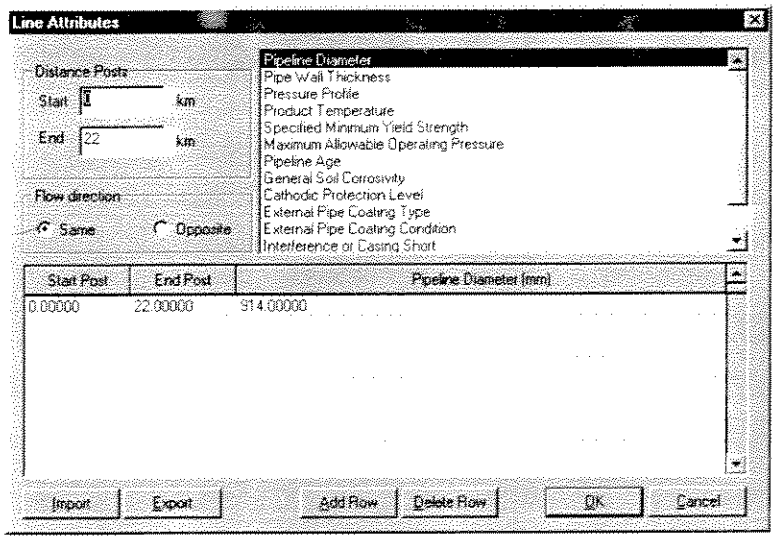
### 4.2.1 Defect Section Node

The node parameter is the section of pipeline that contains the defect, where a section is defined as a length of pipeline for which all related attributes are uniform (see also Section 4.2.1, *PIRAMID* Technical Reference Manual No. 6.x).

#### Defect Section Dialog Box




#### Defect Section Line Attributes Dialog Box

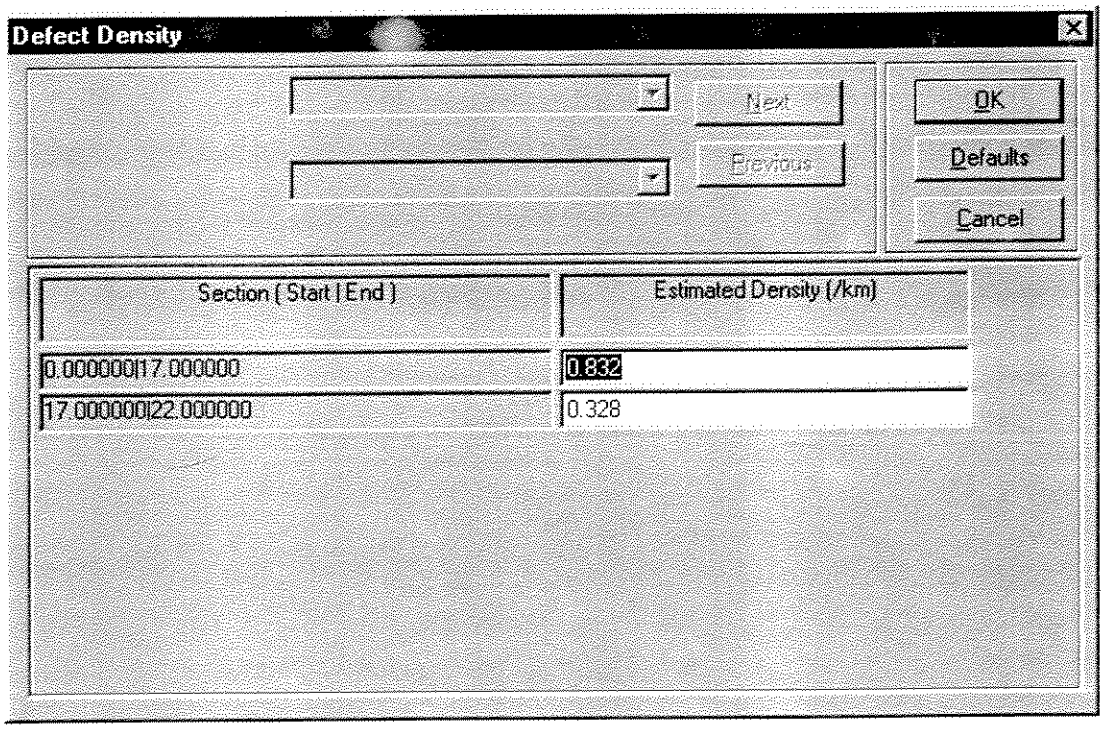



The user must specify the value of each pipeline attribute displayed in the input dialog box. Attributes can be defined interactively via the input window and embedded

spreadsheets or by importing a predefined segment data file (see User's Guide page 20 and Appendix D). Each attribute must be defined along the entire length of the pipeline segment being analyzed. The line attributes required for the default configurations provided by C-FER for different failure causes are described in *PIRAMID* Technical Reference Manual No. 6.x as follows:

| Model Type | Onshore | Offshore |
|---|---|---|
| • Corrosion | Table A.1 | Table A.1 (offshore addendum) |
| • Manufacturing cracks | Table B.1 | to be developed |
| • SCC | Table C.1 | to be developed |
| • Dent/gouges | Table D.1 | to be developed |
| • Ground movements | Table E.1 | to be developed |

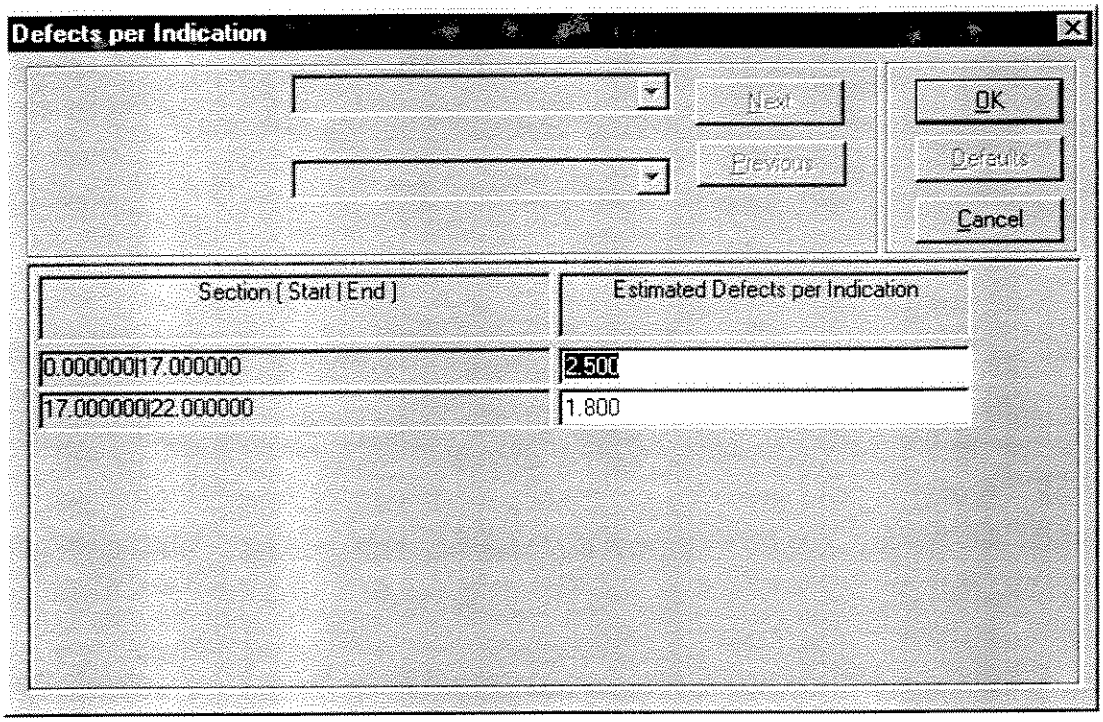
**Defect Density Input Dialog Box**




Defect density is defined as the number of defects per unit length of the pipeline. It is defined separately for each defect section. For a pre-inspection run this parameter is defined as an estimate by the user since measurement of its value would be readily

available. For a post-inspection run the measured defect density would be input directly. For a method that detects groups of defects (*e.g.*, coating damage survey), the defect density must calculated by multiplying the number of defect groups (*e.g.*, the number of coating damage defects) by an estimate of the number of defects per group.

### Defects Per Indication Dialog Box



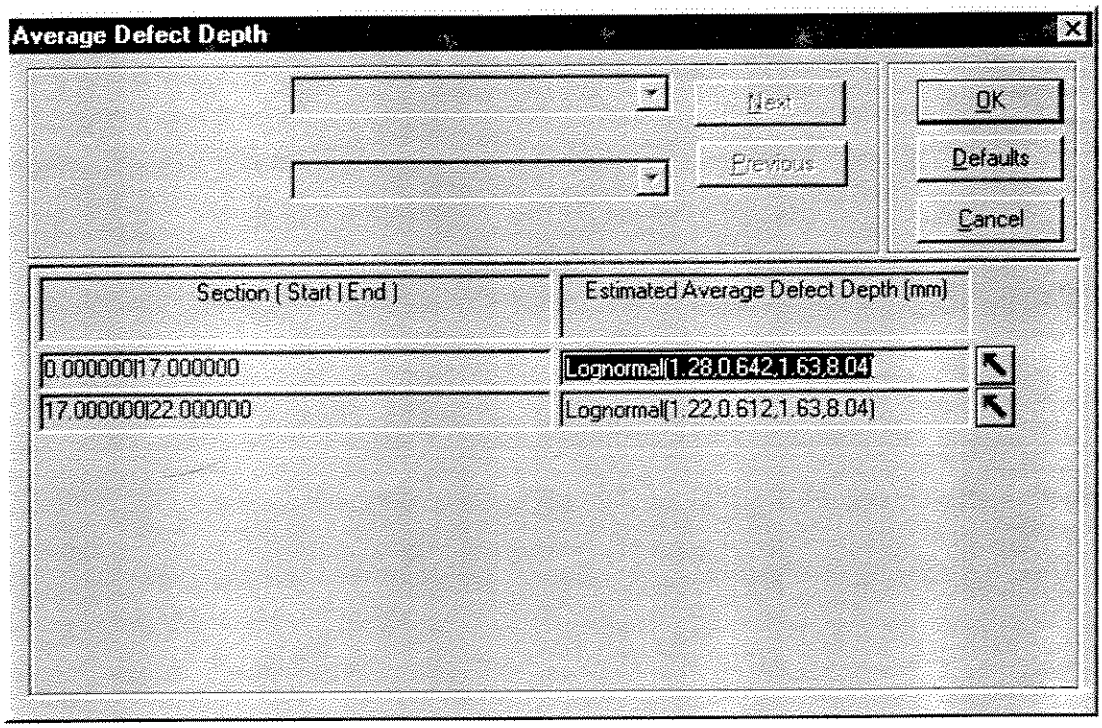

Defects per indications are defined as the number of defects in a given indication detected by the inspection method (*e.g.*, the average number of corrosion pits in a coating damage feature for a coating damage survey). It is defined separately for each defect section. Defects per indication is only required if an inspection method, characterized as measuring multiple defects, has been carried out or is contemplated as a possible choice.

### 4.2.2 Defect Attribute Nodes

The parameters of the nodes in this group represent the defect attributes required for the current configuration (see Section 4.2.2 in *PIRAMID* Technical Reference Manual No. 6.x). Definitions of the defect attributes required for the default configurations associated with different failure causes are given in *PIRAMID* Technical Reference Manual No. 6.x as follows:

- Corrosion Section A.3.1
- Manufacturing cracks Section B.3.1
- SCC Section C.3.1
- Dent/gouges Section D.3.1
- Ground movements Section E.3.1

**Sample Defect Attribute Dialog Box**




The defect attribute input dialog allows the user to define the probability distribution of the defect attribute for each pipeline section. The probability distribution can be modified

using the Distribution Picker. Definition of the input parameter to this node depends on the run type and inspection method used as follows:

- **Estimated Defect Attribute.** The defect attribute input represents a user estimate in a pre-inspection run or a post-inspection run where the previous inspection method does not measure the attribute. This estimate can be based on inspection information for similar lines and/or subjective judgment. For lines that were inspected as part of past maintenance events, this parameter can also be estimated from the results of past inspections, adjusted to take into account repairs that have been carried out and defect growth during the interval between inspections.

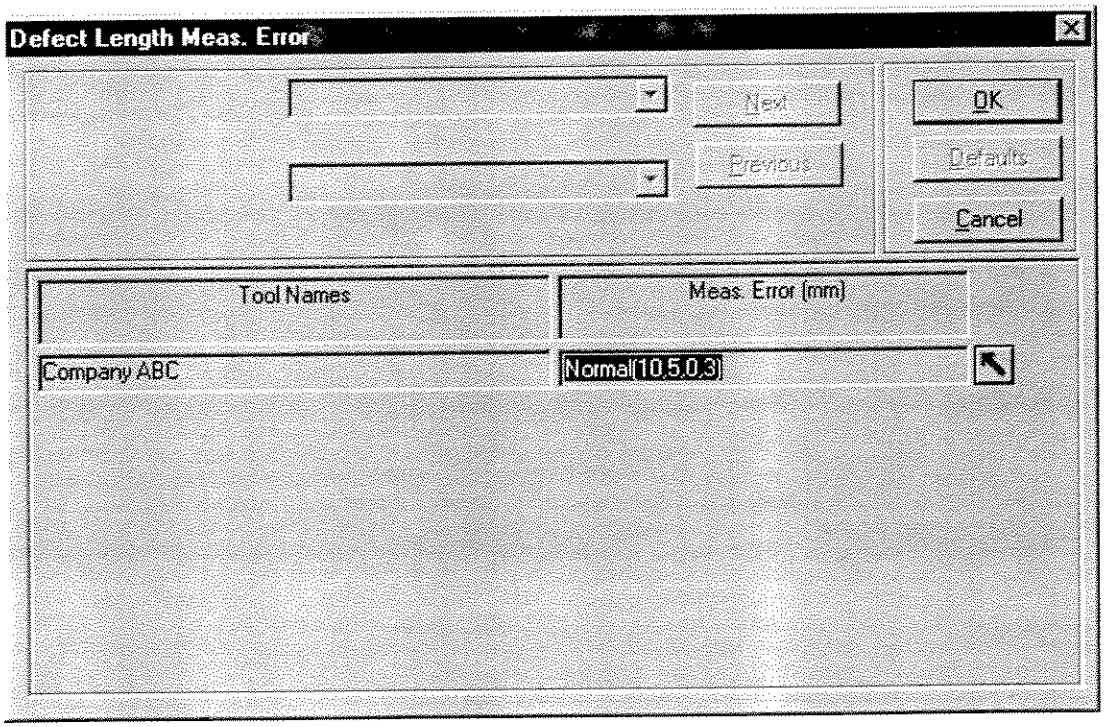
- **Measured Defect Attribute**. In a post-inspection run where the inspection method used measures the attribute in question, the attribute input is defined based on the measurement. The required input probability distribution can be obtained by fitting an appropriate probability distribution to the data points representing the depths of the individual defects detected by the inspection tool. This fitting analysis must be carried out independently of *PIRAMID*.

## 4.3 Remaining Damage Node Group

### 4.3.1 Defect Attribute Measurement Error Nodes

The node parameter is the measurement error associated with a defect attribute value provided by the inspection tool (see Section 5.2, *PIRAMID* Technical Reference Manual No. 6.x). The input for this node is required for all inspection methods that measure the attribute in question.

**Defect Attribute Measurement Error Dialog Box**




The defect attribute measurement error input dialog allows the user to define the corresponding probability distribution for each inspection tool. The probability distribution can be modified using the Distribution Picker.

### 4.3.2 Defect Attribute After repair Node

These nodes contain the calculated distributions of the defect attributes after defect repairs have been carried out (see Section 5.6, *PIRAMID* Technical Reference Manual No. 6.x). No user input is required to specify these nodes.

### 4.3.3 Repair Action Node

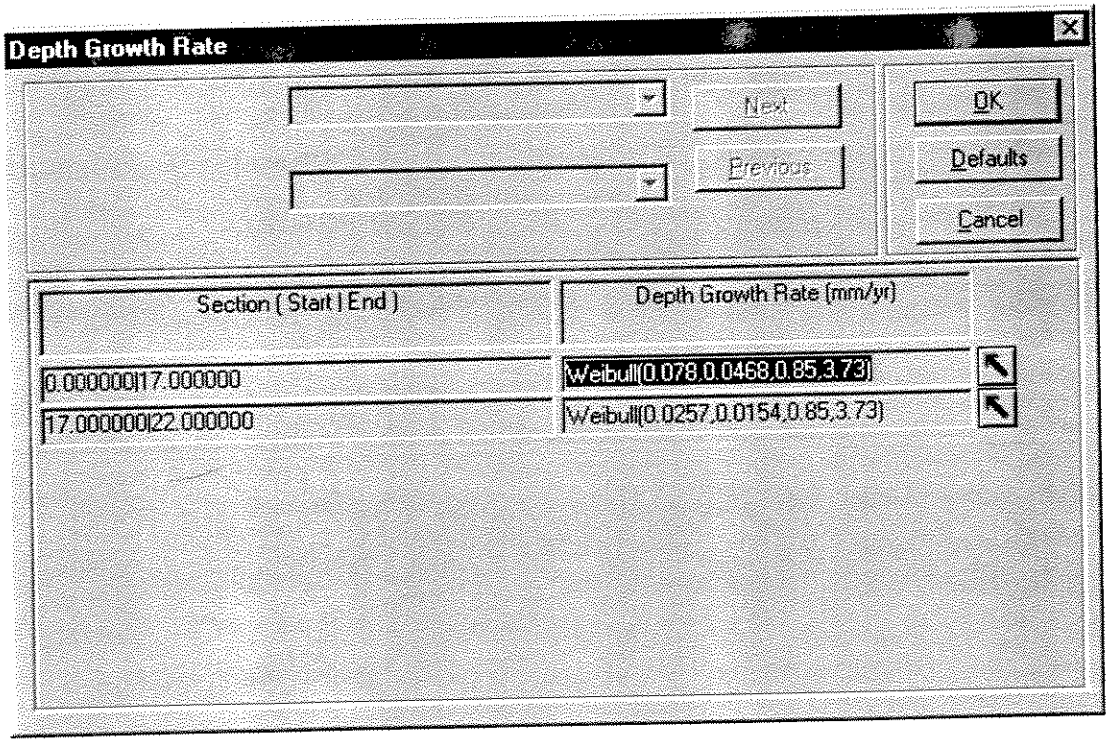
The node parameter is the repair action implemented for a specific corrosion defect (see Section 5.4, *PIRAMID* Technical Reference Manual No. 6.x). No user input is required to specify this node.

## 4.4 Pipe Resistances Node Group

The parameters of the nodes in this group represent the pipe resistance variables required for the current configuration (see Section 6.2 in *PIRAMID* Technical Reference Manual No. 6.x). Definitions of the required parameters for the default configurations associated with different failure causes are given in *PIRAMID* Technical Reference Manual No. 6.x as follows:

- Corrosion Section A.3.2
- Manufacturing cracks Section B.3.2
- SCC Section C.3.2
- Dent/gouges Section D.3.2
- Ground movements Section E.3.2

### Pipe Resistance Node Dialog Box




The input dialog for each pipe resistance parameter allows the user to define the corresponding probability distribution for each pipeline section. The probability distribution can be modified using the Distribution Picker.

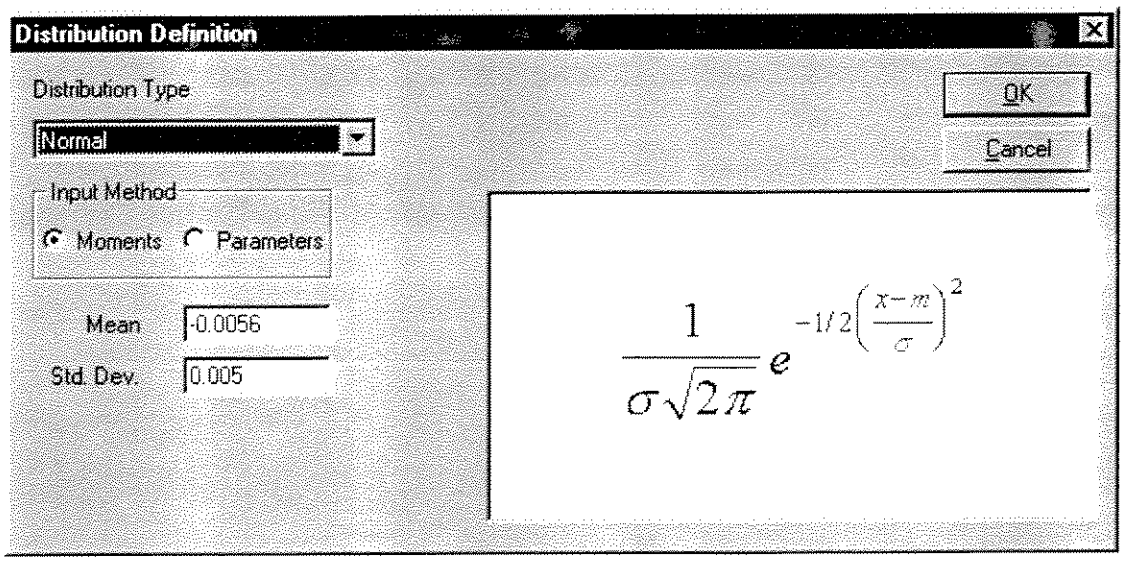
## 4.5 Performance Node Group

### 4.5.1 Model Error Nodes

The parameters of the nodes in this group represent model error factors required for the current configuration (see Section 7.2 in *PIRAMID* Technical Reference Manual No. 6.x). Definitions of the required parameters for the default configurations associated with different failure causes are given in *PIRAMID* Technical Reference Manual No. 6.x as follows:

- Corrosion Section A.3.3
- Manufacturing cracks Section B.3.3
- SCC Section C.3.3
- Dent/gouges Section D.3.3
- Ground movements Section E.3.3

**Model Error Node Dialog Box**




The input dialog for each model error factor allows the user to define the corresponding probability distribution. The probability distribution can be modified using the Distribution Picker.

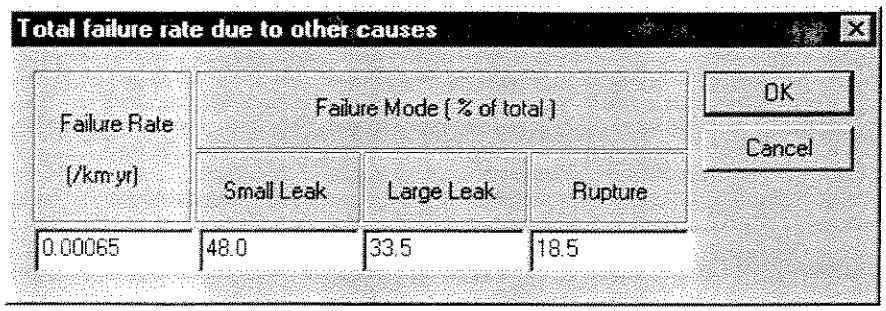
### 4.5.2 Performance at Defect Node

The node parameter is the performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline at a randomly selected defect during the inspection interval (see Section 7.3, *PIRAMID* Technical Reference Manual No. 6.x). No user input is required to specify this node.

### 4.5.3 Segment Performance Node

The node parameter is the performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline segment during the inspection interval (see Section 7.4, *PIRAMID* Technical Reference Manual No. 6.x). The parameter is calculated by combining the calculated failure rates for the cause considered in the current configuration with the failure rate and proportion of failure type for all other failure mechanisms.

#### Segment Performance Dialog Box




The segment performance input dialog box allows the user to define the annual failure rates due to all other causes. In the column titled **Failure Rate** the user must input the failure rate per km · year. In the columns titled **Failure Mode** the user must define the relative frequency in % (*i.e.*, a number between 0 and 100) of each failure mode. The total relative frequencies for small leak, large leak and rupture must add up to 100.

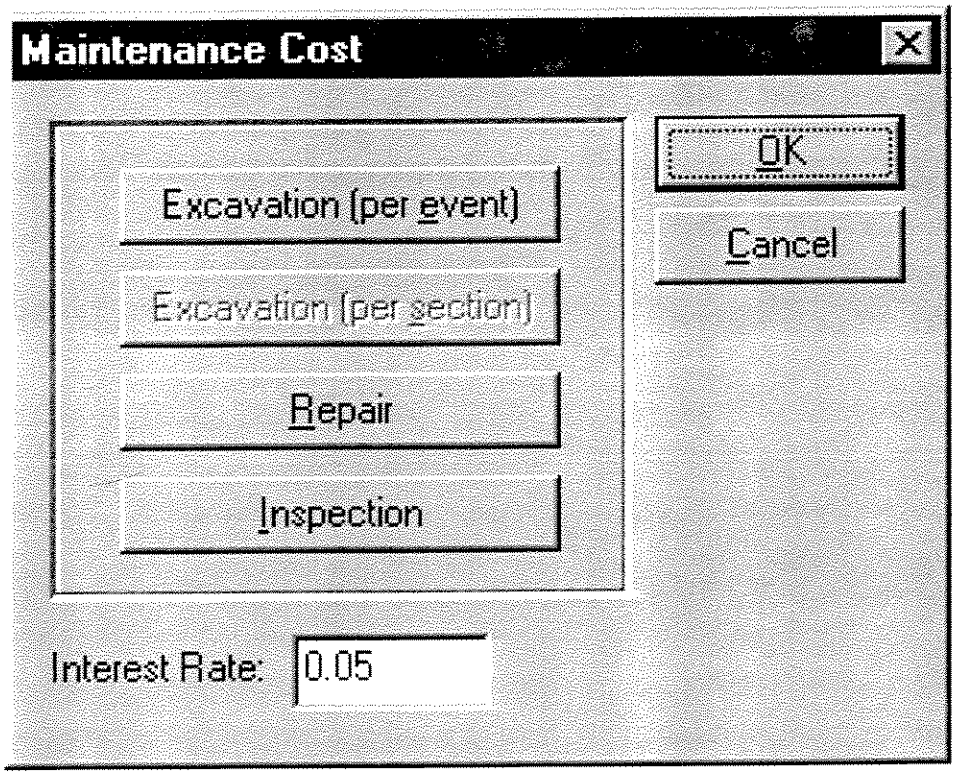
## 4.6 Repair and Interruption Cost Node Group

### 4.6.1 Maintenance Cost Node

The node parameter is the total annual maintenance cost for the whole pipeline segment in present value currency (see Section 9.2, *PIRAMID* Technical Reference Manual No. 6.x). The maintenance cost is calculated from the number of excavations and repairs as calculated from previous nodes and the inspection and unit repair/excavation costs defined as input to this node.

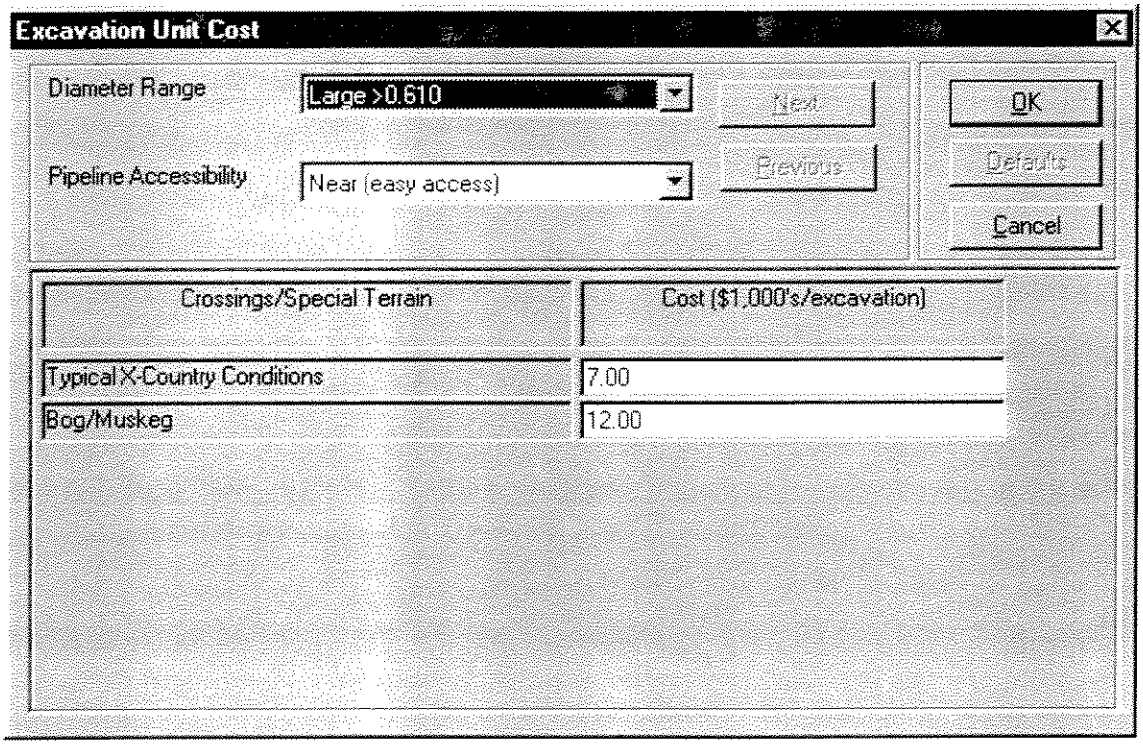
The initial Maintenance Cost dialog box allows the user to input the real interest rate and access other dialog boxes to define the required costs. The real interest rate is defined as the difference between the actual interest rate and the inflation rate.

**Maintenance Cost Dialog Box**



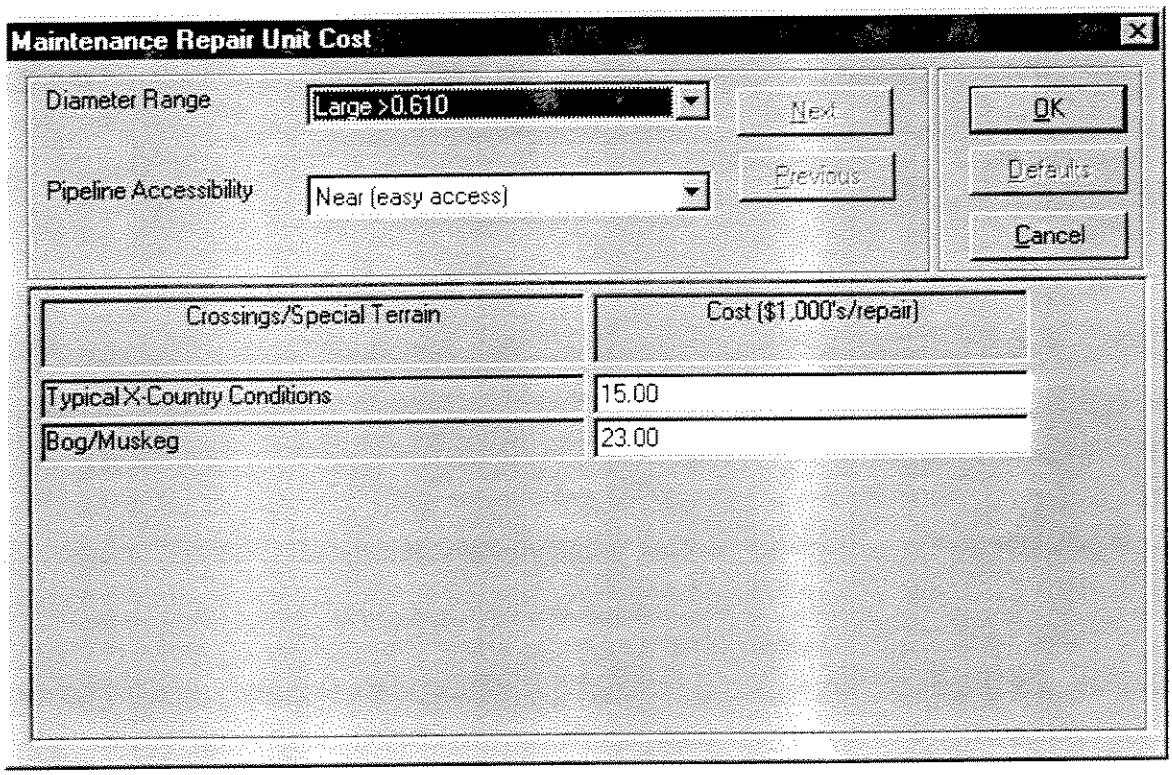
The user must define the unit cost per single excavation for all combinations of diameter range, accessibility, and crossing/special terrain type. It is assumed that an excavation implies at certain minimum amount of investigation/rehabilitation work (*e.g.*, coating repair, or defect grinding). The costs of these activities are included in the excavation cost.

### Excavation Unit Cost Dialog Box



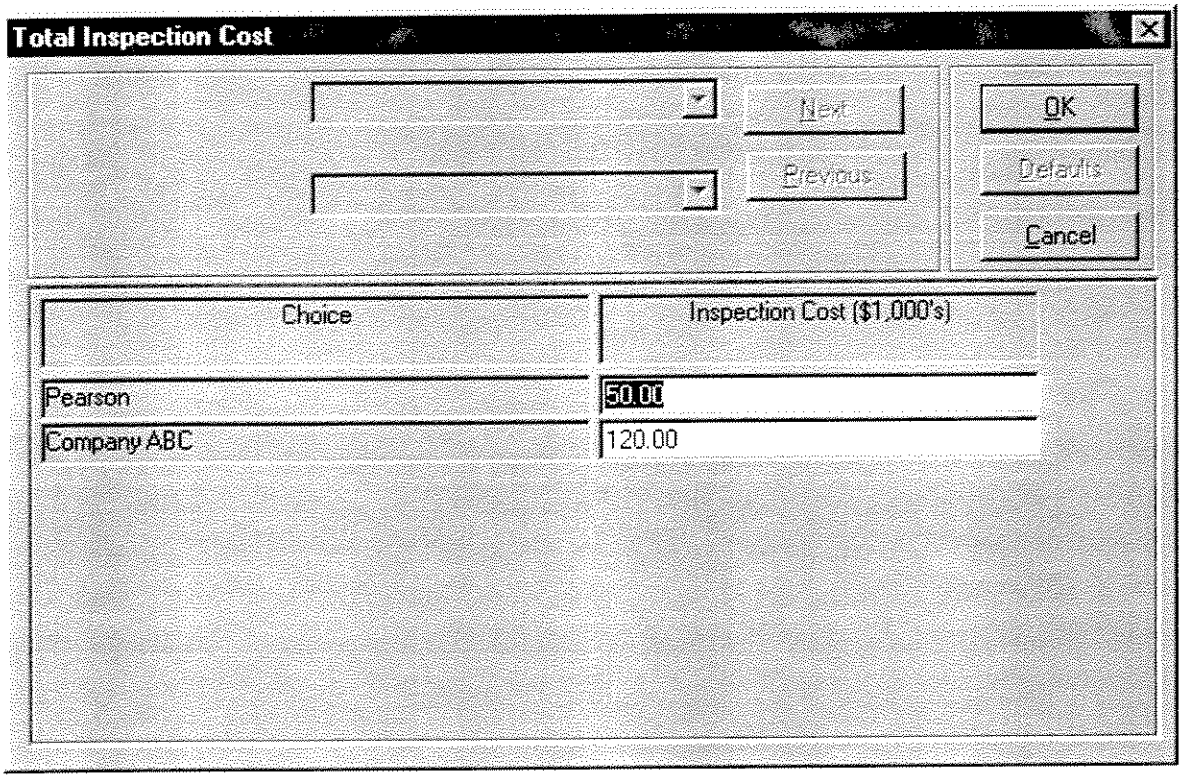
The user must define the unit cost per single pipe body repair for all combinations of diameter range, accessibility, and crossing/special terrain type. The repair cost does not include the cost of coating repair since this parameter is included as part of the excavation cost.

### Maintenance Repair Unit Cost Input Dialog Box



The total inspection cost for the entire segment is defined for the different inspection choices considered.

### Total Inspection Cost Dialog Box



# 5.0 User Module Specification

## 5.1 Module Functions

Each user-defined module performs specific functions that are required by the ***PIRAMID*** **Onshore/Offshore-Inspection** program. The formats of the functions are explained in Section 5.2.

### 5.1.1 Detection Module

The function of the detection module is to calculate the probability of detection for a given defect. The **DetectionProbability** function performs this calculation and thus is a required part of the module. In determining the probability, the function has access to any of the line attributes or defect attributes. This function may also create a number of user-specified constants that are used to distinguish between the accuracy of different tools within a given inspection method. The program will request the values of these constants for different inspection tools at run time. For more detail on the use and interpretation of these constants please see Section 3.2.3 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **GetUsedLineAttributes**, **GetUsedDefectAttributes**, **GetUserConstants**

### 5.1.2 Excavation Module

The function of the excavation module is to calculate the excavation criterion for a specific defect. The **ExcavationCriterion** has access to all line attributes and defect attributes. The values returned by this function are compared to the excavation threshold (which are defined by the user at run time) to determine whether the defect will be excavated. For more detail on the use and interpretation of these constants please see Section 3.3 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **GetUsedLineAttributes**, **GetUsedDefectAttributes**, **GetCalculationResultInformation**

### 5.1.3 Repair Module

The function of the repair module is to calculate the repair criterion for a specific defect. The **RepairCriterion** has access to all line attributes and defect attributes. The values returned by this function are compared to the repair threshold (which are defined by the user at run time) to determine whether the defect will be repaired. For more detail on this function please see Section 3.3 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **GetUsedLineAttributes**, **GetUsedDefectAttributes**, **GetCalculationResultInformation**

### 5.1.4 Performance Module

The function of the performance module is to calculate the time-to-failure and the failure mode of a specific defect. The input to the **PipePerformance** function may include line attributes, defect attributes, pipe resistance parameters, and model error factors. For more detail on this function please see Section 7.3 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **GetUsedLineAttributes**, **GetUsedDefectAttributes**, **GetUsedMechanicalProperties**, **GetUsedModelErrorFactors**

### 5.1.5 Correlation Module

The correlation module is an optional module that can be used to describe a relationship between any pair of defect attributes. The **Correlation** function is a two-way function that returns the value of either of the correlated parameters given the value of the other. This function has access to line attributes, defect attributes, and pipe resistance parameters. For more detail on this function please see Section 4.2.2.3 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **GetCorrelatedDefectAttributes**, **GetUsedLineAttributes**, **GetUsedDefectAttributes**, **GetUsedMechanicalProperties**

### 5.1.6 Defaults Module

The defaults module is an optional module provided as a means for generating default values for the defect density, as well as selected defect attributes, pipe resistance attribute, and model error factors. The **GetDefaultMappings** function has access to line attributes in order to permit the default values to be line attribute dependent. For more detail on this function please see Sections 4.3 and 6.2 of *PIRAMID* Technical Reference Manual No. 6.x.

See also: **DefectDensity**, **GetUsedLineAttributes**

## 5.2 Example User Modules

### 5.2.1 Detection Module

```
#include "UserDLL.h"
#include <math.h>

// C-FER Detection Probability model based on Hi-Resolution (2) In-line
// inspection tool for detecting corrosion defects

//================================

const DEFECT_ATTRIBUTE DefectAttributes[] = {
      { "Average Defect Depth","mm","" },
      { "", "" }
};

extern "C"
const DEFECT_ATTRIBUTE* GetUsedDefectAttributes()
{
    return DefectAttributes;
}

//================================

const USER_CONSTANT UserConstants[] = {
      "p(detect) at havg = 1mm",
      ""
};

extern "C"
const USER_CONSTANT* GetUserConstants()
{
    return UserConstants;
}

//================================

extern "C"
int DetectionProbability( const double* lineAttributes, const double*
defectAttributes,  const double* userConstants, double* probability )
{
    double pd = userConstants[ 0 ];
    double averageDefectDepth = defectAttributes[ 0 ];

    double q = -log( 1.0 - pd );

    if ( q > 0 )
        {
        *probability = 1.0 - exp( -q * averageDefectDepth );
        return USER_SUCCESS; // successful
        }
    *probability = 0.0;

    return USER_FAILURE; // bad value

}
```

### 5.2.2 Excavation Module

```
#include "UserDLL.h"
#include <math.h>

// C-FER Excavation Criteria model based on Hi-Resolution (2) In-line
// inspection tool

//================================

const DEFECT_ATTRIBUTE DefectAttributes[] = {
      { "Average Defect Depth", "mm", "" },
      { "Defect Length", "mm", "" },
      { "", "" }
};

extern "C"
const DEFECT_ATTRIBUTE* GetUsedDefectAttributes()
{
    return DefectAttributes;
}

//================================

const LINE_ATTRIBUTE LineAttributes[] = {
      "Pipeline Diameter",
      "Pipe Wall Thickness",
      "Specified Minimum Yield Strength",
      "Maximum Allowable Operating Pressure",
      ""
};

extern "C"
const LINE_ATTRIBUTE* GetUsedLineAttributes()
{
    return LineAttributes;
}

//================================

const RESULT_INFORMATION ResultInformation[] = {
    { eRESULT_COMPARISON_LESS_THAN_THRESHOLD, "", "Multiple of MAOP" }
};

extern "C"
const RESULT_INFORMATION* GetCalculationResultInformation()
{
    return ResultInformation;
}
//================================

extern "C"
int ExcavationCriterion( const double* lineAttributes, const double*
defectAttributes, double* result )
{
    ( calculation happens here )

    return USER_SUCCESS; // successful
}
```

### 5.2.3 Repair Module

```
#include "UserDLL.h"
#include <math.h>

// C-FER Repair Criteria model

//=================================

const DEFECT_ATTRIBUTE DefectAttributes[] = {
      { "Maximum Defect Depth", "mm", "" },
      { "Average Defect Depth", "mm", "" },
      { "Defect Length", "mm", "" },
      { "", "" }
};

extern "C"
const DEFECT_ATTRIBUTE* GetUsedDefectAttributes()
{
    return DefectAttributes;
}

//=================================

const LINE_ATTRIBUTE LineAttributes[] = {
      "Pipeline Diameter",
      "Pipe Wall Thickness",
      "Specified Minimum Yield Strength",
      "Maximum Allowable Operating Pressure",
      ""
};

extern "C"
const LINE_ATTRIBUTE* GetUsedLineAttributes()
{
    return LineAttributes;
}

//=================================

const RESULT_INFORMATION ResultInformation[] = {
    { eRESULT_COMPARISON_LESS_THAN_THRESHOLD, "", "Multiple of MAOP" }
};

extern "C"
const RESULT_INFORMATION* GetCalculationResultInformation()
{
    return ResultInformation;
}
//=================================

extern "C"
int RepairCriterion( const double* lineAttributes, const double*
defectAttributes, double* result )
{
    ( calculation happens here )

    return USER_SUCCESS; // successful
}
```

### 5.2.4 Performance Module

```
#include "UserDLL.h"
#include <math.h>

//================================

const DEFECT_ATTRIBUTE DefectAttributes[] = {
      { "Average Defect Depth", "mm", "" },
      { "Defect Length", "mm", "" },
      { "Maximum Defect Depth", "mm", "" },
      { "", "" }
};

extern "C"
const DEFECT_ATTRIBUTE* GetUsedDefectAttributes()
{
    return DefectAttributes;
}

//================================

const LINE_ATTRIBUTE LineAttributes[] = {
      "Pipeline Diameter",
      "Pipe Wall Thickness",
      "Pressure Profile",
      ""
};

extern "C"
const LINE_ATTRIBUTE* GetUsedLineAttributes()
{
    return LineAttributes;
}

//================================

const MECHANICAL_PROPERTY MechanicalProperties[] = {
      { "Depth Growth Rate", "mm/yr", "" },
      { "Length Growth Rate", "mm/yr", "" },
      { "Yield Strength", "MPa", "" },
      { "", "" }
};

extern "C"
const MECHANICAL_PROPERTY* GetUsedMechanicalProperties()
{
    return MechanicalProperties;
}

//================================

const MODEL_ERROR_FACTOR ModelErrorFactors[] = {
      { "Additive Model Error", "", "" },
      { "", "" }
};

extern "C"
const MODEL_ERROR_FACTOR* GetUsedModelErrorFactors()
{
```

```
    return ModelErrorFactors;
}

//=============================

extern "C"
int PipePerformance( const double* lineAttributes, const double*
defectAttributes, const double* mechanicalProperties, const double*
modelErrorFactors, double* timeToFailure, FAILURE_TYPE* failureMode )
{
    double diameter = lineAttributes[ 0 ];
    double thickness = lineAttributes[ 1 ];
    double averageDefectDepth = defectAttributes[ 0 ];
    double maximumDefectDepth = defectAttributes[ 2 ];
    double depthGrowthRate = mechanicalProperties[ 0 ];
    double additiveError = modelErrorFactors[ 0 ];

    ( calculation happens here )
    return USER_SUCCESS; // successful!
}
```

### 5.2.5 Defaults Module

```
#include "UserDLL.h"
#include <math.h>

// default functions to export
extern "C" _USERDLL
int DefectDepthDefault( const double*, double*, double*, double*,
double*, DISTRIBUTION_TYPE* );

( declare all others as well … )

// C-FER Defaults
//==============================
const LINE_ATTRIBUTE LineAttributes[] = {
      "Pipe Wall Thickness",
      "Product Temperature",
      "Pipeline Age",
      "General Soil Corrosivity",
      "Cathodic Protection Level",
      "External Pipe Coating Type",
      "External Pipe Coating Condition",
      "Specified Minimum Yield Strength",
      "Interference or Casing Short",
      "Presence of Coating Shielding"
      ""
};

extern "C"
const LINE_ATTRIBUTE* GetUsedLineAttributes()
{
     return LineAttributes;
}

//==============================
// Mapping of entities to functions which calculate the defaults

const DEFAULT_MAPPING DefaultMappings[] = {
      { "Average Defect Depth", "DefectDepthDefault" },
      { "Defect Length", "DefectLengthDefault" },
      { "Yield Strength", "YieldStrengthDefault" },
      { "Depth Growth Rate", "GrowthRateDefault" },
      { "Max. to Avg. Depth Ratio", "DepthRatioDefault" },
      { "Additive Model Error", "AdditiveModelErrorDefault" },
      { "", "" }
};

extern "C"
const DEFAULT_MAPPING* GetDefaultMappings()
{
      return DefaultMappings;
}
extern "C"
int DefectDepthDefault( const double* lineAttributes, double* mean,
double* stdev, double* p1, double* p2, DISTRIBUTION_TYPE* distType )
{
    ( calculation happens here )
    return USER_SUCCESS;
}
```

## 5.3 Function Reference

The following describes the purpose, syntax, inputs, and outputs of the different user functions listed in alphabetical order.

### 5.3.1 Correlation

**Description** Determines the value of one out of a pair of correlated defect attributes. The correlation involves a primary and secondary attribute and the derivation of one from the other may depend on any defect attributes or pipe resistance parameters.

**Syntax** int Correlation( const double* *lineAttributes*, const double* *mechanicalProperties*, const double* *defectAttributes*, double* *primary*, double* *secondary*, CORRELATION_TYPE *type* );

**Parameters** The **Correlation** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *mechanicalProperties* | mechanical property values as requested by **GetUsedMechanicalPropertiess** |
| *defectAttributes* | defect attribute values as requested by **GetUsedDefectAttributes** |
| *primary* | pointer to primary attribute for correlation |
| *secondary* | pointer to secondary attribute for correlation |
| *type* | either eCORRELATION_SECONDARY_FROM_PRIMARY or eCORRELATION_PRIMARY_FROM_SECONDARY |

**Remarks** Required in the correlation module.

**Return Value** USER_SUCCESS if the function produced a valid result;
USER_FAILURE if the function failed to produce a valid result.

### 5.3.2 DefectDensityDefault

**Description** Returns the defect density as a function of line attributes.

**Syntax** double DefectDensityDefault( const double* *lineAttributes*, const double *sectionLength*);

**Parameters** The **DefectDensityDefault** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *sectionLength* | length of the current section |

**Remarks** Only used in the defaults module.

**Return Value** The defect density for the current section of the pipeline.

### 5.3.3 DetectionProbability

**Description** Returns the probability of detection for a given defect.

**Syntax** int DetectionProbability( const double* *lineAttributes*, const double* *defectAttributes*, const double* *userConstants*, double* *probability* );

**Parameters** The **DetectionProbability** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *defectAttributes* | defect attribute values as requested by **GetUsedDefectAttributes** |
| *userConstants* | values as requested by **GetUserConstants** |
| *probability* | pointer to value to receive calculated probability |

**Remarks** Required in the detection module.

**Return Value** USER_SUCCESS if the function produced a valid result;
USER_FAILURE if the function failed to produce a valid result.

### 5.3.4 ExcavationCriterion

**Description** Returns the excavation criterion for a specific defect. This value is to be compared to the corresponding threshold value as given in the Excavation and Repair Threshold input dialog. If the comparison is true, the defect is excavated. The operation for the comparison is defined in the RESULT_INFORMATION structure.

**Syntax** int ExcavationCriterion( const double* *lineAttributes*, const double* *defectAttributes*, double* *result* );

**Parameters** The **ExcavationCriterion** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *defectAttributes* | defect attribute values as requested by **GetUsedDefectAttributes** |
| *result* | pointer to value to receive calculated criteria |

**Remarks** Required in the excavation module.

**Return Value** USER_SUCCESS if the function produced a valid result.
USER_FAILURE if the function failed to produce a valid result.

### 5.3.5 GetCalculationResultInformation

**Description** Returns the information describing the threshold by which comparisons will be made and the comparison operation itself.

**Syntax** const RESULT_INFORMATION* GetCalculationInformation();

**Parameters** The **GetCalculationResultInformation** function requires no parameters.

**Remarks** Required in both the excavation and repair modules. The description part of the returned value will be used in the Excavation and Repair Criteria input dialog.

**Return Value** A pointer to a constant RESULT_INFORMATION structure.

### 5.3.6 GetCorrelatedDefectAttributes

**Description** Returns the defect attributes that are correlated.

**Syntax** const CORRELATION* GetCorrelatedDefectAttributes();

**Parameters** The **GetCorrelatedDefectAttributes** function requires no parameters.

**Remarks** Only used in the correlation module.

**Return Value** A pointer to a constant array of one or more CORRELATION structures.

### 5.3.7 GetDefaultMappings

**Description** Returns the attributes and the functions that will provide default values for them.

**Syntax** const DEFAULT_MAPPING* GetDefaultMappings();

**Parameters** The **GetDefaultMappings** function requires no parameters.

**Remarks** Only used in the defaults module. Applicable to any defect attribute or pipe resistance parameter.

**Return Value** A pointer to a constant array of one or more DEFAULT_MAPPING structures.

### 5.3.8 GetUsedDefectAttributes

**Description** Returns the defect attributes that are to be used by primary function in the module. The order of the attributes listed in the array corresponds to the order in which the values will be stored in the input array passed to the primary function.

**Syntax** const DEFECT_ATTRIBUTE* GetUsedDefectAttributes();

**Parameters** The **GetUsedDefectAttributes** function requires no parameters.

**Return Value** A pointer to a constant array of one or more DEFECT_ATTRIBUTE structures.

### 5.3.9 GetUsedLineAttributes

**Description** Returns the line attributes that are to be used by primary function in the module. The order of the attributes listed in the array corresponds to the order in which the values will be stored in the input array passed to the primary function.

**Syntax** const LINE_ATTRIBUTE * GetUsedLineAttributes();

**Parameters** The **GetUsedLineAttributes** function requires no parameters.

**Return Value** A pointer to a constant array of one or more LINE_ATTRIBUTE structures.

### 5.3.10 GetUsedMechanicalProperties

**Description** Returns the mechanical properties ( pipe resistances ) that are to be used by primary function in the module. The order of the properties listed in the array corresponds to the order in which the values will be stored in the input array passed to the primary function.

**Syntax** const MECHANICAL_PROPERTY* GetUsedMechanicalProperties();

**Parameters** The **GetUsedMechanicalProperties** function requires no parameters.

**Return Value** A pointer to a constant array of one or more MECHANICAL_PROPERTY structures.

### 5.3.11 GetUsedModelErrorFactors

**Description** Returns the model error factors that are to be used by primary function in the module. The order of the model error factors listed in the array corresponds to the order in which the values will be stored in the input array passed to the primary function.

**Syntax** const MODEL_ERROR_FACTOR* GetUsedModelErrorFactors();

**Parameters** The **GetUsedModelErrorFactors** function requires no parameters.

**Return Value** A pointer to a constant array of one or more MODEL_ERROR_FACTOR structures.

### 5.3.12 GetUserConstants

**Description** Returns the user constants that are to be used by detection function. The order of the attributes listed in the array corresponds to the order in which the values will be stored in the input array passed to the function.

**Syntax** const USER_CONSTANT* GetUserConstants();

**Parameters** The **GetUserConstants** function requires no parameters.

**Remarks** Only used in the detection module.

**Return Value** A pointer to a constant array of one or more USER_CONSTANT structures.

### 5.3.13 PipePerformance

**Description** Returns the time to failure and failure mode for the given defect and conditions.

**Syntax** int PipePerformance( const double* *lineAttributes*, const double* *defectAttributes*, const double* *mechanicalProperties*, const double* *modelErrorFactors*, double* *timeToFailure*, FAILURE_MODE* *failureMode* );

**Parameters** The **PipePerformance** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *defectAttributes* | defect attribute values as requested by **GetUsedDefectAttributes** |
| *mechanicalProperties* | mechanical property values as requested by **GetUsedMechanicalPropertiess** |
| *modelErrorFactors* | model error factor values as requested by **GetUsedModelErrorFactors** |
| *timeToFailure* | pointer to value to receive calculated time to failure |
| *failureMode* | pointer to value to receive determined mode of failure<br>Will be one of the following if the function is successful:<br>eFAIL_SMALL_LEAK<br>eFAIL_LARGE_LEAK<br>eFAIL_RUPTURE |

**Remarks** Required in the performance module.

**Return Value** USER_SUCCESS if the function produced a valid result;
USER_FAILURE if the function failed to produce a valid result.

### 5.3.14 RepairCriterion

**Description** Returns the repair criterion for a specific defect. This value is to be compared to the corresponding threshold value as given in the Excavation and Repair Threshold input dialog. If the comparison is true, the defect is repaired. The operation for the comparison is defined in the RESULT_INFORMATION structure.

**Syntax** int RepairCriterion( const double* *lineAttributes*, const double* *defectAttributes*, double* *result* );

**Parameters** The **RepairCriterion** function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *defectAttributes* | defect attribute values as requested by **GetUsedDefectAttributes** |
| *result* | pointer to value to receive calculated criteria |

**Remarks** Required in the repair module.

**Return Value** USER_SUCCESS if the function produced a valid result;
USER_FAILURE if the function failed to produce a valid result.

### 5.3.15 User Default Function

**Description** Returns a distribution for an random parameter (defect attribute, pipe resistance parameter or model error factor).

**Syntax** `int DefaultFunction` ( const double* *lineAttributes*, double* *mean*, double* *stdDev*, double* *p1*, double* *p2*, DISTRIBUTION_TYPE* *distType* );

**Parameters** The User Default function has the following parameters:

| Parameter | Description |
|---|---|
| *lineAttributes* | pipeline attribute values as requested by **GetUsedLineAttributes** |
| *mean* | pointer to a value to receive the mean |
| *stdDev* | pointer to a value to receive the standard deviation |
| *p1* | pointer to a value to receive the first parameter |
| *p2* | pointer to a value to receive the second parameter |
| *distType* | pointer to a value to receive the distribution type. |
| | Will be one of the following if the function is successful: |
| | eDISTRIBUTION_DETERMINISTIC |
| | eDISTRIBUTION_EXPONENTIAL |
| | eDISTRIBUTION_RECTANGULAR |
| | eDISTRIBUTION_NORMAL |
| | eDISTRIBUTION_LOGNORMAL |
| | eDISTRIBUTION_GAMMA |
| | eDISTRIBUTION_GUMBEL |
| | eDISTRIBUTION_RAYLEIGH |
| | eDISTRIBUTION_WEIBULL |
| | eDISTRIBUTION_SHIFTED_LOGNORMAL |
| | eDISTRIBUTION_BETA |

**Remarks** Only used in the defaults module. Not all parameters may be updated as the need depends on the type of distribution. The attribute and the function which calculates a default distribution for it, are described in the **GetDefaultMapping** function. The name of the function is defined by the user.

**Return Value** USER_SUCCESS if the function produced a valid result; USER_FAILURE if the function failed to produce a valid result.

## 5.4 Structure Reference

The following describes the structures used, listed in alphabetical order.

### 5.4.1 CORRELATION Structure

**Definition**

```
typedef struct
   {
   char primary[ REQUEST_NAME_LENGTH + 1 ];
   char secondary[ REQEST_NAME_LENGTH + 1 ];
   } CORRELATION;
```

**Fields** The **CORRELATION** structure has the following fields:

| Field | Description |
|---|---|
| *primary* | name of an attribute |
| *secondary* | name of another attribute, different then the primary |

**Comments** The **CORRELATION** structure is only used in the correlation module.

### 5.4.2 DEFAULT_MAPPING Structure

**Definition**

```
typedef struct
   {
   char name[ REQUEST_NAME_LENGTH + 1 ];
   char function[ REQUEST_NAME_LENGTH + 1 ];
   } DEFAULT_MAPPING;
```

**Fields** The **DEFAULT_MAPPING** structure has the following fields:

| Field | Description |
|---|---|
| *name* | name of the attribute which will be given a default value |
| *function* | name of the function used in providing a default value |

**Comments** The **DEFAULT_MAPPING** structure is only used in the defaults module.

### 5.4.3 LINE_ATTRIBUTE Structure

**Definition**

```
typedef struct
  {
  char name[ LINE_ATTRIBUTE_NAME_LENGTH + 1 ];
  } LINE_ATTRIBUTE;
```

**Fields** The **LINE_ATTRIBUTE** structure has the following fields:

| Field | Description |
|---|---|
| *name* | name of the line attribute for which a value will be retrieved |

### 5.4.4 REQUEST_PARAM Structure

**Definition**

```
typedef struct
  {
  char name[ REQUEST_NAME_LENGTH + 1 ];
  char units[ REQUEST_UNIT_LENGTH + 1];
  char attribute[ REQUEST_NAME_LENGTH + 1];
  } REQUEST_PARAM;
```

**Fields** The **REQUEST_PARAM** structure has the following fields:

| Field | Description |
|---|---|
| *name* | name of the attribute for which a value will be retrieved |
| *units* | units that the value will be entered in |
| *attribute* | Future development |

**Comments** The following parameter types are equivalent to this structure:
DEFECT_ATTRIBUTE
MECHANICAL_PROPERTY
MODEL_ERROR_FACTOR

### 5.4.5 RESULT_INFORMATION Structure

**Definition**

```
typedef struct
  {
  RESULT_COMPARISON_TYPE comparison;
  char units[ REQUEST_UNIT_LENGTH + 1 ];
  char description[ RESULT_DESCRIPTION_LENGTH + 1 ];
  } RESULT_INFORMATION;
```

**Fields**

The **RESULT_INFORMATION** structure has the following fields:

| Field | Description |
|---|---|
| *comparison* | operation for comparison with the corresponding threshold value<br>May be one of the following:<br>eRESULT_COMPARISON_GREATER_THAN_THRESHOLD<br>eRESULT_COMPARISON_LESS_THAN_THRESHOLD |
| *units* | units label pertaining to the corresponding threshold value |
| *description* | label describing value required for threshold |

**Comments**

The **RESULT_INFORMATION** structure is only used in the excavation and repair modules. If the value returned by the excavation or repair function, when compared to the corresponding threshold criterion, meets the comparison operation constraint, the defect being analyzed is excavated or repaired respectively.

### 5.4.6 USER_CONSTANT Structure

**Definition**

```
typedef struct
  {
  char name[ CONSTANT_NAME_LENGTH + 1 ];
  } USER_CONSTANT;
```

**Fields**

The **USER_CONSTANT** structure has the following fields:

| Field | Description |
|---|---|
| *name* | name of the constant for which a value will be retrieved |

**Comments**

The **USER_CONSTANT** structure is only used in the detection module.

# Appendix D

## MODEL DATA FILE FORMAT SPECIFICATION

# ONSHORE MODEL DATA FILE FORMAT SPECIFICATION

# Onshore Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>            ::= <Segments>
<Segments>             ::= <Segment> | <Segment> + <Segments>
<Segment>              ::= [Segment] + NL + <SegmentAttributes>

<SegmentAttributes>    ::= <SegmentName> + <Start> + <End> + <PipelineAttributes>
<PipelineAttributes>   ::= <PipelineAttribute> | <PipelineAttribute> + <PipelineAttributes>

<PipelineAttribute>    ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>      ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>       ::= <AssignedValue> + NL
<AssignedValue>        ::= <S1Data> | <C1Data> | <S2Data>
<S1Data>               ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>              ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>               ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>              ::= <KPSValue> + : + <Value>
<S2Data>               ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>              ::= <KPSValue> + : + <KPEValue> + : + <Choice>

<SegmentName>          ::= Segment_Name + = + <StringValue> + NL
<Start>                ::= KP_Start + = + <KPSValue> + NL
<End>                  ::= KP_End + = + <KPEValue> + NL
<KPSValue>             ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>             ::= e REAL | e INTEGER defined by e > s
<Value>                ::= x REAL | x INTEGER
<Choice>               ::= x element of set of { A ... Z } as appropriate to list choice
<StringValue>          ::= any sequence of printable characters including a space

<Label>                ::= Pipe_Diameter | Pipe_Wall_Thickness |
                           Operating_Temperature | Product_Flow_Rate |
                           Block_Valve_Spacing | Time_To_Block_Valve_Closure |
                           Detectable_Release_Volume | Time_To_Leak_Detection |
                           Time_To_Leak_Stoppage | Line_Elevation |
                           Operating_Pressure | Accessibility | Far_Field_Terrain |
                           Surface_Containment | Surface_Topography |
                           Annual_Rainfall | Flood_Potential | Near_Field_Terrain |
                           Land_Use_Within_5km | Pipeline_Orientation |
                           Adjacent_Land_Use | Crossing_Type/Special_Terrain |
                           Surface_Water_Within_300m | Drinking_Water_Within_5km |
                           Other_Water_Within_5km | Land_Use_Within_5km |
                           Sensitive_Environment_Within_10km |
                           Sensitive_Groundwater_Within_10km | Aquifer_Conductivity |
                           Confining_Layer_Thickness | Confining_Layer_Conductivity
```

# OFFSHORE MODEL DATA FILE FORMAT SPECIFICATION

# Offshore Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>            ::= <Segments>
<Segments>             ::= <Segment> | <Segment> + <Segments>
<Segment>              ::= [Segment] + NL + <SegmentAttributes>

<SegmentAttributes>    ::= <SegmentName> + <Start> + <End> +
                           <LaunchZoneNames> + <PipelineAttributes>

<SegmentName>          ::= Segment_Name + = + <StringValue> + NL
<Start>                ::= KP_Start + = + <KPSValue> + NL
<End>                  ::= KP_End + = - <KPEValue> - NL
<KPSValue>             ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>             ::= e REAL | e INTEGER defined by e > s
<Value>                ::= x REAL | x INTEGER
<Choice>               ::= x element of set of { A ... Z } as appropriate to list choice
<StringValue>          ::= any sequence of printable characters including a space

<LaunchZoneNames>      ::= Launch_Zone_Names + = + <LaunchZoneList> + NL
<LaunchZoneList>       ::= <LaunchZone> | <LaunchZone> + <LaunchZoneList>
<LaunchZone>           ::= <StringValue>

<PipelineAttributes>   ::= <PipelineAttribute> | <PipelineAttribute> + <PipelineAttributes>
<PipelineAttribute>    ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>      ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>       ::= <AssignedValue> - NL
<AssignedValue>        ::= <S1Data> | <C1Data> | <S2Data> | <S3Data> |
                           <P1Data> | <P2Data> | <P3Data> | <P4Data>
<S1Data>               ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>              ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>               ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>              ::= <KPSValue> + : + <Value>
<S2Data>               ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>              ::= <KPSValue> + : + <KPEValue> + : + <Choice>
<S3Data>               ::= <S3Datum> | <S3Datum> + , + <S3Data>
<S3Datum>              ::= <KPSValue> + : + <KPEValue> + : + <LaunchZone>
<P1Data>               ::= <P1Datum> | <P1Datum> + , + <P1Data>
<P1Datum>              ::= <KPSValue> + : + <Choice> + : + <Value>

<Label>                ::= Pipe_Diameter | Pipe_Wall_Thickness |
                           Line_Elevation | Operating_Pressure |
                           Operating_Temperature | Product_Flow_Rate |
                           Block_Valve_Spacing | Time_To_Block_Valve_Closure |
                           Detectable_Release_Volume | Time_To_Leak_Detection |
                           Time_To_Leak_Stoppage | Pipeline_Orientation |
                           Vessel_Traffic_Density | Adjacent_Platform | Launch_Zones
```

# ONSHORE INSPECTION DATA FILE FORMAT SPECIFICATION

# Onshore Inspection Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>       ::= <Pipeline>
<Pipeline>            ::= [Pipeline] + NL + <PipelineInfo>

<PipelineInfo>        ::= <Start> + <End> + <PipelineAttributes>
<PipelineAttributes> ::= <PipelineAttribute> | <PipelineAttribute> +
<PipelineAttributes>

<PipelineAttribute>  ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>    ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>     ::= <AssignedValue> + NL
<AssignedValue>        ::= <S1Data> | <C1Data> | <S2Data> | <P5Data>
<S1Data>          ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>          ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>         ::= <KPSValue> + : + <Value>
<S2Data>          ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Choice>
<P5Data>          ::= <P5Datum> | <P5Datum> + , + <P5Data>
<P5Datum>         ::= <KPSValue> + : + <StringValue> + : <Value>

<Start>               ::= Start + = + <KPSValue> + NL
<End>             ::= End + = + <KPEValue> + NL
<KPSValue>        ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>        ::= e REAL | e INTEGER defined by e > s
<Value>               ::= x REAL | x INTEGER
<Choice>          ::= x element of set of { A ... Z } as appropriate to list
choice
<StringValue>         ::= any sequence of printable characters including a
space

<Label>               ::= Pipe Diameter | Pipe Wall Thickness |
                        Pipeline Orientation | Elevation Profile |
                    Pressure Profile | Flow Rate | Product Temperature |
                    Block Valve_Spacing | Time to Block Valve Closure |
                    Detectable Release Volume | Time to Leak Detection |
                    Time to Leak Stoppage | Adjacent Land Use |
                    Pipeline Accessibility | Crossings/Special Terrain |
                    Near Field Terrain | Far Field Terrain |
                    Surface Containment | Surface Topography |
                    Annual Rainfall | Flood Potential (return period) |
                    Surface Water within 300 m | Drinking Water within 5 km |
                    Other Water within 5 km | Direct Exposure due to Land Use
within 5 km |
                    Sensitive Environment within 10 km | Defects per Indication |
                    Sensitive Groundwater within 10 km | Defect Density |
```

Structure Type | <UserDefinedLabel>

For user defined modules, new line attributes may be specified. The name of the new line attribute will correspond directly to the label used in the MDL specification for that attribute.

<UserDefinedLabel> ::= <StringValue>

# OFFSHORE INSPECTION DATA FILE FORMAT SPECIFICATION

# Offshore Inspection Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>       ::= <Pipeline>
<Pipeline>            ::= [Pipeline] + NL + <PipelineInfo>

<PipelineInfo>        ::= <Start> + <End> + <PipelineAttributes>
<PipelineAttributes> ::= <PipelineAttribute> | <PipelineAttribute> +
<PipelineAttributes>

<PipelineAttribute>   ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>     ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>      ::= <AssignedValue> + NL
<AssignedValue>         ::= <S1Data> | <C1Data> | <S2Data> | <P5Data>
<S1Data>          ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>          ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>         ::= <KPSValue> + : + <Value>
<S2Data>          ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Choice>
<P5Data>          ::= <P5Datum> | <P5Datum> + , + <P5Data>
<P5Datum>         ::= <KPSValue> + : + <StringValue> + : <Value>

<Start>               ::= Start + = + <KPSValue> + NL
<End>             ::= End + = + <KPEValue> + NL
<KPSValue>        ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>        ::= e REAL | e INTEGER defined by e > s
<Value>               ::= x REAL | x INTEGER
<Choice>          ::= x element of set of ( A ... Z ) as appropriate to list
choice
<StringValue>         ::= any sequence of printable characters including a
space

<Label>               ::= Pipe Diameter | Pipe Wall Thickness |
                        Pipeline Orientation | Elevation Profile |
                    Pressure Profile | Flow Rate | Product Temperature |
                    Block Valve_Spacing | Time to Block Valve Closure |
                    Detectable Release Volume | Time to Leak Detection |
                    Time to Leak Stoppage | Vessel Traffic Density |
                    Defect Density | Structure Type | <UserDefinedLabel>
```

For user defined modules, new line attributes may be specified. The name of the new line attribute will correspond directly to the label used in the MDL specification for that attribute.

```
<UserDefinedLabel>   ::= <StringValue>
```

# ONSHORE MECHANICAL DAMAGE MODEL DATA FILE FORMAT SPECIFICATION

Onshore Mechanical Damage Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>        ::= <Segment>
<Segment>              ::= [Segment] + NL + <SegmentAttributes>

<SegmentAttributes>  ::= <SegmentName> + <Start> + <End> + <PipelineAttributes>
<PipelineAttributes> ::= <PipelineAttribute> | <PipelineAttribute> +
<PipelineAttributes>

<PipelineAttribute>  ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>    ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>     ::= <AssignedValue> + NL
<AssignedValue>        ::= <S1Data> | <C1Data> | <S2Data> | <P5Data>
<S1Data>           ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>          ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>           ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>          ::= <KPSValue> + : + <Value>
<S2Data>           ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>          ::= <KPSValue> + : + <KPEValue> + : + <Choice>
<P5Data>           ::= <P5Datum> | <P5Datum> + , + <P5Data>
<P5Datum>          ::= <KPSValue> + <StringValue> + : + <Value>

<Start>                ::= KP_Start + = + <KPSValue> + NL
<End>              ::= KP_End + = + <KPEValue> + NL
<KPSValue>         ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>         ::= e REAL | e INTEGER defined by e > s
<Value>                ::= x REAL | x INTEGER
<Choice>           ::= x element of set of { A ... Z } as appropriate to list
choice
<StringValue>          ::= any sequence of printable characters including a
space

<Label>                ::= Pipe_Diameter | Pipe_Wall_Thickness |
                     Pipe_Body_Yield_Strength | Notch_Toughness |
                     Operating_Temperature | Product_Flow_Rate |
                     Block_Valve_Spacing | Time_To_Block_Valve_Closure |
                     Detectable_Release_Volume | Time_To_Leak_Detection |
                     Time_To_Leak_Stoppage | Line_Elevation |
                     Operating_Pressure | Cover_Depth_Range |
                     Depth_Of_Cover | Adjacent_Land_Use |
                     Accessibility | Far_Field_Terrain |
                     Crossing_Type/Special_Terrain | Surface_Water_Within_300m |
                     One_Call_System | Dig_Notification_Requirement |
                     Dig_Notification_Response | Public_Awareness |
                     ROW_Indication | Pipeline_Signage | Permanent_Markers |
                     Buried_Markers | ROW_Surveillance_Interval_Summer |
                     ROW_Surveillance_Interval_Winter |
```

ROW_Surveillance_Method_Summer | Mechanical_Protection |
ROW_Surveillance_Method_Winter |
Surface_Containment | Surface_Topography |
Annual_Rainfall | Flood_Potential | Near_Field_Terrain |
Land_Use_Within_5km | Pipeline_Orientation |
Drinking_Water_Within_5km | Other_Water_Within_5km |
Sensitive_Environment_Within_10km |
Sensitive_Groundwater_Within_10km |
Confining_Layer_Thickness | Confining_Layer_Conductivity |
Aquifer_Conductivity | Structure

# OFFSHORE MECHANICAL DAMAGE MODEL DATA FILE FORMAT SPECIFICATION

# Offshore Mechanical Damage Model Data File Format Specification

Note: NL represents a new line character. Blank lines in the input file will be ignored. Lines starting with a semicolon will be regarded as comment lines and will be ignored.

Notation: + represents a logical AND meaning "and must include"
| represents a logical OR. Choose only one of the options separated by the |.

A model data file consists of one <ModelData> description.

```
<ModelData>       ::= <Segment>
<Segment>             ::= [Segment] + NL + <SegmentAttributes>

<SegmentAttributes>  ::= <Start> + <End> +
                      <LaunchZoneNames> + <PipelineAttributes>

<Start>                  ::= KP_Start + = + <KPSValue> + NL
<End>             ::= KP_End + = + <KPEValue> + NL
<KPSValue>        ::= s REAL | s INTEGER defined by s >= 0.0
<KPEValue>        ::= e REAL | e INTEGER defined by e > s
<Value>                  ::= x REAL | x INTEGER
<Choice>          ::= x element of set of { A ... Z } as appropriate to list
choice
<StringValue>            ::= any sequence of printable characters including a
space

<LaunchZoneNames>     ::= Launch_Zone_Names + = + <LaunchZoneList> + NL
<LaunchZoneList>      ::= <LaunchZone> | <LaunchZone> + <LaunchZoneList>
<LaunchZone>          ::= <StringValue>

<PipelineAttributes> ::= <PipelineAttribute> | <PipelineAttribute> +
<PipelineAttributes>
<PipelineAttribute>   ::= <Label> + = + <AssignmentLine> | <AssignmentLines>
<AssignmentLines>     ::= <AssignmentLine> | <AssingmentLine> + <AssignmentLines>
<AssignmentLine>      ::= <AssignedValue> + NL
<AssignedValue>         ::= <S1Data> | <C1Data> | <S2Data> | <S3Data> |
                          <P1Data> | <P2Data> | <P3Data> | <P4Data>
<S1Data>          ::= <S1Datum> | <S1Datum> + , + <S1Data>
<S1Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Value>
<C1Data>          ::= <C1Datum> | <C1Datum> + , + <C1Data>
<C1Datum>         ::= <KPSValue> + : + <Value>
<S2Data>          ::= <S2Datum> | <S2Datum> + , + <S2Data>
<S2Datum>         ::= <KPSValue> + : + <KPEValue> + : + <Choice>
<S3Data>          ::= <S3Datum> | <S3Datum> + , + <S3Data>
<S3Datum>         ::= <KPSValue> + : + <KPEValue> + : + <LaunchZone>
<P1Data>              ::= <P1Datum> | <P1Datum> + , + <P1Data>
<P1Datum>             ::= <KPSValue> + : + <Choice> + : + <Value>

<Label>                 ::= Pipe_Diameter | Pipe_Wall_Thickness |
                          Pipe_Body_Yield_Strenght |
                    Line_Elevation | Operating_Pressure |
                    Operating_Temperature | Product_Flow_Rate |
                    Block_Valve_Spacing | Time_To_Block_Valve_Closure |
                    Detectable_Release_Volume | Time_To_Leak_Detection |
                    Time_To_Leak_Stoppage | Pipeline_Orientation |
```

Vessel_Traffic_Density | Adjacent_Platform |
Launch_Zones | Public_Awareness | Reference_Cover |
Cover_Erosion_Potential | Cover_Monitoring_Action |
Response_to_Cover_Monitoring | Mechanical_Protection

# Appendix E

## PIRAMID Onshore-Mechanical Damage Version 1.0 Model Input Definitions

### 1.0 Scope

PIRAMID Onshore-Mechanical Damage 1.0 was developed by expanding the Performance node in PIRAMID Onshore into a number of nodes that calculate performance from the potential for mechanical damage in the pipeline, and modifying the Failure Segment and Maintenance Cost nodes to make them consistent with the changes made to the Performance node. This Appendix defines model inputs for nodes that were added to or changed from the PIRAMID Onshore model described in Appendix A.

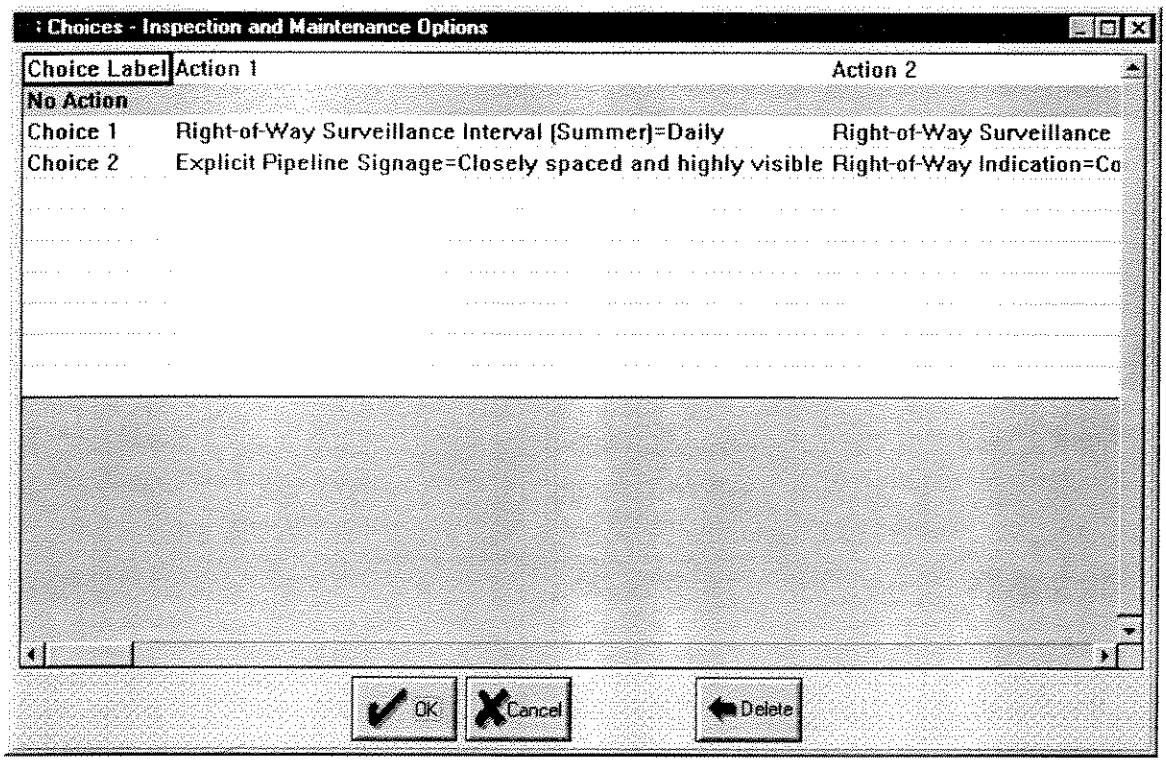
# 2.0 Influence Diagram Parameters

## 2.1 Choices Node Group

### 2.1.1 Choices Node

The parameter of this node is represented by a discrete set of preventative maintenance choices that are to be considered in the run.

**Choices Input Dialog Box**



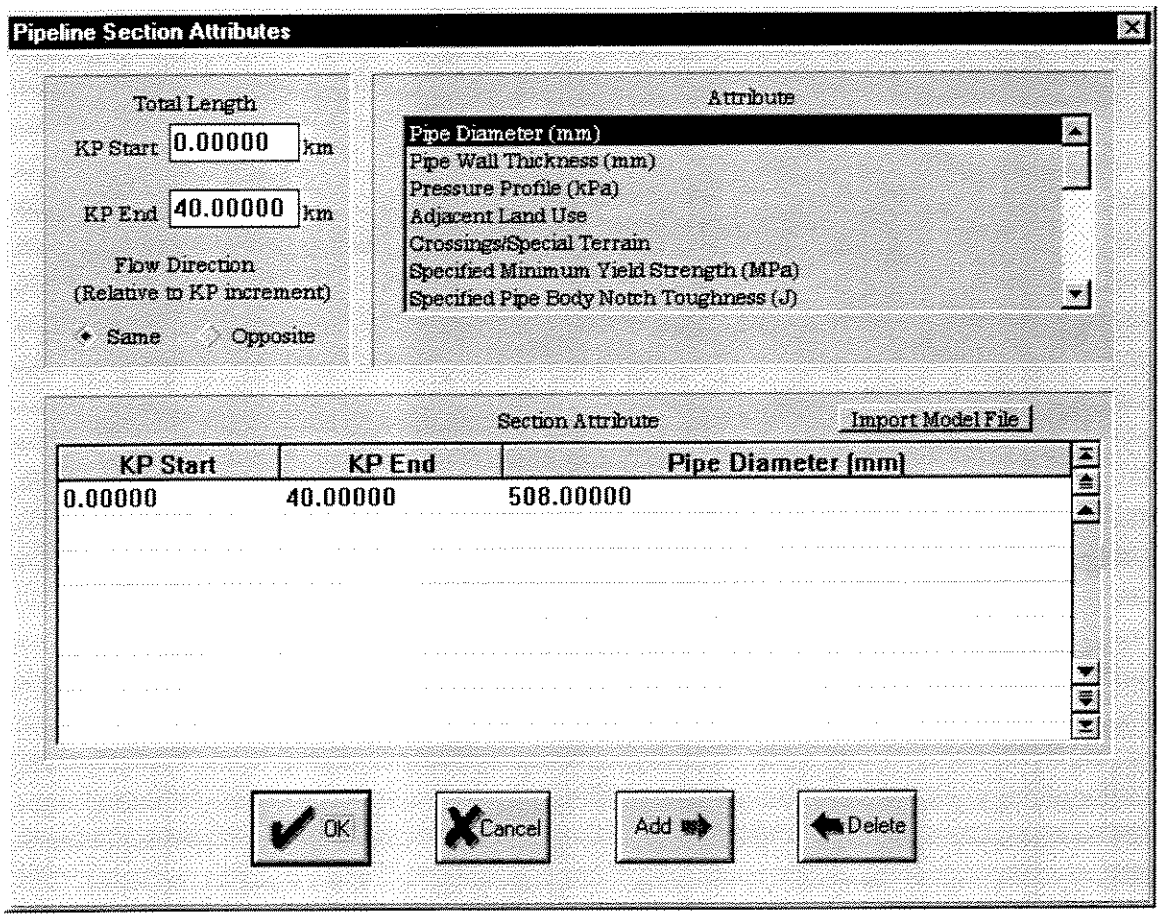

For each choice that is to be considered (except for the base case, identified by the choice label 'No Action'), the user must define the set of individual maintenance actions that are to be associated with the choice. Each choice is identified by a user-defined text string label entered in the first column of the table. Maintenance actions are selected from pull-down menus activated by clicking on cells in the row corresponding to the choice being specified.

## 2.2 Damage Potential Node Group

### 2.2.1 Damage Section Node

The node parameter is the section of pipeline that is associated with a mechanical interference event, where a section is defined as a length of pipeline for which all mechanical damage-related attributes are uniform (see also Section 4.2, PIRAMID Technical Reference Manual No. 8.x).

**Section Attributes Input Dialog Box**




The user must specify the value of each pipeline attribute displayed in the input dialog box. Attributes can be defined interactively via the input window and embedded spreadsheets or by importing a predefined segment data file.

Each attribute must be defined along the entire length of the pipeline segment being analyzed. Mechanical Damage related line attributes are as follows:

- **Pipe Diameter.** Outside diameter of the line pipe.
- **Pipe Wall Thickness.** Wall thickness of the line pipe.
- **Pressure Profile.** The maximum pipeline operating pressure. Since the operating pressure can potentially vary along the line length, this parameter is defined at the beginning, end, and user-selected points along the line segment. The program will define the maximum operating pressure at any point by linear interpolation between the points defined by the user.
- **Adjacent Land Use.** The land use on or adjacent to the pipeline right-of-way (*e.g.*, industrial, residential-urban, agricultural, remote).
- **Crossings/Special Terrain.** Crossings that intersect the right-of-way or special terrain features through which the right-of-way passes (*e.g.*, roadway/railway crossing, river/stream crossing, bog/marsh, marsh/swamp).
- **Specified Minimum Yield Strength.** Specified minimum yield strength of the line pipe.
- **Specified Pipe Body Notch Toughness.** Specified Charpy V-notch impact energy of the line pipe (upper plateau energy for a full-sized Charpy specimen).
- **Depth of Cover.** The pipeline burial depth.
- **One-call System Type.** The type of one call system in-place (*e.g.*, none, multiple systems, unified system).
- **Dig Notification Requirement.** The legislative requirements in force pertaining to the need for parties planning excavation activity to use a one call system (*i.e.*, not required, required but not enforced, required and enforced).
- **Dig Notification Response.** The action taken by the operator in response to a pending dig notification (*e.g.*, provide location information only, locate and mark, excavation by operator).
- **Level of Public Awareness.** A relative assessment of the level of public awareness regarding the presence of and hazards posed by buried pipelines (*i.e.*, below average, average, above average).

- **Right-of-Way Indication.** An assessment of the degree to which the condition of the right-of-way provides an indication of the existence of a buried pipeline in the immediate area (*i.e.*, none, limited, intermittent or variable, continuous, and distinctive).

- **Explicit Pipeline Signage.** An assessment of the number and placement of explicit signage that describes the pipeline and provides a telephone notification number (*i.e.*, none, limited, at selected strategic locations, closely spaced, and highly visible).

- **Permanent Alignment Markers.** An indication of whether or not permanent above ground markers (such as fencing or concrete curbs) exist to provide an obvious indication of the location of the pipeline alignment.

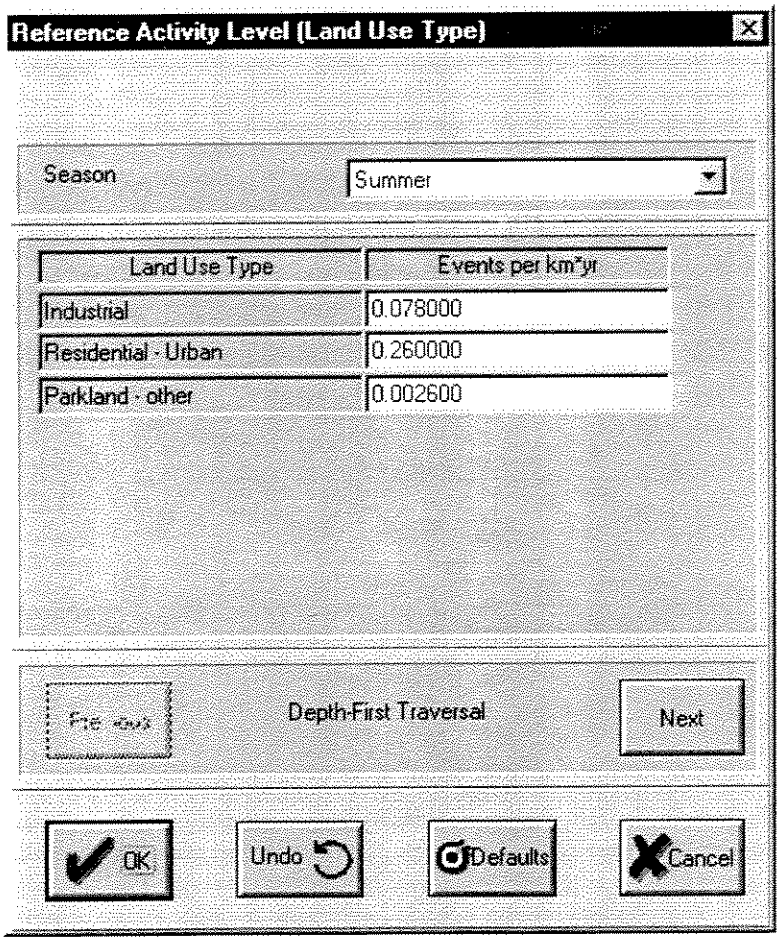
- **Buried Alignment Markers.** An indication of whether or not buried markers (such as coloured tape) exist to provide an indication of the location of the pipeline alignment after the start of excavation activity.

- **Right-of-Way Surveillance Interval.** An indication of the time interval between right-of-way patrols (*e.g.*, daily, weekly, monthly).

- **Right-of-Way Surveillance Method.** An indication of the method employed to patrol the right-of-way (*i.e.*, aerial patrol, ground-based patrol)

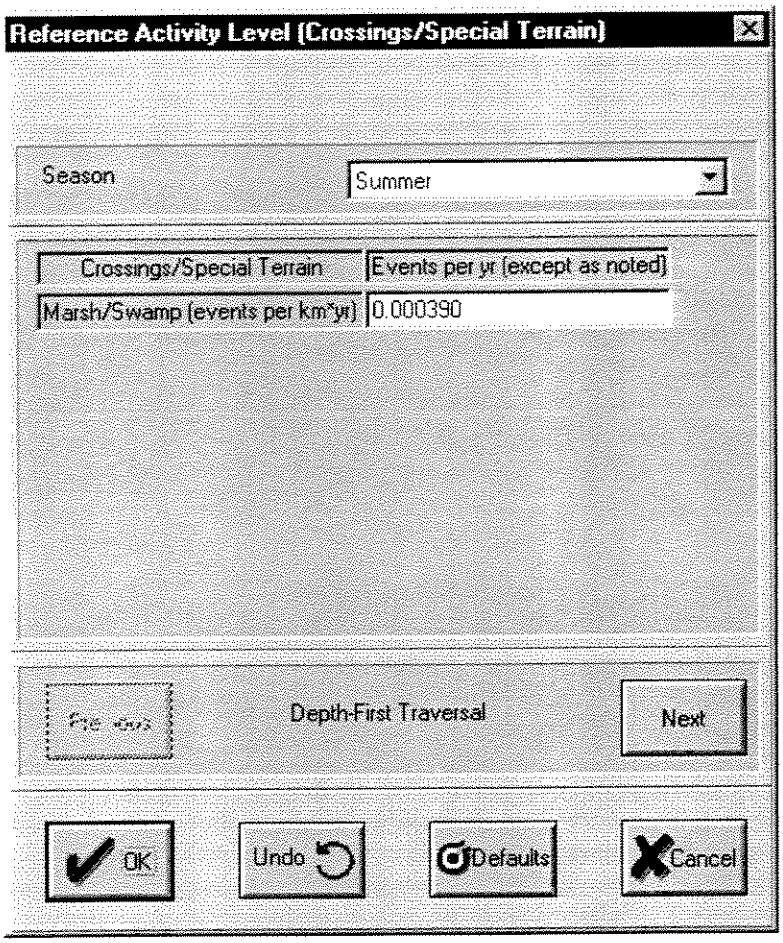
- **Mechanical Protection.** An indication of whether or not buried physical protection (such as concrete slabs or steel plates) exists to effectively prevent direct contact between the pipe body and any excavation equipment.

### Land Use Activity Input Dialog Box




The user must specify the frequency of occurrence of excavation or digging activity on the right-of-way for each land use type specified via the Section Attributes input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

### Crossing/Special Terrain Activity Level Input Dialog Box



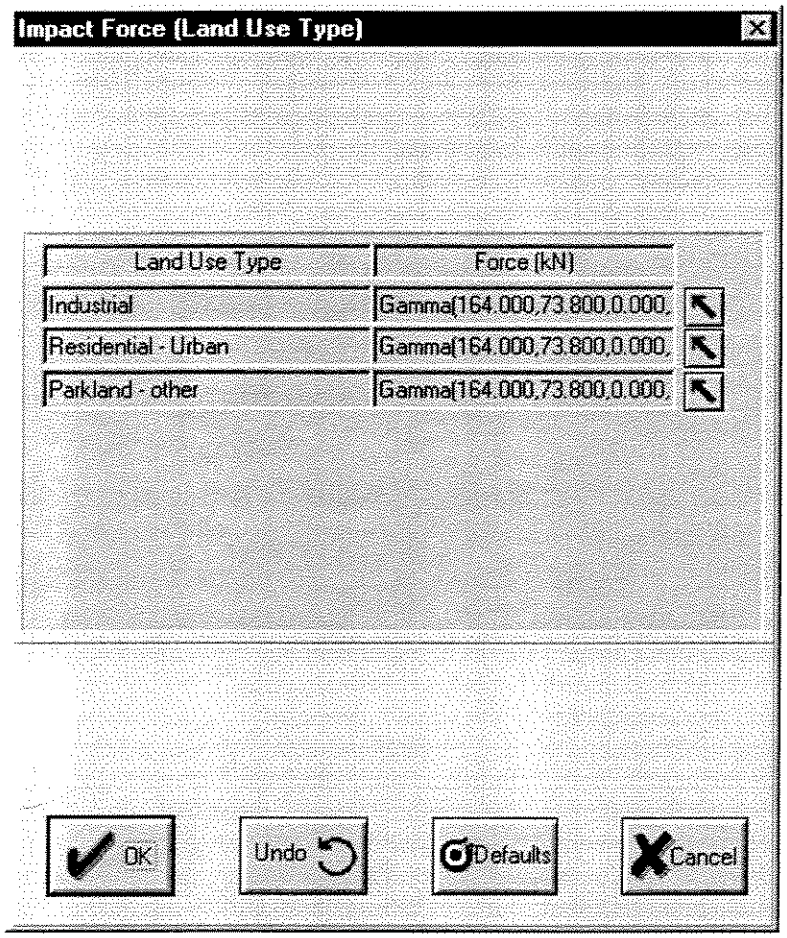

The user must specify the frequency of occurrence of excavation or digging activity on the right-of-way for each crossing and special terrain type specified via the Section Attributes input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

## 2.3 Damage Characteristics Node Group

### 2.3.1 Impact Force Node

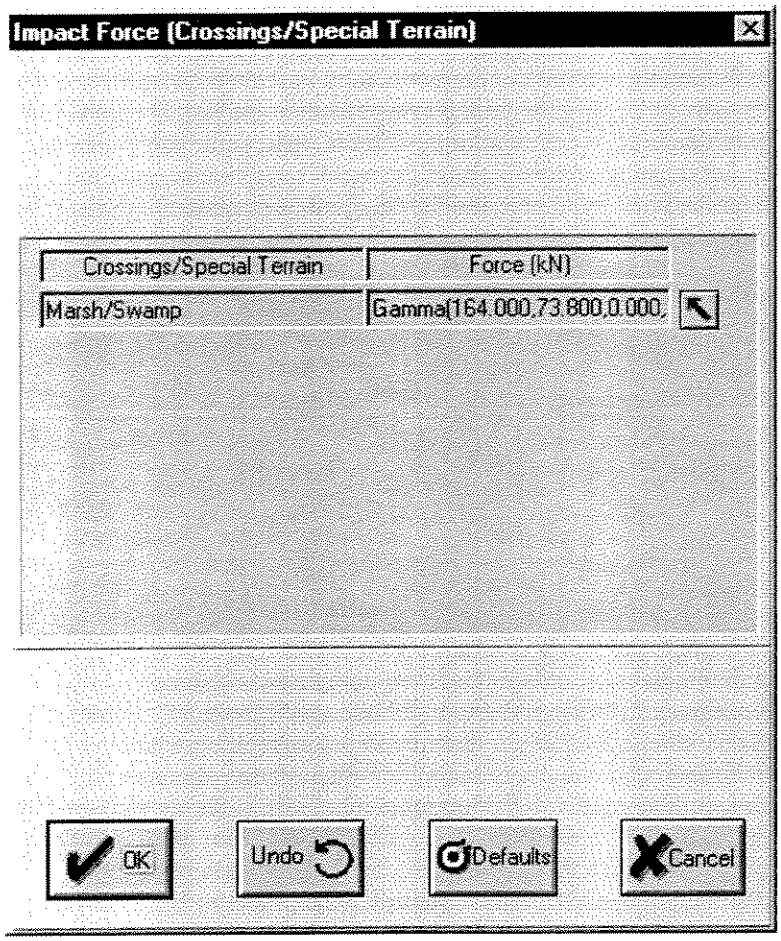
The node parameter is the force associated with an impact event (see Section 5.2.1, PIRAMID Technical Reference Manual No. 8.x).

**Impact Force (Land Use Type) Input Dialog Box**




The user must specify a probability distribution for the impact force applicable to each land use type specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

### Impact Force (Crossing/Special Terrain) Input Dialog Box

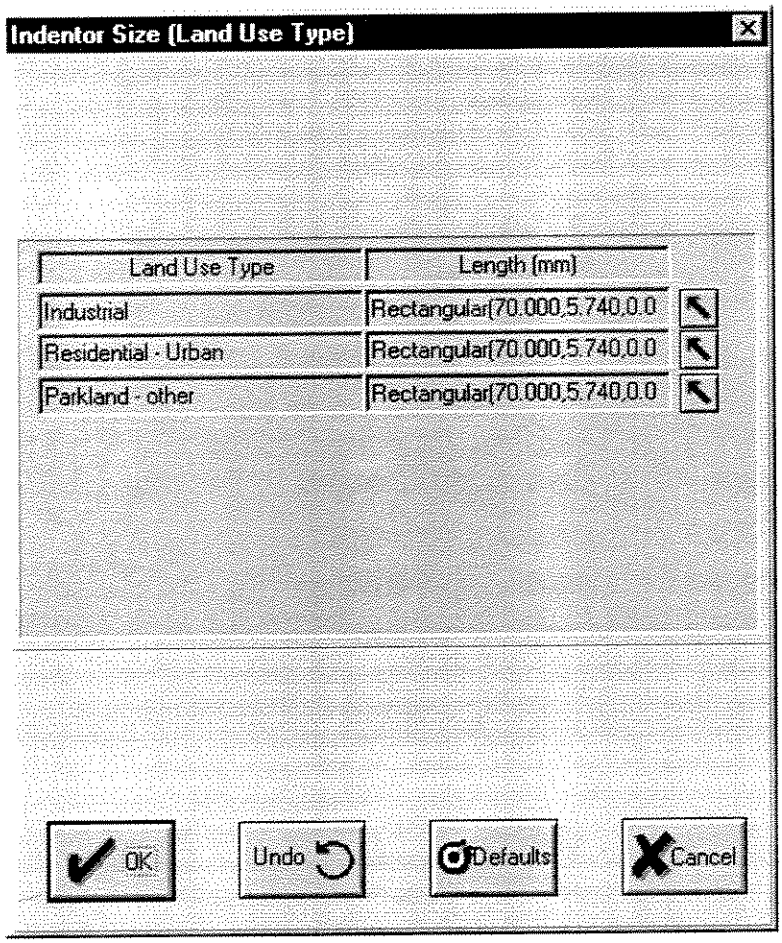



The user must specify a probability distribution for the impact force applicable to each crossing and special terrain type specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

## 2.3.2 Indentor Size Node

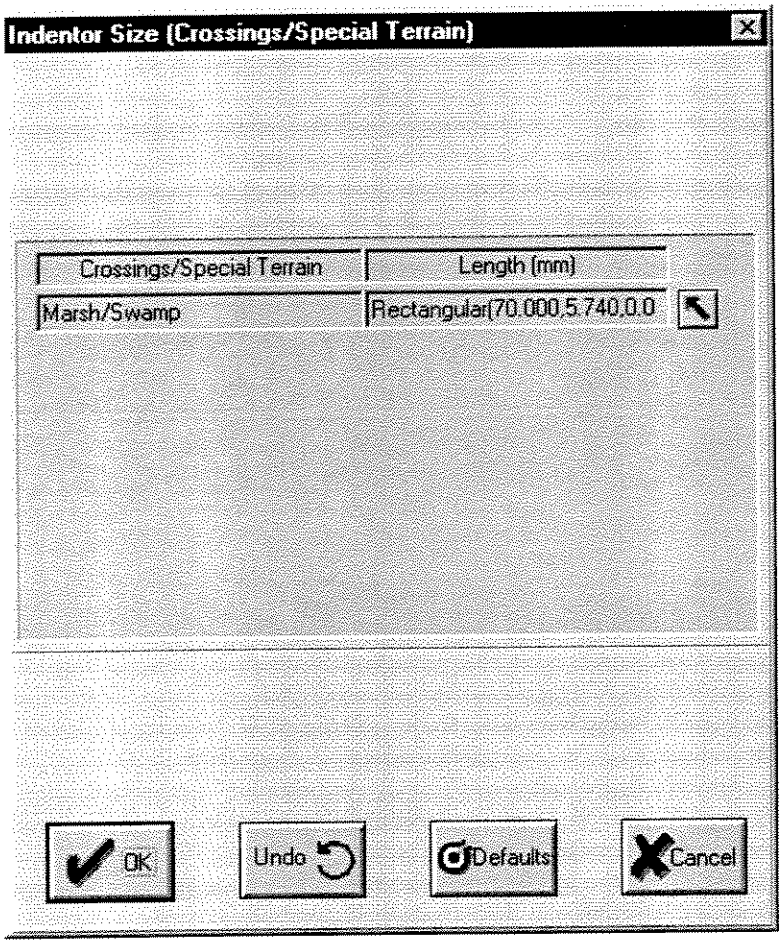
The node parameter is the length of the indentor (see Section 5.2.2, PIRAMID Technical Reference Manual No. 8.x).

### Indentor Size (Land Use Type) Input Dialog Box




The user must specify a probability distribution for the indentor length applicable to each land use type specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

### Indentor Size (Crossing/Special Terrain) Input Dialog Box



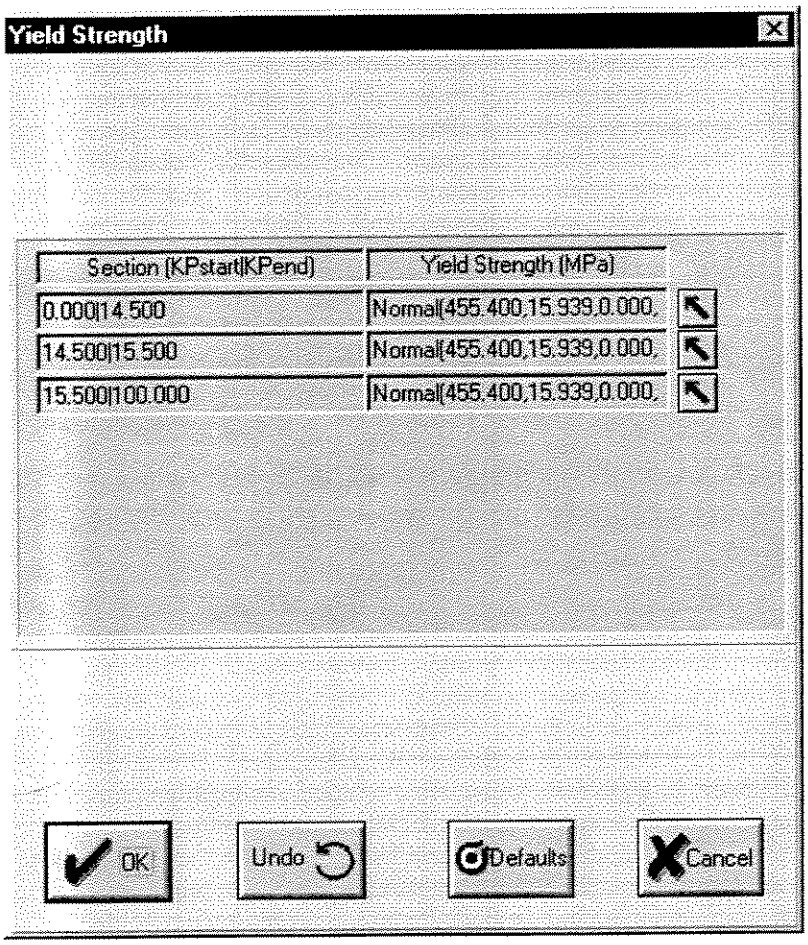

The user must specify a probability distribution for the indentor length applicable to each land use type specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

## 2.4 Mechanical Properties Node Group

### 2.4.1 Yield Strength Node

The node parameter is the actual yield strength of the pipe body. It is noted that the actual yield strength is related (but not equal) to the specified minimum yield strength (see Section 6.2.1, PIRAMID Technical Reference Manual No. 8.x).

**Yield Strength Input Dialog Box**

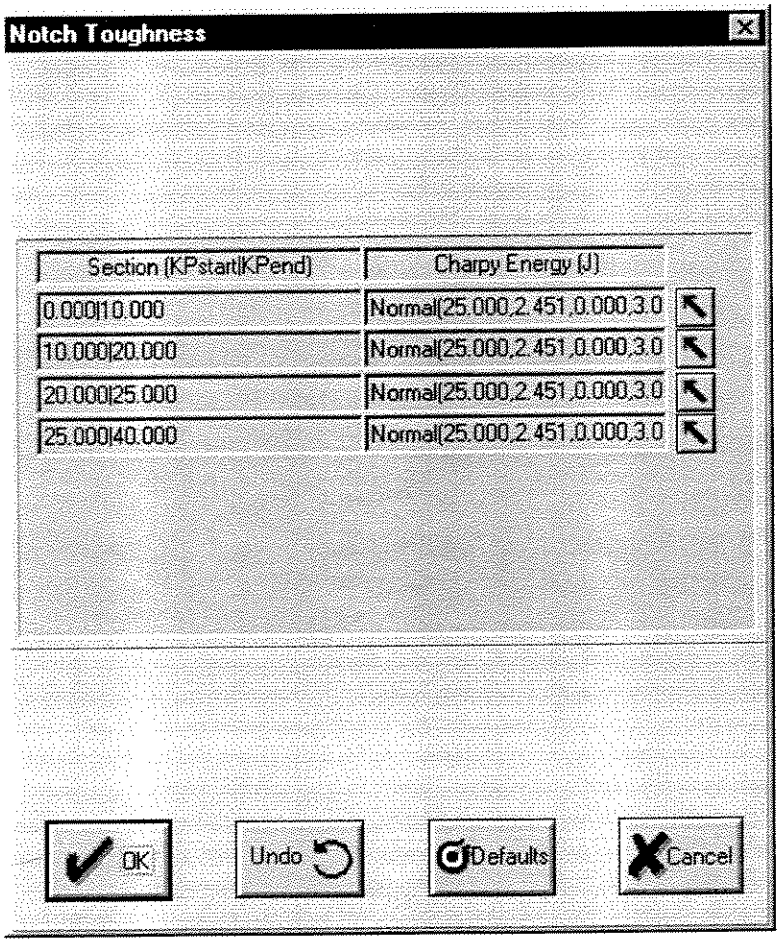



The user must specify a probability distribution for the pipe body yield strength applicable to each damage section defined by the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, a default distribution for each section, estimated from the specified yield strength, will be loaded into the model.

## 2.4.2 Notch Toughness Node

The node parameter is the actual notch toughness of the line pipe. It is noted that the actual toughness is related (but not equal) to the specified pipe body notch toughness (see Section 6.2.2, PIRAMID Technical Reference Manual No. 8.x).

### Notch Toughness Input Dialog Box



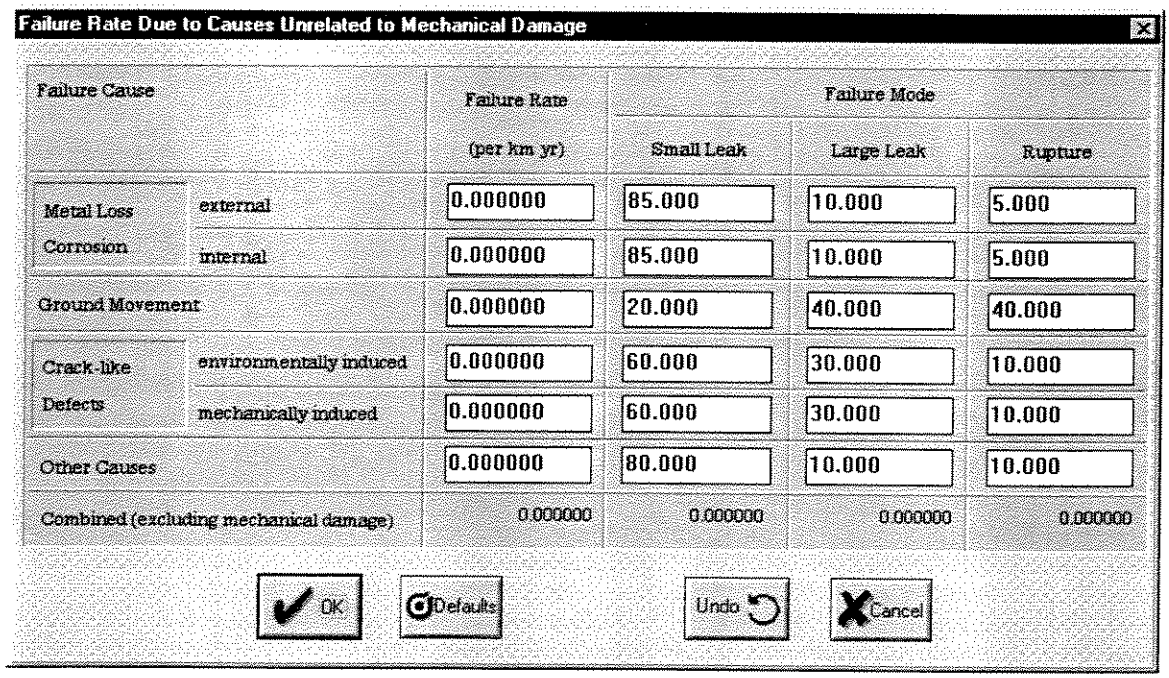

The user must specify a probability distribution for the pipe body notch toughness (in terms of the full-size Charpy V-notch plateau impact energy) applicable to each damage section defined by the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, a default distribution for each section, estimated from the specified notch toughness, will be loaded into the model.

## 2.5 Pipe Performance Node Group

### 2.5.1 Segment Performance Node

The node parameter is the annual performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline segment (see Section 7.3, PIRAMID Technical Reference Manual No. 8.x). The parameter is calculated by combining the mechanical damage related performance (as calculated from previous nodes of the diagram) with the performance related to other failure mechanisms as defined in the input dialog box for this node.

**Segment Performance Input Dialog Box**




The segment performance input dialog box allows the user to define the annual failure rates due to different causes excluding mechanical damage. In the column titled *Failure Rate* the user must input the failure rate per km year for each failure cause indicated in the Table. In the columns titled *Relative Failure Mode Frequency* the user must define the relative frequency in % (a number between 0 and 100) of each failure mode for each failure cause. (Note that the total relative frequencies for small leak, large leak, and rupture must add up to 100 for each failure cause.) Alternatively, upon request by the user, default data stored in the PIRONMD.INI file will be loaded into the model.

Once the user input is complete, the program will calculate the total failure rate and relative frequencies associated with different failure causes (excluding mechanical damage) and display them on the bottom row of the table.

### 2.5.2 Performance Given Damage Node

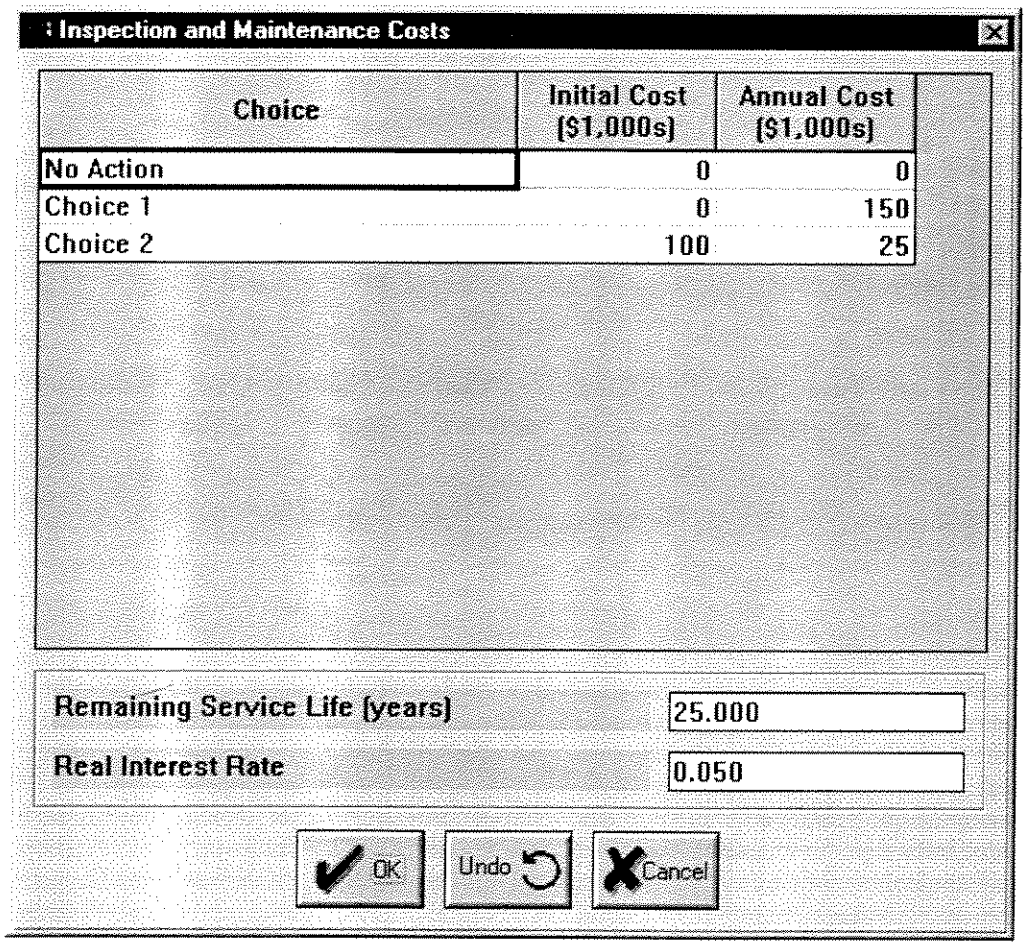
The node parameter is the performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline at a randomly selected mechanical interference event (see Section 7.2, PIRAMID Technical Reference Manual No. 8.x). No user input is required to specify this node.

## 2.6 Repair and Interruption Cost Node Group

### 2.6.1 Maintenance Cost Node

The node parameter is the total annual maintenance cost for the whole pipeline segment in present value currency (see Section 9.2, PIRAMID Technical Reference Manual No. 6.x). The maintenance cost is calculated as the sum of the annual cost and the annualized equivalent of the initial cost associated with implementing the prescribed set of maintenance actions.

**Maintenance Cost Input Dialog Box**




The initial maintenance cost dialog box requires the user to define the individual cost components associated with each choice. In addition, the user must input the expected remaining service life of the pipeline and the real interest rate (where the real interest rate is defined as the difference between the actual interest rate and the inflation rate).

# Appendix F

## PIRAMID Offshore-Mechanical Damage Version 1.0 Model Input Definitions

### 1.0 Scope

PIRAMID Offshore-Mechanical Damage 1.0 was developed by expanding the Performance node in PIRAMID Offshore into a number of nodes that calculate performance from the potential for mechanical damage in the pipeline and modifying the Failure Segment and Maintenance Cost nodes to make them consistent with the changes made to the Performance node. This Appendix defines model inputs for nodes that were added to or changed from the PIRAMID Offshore model described in Appendix B.

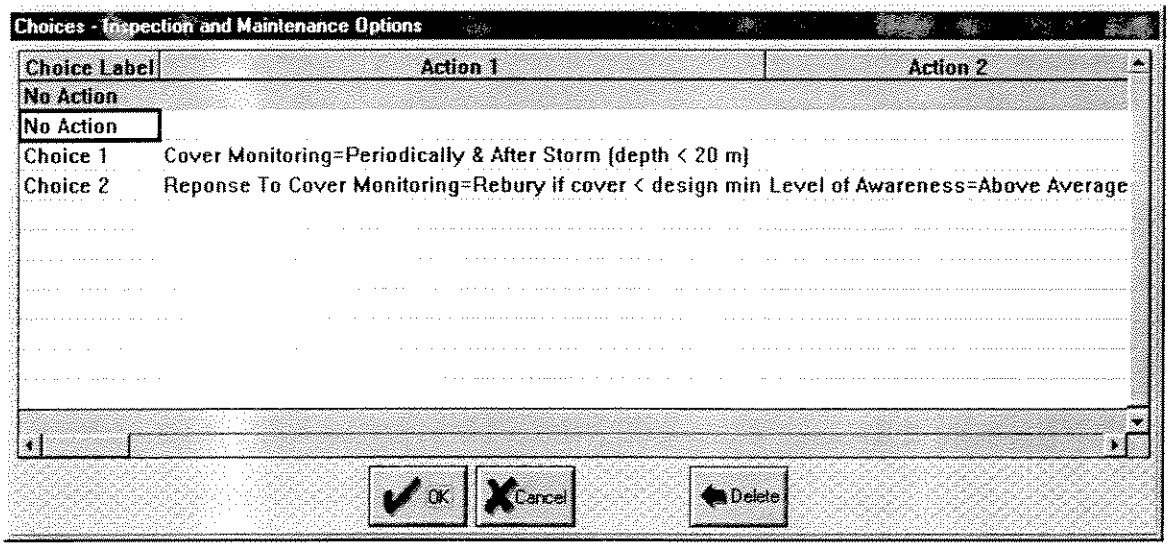
# 2.0 Influence Diagram Parameters

## 2.1 Choices Node Group

### 2.1.1 Choices Node

The parameter of this node is represented by a discrete set of preventative maintenance choices that are to be considered in the run.

**Choices Input Dialog Box**



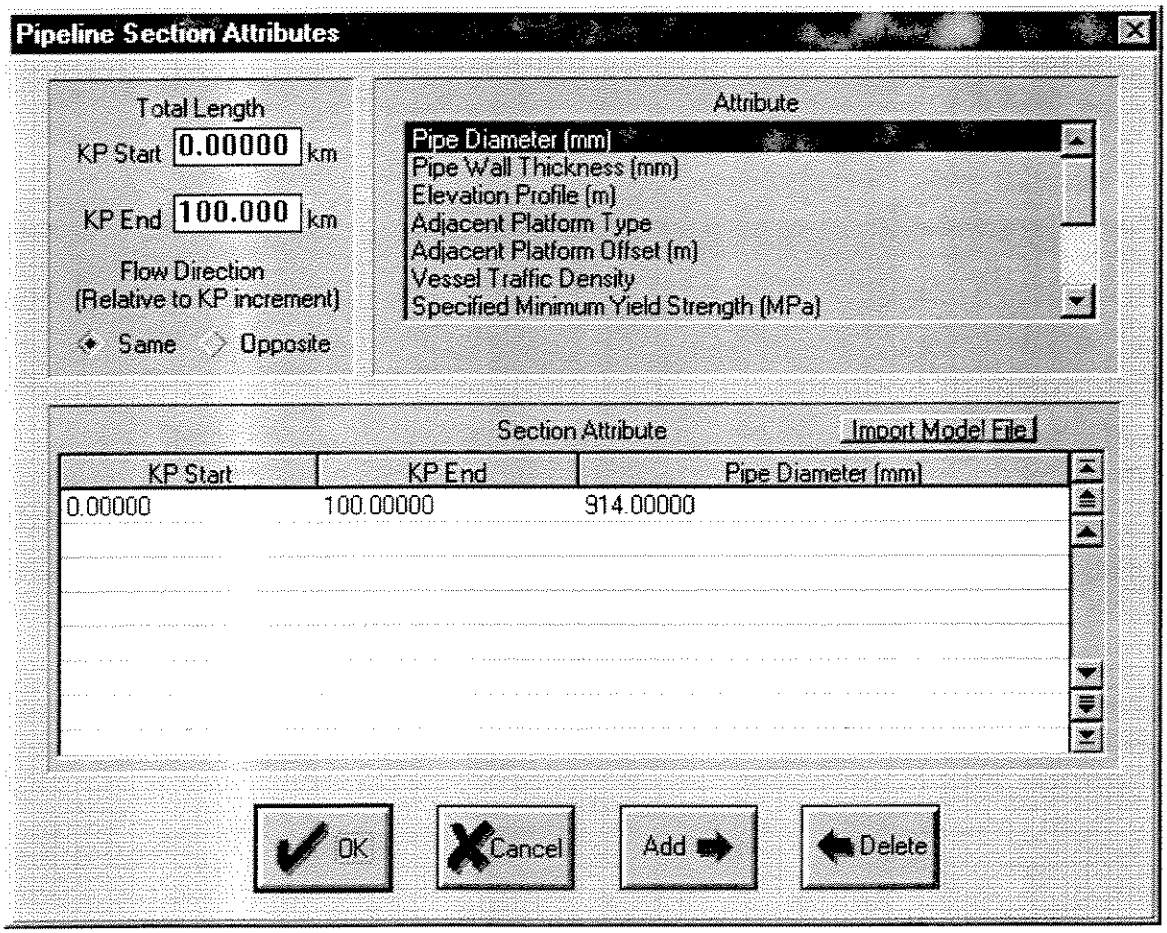

For each choice that is to be considered (except for the base case, identified by the choice label 'No Action'), the user must define the set of individual maintenance actions that are to be associated with the choice. Each choice is identified by a user-defined text string label entered in the first column of the table. Maintenance actions are selected from pull-down menus activated by clicking on cells in the row corresponding to the choice being specified.

## 2.2 Damage Potential Node Group

### 2.2.1 Damage Section Node

The node parameter is the section of pipeline that is associated with a mechanical interference event, where a section is defined as a length of pipeline for which all mechanical damage-related attributes are uniform (see also Section 4.2, PIRAMID Technical Reference Manual No. 9.x).

**Section Attributes Input Dialog Box**




The user must specify the value of each pipeline attribute displayed in the input dialog box. Attributes can be defined interactively via the input window and embedded spreadsheets or by importing a predefined segment data file.

Each attribute must be defined along the entire length of the pipeline segment being analyzed. Mechanical Damage related line attributes are as follows:

- **Pipe Diameter.** Outside diameter of the line pipe.

- **Pipe Wall Thickness.** Wall thickness of the line pipe.

- **Elevation Profile.** The relative elevation (-ve values implying depth below sea level) of the start and end points of the line segment and selected reference points along the length of the line (the location of intermediate points shall be chosen to adequately characterize the elevation profile given that the program uses linear interpolation to infer the elevation at all locations between specified reference points)

- **Adjacent Platform Type.** The type (*i.e.,* manned vs. unmanned, major vs. minor) and the location (*i.e.,* reference kilometer post) of permanent offshore facilities adjacent to the pipeline.

- **Adjacent Platform Offset.** The shortest perpendicular distance from a point on the sea surface directly above the pipeline to the centre of a permanent offshore facility adjacent to the pipeline.

- **Vessel Traffic Density.** The type (i.e., fishing vs. shipping) and density (i.e., low, medium, or high) of vessel traffic on the sea surface above the pipeline.

- **Specified Minimum Yield Strength.** Specified minimum yield strength of the line pipe.

- **Level of Awareness.** A relative assessment of the level of awareness of vessel and platform operators (*i.e.,* below average, average, or above average) regarding the presence of and hazards posed by buried pipelines.

- **Reference Cover.** The extent of seabed cover over the pipeline (*i.e.,* none, limited or variable, or greater than or equal to minimum requirements) at the start of the current inspection and maintenance cycle.

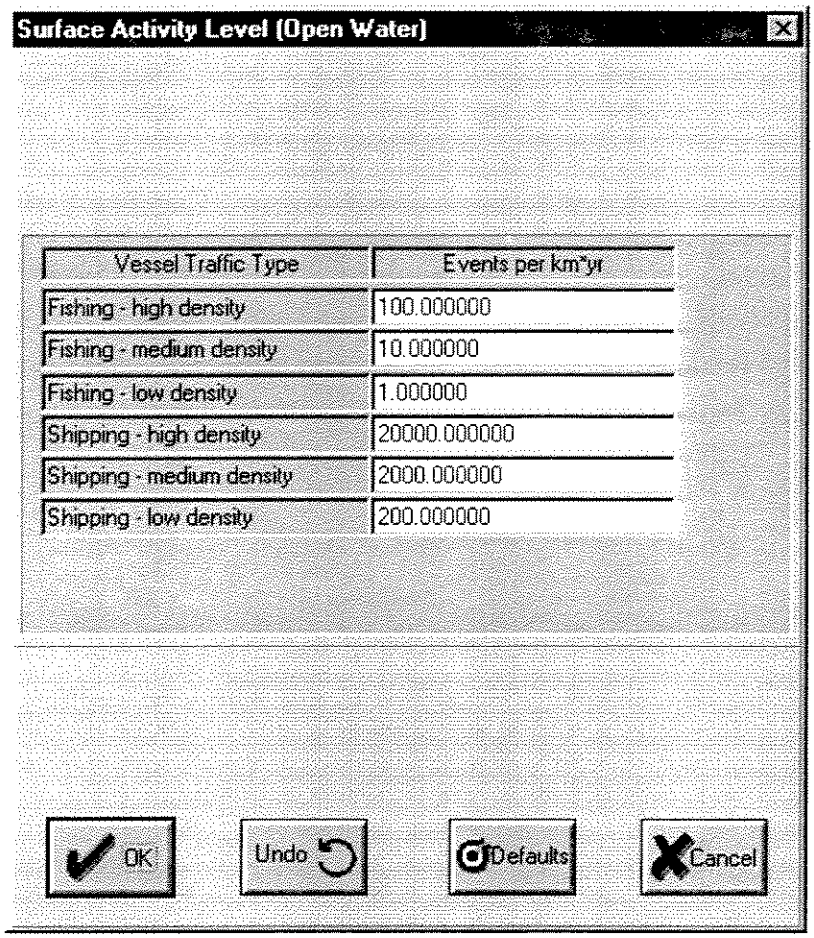
- **Cover Erosion Potential.** The potential (*i.e.,* yes or no) for the loss of seabed cover over the pipeline due to natural seabed degradation.

- **Cover Monitoring.** The frequency (*i.e.,* periodically and/or after storms) of subsea monitoring activities, as a function of water depth (*i.e.,* if less than 5 m or if less than 20 m), undertaken to verify the extent of pipe cover.

- **Response to Cover Monitoring.** The criteria that triggers the reburial of a pipeline following a subsea inspection to verify the extent of pipeline cover (*i.e.*, rebury if line exposed or rebury if cover is less than design minimum).

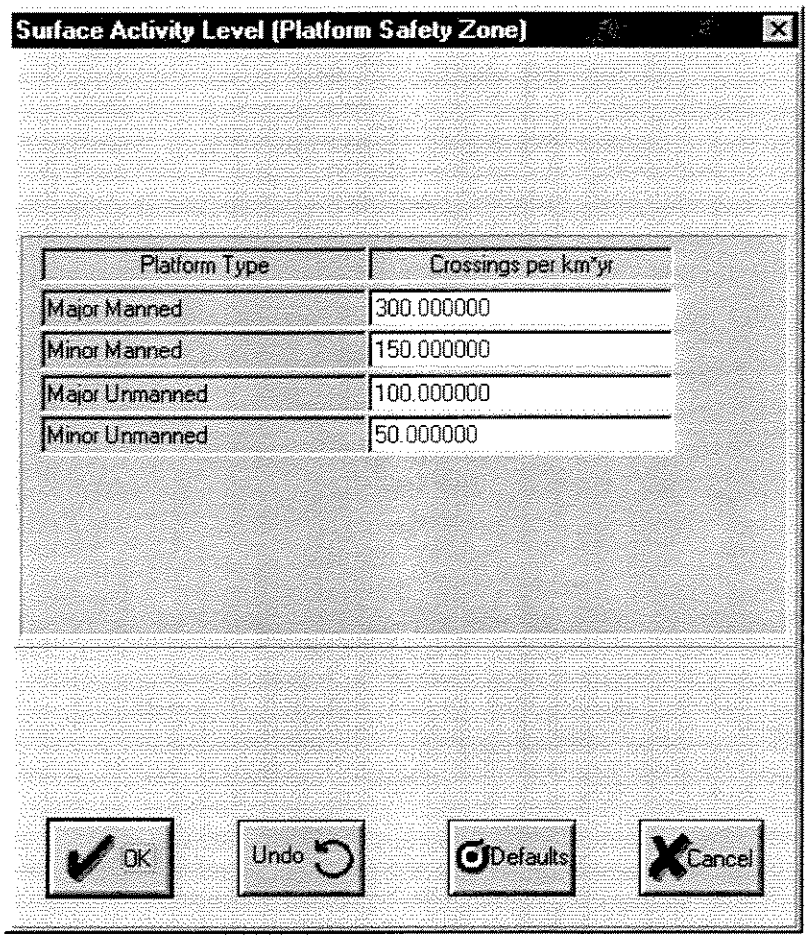
- **Mechanical Protection.** An indication of the type of physical protection, if any, that exists to effectively prevent direct contact between the pipe body and a grounded vessel hull or dragged equipment (*i.e.*, none, armoured jacket, or engineered backfill).

### Surface Activity Level (Open Water) Input Dialog Box




The user must specify the frequency of occurrence of pipeline crossing events for each surface vessel traffic type and density category specified via the Section Attributes input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

### Surface Activity Level (Platform Safety Zone) Input Dialog Box



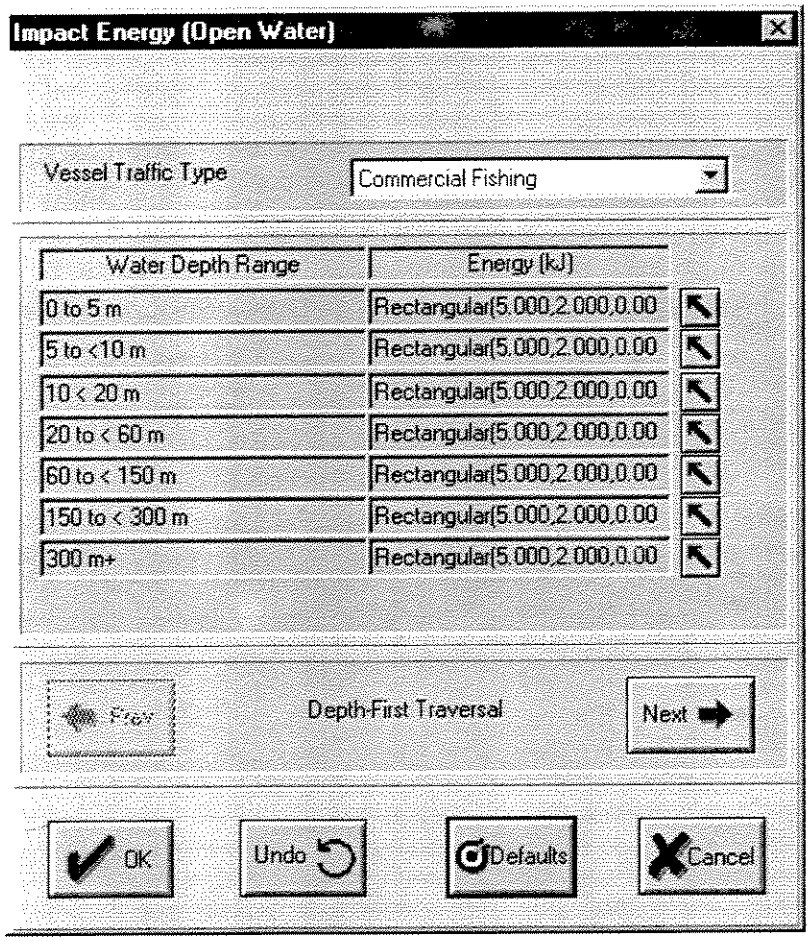

The user must specify the frequency of occurrence of pipeline crossing events for platform supply vessels servicing each platform type specified via the Section Attributes input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

## 2.3 Damage Characteristics Node Group

### 2.3.1 Impact Energy Node

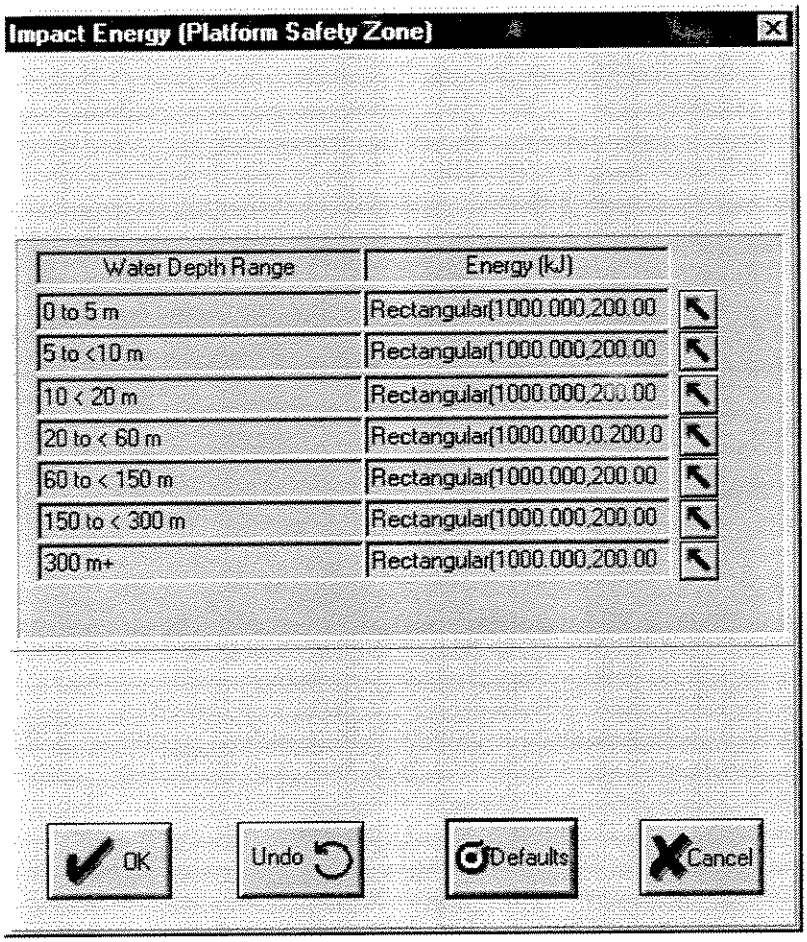
The node parameter is the energy associated with an impact event caused by dragged objects (see Section 5.2.1, PIRAMID Technical Reference Manual No. 9.x).

**Impact Energy (Open Water) Input Dialog Box**




The user must specify a probability distribution for the impact energy associated with dragged objects (*i.e.*, anchors or fishing net gear) for each vessel traffic type and water depth range specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

### Impact Energy (Platform Safety Zone) Input Dialog Box

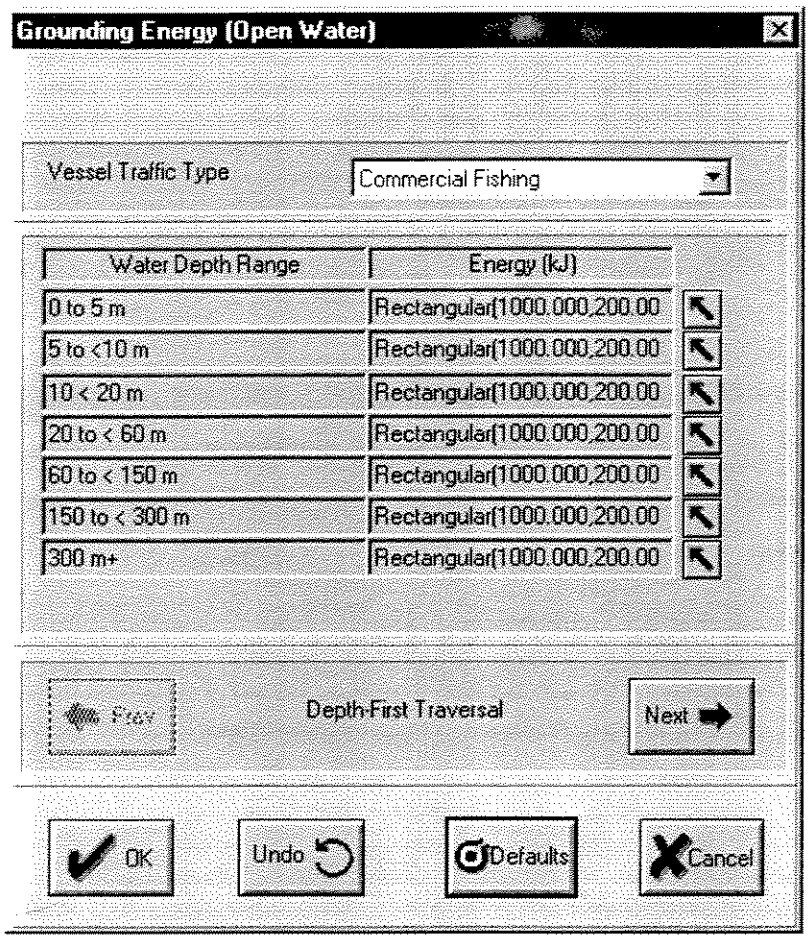



The user must specify a probability distribution for the impact energy associated with supply vessel anchor drag for each water depth range specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

## 2.3.2 Grounding Energy Node

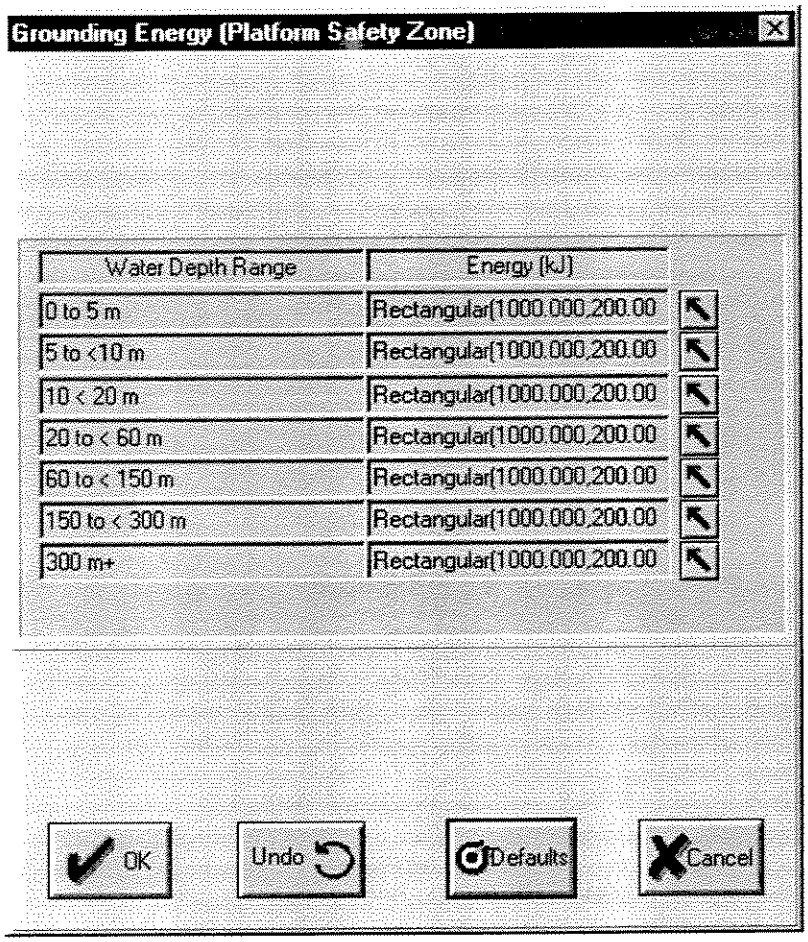
The node parameter is the energy associated with an impact event caused by vessel grounding (see Section 5.2.2, PIRAMID Technical Reference Manual No. 9.x).

**Grounding Energy (Open Water) Input Dialog Box**




The user must specify a probability distribution for the impact energy associated with vessel grounding for each vessel traffic type and water depth range specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

### Grounding Energy (Platform Safety Zone) Input Dialog Box



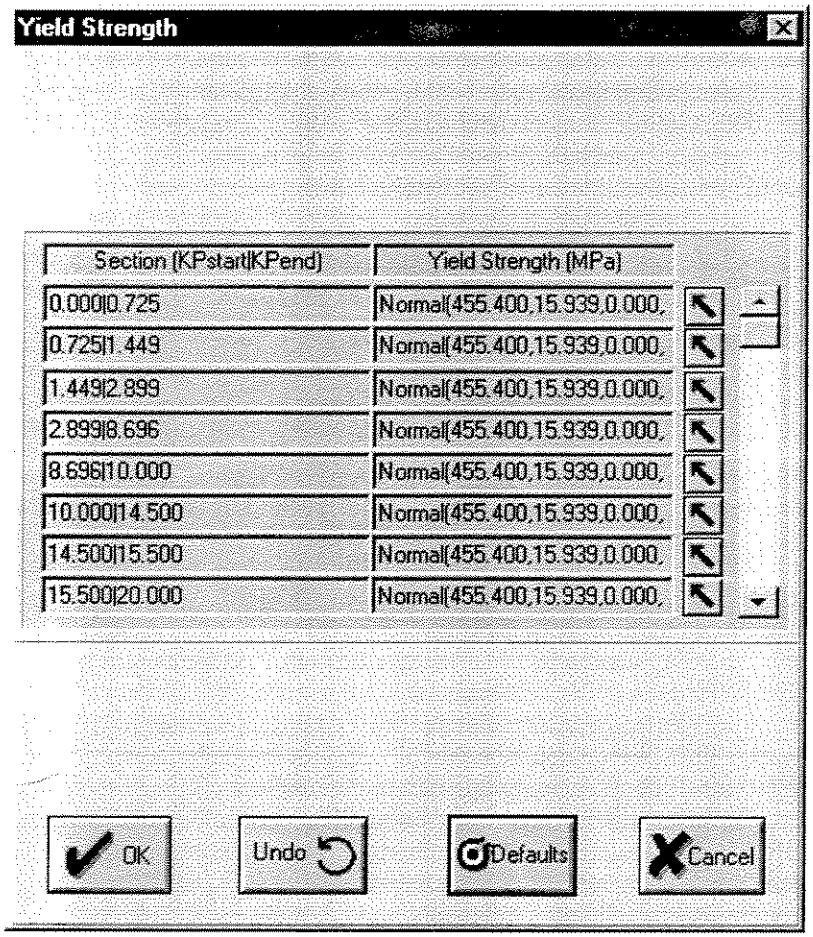

The user must specify a probability distribution for the impact energy associated with supply vessel grounding for each water depth range specified via the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

## 2.4 Mechanical Properties Node Group

### 2.4.1 Yield Strength Node

The node parameter is the actual yield strength of the pipe body. It is noted that the actual yield strength is related (but not equal) to the specified minimum yield strength (see Section 6.2.1, PIRAMID Technical Reference Manual No. 9.x).

**Yield Strength Input Dialog Box**



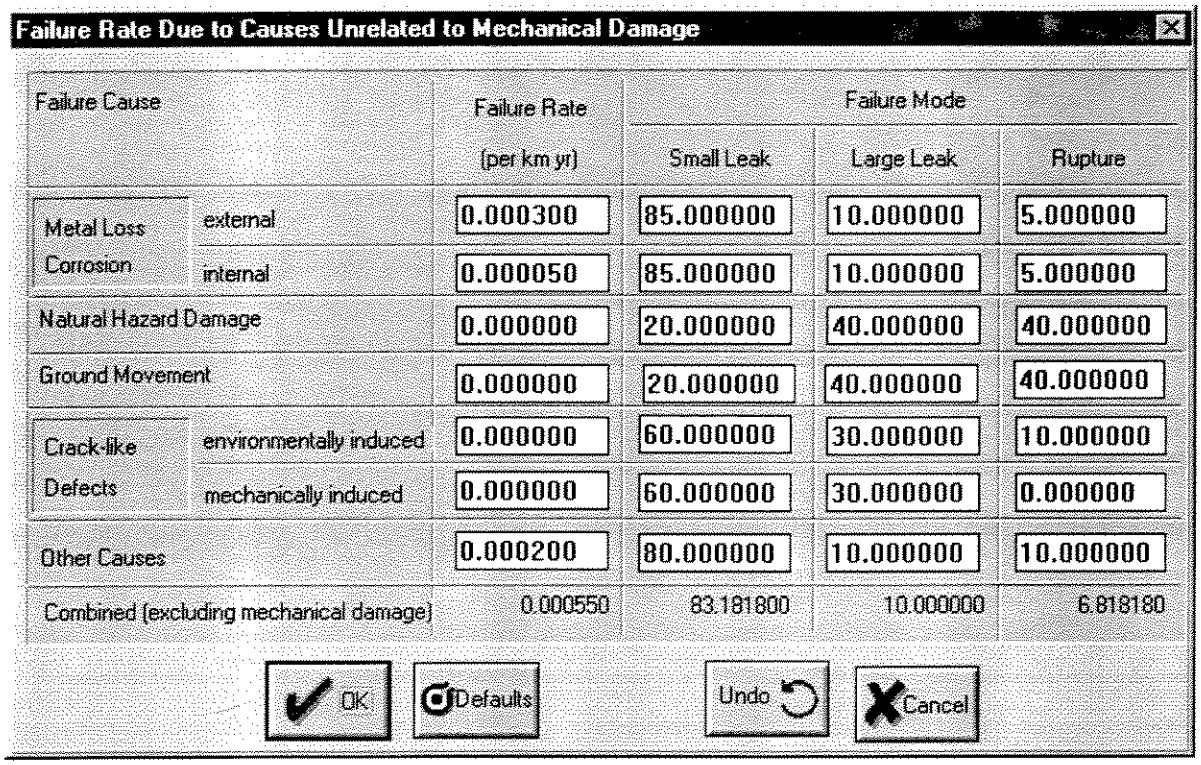

The user must specify a probability distribution for the pipe body yield strength applicable to each damage section defined by the Section Attributes input dialog box. The distribution type and distribution parameters are entered or modified via the Distribution Picker input dialog box. Alternatively, upon request by the user, a default distribution for each section, estimated from the specified yield strength, will be loaded into the model.

## 2.5 Pipe Performance Node Group

### 2.5.1 Segment Performance Node

The node parameter is the annual performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline segment (see Section 7.4, PIRAMID Technical Reference Manual No. 9.x). The parameter is calculated by combining the mechanical damage related performance (as calculated from previous nodes of the diagram) with the performance related to other failure mechanisms as defined in the input dialog box for this node.

**Segment Performance Input Dialog Box**




The segment performance input dialog box allows the user to define the annual failure rates due to different causes excluding mechanical damage. In the column titled *Failure Rate* the user must input the failure rate per km year for each failure cause indicated in the Table. In the columns titled *Relative Failure Mode Frequency* the user must define the relative frequency in % (a number between 0 and 100) of each failure mode for each failure cause. (Note that the total relative frequencies for small leak, large leak, and rupture must add up to 100 for each failure cause.) Alternatively, upon request by the user, default data stored in the PIROFMD.INI file will be loaded into the model.

Once the user input is complete, the program will calculate the total failure rate and relative frequencies associated with different failure causes (excluding mechanical damage) and display them on the bottom row of the table.

### 2.5.2 Performance Given Impact Node

The node parameter is the performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline at a randomly selected mechanical interference event caused by a dragged object (see Section 7.2, PIRAMID Technical Reference Manual No. 9.x). No user input is required to specify this node.

### 2.5.3 Performance Given Grounding Node

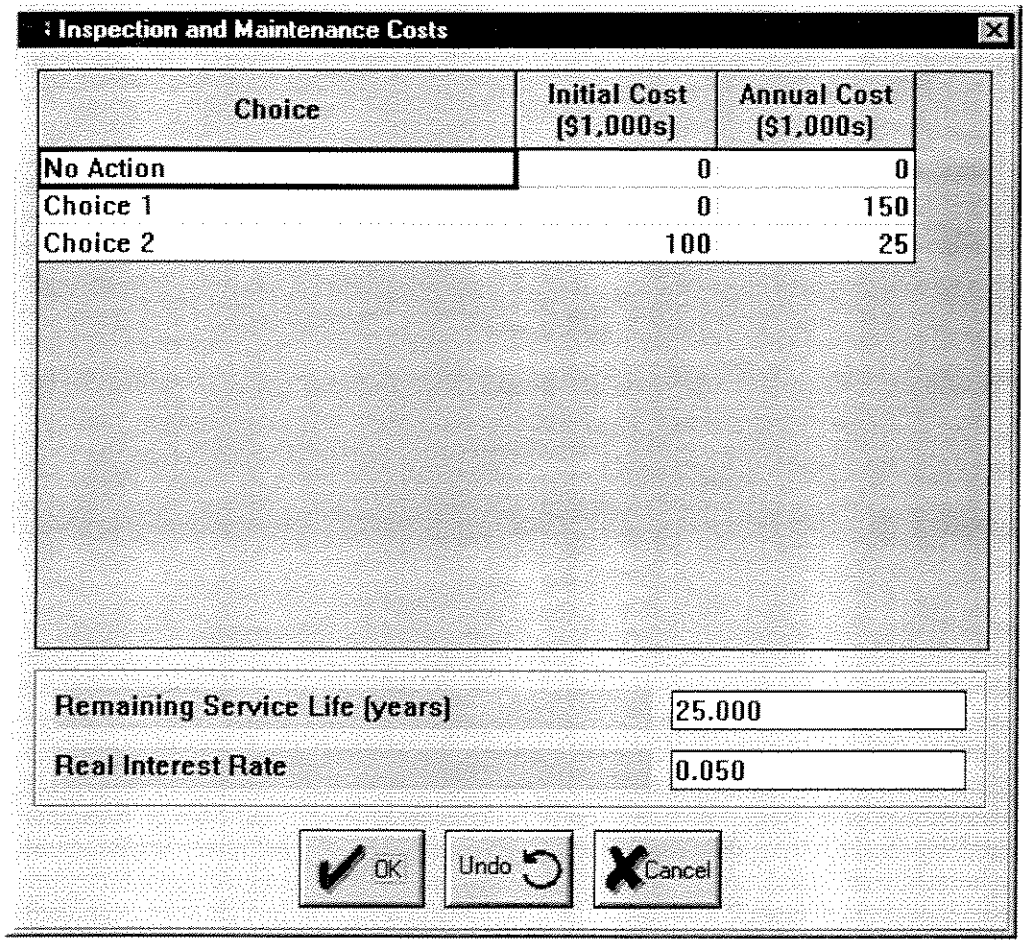
The node parameter is the performance (*i.e.*, safe, small leak, large leak, or rupture) of the pipeline at a randomly selected mechanical interference event caused by vessel grounding (see Section 7.3, PIRAMID Technical Reference Manual No. 9.x). No user input is required to specify this node.

## 2.6 Repair and Interruption Cost Node Group

### 2.6.1 Maintenance Cost Node

The node parameter is the total annual maintenance cost for the whole pipeline segment in present value currency (see Section 9.2, PIRAMID Technical Reference Manual No. 9.x). The maintenance cost is calculated as the sum of the annual cost and the annualized equivalent of the initial cost associated with implementing the prescribed set of maintenance actions.

**Maintenance Cost Input Dialog Box**




The initial maintenance cost dialog box requires the user to define the individual cost components associated with each choice. In addition, the user must input the expected remaining service life of the pipeline and the real interest rate (where the real interest rate is defined as the difference between the actual interest rate and the inflation rate).