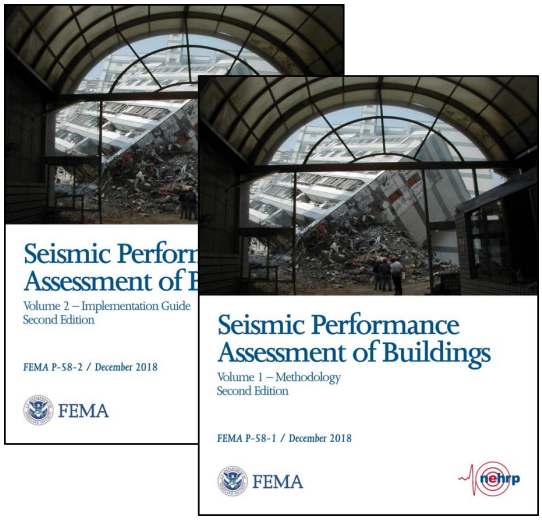


Background Document

FEMA P-58/BD-3.7.1

PACT Technical Manual

Version 3.1.2

Prepared by

Scott Hagie
John A. Martin and Associates, Inc.
950 S. Grand Ave. 4th Floor
Los Angeles, California 90015

Submitted to

APPLIED TECHNOLOGY COUNCIL
201 Redwood Shores Parkway, Suite 240
Redwood City, California 94065
www.ATCouncil.org

Prepared for

FEDERAL EMERGENCY MANAGEMENT AGENCY
U.S. Department of Homeland Security
500 C Street, SW
Washington, D.C. 20472

June 2018

FEMA

nehrp

Background Documentation

FEMA P-58 Background Documents are a series of reports documenting the technical background and source information for key aspects of the FEMA P-58 methodology and its implementation. This report was developed over the course of the 5-year ATC-58-2 Project funded under FEMA Contract HSFE60-12-C-0243.

Background Documents were developed by consultants, serving at various levels within the project hierarchy, reporting the results of: (1) decisions on technical development protocols; (2) focused studies on the development of key aspects of the methodology; (3) documentation of recommended procedures; and (4) collection of available data for the development of structural and nonstructural fragilities. They were initially intended to serve as a record of the technical state-of-knowledge at the time they were produced, and as resources for the development of the eventual project reports. As such, they represent a snapshot in time, and may, or may not, match the technical content, recommended procedures, or data incorporated into the final methodology and its implementation.

This Background Document is intended for the purpose of providing supplemental knowledge to users of the FEMA P-58 methodology. Information contained herein has not been independently verified for accuracy as a stand-alone document, and may have been superseded in its final implementation within the methodology. Specifically in the case of certain nonstructural component fragilities, the NISTIR fragility classification numbering scheme was modified over the course of the project, and the fragility classification number assigned in this document might be different from numbers assigned in the final fragility database. Users of information in this document assume all liability arising from such use.

Notice

Cover photograph – Primary resource documents for the FEMA P-58 *Seismic Performance Assessment of Buildings, Methodology and Implementation* series of products: FEMA P-58-1, *Volume 1 – Methodology, Second Edition*, and FEMA P-58-2, *Volume 2 – Implementation Guide, Second Edition*.

Pact Technical Manual
Version 3.0.3
(September 2016)

including Appendix B: Environmental Loss Computation in PACT 3.1.2 (February 2017)

updated June 2018

INDEX

1 Programming Structure Overview

Terminology notes:

For various reasons, some terminology in the source code and some files is not the official terminology used in the ATC58 guidelines and in the Pact 2 user interface. Some effort has been made to correct these inconsistencies in the code, but many issue still remain. Here is a list of the more common ones:

- "Analysis" in the code most often refers to a scenario or intensity.
- "Run" and "Runs" also generally refer to scenarios or intensities.
- "Percent" is frequently used for probability, instead of the more correct "fraction". All internal probability variables are between 0 and 1 (0 <= r < 1), regardless of the name, not 0 and 100 is the name may imply.
- Red Tag is still sometimes used instead of the newer terminology, Unsafe Placard.

Other notes:

C# is always zero based for arrays and lists. That means floor 1 is floor[0], realization 1 is Realization[0], etc.

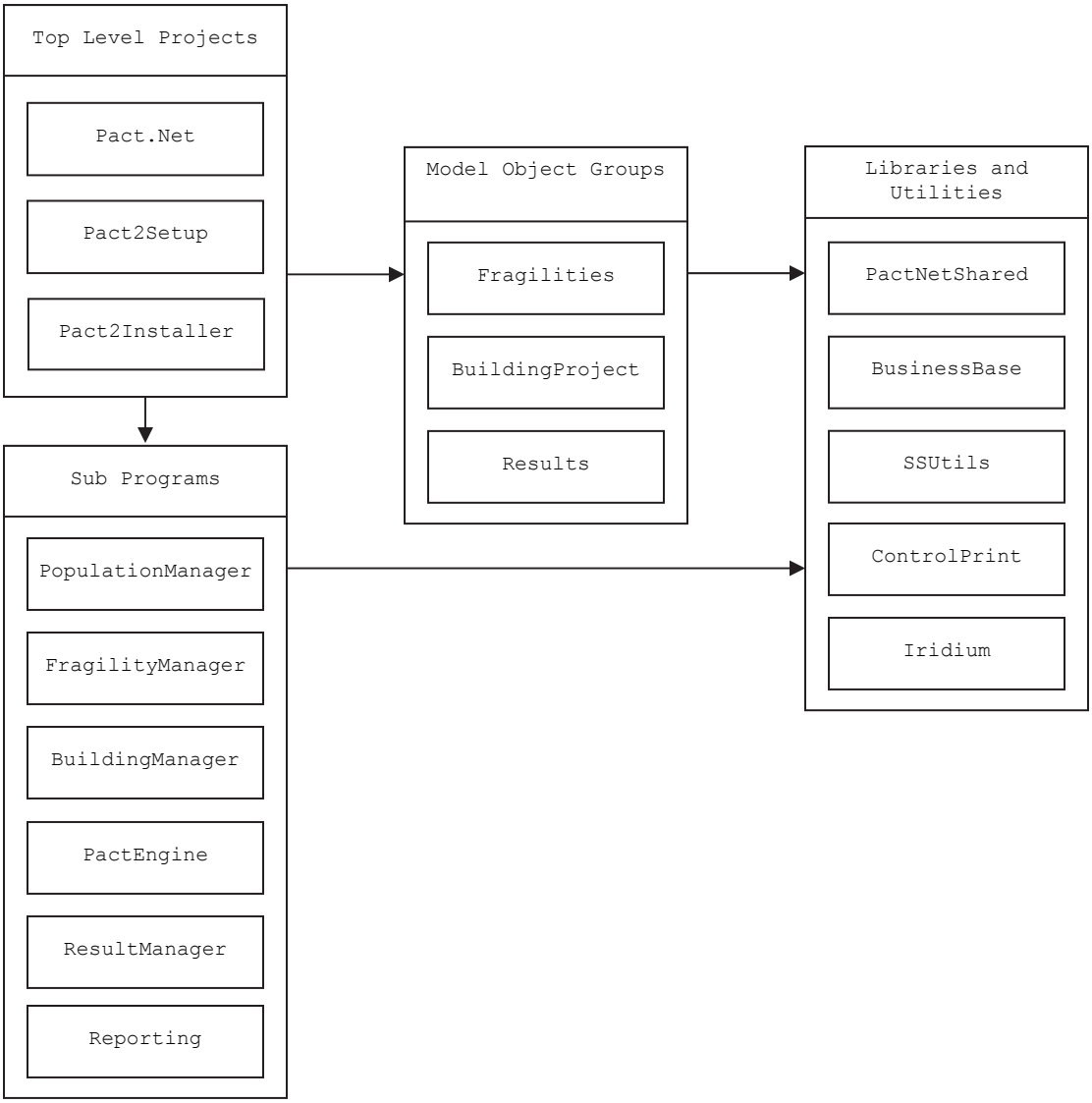
Direction is internally referred to with an integer number. 0 is direction 1, 1 is direction 2, and 2 is non-directional.

Performance Group Correlation is sometimes defined as a Boolean, i.e. are we correlated or are we not. Other times it is defined as a double, with 0 being not correlated and 1 being fully correlated. When it is defined as a double, 0 and 1 are the only allowable values, since the program does not deal with any partial correlation.

1.1 High Level Program Overview

Visual Studio.Net programs are called "solutions", and a solution can contain multiple "projects", which are logically independent units containing programming code, forms, and resources such as graphics or files. While each project is logically separate, they can have dependencies on other projects.

The Pact2 solution, which is called PactNet internally, is divided into sixteen projects, with a web of dependencies between projects. The projects can be divided into several different categories.



1. Top Level Projects – Master Projects that contain references to almost everything else in solution.
    a. Pact.Net – The main Pact2 executable, the main menu and the splashscreen.
    b. PactSetup – The Microsoft installer project.
2. Sub Programs – Each of these five projects is a logically separate sub-program that is normally run from the main menu control panel. Each should also be able to be compiled and run without requiring any of the other sub-programs or the top level projects. The will usually have dependency references to the types of objects they work on (buildings, fragilities, etc) as well as references to the libraries and utilities.
    a. PopulationManager – The Population Modeler forms
    b. FragilityManager – The Fragility Manager forms
    c. BuildingManager – The Building Manager forms
    d. PactEngine – The Main Engine for running Pact 2 performance evaluations
    e. ResultManager – The Result Manager, for viewing evaluation results
    f. Reporting – For generating the automated reports
3. Logical Object Groups – Each of these is a group of classes that model a particular set of object types. They will have dependency references to other logical object groups and libraries and utilities.
    a. Fragilities – Classes describing Fragilities, Damage States, and related data
    b. BuildingProject – Classes describing buildings, floors, performance groups, hazard curves, intensities, and the project files that contain them.
    c. Results – Classes describing evaluation results and the calculations required to come up with them.
4. Libraries and Utilities – Various libraries and utility projects created to support the other projects and classes. May contain references to other library and utility projects, but most contain no solution references.
    a. PactNetShared – A class containing various solution specific code and forms, as well as most global information and most of the resources, such as sample projects and fragilities.
    b. BusinessBase – A collection of base classes for many of the other classes in the solution. Based on Rockford Lhotka's Expert Business Objects.
    c. SSUtils – A large library routine of low-level programming routines, custom controls, and control extensions.
    d. ControlPrint – A custom control class to handle printing in forms
    e. Iridium – A set of open source classes to do math routines, including matrix operations and statistical analysis. It has been slightly modified from the original code, which may have copyright implications (see Third Party Control section for details.)
    f. OpenXMLUtils – A set of functions to help generate Microsft Word files for the reporting

1.2 Top Level Projects Overview

1.2.1 Pact.Net Project

The Pact.Net Project is the highest level project in the solution, not counting the installer project. It contains the following:

- frmSplashScreen – The splash screen form
- MenuForm – The Main Menu (or Control Panel) form
- MiscResources.resx – Contains the splash screen image
- StartUp.cs – A simple for starting the program
- app.config – The default configuration file for the user options

The project contains references to most of the other projects in the solution

1.2.2 PactSetup Project

This is a Microsoft setup project for the solution. Use this to generate an installable file.

It has a (non-standard) dependency reference to Pact.Net.

1.3 Sub Programs Overview

1.3.1 PopulationManager Project

This contains the forms for the population manager.

- frmPopulationManager – The main form for the population manager
- frmPopulationModel – A subform that shows the data for a single population model. This is used both by frmPopulationManager and frmBuildingManager.cs

It references Fragilities, PactNetShared, BusinessBase, ControlPrint, SSUtils, log4net, and the Nevron Chart dlls.

1.3.2 FragilityManager Project

This contains the forms for the Fragility Manager.

- frmEditFragility – The main form for the fragility manager
- frmAddNewFragility – The form used when adding a new fragility to request the ID
- frmAddNewEDPType – The form used when adding or editing a new EDP type
- frmImportFromExcel – The form to input fragilities from Excel

It references Fragilities, PactNetShared, BusinessBase, ControlPrint, SSUtils, log4net, and the Nevron Chart dlls.

1.3.3 BuildingManager Project

This contains the forms for the Building Manager.

- frmBuildingManager – The main form for the building manager
- frmBuildingManagerPGChooser – The form used when choosing a new Performance Group
- frmEDPGroupNameCreate – The form used when creating a new custom EDP Group Name
- frmBuildingImportEDPs – A form to import EDP information
- frmBuildingExportEDPs – A form to export EDP information

- frmHazardCurveRanges – A form to input hazard curves

It references PopulationManager, BuildingProject, Fragilities, PactNetShared, BusinessBase, ControlPrint, SSUtils, log4net, and the Nevron Chart dlls.

1.3.4 PactEngine Project

Contains the form for the Main Engine.

- frmEngine – The main form for the engine

It references BuildingProject, Results, Fragilities, PactNetShared, BusinessBase, SSUtils, log4net, and Iridium.

1.3.5 ResultManager Project

Contains the forms for the Results Manager

- frmResults1 – The main form for the results
- frmResultsTimeBased – The form for showing time-based results
- frmResultDrillDown – The data drill down form
- frmResultsOneDSGroup – The data drill down form showing one damage state group
- frmResultOneConsequence – The data drill down for showing one consequence
- frmResultRedTag – A form to show the results of Unsafe Placards

It references BuildingProject, Results, Fragilities, PactNetShared, BusinessBase, SSUtils, log4net, and the Nevron Chart dlls.

1.3.6 Reporting

Contains the form for generating automatic Microsoft Word reports

- frmCeateInputReport – The main form used in creating reports
- frmReportOptions – A form to change user options when creating the report
- clsResultReport – A class representing a Result Report
- clsProjectReport – A class representing a Project report

Logical Object Groups Overview

1.3.7 Fragilities Project

Contains all the classes having to do with fragilities.

clsFragilityCurveLibraries – All of the fragility curve libraries
    clsFragilityCurveLibrary – A single fragility curve library
        clsFragilityCurve – A single fragility curve
            clsEDPType - The EDP type information
                clsFragilityRating - The rating of a particular fragility
                clsFragDamageStates – All the damage states
                    clsFragDamageState - A single damage state base class
                    clsFragSeqDamageState - A single sequential damage state
                    clsFragMutExDamageState – A single mutually exclusive damage state
                    clsFragSimultDamageState – single Simultaneous damage state
                        clsFragConsequenceGroup - A single group of consequences
                            clsFragStandardConsequence – Consequences base class
                            clsFragCostConsequence - The cost consequences
                            clsFragTimeConsequence – The time consequences

clsElementCodes - The master list of element codes
clsFormulaEntry – Not Used
clsFormulaVariable – Not Used
clsOccupancyTemplate – Not Used
Utilities.cs – Misc. utilities for the fragility functions

Also included in this project are some data files:

ATCCurves/*.* - All of the Fragility Curves
DataFiles/ElementLibraryCodes.xml – All of the Uniformat codes
DataFiles/Measures.xml – Not Used
DataFiles/AllElementLibraryDescriptions.xml – Unused
Resources/BasicEDPTypes.xml – The default EDP types
Resources/OpeningScreen.pdn – Not Used
Resources/ucum-essence.xml – Not Used

1.3.8 BuildingProject Project

Contains all the classes describing the building and building project.

clsProject - The root Project object, containing the building and runs info

- clsBuilding - The Building object
  - clsBuildingFloor - A Building Floor.
    - clsBuildingDirection - A particular floor in a particular direction
      - clsPerformanceGroup - The quantity of a particular floor/direction and fragility from a particular population and EDP Group Name
    - clsPopulationPercent - The population percent of the floor that it affects
  - clsCollapseModes - The collection of possible collapse modes
    - clsCollapseMode - A single collapse mode
  - clsDefaults - The defaults for the items (Core and shell or nonstructural)
    - clsDefaultItem - The defaults for an item
- clsRuns - All the intensities or scenarios
  - clsRun – A single intensity or scenario
    - clsEDPInput - The base type for the EDP input for a particular direction and Intensity or Scenario
    - clsEDPInputNonlinear – The non-linear EDP input for a particular direction and Intensity or Scenario
    - clsEDPInputSimplified - The simplified EDP input for a particular direction and Intensity or Scenario
      - clsEDPArray - The EDP input for a particular intensity, direction, group name, and EDP type (drift, acc, etc.)

clsHazardCurve - Not Used

clsHazardPoint - Not Used

clsShellTemplate - Not Used

1.3.9 Results Project

Contains all the classes showing the result information.

clsResultSet - One complete result set, for all intensities

clsResultAnalysis – The results for one intensity or scenario

clsResultAnalysisEDPs - The base class for a collection of all calculated EDP values for a intensity

clsResultAnalysisEDPEigenDecomp2 - A collection of all calculated EDP values for this analysis case using Farzin's alternative decomposition method

clsResultAnalysisEDPCommon - All the generated values for all the EDPs for a particular Analysis and Direction. The base class for clsResultDirectionEDPEigenDecomp

clsResultDirectionEDPEigenDecomp - All the generated values for all the EDPs for a particular Analysis and Direction for the Eigen value decomposition method.

clsResultAnalysisEDPSimplified - The class for a collection of all (normally distributed) calculated EDP values for this analysis case

clsResultRealizationEDPCollection - A class holding all the original info for a particular EDP type (Drift, accel, etc) for a particular analysis and realization (all directions, and all floors)

clsResultReal - One Result/Analysis Run/Realization

clsResultRealFrag - Every single unique fragility used in this realization. Needed to help with RedTags and quantity discounts

clsResultRealDS - Every unique damage state used in this fragility for this realization, used for unsafe placards

clsResultFloor - One Result/Analysis Run/Realization/Floor

clsResultDirection - One Result/Analysis Run/Realization/Floor/Direction

clsResultPG - One Result/Analysis Run/Direction/Realization/Floor/PG

clsResultItemGroup - One Result/Analysis Run/Direction/Realization/ Floor/PG/item group

clsResultDSGroup - Base class for one Result/Analysis Run/Direction/Realization/ Floor/PG/(dsgroup/DS)*/Damage State Group

clsResultDSGroupMutExCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Mutually Exclusive Correlated Damage State Group

clsResultDSGroupSeqCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Sequential Correlated Damage State Group

clsResultDSGroupSimultCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Sequential Correlated Damage State Group

clsResultDS – Base class for One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Damage State

clsResultDSMutExCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Correlated Mutually Exclusive Damage State

clsResultDSSeqCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Seq Damage State

clsResultDSSimultCor - One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Simultaneous Damage State

clsResultConsequence - One Result consequence

clsResultAnalysisView - A single result set for a single analysis case and a specified set of floors, dirs, and PGs

clsResultAnalysisViewCurve - A transformation into a curve for a single analysis case view Base class for the Log normal fitted curve and histogram

clsResultAnalysisViewFittedCurve - A log normally fitted curve for a single analysis case and a specified set of floors, dirs, and PGs

clsResultAnalysisViewHistogram - A single result set of binned data

clsResultRedTagItem - A class representing a single fragility and damage state, to help determine quantity for unsafe placards

clsResultSetView - A class that holds the result view for all analysis cases

clsResultSetViewCurve - The collection of all results in a set...i.e. all analysis runs

clsResultSetViewFittedCurve - A class that holds all the fitted curves view, and the adjusted total, for all analysis cases

clsResultSetViewHistogram - A class that holds all the fitted curves view, and the adjusted total, for all analysis cases

SSRandom – A repeatable random number generator

clsResultAnalysisEDPCholesky – Unused

clsResultAnalysisEDPNormal – Unused

clsResultAnalysisEDPSimplified – Unused

clsResultDirectionEDPCholesky – Unused

1.4 Libraries And Utilities

1.4.1 PactNetShared Project

This contains several miscellaneous forms used by many other forms

- frmAbout – The about form of the help menu
- frmUserOptions – The user options form to change measurement systems

It also contains several miscellaneous classes

- clsUserOptions – A class to help save and load user options
- NevronRoutines – A class with helper routines for the Nevron graph controls
- Globals – A class containing global variable and routines

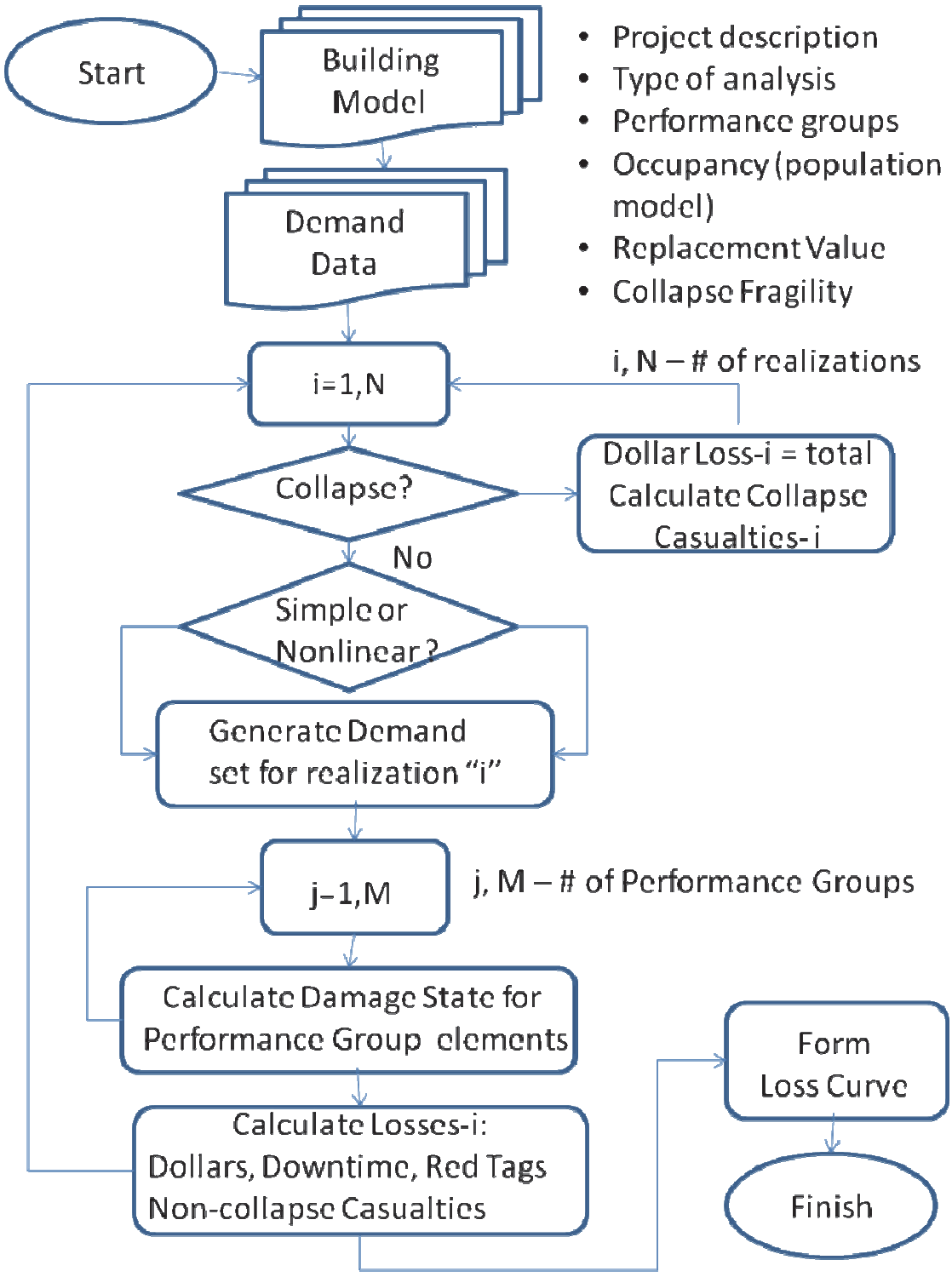
And several other files

- app.config – The main configuration and user settings file
- Log4Net.config – The configuration file for the Log4Net logging routines
- Settings1.settings – The default user options file

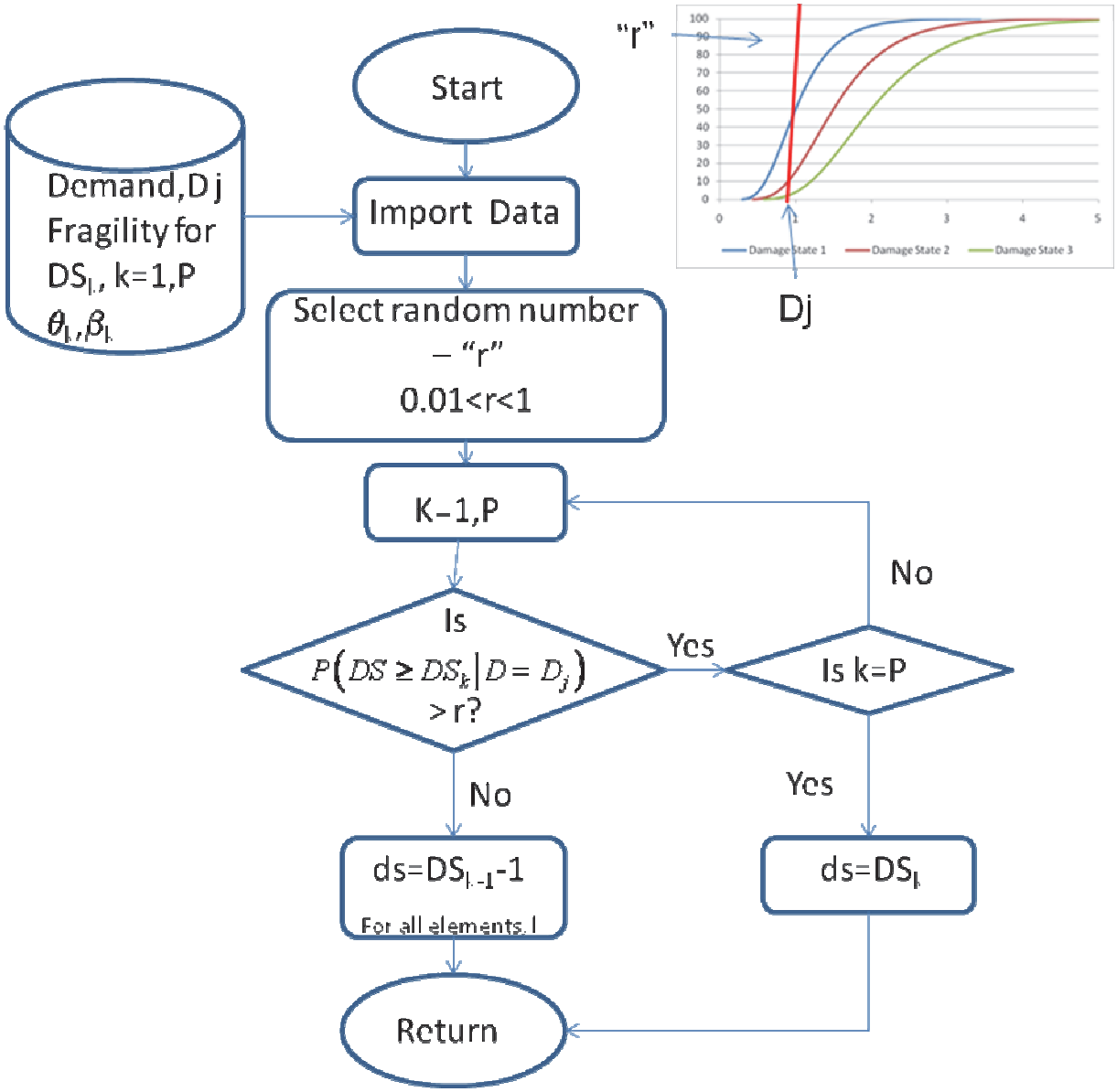
Finally, it is where all of the sample projects and population models are located, in the Projects and Templates directories respectively.

2 Flowcharts

2.1 Global Logic – Intensity Based

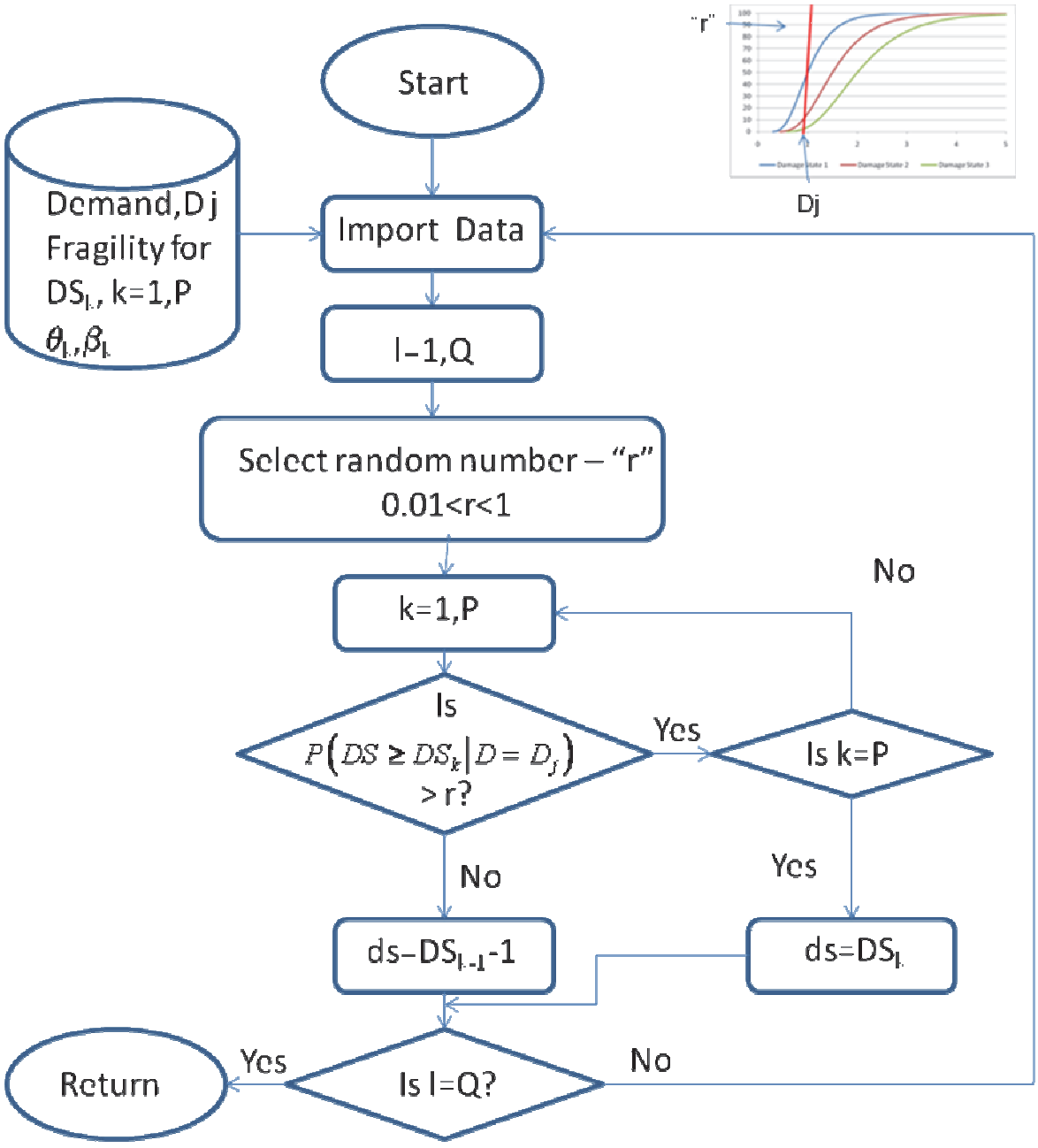


2.2 Calculate Performance Group Damage: Correlated, Sequential



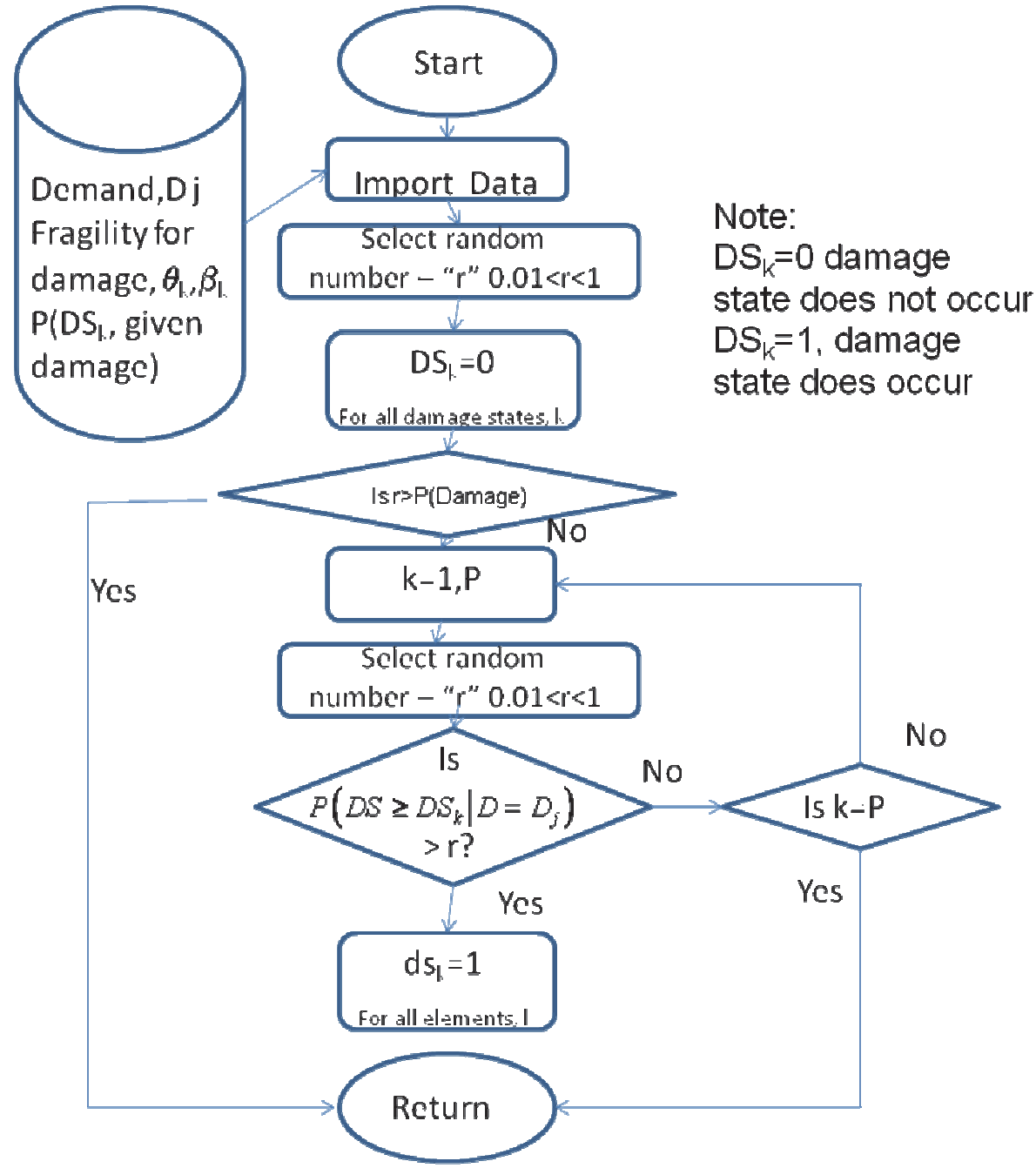
k= damage state index ordered from no damage (k=0) to highest (k=P)

2.3 Calculating Performance Group Damage: Uncorrelated, Sequential

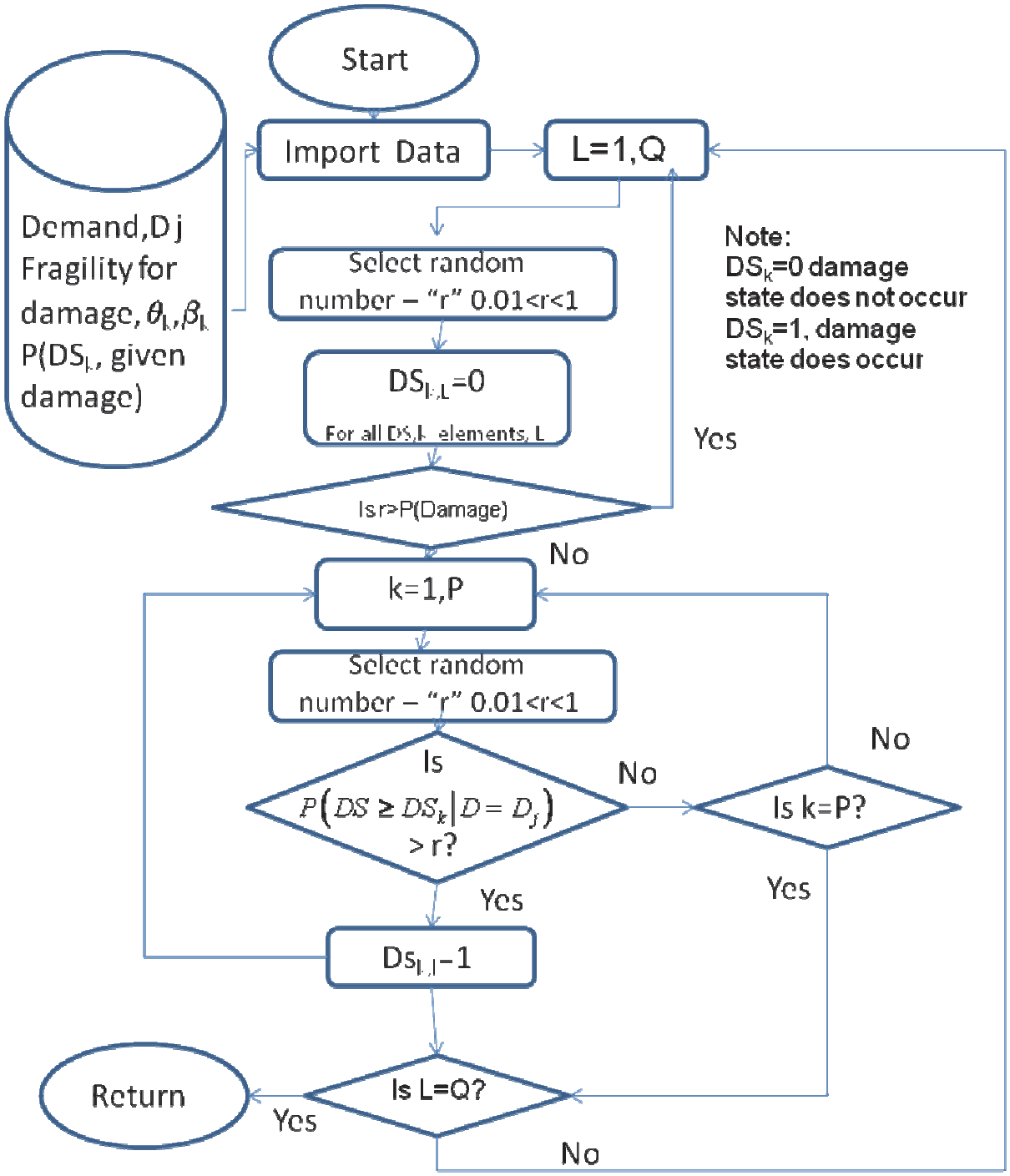


k= damage state index ordered from no damage (k=0) to highest (k=P)
l = element index, from 1 to Q, where Q is # of elements in group

2.4 Calculate Performance Group Damage: Correlated, Simultaneous

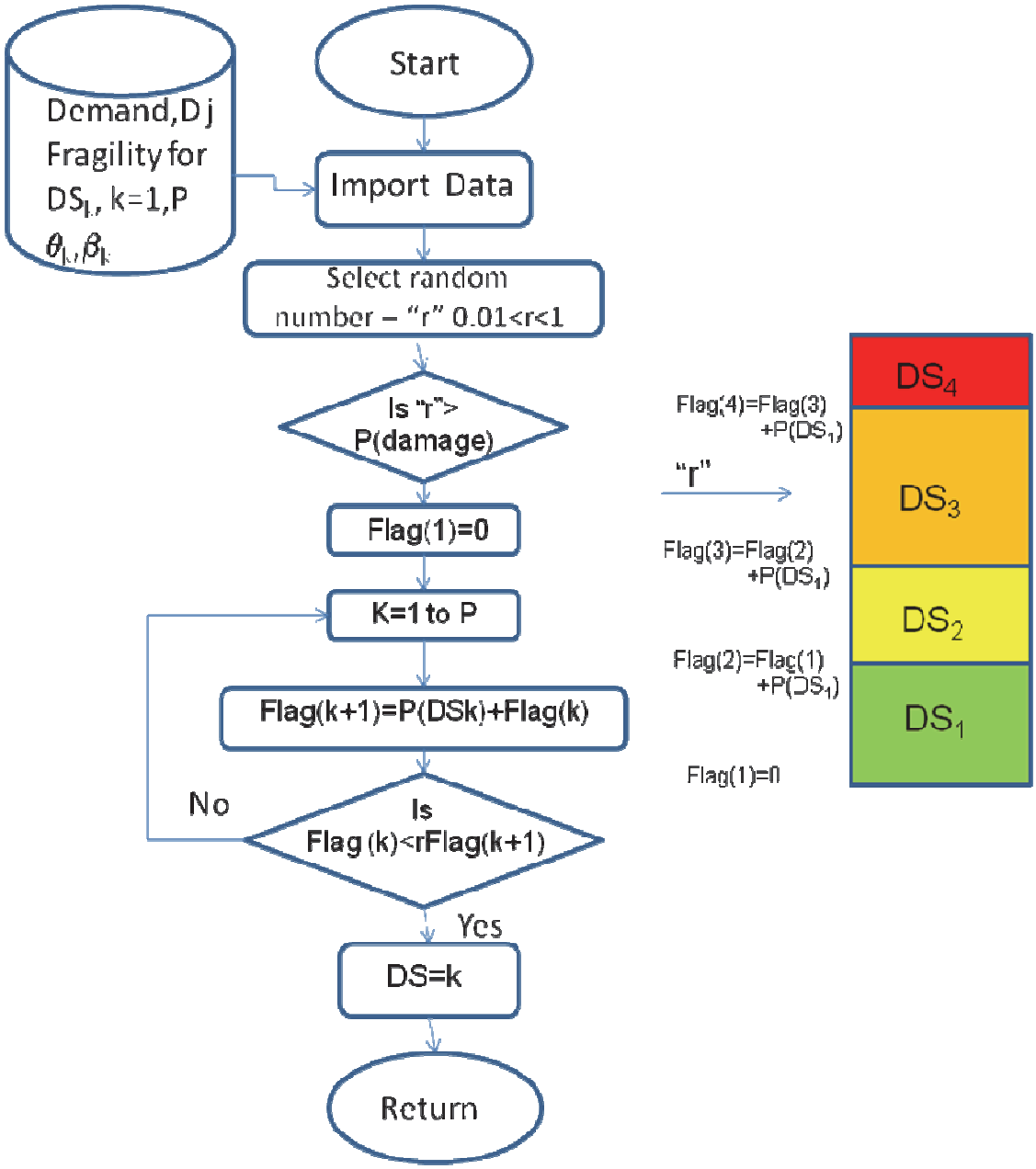


2.5 Calculate Performance Group Damage: Uncorrelated, Simultaneous



k= damage state index k=1 to P; l = element index from 1 to Q

2.6 Calculate Performance Group Damage: Correlated, Mutually Exclusive



k= damage state index 1, to p; Flag(k) = cumulative probability that DS has occurred, ordered from 0 to 100

3 Algorithm Overview

3.1 Engine Logic Overview

```
Read Building Model
Read All Fragility Definitions

Read Population Model

Create List of all Realizations

Create list of all Performance Groups

For Each Realization
    Create Matrix of all EDP Values
Next Realization

For Each Realization
    Do Realization Calculations of damage states and consequences
Next Realization
```

3.2 Realization Calculations Overview

```
Pick a random day/time

For Each Floor
    Calculate Population
Next Floor

Determine if collapsed, and if so
        Get cost and time from replacement values
        Get casualties from population model(s) and collapse mode
        Mark as unsafe placard
        skip to next realization

For Each Floor
    For Each Direction
        For Each Performance Group
            Determine actual quantity

            If Correlated, create one Item Group
            If Uncorrelated, create multiple Item Groups

            For Each Item Group
                Determine Damage State(s)
            Next Item Group
        Next Performance Group
    Next Direction
Next Floor

To determine cost and time discounts, get the total quantity of all damaged items,
grouped by fragility

For Each Floor
    For Each Direction
        For Each Performance Group
            Determine cost discount and calculate cost
            Determine time discount and calculate time
            Determine Death and Injury areas, grouped by population
        Next Performance Group
    Next Direction
```

```
    Sum up the total man-hours required by floor
    Calculate the total number of workers per floor by multiplying the floor area times
        the Max Workers Per Square Foot
    Calculate the total time by dividing the man-hours required by the number of workers
    Sum up Death and Injury areas by floor
    Use Population to determine casualties
    Apply floor multipliers to cost
Next Floor

Sum up cost
Sum up total time
Sum up casualties

If Residual Drift fail
    Set cost equal to Building Replacement Cost.
    Set time to Building Replacement Time.
Otherwise
    Apply date and region multipliers to cost
    If cost is more than Building Replacement Cost * Percent Loss Threshold, set cost
    equal to Building Replacement Cost

Determine Red Flag consequences

Create list of fragilities that were long lead flagged
```

3.3 Determining Simplified EDP values for all realizations

```
For each Direction

    Get NumValues - The total number of floor values for all possible EDP types (i.e. if
        we are looking at just acceleration and drift with one population, it will be the
        number of floors * 2 + 1)
    Get NumRealizations – The number of realizations

    Create a matrix of size NumValues x NumRealizations

    For each Realization
        For each NumValue
            Create a random number between 0 and 1
            Get the median value and dispersion for this EDPType
            Use the Inverse LogNormal function with the median, dispersion, and random
                number and fill the appropriate matrix value
        Next NumValue
    Next Realization
Next Direction
```

3.4 Calculating the population for a realization

```
Find Date and Time for realization
    Generate a random date between Jan 1 and Dec 31

For every population type, calculate the population per square foot for that type
    For each Population Type
        Get the median population per square foot based on the date and time
        Use the Lognormal Inverse function with the median population per square foot,
            dispersion, and a random number to calculate a total population per square
            foot
    Next Population Type

Calculate Floor Population for every floor, grouped by population type
    For Each Floor
        For Each Population Type in Floor
            Get the total floor area marked as this population type
            Multiply that by the calculated total population per square foot for this
Population Type
```

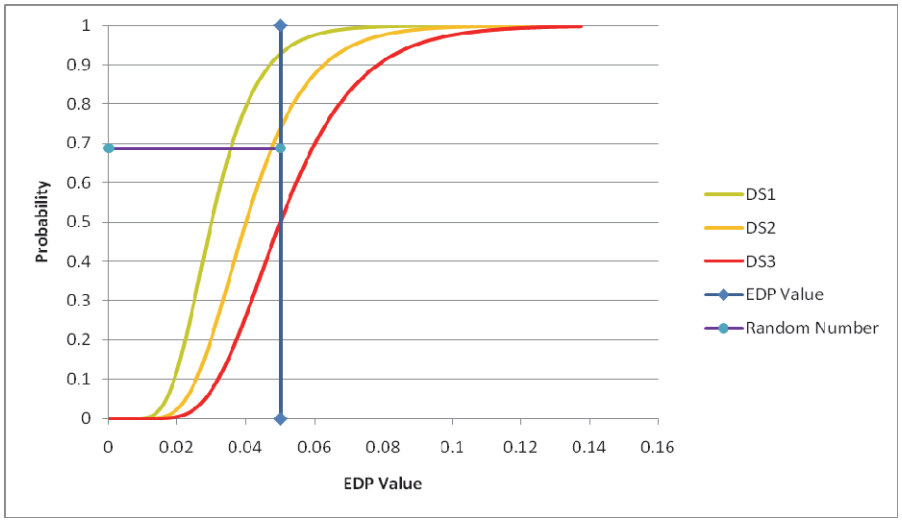
```
        Next Population
    Next Floor
```

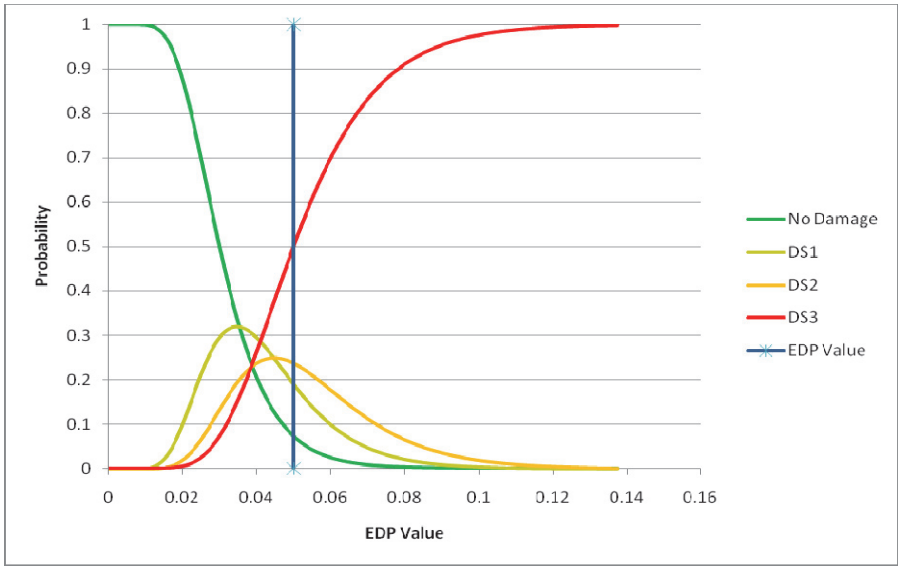
3.5 Calculating the collapse and mode for a realization

```
Calculate an SA(T1) value for this realization
    Use the Lognormal Inverse function with the SA(T1) median entered, the SA(T1)
    dispersion entered, and a randomly generated number

Calculate the chance of collapsing based on the calculated SA(T1) value
    Use the LogNorm function with the Collapse fragility median, dispersion, and the
SA(T1) value

Generate a random number and see if it is greater than the chance of collapsing

If it is collapsed find the collapse mode it is in
    Get the list of collapse modes and find the chance of each
    Generate a random number to determine which mode it is in
```

3.6 Calculating Sequential Damage States

```
For Each Performance Group
        Get the EDP value that applies to this Performance Group

        For Each Item Group
                Generate a random number between 0 and 1

                For Each Possible Damage State
                      Calculate the percent chance of being in this damage state given the
                            EDP Value
                      If the random number is less than the percent chance of being in that
                            damage state, mark this item group as being in that state
                Next Damage State

                If no damage state was found, mark as having no damage
        Next Item Group
Next Performance Group
```





3.7 Calculating Simultaneous Damage States

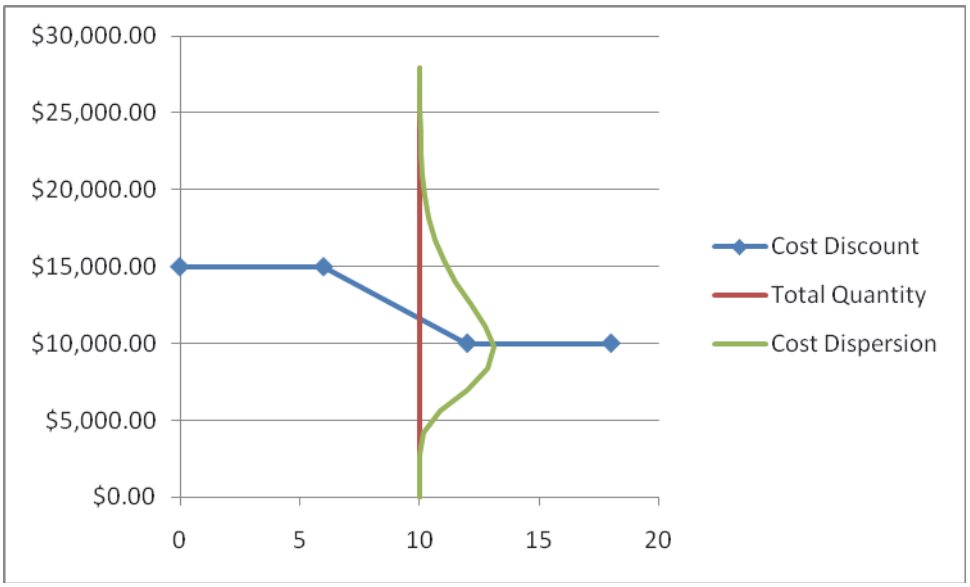
```
For Each Performance Group
    Get the EDP value that applies to this Performance Group

    For Each Item Group
        Generate a random number between 0 and 1

        Calculate the percent chance of being in any damage state given the EDP Value
        If the random number is less than the percent chance of being in a damage state

            Do Loop
                For Each Sub Damage State
                    Generate a random number between 0 and 1

                    If random number is less than the percent chance of being in this
                    damage state, add this damage state to the list of actual damage states
                Next Sub Damage State
            Loop until at least one sub damage state has been indicated

        End If
    Next Item Group
Next Performance Group
```

3.8 Calculating Mutually Exclusive Damage States

```
For Each Performance Group
    Get the EDP value that applies to this Performance Group

    For Each Item Group
        Generate a random number between 0 and 1

        Calculate the percent chance of being in any damage state given the EDP Value
        If the random number is less than the percent chance of being in a damage state

            Generate a random number between 0 and 1
            Determine which damage state the item is in
        End If
    Next Item Group
Next Performance Group
```

3.9 Calculating Cost or Downtime for a Damage State

```
Find mean cost per unit by using the quantity and cost step functions and the total
quantity of items that are in any damage state for the building.

Generate random number between 0 and 1

If cost dispersion is lognormally distributed
    Convert mean unit cost to median cost

    Use the random number, the median cost, and the dispersion to get a final unit cost

If cost dispersion is normally distributed

    Use the random number, the mean cost, and the COV value to get a final unit cost

Multiply the final unit cost by the quantity of items in this item group
```


$30,000.00
$25,000.00
$20,000.00
$15,000.00
$10,000.00
$5,000.00
$0.00
0
5
10
15
20
Cost Discount
Total Quantity
Cost Dispersion

3.10 Calculating Realization Deaths and Injuries for a realization

```
Calculate the population for the realization

Calculate if the building has collapsed, and if so, in what mode

If collapsed
    Calculate total deaths based on death rate for mode of collapse and total floor
        population
    Calculate total injuries based on injury rate for mode of collapse and total floor
        population
    End calculation

If not collapsed
    For every floor
        For every population type on the floor
            Get a total of the area where everyone is dead by adding all death areas for
                all performance groups in all directions that use this population type and
                have casualty implications
            Get the percent of the total area of this population where everyone is dead by
                dividing the sum of all deaths areas for that population by the total area
                of the population
            Get the total dead from this population by multiplying the area percent by the
                total number of people of this population

            Get a total of the area where everyone is injured by adding all injury areas
                for all performance groups in all directions that use this population type
                and have casualty implications
            Get the percent of the total area of this population where everyone is injured
                by dividing the sum of all injury areas for that population by the total
                area of the population
            Get the total injuries from this population by multiplying the area percent by
                the total number of people of this population

        Next Population Type

        Get total deaths by adding up all of the total dead from all populations on this
            floor.
        Get total injuries by adding up all of the total injured from all populations on
            this floor.

        If the total deaths and total injuries is more than the total floor population,
                then make total injuries for the floor equal to the total population minus
                total deaths
    Next floor

    Get total deaths by adding up all of the total dead from all floors.
    Get total injuries by adding up all of the total injured from all floors.

    End calculation
```

3.11 Calculating the Unsafe Placard Consequences for a Realization

```
For each Fragility used in any Performance Group
    Get a count of how many items of that fragility type are in any Performance groups in
        any Floor and Direction.
    Create a list of all Possible Damage States (and sub-states) for the Fragility

    For each Possible Damage State
        Get the Unsafe Placard median and dispersion values for that damage state
        Use the Lognormal Inverse function with the median, dispersion, and a random
            number to get the percent of items required to trigger an unsafe placard

        Go through each Floor, Direction, Performance Group, Item Group, and actual Damage
            State (or sub damage state) and sum up how many items are in this damage state

        Divide the amount that are actually in this damage state by the total quantity of
            items of this fragility type, and compare that to the percent of items
            required to trigger an unsafe placard. If it is greater, mark this realization
            as causing an unsafe placard
    Next Possible Damage State
Next Fragility
```

3.12 Getting the list of Long Lead Flags for a realization

```
For each Floor
    For each Direction
        For each Performance Group
            For each actual Damage State (and Sub-Damage State)
                If Damage State has a long lead flag
                    Add FragilityID to Long Lead Flag List
                End If
            Next Damage State
        Next Performance Group
    Next Direction
Next Floor

Sort the Long Lead Flag List and remove any duplicates
```

3.13 Get the total quantity of all damaged items for a realization, grouped by fragility

```
Create a list of each unique fragility in the building that includes a count variable

For each Floor
    For each Direction
        For each Performance Group
            For each Item Group
                Determine the Fragility used
                Get the count of items in this item group

                If this Item Group is in any damage state
                    Increment the count variable for this fragility by the number of items
                End If

            Next Item Group
        Next Performance Group
    Next Direction
Next Floor
```

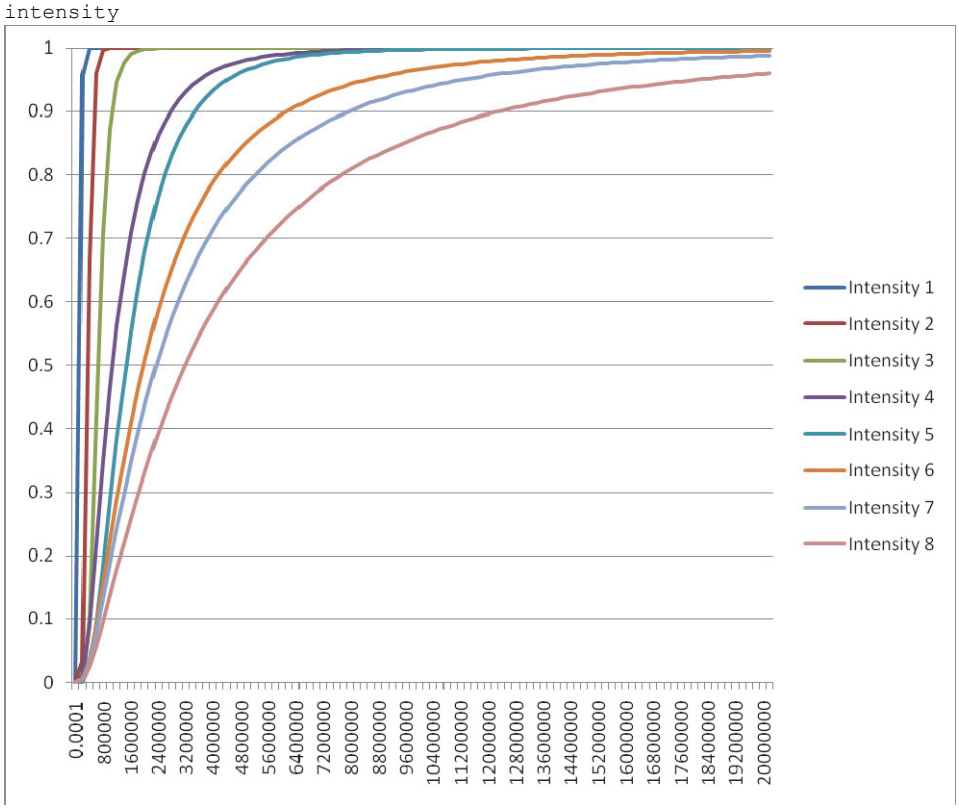
3.14 Calculating Lognormal Integrated Results (Cost, Time, or Casualty)

Get the array of MAFE values for all intensities

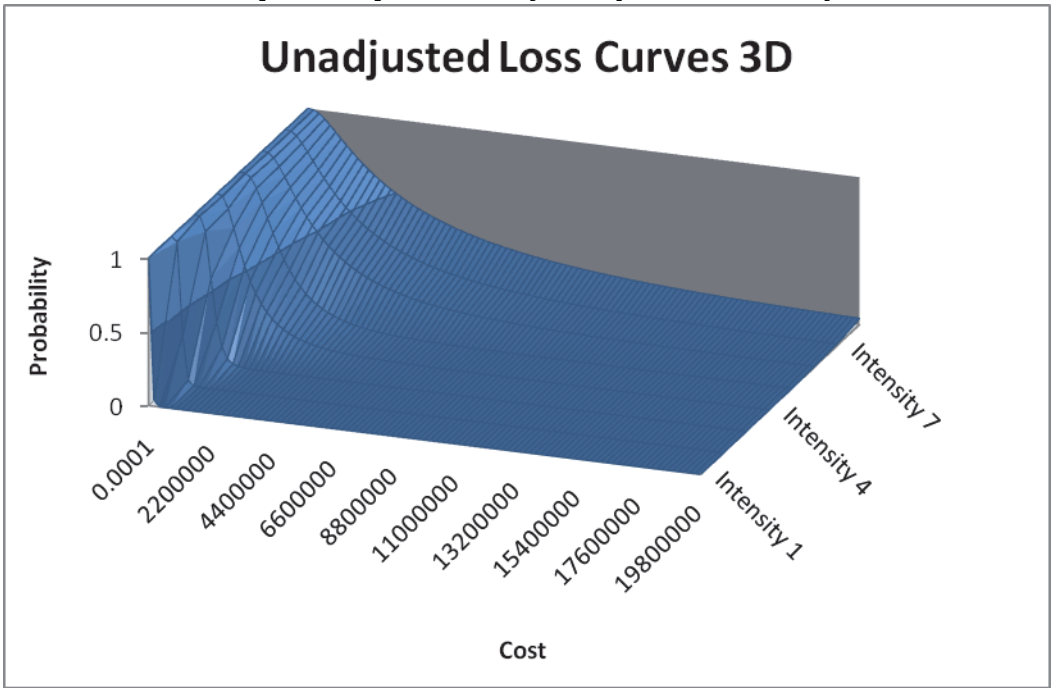
Calculate Intensity Adjustment Totals
    Find the midpoint values values of each MAFE and the MAFE before and after it
    (The midpoint values before the 1st point is equidistant to its midpoint value after)
    (The midpoint values after the last point is equidistant to its midpoint value before)
    Find the distances for each MAFE from the midpoint before to the midpoint after. This
        is the probability of occurance.

Find median value and dispersion for each intensity

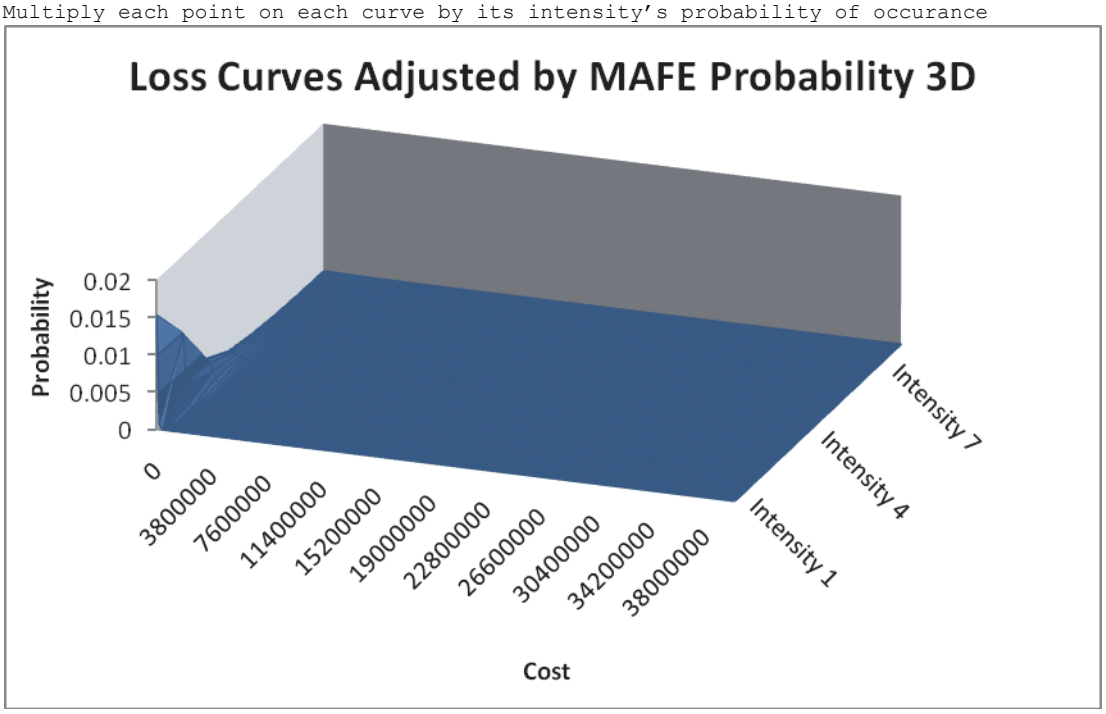
Create a cumulative lognormal fitted curve using the median and dispersion for each intensity



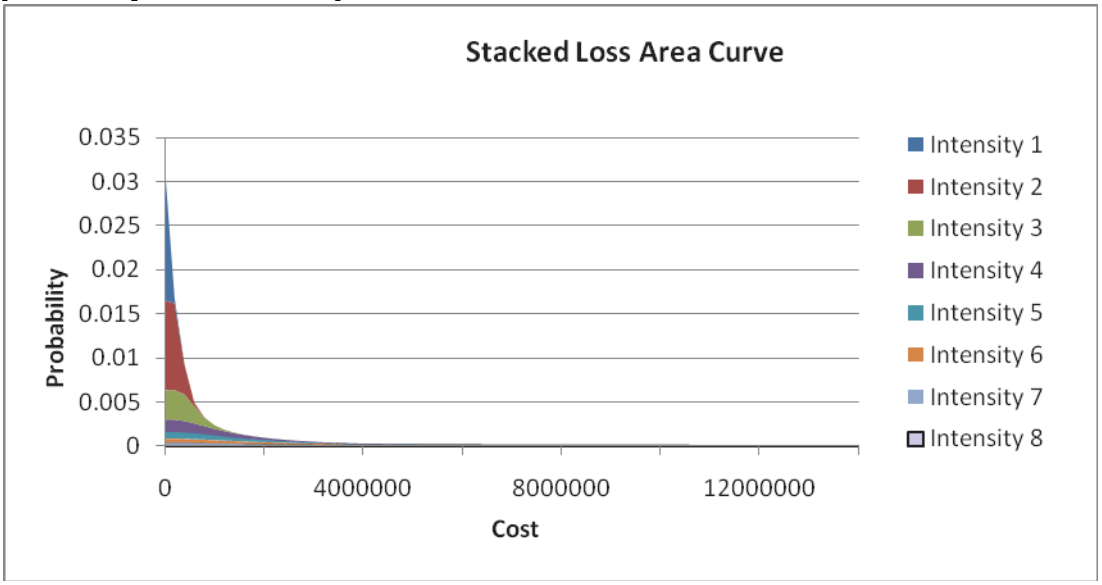
Invert each intensity curve by subtracting each point on the original curve from one.



Multiply each point on each curve by its intensity's probability of occurance



Add each cost point on the curve together across all intensities to get the total probability for each cost point



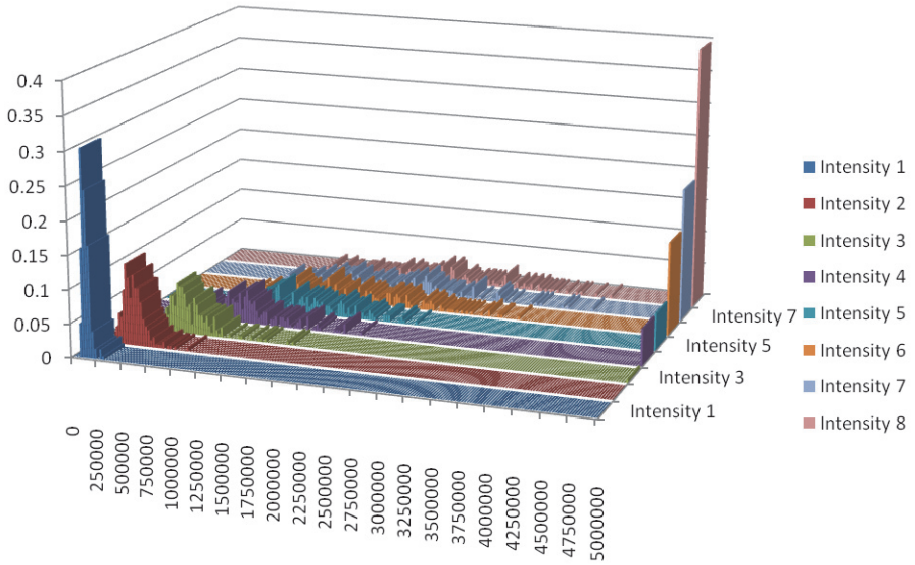
Finally, add all the costs together to get a final value.

3.15 Calculating Binned Results (Cost, Time, or Casualty)

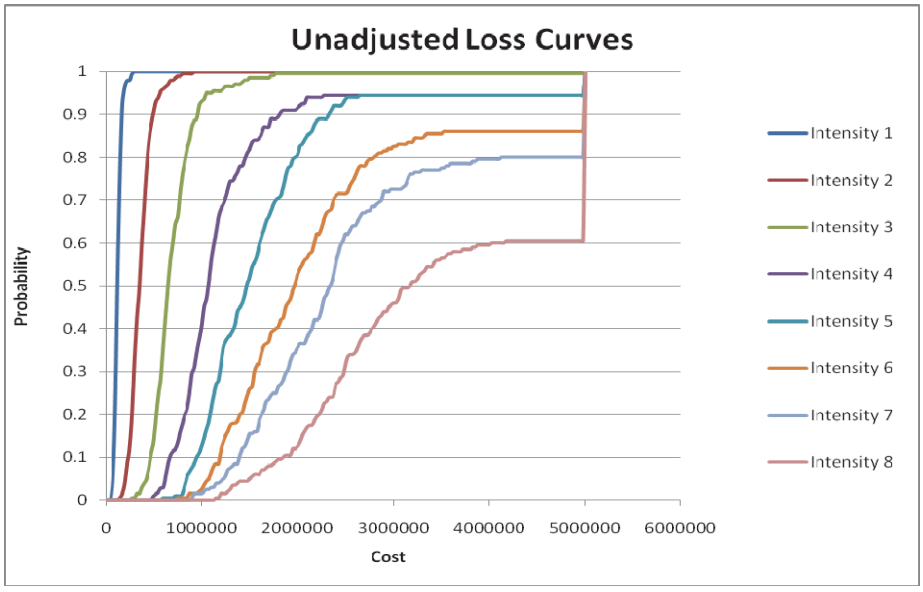
Get the array of MAFE values for all intensities

Calculate Intensity Adjustment Totals
    Find the midpoint values values of each MAFE and the MAFE before and after it
    (The midpoint values before the 1st point is equidistant to its midpoint value after)
    (The midpoint values after the last point is equidistant to its midpoint value before)
    Find the distances for each MAFE from the midpoint before to the midpoint after. This is the probability of occurance.

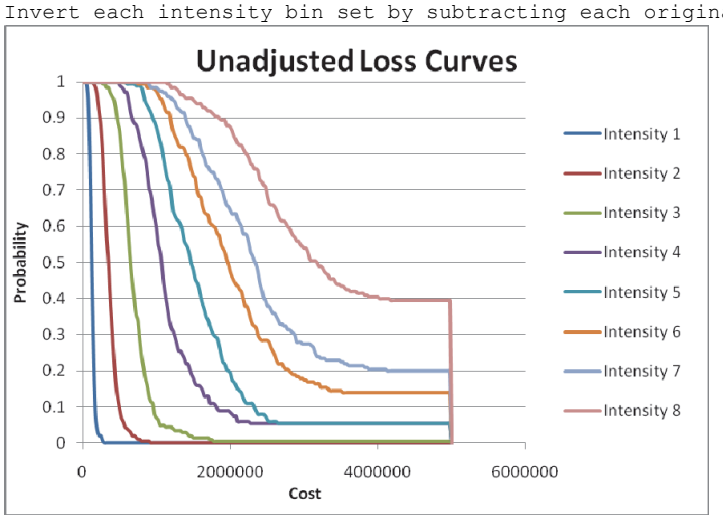
Bin the values of each intensity using the same bin values



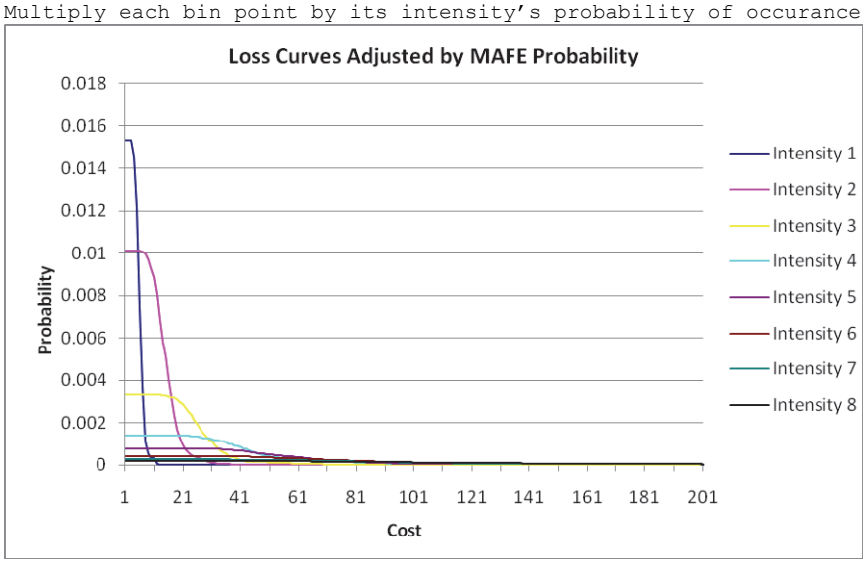
Create a set of cumulative bins



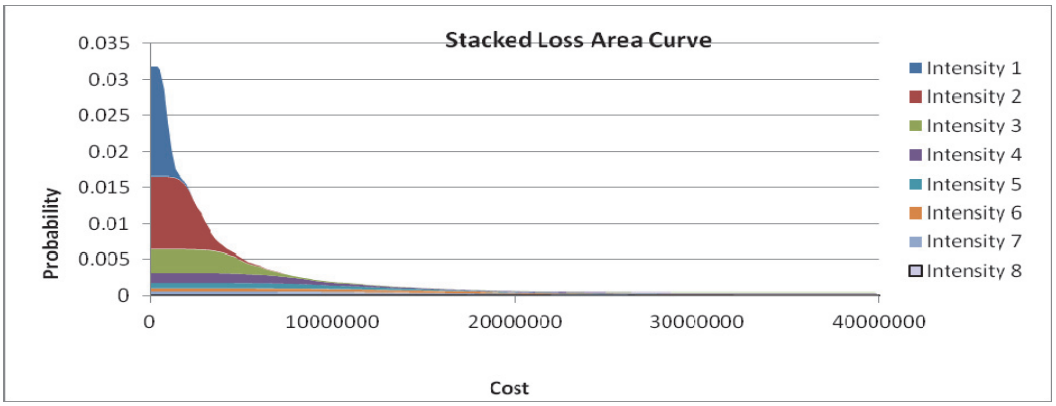
Invert each intensity bin set by subtracting each original bin from one.



Multiply each bin point by its intensity's probability of occurance



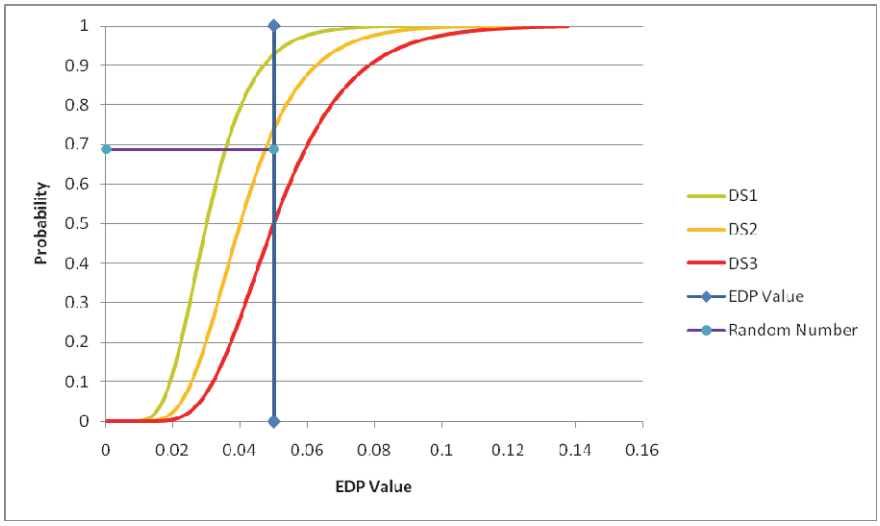
Add each bin cost together across all intensities to get the total probability for each cost bin

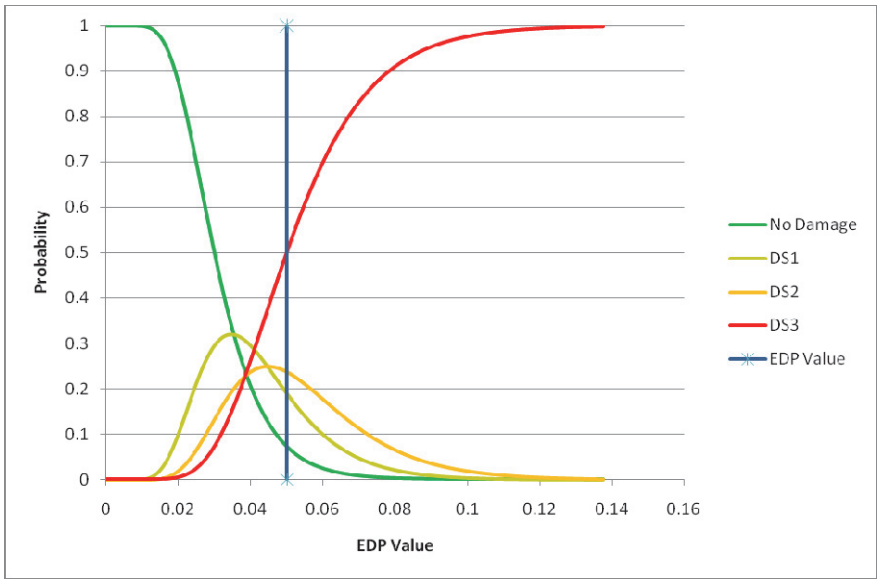


Finally, add all the costs together to get a final value.

3.16 Calculating Sequential Damage States

```
For Each Performance Group
        Get the EDP value that applies to this Performance Group

        For Each Item Group
                Generate a random number between 0 and 1

                For Each Possible Damage State
                      Calculate the percent chance of being in this damage state given the
                            EDP Value
                      If the random number is less than the percent chance of being in that
                            damage state, mark this item group as being in that state
                Next Damage State

                If no damage state was found, mark as having no damage
        Next Item Group
Next Performance Group
```





3.17 Calculating Simultaneous Damage States

```
For Each Performance Group
    Get the EDP value that applies to this Performance Group

    For Each Item Group
        Generate a random number between 0 and 1

        Calculate the percent chance of being in any damage state given the EDP Value
        If the random number is less than the percent chance of being in a damage state

            Do Loop
                For Each Sub Damage State
                    Generate a random number between 0 and 1

                    If random number is less than the percent chance of being in this
                    damage state, add this damage state to the list of actual damage states
                Next Sub Damage State
            Loop until at least one sub damage state has been indicated

        End If
    Next Item Group
Next Performance Group
```

3.18 Calculating Mutually Exclusive Damage States

```
For Each Performance Group
    Get the EDP value that applies to this Performance Group

    For Each Item Group
        Generate a random number between 0 and 1

        Calculate the percent chance of being in any damage state given the EDP Value
        If the random number is less than the percent chance of being in a damage state

            Generate a random number between 0 and 1
            Determine which damage state the item is in
        End If
    Next Item Group
Next Performance Group
```

4 File Formats

Almost all input and output files are in the XML format, which is a hierarchical text-based format similar to HTML, and is easily readable by both humans and machines. Any XML file can be edited with a simple text editor (although a dedicated XML editor will help prevent certain problems.)

The format of the files closely follows the internal class hierarchy outlined above. This manual will not detail every field, but the fields should be fairly self explanatory for the most part. If you are creating a new file by hand, it is advisable to modify an existing file rather than try and create a new file from scratch. If you do edit files, be sure to always close your XML tags. The XML parser built into Visual Studio.Net is, like most XML parsers, very unforgiving of even small mistakes or typos.

Typically, if you have an otherwise valid XML file, but leave a particular XML node out, the program will use default values for the missing data. In some cases this will work, but in other cases it will probably cause a crash. Be careful when hand editing files to not miss important sections.

4.1 Common Sections

There are some common sections in most of the XML files to take note of.

Most start with an optional line that reads:

```
<?xml version="1.0" encoding="utf-8"?>
```

This can safely be ignored or excluded.

Many measurements have the following format:

```
<MeasurementType>
    <Unit>Unit Type Name</Unit>
    <Value>#####</Value>
</MeasurementType >
```

Example:

```
<Area>
    <Unit>Square Foot</Unit>
    <Value>22736</Value>
</Area>
```

It is generally not advised to change the measurement type because the program is likely expecting a certain type, but you can change the unit type name in the file to a different unit of the same type if you so desire, such as changing Inch to Millimeter. Valid measurement types and unit type names are found in the Measures.resx file in the SSUtils project.

Another common feature to note is that some XML fields contain arrays in a delimited format, with rows separated by carriage returns and columns by commas, similar to CSV files. One example:

```
        <EDPArray>
          <NumEQs>11</NumEQs>
          <NumEntries>4</NumEntries>
          <EDPTypeName>Acceleration</EDPTypeName>
          <DispersionValue>0</DispersionValue>
          <EDPGroupName>Acceleration</EDPGroupName>
          <ArrayValues>0.1,0.16,0.15,0.21
0.15,0.22,0.25,0.25
0.09,0.23,0.25,0.23
0.07,0.11,0.12,0.17
0.17,0.22,0.17,0.32
0.08,0.11,0.13,0.19
0.11,0.18,0.23,0.25
0.13,0.19,0.19,0.22
0.12,0.17,0.15,0.16
```

```
0.12,0.28,0.23,0.24
0.1,0.26,0.28,0.28
</ArrayValues>
        </EDPArray>
```

It is very inelegant, but XML does not handle tabular data gracefully. In these examples, any excess spaces or tabs will not matter, but the XML generator built into Visual Studio.Net does not insert them, hence the loss of indentation.

There are some places where there is repeated data. In almost all cases this is required for the proper running of the program

4.2 Fragility File Format

All fragility files are in subdirectories of the ATCCurves directory, which is directly underneath the Pact 2 root directory.

It is recommended that each fragility be in a separate directory, the name of which is the same as the fragility ID. Inside this directory should be an XML file with the same name as the directory, but with an xml file extension. Any images the fragility references should also be in the same directory. There should not be any other files with the xml extension in the directory. Example:

```
ATCCurves
|
+--B1035.000 (Directory)
    |
    +--B1035.000.xml (File)
    +--B1035.000_DS1.JPG
    +--B1035.000_DS2.JPG
    +--B1035.000_DS3.JPG
```

That being said, the program will try and adapt to bending of most of those rules though. If you have any directory underneath the ATCCurves directory that contains any file with the extension of xml, it will try and read it. This is of particular note when you are making backups of fragility xml files or directories if you are modifying them, in that you should probably move the entire backup directory out of the ATCCurves directory.

The file format that Pact 2 uses is similar but not identical to the Pact 1 file format. The format skeleton of the file is:

```
<FragilityCurve>
    <EDPType>
    </EDPType>
    <Ratings>
    </Ratings>
    <DamageStates>
        <DamageState>
            <ConsequenceGroup>
                <CostConsequence>
                </CostConsequence>
                <TimeConsequence>
                </TimeConsequence>
            </ConsequenceGroup>
        </DamageState>
</DamageStates>
</FragilityCurve>
```

FragilityCurve is the top level XML node (clsFragilityCurve). It should include the attribute FileVersion="2.0" to distinguish a Pact 2 file from the Pact 1 fragility files. Under this node is the data for the clsFragilityCurve class. There is a UniqueID field, which contains a generated unique 128-bit integer ID for the curve. It is recommended that if you create a fragility by hand, that you attempt to generate a UniqueID in this format.

EDPType is the data for the expected EDP Type of this fragility (clsEDPType). You may see some files with a subnode labeled EDPAdjustment, but this function is not used anymore.

Ratings is the data for the fragility ratings (clsFragilityRatings).

DamageStates is the data for the damage state groups (clsFragDamageStates). It has a data node called DSGroupType, which should be the type of Damage State Group this is. It will contain one or more DamageState subnodes.

DamageState is the data for an individual damage state (clsFragDamageState, and its inheritors clsFragSeqDamageState, clsFragSimultDamageState, and clsFragMutExDamageState.cs). It will contain slightly different data depending on which kind of damage state it is. In a Sequential damage state, it will contains a Median and a Beta field. If it is a Simultaneous or Mutually Exclusive damage state, it will contain a Percent field, which actually holds a fraction value between 0 and 1. It will have either a ConsequenceGroup subnode, or a DamageStates subnode.

ConsequenceGroup is the data for a consequence function (clsFragConsequenceGroup). It holds the data for the casualty, unsafe placard, and long lead consequences, and a subnodes for the cost and downtime consequences.

CostConsequence is the data for the cost consequences (clsFragCostConsequence).

TimeConsequence is the data for the downtime consequences (clsFragTimeConsequence).

4.3 Building Project File Format

The file format for the Pact 2 building project is also similar to the file format of the Pact 1 projects, but there are some major differences. The format skeleton of the file is:

```
<Project>
    <Building>
        <StructuralDefaults>
            <TypicalDefaults>
                <Item>
                </Item>
            </TypicalDefaults>
        </StructuralDefaults>
        <NonStructuralDefaults>
            <TypicalDefaults>
                <Item>
                </Item>
            </TypicalDefaults>
        </NonStructuralDefaults>
        <CollapseModes>
            <CollapseMode>
            </CollapseMode>
        </CollapseModes>
        <TypicalOccupancyTemplates>
            <OccupancyTemplatePercentage>
                <PopulationModel>
                </PopulationModel>
            </OccupancyTemplatePercentage>
        </TypicalOccupancyTemplates>
        <Floor>
            <StructuralDefaults>
            </StructuralDefaults>
            <NonStructuralDefaults>
            </NonStructuralDefaults>
            <OccupancyTemplates>
                <OccupancyTemplatePercentage>
                </OccupancyTemplatePercentage>
            </OccupancyTemplates>
            <Direction>
                <PerformanceGroup>
                </PerformanceGroup>
            </Direction>
        </Floor>
    </Building>
    <AnalysisCases>
        <Analysis>
            <EDP>
                <EDPArray>
```

```
            </EDPArray>
         </EDP>
      </Analysis>
   </AnalysisCases>
</Project>
```

Project is the top level XML node, and describes the overall project(clsProject). It should include the attribute FileVersion="2.0" to distinguish a Pact 2 file from the Pact 1 project files.

Building is the data for the building object (clsBuilding).

StructuralDefaults is the holder for the core and shell typical defaults for the building. It always has a sole TypicalDefaults subnode.

NonStructuralDefaults is the holder for the nonstructural typical defaults for the building. It always has a sole TypicalDefaults subnode.

TypicalDefaults is the collection for the individual defaults (clsDefaults). whether it be the core and shell or nonstructural, or if it is the building defaults or floor defaults, the format is the same. In all cases it is just a collection for multiple Item nodes with individual defaults.

Item is the data for a single default (clsDefaultItem). In addition to containing Fragility IDs, floor percentages, and population names, it contains a unique ID. This unique ID ties the defaults for the building to the defaults for a single floor, so when you change a building default the floor defaults with the same unique ID also change.

If you are hand making project files, you should not need any of the StructuralDefault, NonStructuralDefault, TypicalDefaults, and Item nodes. It is acceptable to completely leave them out.

CollapseModes is the information about collapse modes (clsCollapseModes). In addition to the collapse fragility and other collapse data, it also contains a collection of collapse modes.

CollapseModes is the data for a single collapse mode (clsCollapseMode). It contains, among other things, several sets of comma separated values indicating floor percentages, casualty rates, and casualty rate betas. Each of these should contain as many items as you have floors, but they may each contain an additional "0" on the end. While they should not be there, they do not appear to harm anything and will not be used by the program.

TypicalOccupancyTemplates is a misnamed due to legacy reasons, but it is the holder for the typical building populations and floor percentages. It will just contain a one or more OccupancyTemplatePercentage nodes.

OccupancyTemplatePercentage, which is also misnamed due to legacy reasons, holds information about a particular population model, and the percentages of the floor that it covers (clsPopulationPercent). Like "Item" above, it contains a unique ID that ties the typical building population defaults to the population information for each individual floor.

PopulationModel contains a copy of the population model (clsPopulationModel). Changing this copy will not change the original population model.

Floor contains information about a particular building floor (clsBuildingFloor). In addition to other things, it will contain three direction subnodes.

StructuralDefaults is the holder for the core and shell defaults for this floor. It is identical in format and has the same subnodes as the typical building defaults above.

NonStructuralDefaults is the holder for the nonstructural defaults for this floor. It is identical in format and has the same subnodes as the typical building defaults above.

OccupancyTemplates is the holder for the floor populations and percentages. It is identical in format and has the same subnodes as the TypicalOccupancyTemplates above.

Direction is the data for a single direction (clsBuildingDirection). It will contain one or more PerformanceGroup subnodes.

PerformanceGroup is the for a single performance group (clsPerformanceGroup).

AnalysisCases holds all the information about the analysis results (clsRuns). It should have an attribute indicating if it is NonLinear or Simplified. It will have one or more Analysis subnodes.

Analysis holds the information about a single intensity or scenario, as well as the hazard curve information (clsRun). It also contains a collection of EDP subnodes, one for each direction.

EDP contains all the EDP information about a particular direction (clsEDPInput). It contains a collection of EDPArray nodes, one for each EDP Type or EDP group name.

EDPArray contains all the analysis results for a particular intensity, direction, and EDP Group (clsEDpArray).

4.4 Result Project XML File Format

Evaluation results can be saved as either a bin file or an XML file. The bin file is smaller and faster to load, but not user editable in any way, and so will not be covered here. The XML file is typically twice as big, with saving and loading time to match, but can be altered or viewed in a text editor or XML editor, although because of the size it will give many editors a very hard time.

The results file is also the most complicated of the files and has a large structure. Also, in an attempt to reduce the file size, many of the data labels are a little more cryptic than is typical in the other XML file formats.

The file contains not only damage quantities and consequences, it also contains all the original data and all the random numbers used in any calculation. The goal of containing so much information in the file is that you can hand calculate any and all information from the data given in this file and return identical answers.

The format skeleton of the file is:

```
<ResultSet>
    <Project>
    </Project>
    <AllUniqueFragilites>
    </AllUniqueFragilites>
    <Analysis>
        <AnalysisEDPInfo EDPType="Cholesky">
            <AllEDPTypesAndNames>
            </AllEDPTypesAndNames>
            <ResultDirectionEDPCholesky>
            </ResultDirectionEDPCholesky>
        </AnalysisEDPInfo>
        <AnalysisEDPInfo EDPType="Simplified">
            <AllEDPTypesAndNames>
            </AllEDPTypesAndNames>
            <Real>
            </Real>
        </AnalysisEDPInfo>
        <Realization>
            <RealFragility>
                <RealDamageState>
                </RealDamageState>
            </RealFragility>
            <PopulationModelRands>
                <Pop>
```

```
            </Pop>
        </PopulationModelRands>
        <PopulationModel_PopPerSF>
            <Pop>
            </Pop>
        </PopulationModel_PopPerSF>
        <Floor>
            <Direction>
                <PG>
                    <ItemGroup>
                        <DSGroup>
                            <DamageState>
                                <Consequence>
                                </Consequence>
                            </DamageState>
                        </DSGroup>
                    </ItemGroup>
                </PG>
            </Direction>
        </Floor>
    </Realization>
</Analysis>
</ResultSet>
```

ResultSet is the root node and contains all the input and output data (clsResultSet). It will contain one or more Analysis nodes.

Project contains a copy of the original building project, and is identical to the Building Project file format given above.

AllUniqueFragilities contains copies of every single fragility used in the project. It has multiple FragilityCurve nodes that are each identical to the Fragility file format given above.

Analysis is the data for a single intensity or scenario (clsResultAnalysis). It will contain EDP information as well as one or more Realization nodes.

AnalysisEDPInfo is all of the calculated EDP information for all realizations and EDP types (clsResultAnalysisEDPs, and its inherited classes clsResultAnalysisEDPCholesky and clsResultAnalysisEDPSimplified). An EDPType attribute is required, specifying if the EDP information is of a Cholesky type (i.e. non-linear) or a Simplified type. The data structures underneath will be different depending on the type.

AllEDPTypesAndNames is a list of EDPTypes (clsEDPType) and EDPGroupNames.

ResultDirectionEDPCholesky is a list of all the generated Cholesky EDP values for a particular analysis and direction (clsResultDirectionEDPCholesky). It contains a collection of matrix arrays with every step of the Cholesky decomposition process and creation of the final EDP values for all realizations.

Real is a list of all realizations for a simplified analysis, and the generated EDP values for each.

Realization is the results of one realization for a particular analysis (clsResultReal).

RealFragility is a collection of all fragilities used in any performance group of this realization (clsResultRealFrag). It is used to calculate totals for cost discounts and unsafe placards.

RealDamageState is a collection of all possible damage states for all fragilities used (clsResultRealDS). It is used to calculate unsafe placard totals.

PopulationModelRands is a collection of all random numbers used in calculating the population for the building.

PopulationModel_PopPerSF is a collection of all the population per square foot results.

```
Floor is the results for a single floor (clsResultFloor).

    Direction is the results of a single direction (clsResultDirection).

        PG is the results of a single performance group (clsResultPG).

            ItemGroup is the results of a single item group (clsResultItemGroup).

                DSGroup is the results of a single damage state group
                (clsResultDSGroup, and its inherited classes
                clsResultDSGroupSeqCor, clsResultDSGroupSimultCor, and
                clsResultDSGroupMutExCor)

                    DamageState is the results of a single damage state
                    (clsResultDS).

                        Consequence is the results of a single consequence
                        (clsResultConsequence).
```

4.5 Population Model

The population model consists of only one level, called <PopulationModel>, which contains all of the fields for the population model. Among others, it contains four fields containing comma separated values for weekday and weekend occupancy fractions by both hour and month.

4.6 Element Library Codes

The ElementLibraryCodes.xml file, found in the DataFiles subdirectory, contains the Uniformat descriptions for each level. The skeleton of the file is:

```
<Elements Source="ATSM Standard E1557 Uniformat II">

  <L1Element>

    <L2Element>

      <L3Element>

        <L4Element>

        </L4Element>

      </L3Element>

    </L2Element>

  </L1Element>
```

Each level contains a Uniformat Code fragment and a description, as well as one or more sublevels.

5 Imported Code and Third Party Controls

Pact uses several third party libraries. Most, with the exception of the Nevron Graph Control, use some variant of an open source or free license.

5.1 Nevron Graph Control

The Nevron Graph controls are controls used to draw and manipulate graphs.

Link:

http://www.nevron.com/

License:

http://www.nevron.com/Purchase.Buy.Licensing.aspx

It is a commercial license, and is therefore required to be purchased to develop for it or compile it into your programs. There is a trial version available that may or may not allow you to fully compile the program. See their website for details on the trial version and its licensing.

5.2 Control Print Project

The Control Print class is a control to allow printing of forms and controls. It is adapted from the ControlPrint source code on codeproject.com by Nader Elshehabi

Link:

http://www.codeproject.com/KB/printing/ControlPrint.aspx

License:

It is released under the Code Project Open License:

http://www.codeproject.com/info/cpol10.aspx

The license allows for use of the code for almost any purpose, commercial or otherwise. It allows for releasing the source code, as long as a link to the license is included, and any changes have been documented. Accordingly, a link is included in the source code, as well as a meticulous description of all changes that have been made.

5.3 Iridium MathNet Numerics Library

Iridium MathNet Numerics is a collection of classes to allow for various higher level math functions, including matrix operations and statistical analysis.

Link:

http://www.mathdotnet.com/Iridium.aspx

License:

The source code is released under the GNU Lesser General Public License (LGPL):

http://www.opensourcedotnet.info/LGPL.aspx

The license allows for distribution of the library, but the source code of the library must be made available and a copyright notice for the library must be put alongside the copyright notice in the program.

5.4 BusinessBase Project

The BusinessBase project is a set of low level classes to facilitate common functionality shared by most other classes in the program. The classes in the BusinessBase Project have been adapted from the CSLA.Net project, as written in the book “Expert C# 2005 Business Objects”, written by Rockford Lhotka.

Link:

http://www.lhotka.net/

License:

The source code is released under a custom open source license:

http://www.lhotka.net/cslanet/License.aspx

The license allows for virtually any use of the code, with one exception that does not apply to this project.

Section 4 of the license stipulates that if the source code is released, then a copy of the license must be included with it. Accordingly, the license has been included in the source files themselves.

5.5 NUnit

Nunit is a class to create unit tests. While not used extensively, and not strictly necessary, I do use it in a few places.

Link:

http://www.nunit.org/index.php?p=home

License:

The code is release under a the open source zlib/libpng license, which allows for almost any use as long as the source of the software is not misrepresented as your own.

http://www.nunit.org/index.php?p=license&r=2.2.10

5.6 Open XML SDK

Open XML SDK is a library to allow creation of Microsoft Office files.

Link:

http://www.microsoft.com/en-us/download/details.aspx?id=5124

License:

You are allowed to distribute the DLL files freely

http://social.msdn.microsoft.com/Forums/en-US/oxmlsdk/thread/19032efe-80af-4a16-b954-1f5a6d2f2236#x_Q3

5.7 Log4Net Control

log4net is a tool to help output log errors.

Link:

http://logging.apache.org/log4net/

License:

It is released under the Apache License:

http://logging.apache.org/log4net/license.html

The license allows for redistribution of their work, provided you include a copy of their license. Pact has not changed the code in anyway, and only uses the compiled object version of their software.

For compliance, the license is shown in full below:

Overview

Typically the licenses listed for the project are that of the project itself, and not of dependencies.

- **Project License**

- **Apache License, Version 2.0**

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or

---

Appendix A – Classes and Members

BuildingManager Assembly

frmBuildingManagerPGChooser

summary: The form to choose a new fragility for a performance group

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

FillTreeView

summary: Fill the tree view

Parameters:

CurrentDirNumber(System.Int32):

tvPerfGroups_AfterSelect

summary: Called when selecting a fragility

Parameters:

sender(System.Object):

e(System.Windows.Forms.TreeViewEventArgs):

frmBuildingManagerPGChooser_Load

summary: Called when loading the form

Parameters:

sender(System.Object):

e(System.EventArgs):

btnClose_Click

summary: The user clicked the close button

Parameters:

sender(System.Object):

e(System.EventArgs):

Constructor

summary: Constructor

Fields

components

summary: Required designer variable.

CurrentProject

summary: The current project

CurrentDirNum

summary: The current direction number

CurrentDir

summary: The current direction object

OldFrag

summary: The original performance group (or null if new)

NewFragility

summary: The newly chosen fragility ID

InStartup

summary: Are we in the startup code

frmBuildingManager

summary: The building manager main form

Methods

Constructor

summary: Constructor

MeasureSystemChanged

summary: If the measure system changed (i.e. imperial to metric), refill-out some of the info

Parameters:

NewMeasureSystem(SSUtils.SSLocalization.StandardSystems):

AddUnitConvertParseAndFormat

summary: Bind a datasource/datamember to a control, and then add the parse and format events for automatic metric/imperial conversion

Parameters:

dataSource(System.Object):

dataMember(System.String):

ControlObject(System.Windows.Forms.Control):

frmBuildingManager_Load

summary: Load Command

Parameters:

sender(System.Object):

e(System.EventArgs):

SetupControls

summary: Main control setup

OpenProject

summary: Open a project using the openfiledialog

LoadProject

summary: Load a project from a filename

Parameters:

ProjectPath(System.String):

ProjectName(System.String):

LoadProject

summary: Load a project from a path and filename

Parameters:

ProjectPathAndName(System.String):

NewCurrentProject

summary: Start a new project

IsOKToCloseProject

summary: Ask the user to save the current project (if needed)

returns: Should we continue

SaveCurrentProject

summary: Save the current project with the current path name

SaveAsCurrentProject

summary: Save the current project, using the SaveFileDialog

SaveCurrentProject

summary: Save the current project using a filename

Parameters:

FilePath(System.String):

FileName(System.String):

InitProject

summary: Init a newly loaded or created project, and reset all the controls

Parameters:

MainProject(PactNet.BuildingProject.clsProject):

tmrCheckStatus_Tick

summary: Check every ten seconds to see if the project has been changed...update the status accordingly

Parameters:

sender(System.Object):

e(System.EventArgs):

UpdateFormText

summary: Update the text on the top of the form

FillRecentProjects

summary: Fill out the recent projects menu item

RecentBuildingProjects_Click

summary: If someone clicks on one of the recent projects from the menu

Parameters:

sender(System.Object):

e(System.EventArgs):

FillDefaultLabels

summary: Fill the labels that reference the measurment type

FillDefaultBuildingLabels

summary: Fill the building labels that reference the measurment type

FillDefaultFloorLabels

summary: Fill the floor related labels that reference the measurment type

dgvFloorDefaults_CellFormatting

summary: Make sure the floor defaults are in the correct measurement type

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellFormattingEventArgs):

dgvFloorDefaults_CellParsing

summary: Make sure the floor defaults are saved in the default measurement type

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellParsingEventArgs):

FormatUnitMeasurements

summary: Format a measurement in the correct measurement type

Parameters:

OldValue(SSUtils.SSLocalization.Measurement):

FormatString(System.String):

ParseUnitMeasurements

summary: Parse a measurement value into the default measurement type

Parameters:

NewValue(System.String):

CurrentColumn(System.Windows.Forms.DataGridViewColumn):

ParseUnitMeasurements

summary: Parse a measurement value into the default measurement type

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

FormatUnitMeasurements

summary: Format a measurment in the correct measurement type

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

dgvBuildingInfoFloorDefaults_CellPainting

summary: Make sure the roof's height is blacked out

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellPaintingEventArgs):

bsFloorsPopulationPercents_CurrentChanged

summary: handler for if the currently selected floor occupancy template has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsFloorsPopulationPercents_CurrentItemChanged

summary: Handler if any item in the floor occupancy templates has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

FillPopulationPercentsComboBoxes

summary: Fill the population percents combo boxes with the names of the default population models

FillTypicalPopulationPercents

summary: Fill the typical (building) population percents

FillFloorPopulationPercents

summary: Fill the floor population percents

FillPopulationPercent

summary: Fill in population percents (typical or floor)

Parameters:

ThisGrid(System.Windows.Forms.DataGridView):

CurrentPopulationPercents(SSUtils.SortableBindingList{PactNet.BuildingProject.clsPopulationPercent}):

dgvPopulationPercents_CellValueChanged

summary: Handler if the cell value of either of the population percents grids has changed

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvPopulationPercents_UserAddedRow

summary: Handler if the user add a row to either of the population percents grids

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewRowEventArgs):

dgvPopulationPercents_UserDeletedRow

summary: Handler if the user deleted a row in either of the occupancy percents grids

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewRowEventArgs):

dgvPopulationPercents_RowEnter

summary: Handle the row enter event in either of the occupancy percents grids Used to show the population info in the rightmost panel

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

btnApplyPopulationDefaults_Click

summary: UNUSED (Hopefully)

Parameters:

sender(System.Object):

e(System.EventArgs):

ShowPopulationModel

summary: Show the preview of the fragilities in this template

Parameters:

PopModel(PactNet.Fragilities.clsPopulationModel): The population model

FillDefaultsGrid

summary: Fill the very complicated defaults grid

Parameters:

dgvControl(System.Windows.Forms.DataGridView): The data grid view control

CurrentDefaults(PactNet.BuildingProject.clsDefaults): The defaults

DefaultsDGV_CellValueChanged

summary: The cell value changed handler that will work with the building defaults, or the floor specific defaults

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

CurrentDefaults(PactNet.BuildingProject.clsDefaults):

DeleteDefaultDGVRow

summary: Delete a row from Structural/Nonstructural Building/Floor defaults grids

Parameters:

CurrentDGV(System.Windows.Forms.DataGridView): The current DataGridView

CurrentCell(System.Windows.Forms.DataGridViewCell): The current DataGridViewCell

CurrentDefaults(PactNet.BuildingProject.clsDefaults): The Current Defaults object

dgvDefaultTemplate_CellValueChanged

summary: The templates grids cell value changed event handler...works with all template grids

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvDefaultTemplate_KeyDown

summary: The templates grids KeyDown event handler...works with all template grids

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

dgvDefault_EditingControlShowing

summary: The handler used when the combo box is chosen in any of the defaults grid

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewEditingControlShowingEventArgs):

dgvDefaultComboBox_SelectedIndexChanged

summary: Handler for when someone selects a new fragility in any of the defaults datagrids

Parameters:

sender(System.Object):

e(System.EventArgs):

FillPerfGroups

summary: Use the currently selected performance group and repopulate the table

FillPerfGroups

summary: Fill the performance groups information

Parameters:

TheBuilding(PactNet.BuildingProject.clsBuilding):

Direction(System.Int32):

Floor(System.Int32):

dgvPerfGroups_CellValueChanged

summary: Write info back to the performance groups

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

rdoPerfDir_CheckedChanged

summary: If the direction changes

Parameters:

sender(System.Object):

e(System.EventArgs):

ClearPerfGroups

summary: Clear the performance groups datagrid

GetCurrentPerfDirection

summary: Get the currently selected direction in the performance group tab

UserChooseFragility

summary: When the user tries to change a fragility in an existing PG

returns: The new fragility curve chosen

Parameters:

ExistingPG(PactNet.BuildingProject.clsPerformanceGroup): The existing performance group

btnToggleCorrelation_Click

summary: (UNUSED) When a user click the toggle correlation button

Parameters:

sender(System.Object):

e(System.EventArgs):

dgvPerfGroups_UserDeletingRow

summary: When a user is deleting a row from the performance group datagridview

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewRowCancelEventArgs):

dgvPerfGroups_CellEndEdit

summary: Called when a user ends an edit on the Perf groups Datagridview

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvPerfGroups_RowValidating

summary: Called when a row on the perf groups datagridview is validating

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellCancelEventArgs):

dgvPerfGroups_CellClick

summary: Called when a user clicks on a cell on the performance group datagrid view

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvPerfGroups_CellValidating

summary: Called when a cell is being validated on the perf groups datagridview

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellValidatingEventArgs):

btnAddPG_Click

summary: Called when the user clicks the add performance group button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnFillPGDefaults_Click

summary: Called when the user clicks the fill PG defaults (Update Table) button

Parameters:

sender(System.Object):

e(System.EventArgs):

FillPGDefaultsForAnItem

summary: Fill the performance group defaults for an item

Parameters:

TempFloor(PactNet.BuildingProject.clsBuildingFloor):

TempDefault(PactNet.BuildingProject.clsDefaultItem):

WriteAnalysisResultsToCSV

summary: Write the Analysis Results to a CSV file

Parameters:

FileName(System.String):

ReadAnalysisResultsFromCSV

summary: Read the analysis results from the CSV file

Parameters:

FileName(System.String):

tsbSaveStructResults_Click

Parameters:

sender(System.Object):

e(System.EventArgs):

dgvPerfGroups_DataError

summary: Called when there is an error in the performance group data grid view

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

FillCollapseDGVs

summary: Fill the collapse mode grids

Parameters:

CollapseModes(PactNet.BuildingProject.clsCollapseModes):

FillCollapseFloorPercentsDGV

summary: Fill the Collapse Floor Percents percentages of each floor) grid

Parameters:

CollapseModes(PactNet.BuildingProject.clsCollapseModes):

FillCollapseModeProbabilityDGV

summary: Fill the Mode Probability (percentages of each mode occuring) grid

Parameters:

CollapseModes(PactNet.BuildingProject.clsCollapseModes):

FillCollapseConsequenceDGV

summary: Fill the collapse consequences grid

Parameters:

CollapseMode(PactNet.BuildingProject.clsCollapseMode):

dgvCollapseFloorPercents_CellValueChanged

summary: Handle the changing of a collapse floor percent value

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvCollapseModeProbability_CellValueChanged

summary: Handle the changing of a collapse mode probability grid change event

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvCollapseConsequence_CellValueChanged

summary: Handle the collapse consequence grid cell changing event

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvCollapseModeProbability_DataError

summary: Called when there is an error in the dgvCollapseModeProbability

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

dgvCollapseFloorPercents_DataError

summary: Called when there is an error in dgvCollapseModeProbability

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

dgvCollapseConsequence_DataError

summary: Called when there is an error in dgvCollapseConsequence

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

cboAnalysisDirection_SelectedIndexChanged

summary: Changing the analysis direction

Parameters:

sender(System.Object):

e(System.EventArgs):

cboEDPType_SelectedIndexChanged

summary: Changing the analysis EDP type

Parameters:

sender(System.Object):

e(System.EventArgs):

FillAnalysisGrid

summary: Fill the analysis grid

ClearAnalysisGrid

summary: Clear the analysis (EDP) grid

FillAnalysisDGVNonLinear

summary: Fill the analysis grid with a particular NonLinear EDP group

Parameters:

EDPInput(PactNet.BuildingProject.clsEDPInput):

EDPGroupName(System.String):

FillAnalysisDGVSimplified

summary: Fill the analysis grid with a particular simplified EDP group

Parameters:

EDPInput(PactNet.BuildingProject.clsEDPInputSimplified):

EDPGroupName(System.String):

dgvAnalysisSet_CellValueChanged

summary: Called when a cell has changed in the analysis grid

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

AnalysisGridChanged

summary: When the analysis grid has been changed, change the underlying object to match

Parameters:

Row(System.Int32):

Col(System.Int32):

txtSimplifiedDispersion_Validating

summary: Called when the simplified dispersion textbox has been changed

Parameters:

sender(System.Object):

e(System.ComponentModel.CancelEventArgs):

rdoAnalysisSimplified_CheckedChanged

summary: Called when the Simplified Analysis radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoAnalysisNonLinear_CheckedChanged

summary: Called when the NonLinear Analysis radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

navAnalysisAddNewItem_Click

summary: Called when the add new item button in the Analysis navbar has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoAssesIntensity_CheckedChanged

summary: Called when the Intensity assesment radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoAssesScenario_CheckedChanged

summary: Called when the Scenario assesment radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

btnDivideAllDrifts_Click

summary: A hidden helper routine to divide all drifts by the story height to help convert some data

Parameters:

sender(System.Object):

e(System.EventArgs):

UpdateAnalysisControls

summary: Update the enabled status of all the analysis controls

dgvAnalysisSet_DataError

summary: Called when dgvAnalysisSet has an error

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

FillResidualDriftNonLinear

summary: Fill the residual grid

dgvResidualDrift_CellValueChanged

summary: Called when a cell has changed in the residual drift grid

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

ResidualDriftChanged

summary: When the Residual Drift grid has been changed, change the underlying object to match

Parameters:

Row(System.Int32):

Col(System.Int32):

DrawResidualCurve

summary: Draw the residual drift curve

Parameters:

DSGraph(Nevron.Chart.WinForm.NChartControl):

Median(System.Double):

Dispersion(System.Double):

dgvResidualDrift_DataError

summary: Called when dgvResidualDrift has an error

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

DrawHazardCurve

summary: Draw the hazard curve graph

Parameters:

GraphControl(Nevron.Chart.WinForm.NChartControl):

AllRuns(PactNet.BuildingProject.clsRuns):

btnHazardDraw_Click

summary: Draw the hazard curve

Parameters:

sender(System.Object):

e(System.EventArgs):

dgvHazardCurve_RowsAdded

summary: Called when a row has been added to the hazard curve DGV

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewRowsAddedEventArgs):

dgvHazardCurve_RowsRemoved

summary: Called when a row has been removed from the hazard curve DGV

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewRowsRemovedEventArgs):

dgvHazardCurve_DataError

summary: Called when the hazard curve DGV has an error

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewDataErrorEventArgs):

OpenToolStripMenuItem_Click

summary: User clicked the open menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveAsToolStripMenuItem_Click

summary: User clicked the Save As menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

NewToolStripMenuItem_Click

summary: User clicked the New menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveToolStripMenuItem_Click

summary: User clicked the Save menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

ExitToolStripMenuItem_Click

summary: User clicked the Exit menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

OptionsToolStripMenuItem_Click

summary: User clicked the Options menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

pageSetupToolStripMenuItem_Click

summary: User clicked the Page Setup menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PrintToolStripMenuItem_Click

summary: User clicked the Print menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PrintPreviewToolStripMenuItem_Click

summary: User clicked the Print Preview menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

undoToolStripMenuItem_Click

summary: (Unused) User click the Undo menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

redoToolStripMenuItem_Click

summary: (Unused) User clicked the Redo menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

CopyToolStripMenuItem_Click

summary: User clicked the Copy menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PasteToolStripMenuItem_Click

summary: User clicked the Paste menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

CutToolStripMenuItem_Click

summary: User clicked the Cut menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SelectAllToolStripMenuItem_Click

summary: User clicked the Select All menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

AboutToolStripMenuItem_Click

summary: User clicked the About menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsProject_CurrentChanged

summary: The project changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsBuilding_CurrentChanged

summary: The Building changed (Should be same as project)

Parameters:

sender(System.Object):

e(System.EventArgs):

CurrentBuilding_PropertyChanged

summary: The current building item changed

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

CurrentBuilding_EDPsChanged

summary: Event handler for the EDPs changing

Parameters:

sender(System.Object):

e(PactNet.BuildingProject.clsBuilding.EDPsChangedEvent):

FillCboEDPType

summary: Fill the EDP Type combo box

FloorsChanged

summary: Event handler for the floor changing event current building

Parameters:

sender(System.Object):

e(PactNet.BuildingProject.clsBuilding.BuildingEvent):

bsClsCollapseModes_CurrentChanged

summary: The collapse modes changed (Should be same as project/building)

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsCollapseMode_CurrentChanged

summary: A single collapse mode

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsRuns_CurrentChanged

summary: The Analysis Runs object changed (should be same as project)

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsRun_CurrentChanged

summary: The current run changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsBuildingSpecFloorDefaults_CurrentChanged

summary: The current floor in the defaults tab

Parameters:

sender(System.Object):

e(System.EventArgs):

bsFloorsDefaults_CurrentItemChanged

summary: A property of the current Floors object changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsCollapseModes_CurrentItemChanged

summary: A property of the current collapse modes object changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsFloorsPGs_CurrentChanged

summary: The floor changed in the performance group tab

Parameters:

sender(System.Object):

e(System.EventArgs):

CurrentFloor_OccupancyListChanged

summary: Handler for when the occupancy (population) list changes for the current floor

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsProject_DataError

summary: Called when there is an error in the clsproject binding

Parameters:

sender(System.Object):

e(System.Windows.Forms.BindingManagerDataErrorEventArgs):

bsClsProject_BindingComplete

summary: Called when the binding of clsproject is complete

Parameters:

sender(System.Object):

e(System.Windows.Forms.BindingCompleteEventArgs):

tabMain_DrawItem

summary: Custom drawing of the main tab

Parameters:

Sender(System.Object):

e(System.Windows.Forms.DrawItemEventArgs):

frmBuildingManager_FormClosing

summary: Called when the form is closing

Parameters:

sender(System.Object):

e(System.Windows.Forms.FormClosingEventArgs):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

PGGridFilling

summary: True of the Performance grid is being changed by the program (vs the user)

CollapseModesFilling

summary: True if the collapse mode grids are being changed by the program (vs the user)

DefaultsFilling

summary: True if the defaults grids are being changed by the program (vs the user)

PopulationPercentsFilling

summary: True if the population percents grids are being changed by the program (vs the user)

CurrentProject

summary: The current project

PopulationForm

summary: The Population editor form

Mainlog

summary: The log file

LastDefaultsRowUsed

summary: When I hook the Default Grids' Combobox's selectedindexchanged event, I don't get a reference to the row, or even datagrid, that I am using. I could try and use the combobox's parent ref to get it, but it is just cleaner to keep track of what row (and therefore datagrid) that I am referencing

components

summary: Required designer variable.

frmBuildingExportEDPs

summary: UNUSED: A class to export EDP information

Methods

Constructor

summary: Constructor

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

components

summary: Required designer variable.

frmBuildingImportEDPs

summary: A form to help import EDP data NOTE: This form has been abandoned in favor of just cutting and pasting due to the complexity of the EDP information

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Constructor

Fields

components

summary: Required designer variable.

frmHazardCurveRanges

summary: A form to edit the hazard curve using Delta ranges, rather than points

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Basic constructor

Parameters:

AllRuns(PactNet.BuildingProject.clsRuns):

FillHazardGrid

summary: Fill the grid

Parameters:

AllRuns(PactNet.BuildingProject.clsRuns):

btnOK_Click

summary: If the user hit OK,

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCancel_Click

summary: If the user hit cancel

Parameters:

sender(System.Object):

e(System.EventArgs):

Fields

components

summary: Required designer variable.

AllRuns

summary: The clsRuns object

RunsChanged

summary: A boolean indicating the runs have been changed by the form

frmEDPGroupNameCreate

summary: The form for creating a new EDP group name

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Constructor

frmEDPGroupNameCreate_Load

summary: Called when loading the form

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveAndExit

summary: Save the EDP group name and exit the form

btnOK_Click

summary: Called when the user clicks OK

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCancel_Click

summary: Called when the user clicks cancel

Parameters:

sender(System.Object):

e(System.EventArgs):

txtGroupName_KeyDown

summary: Called when the user types anything in the textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

Fields

components

summary: Required designer variable.

EDPType

summary: The EDP type of the performance group

CustomName

summary: The custom name chosen by the user

BuildingProject Assembly

clsRuns

summary: All the intensities or scenarios

remarks: Parent: clsProject Children: clsRun

Methods

GetDeltaSAStart

summary: Get the SA of the start of the Delta of the point

Parameters:

AnalysisNum(System.Int32):

GetDeltaSAEnd

summary: Get the SA of the end of the Delta of the point

Parameters:

AnalysisNum(System.Int32):

GetDeltaMAFEStart

summary: Get the MAFE of the start of the Delta of the point

Parameters:

AnalysisNum(System.Int32):

GetDeltaMAFEEnd

summary: Get the MAFE of the end of the Delta of the point

Parameters:

AnalysisNum(System.Int32):

GetAllMAFEPoints

summary: Get all the MAFE points

Constructor

summary: Constructor

SetFloorHeights

summary: Set the heights of the floors (DON'T CALL UNLESS YOU KNOW WHAT YOU ARE DOING)

Parameters:

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}): The heights of all the floors

SetAllEDPNamesUsed

summary: Set all the EDP Types and name used

Parameters:

AllEDPsUsed(System.Collections.Generic.Dictionary{System.String):

GetAllEDPNamesUsed

summary: Get a unique list of all the EDPs used

ValidateHazardCurve

summary: Validate the hazard curve

returns: Is the hazard curve valid

AddNewCore

summary: Create a new run...used by the UI

CreateNewRun

summary: Create a brand new run and append it to the end

InputXML

summary: Input the XML info and create the object from it

Parameters:

AnalysisCasesNode(System.Xml.Linq.XElement):

InputXMLOldHazardCurve

summary: Input the old Pact 1 XML file

Parameters:

HazardCurveNode(System.Xml.XmlNode):

OutputXML

summary: Creates an XML representation of the project and all sub nodes

returns: The AnalysisCases Node containing all subnodes

Properties

NonDirConversionFactor

summary: The conversion factor used to create the non-directional EDP parameters

MinPointSA

summary: The SA of the minimum point

MaxPointSA

summary: The SA of the maximum point

MinPointMAFE

summary: The MAFE of the minimum point

MaxPointMAFE

summary: The MAFE of the maximum point

AllRuns

summary: Gets the list of all the runs

NumRuns

summary: The number of analysis runs (or intensities, or seenarios)

DefaultNumEqs

summary: Gets or sets the default number of EQs for new runs

remarks: The data will propagate down to the runs

IsNonLinearAnalysis

summary: Is this a nonlinear analysis

IsSimplifiedAnalysis

summary: Is this a simplified (linear) analysis

RunType

summary: The run type (nonlinear or simplified)

NumRealizations

summary: Gets or sets the number of realizations

remarks: The data will propagate down to the runs

clsHazardCurve

summary: NOT USED CURRENTLY: The Hazard Curve class, holds information about the intensity levels

remarks: Parent: ClsProject Children: clsHazardPoint

HazardPointRule

summary: A business rule

Methods

Constructor

summary: Constructor

Parameters:

PropertyName(System.String):

ValidateRule

summary: Validate a rule

Parameters:

domainObject(System.Object):

Fields

FailedRowNum

summary: The row that failed

Methods

PointsToSlope

summary: Misc Reference routine to find the slope given some points

Parameters:

X1(System.Double):

X2(System.Double):

Y1(System.Double):

Y2(System.Double):

InputXML

summary: Input an XML representation and create the hazard curve object from it

Parameters:

HazardNode(System.Xml.Linq.XElement):

InputXMLOld

summary: (Depreciated) Input an old format XML representation and create the hazard curve object from it

Parameters:

CurveNode(System.Xml.XmlNode):

OutputXML

summary: Output an XML representation of this object

GetPoint

summary: Get a hazard point by index

Parameters:

PointNum(System.Int32):

GetPoint

summary: Get a hazard point by pointtype and AnalysisNUm

NumPoints

summary: Return the number of hazard points defined

CreateHazardPoint

summary: Create a new hazard point

Parameters:

PointType(PactNet.BuildingProject.clsHazardPoint.PointTypes): The point type

sa(System.Double): Spectral Acceleration

MAFE(System.Double): Mean Annual Freq. of Exceedence

AnalysisNum(System.Int32): The Analysis Number associated with this point

FindIndexOf

summary: Find the index of a specific type of analysis point

returns: The index of the point found, or 0 if not found

Parameters:

PointType(PactNet.BuildingProject.clsHazardPoint.PointTypes): The point type to find, if "AnalysisPoint" include AnalysisNum

AnalysisNum(System.Int32): If looking for AnalysisPoint, specify which one, else leave blank

SortCollection

summary: Sort the collection so all of the points are in order

FindSlope

summary: Find the slope of a specific point (by analysis number)

Parameters:

AnalysisNum(System.Int32):

CreateRules

summary: The override to determine business rules and IsValid

Properties

HazardPoints

summary: Get the list of all the Hazard Points

Fields

ParentProject

summary: The project

clsDefaults

summary: The defaults for the items Parent: clsBuilding, clsBuildingFloor Children: clsDefaultItem

Methods

GetItemsByLevelID

summary: Get a list of items that match are under a certain level ID

Parameters:

LevelID(System.String):

Add

summary: Add a new default item

Parameters:

NewItem(PactNet.BuildingProject.clsDefaultItem):

Add

summary: Add a new fragility as a default item

Parameters:

NewFrag(PactNet.Fragilities.clsFragilityCurve): The fragility

NewPopulationName(System.String): The population name

Remove

summary: Remove based on GUID

Parameters:

IDToRemove(System.Guid): GUID to remove

GetItem

summary: Get the default item using the ID

returns: The Default Item, or null is not found

Parameters:

TheID(System.Guid): The GUID to get

DoesItemExist

summary: Determine if a default item with a specific GUID exists

returns: True if found

Parameters:

IDToLookup(System.Guid): The GUID to look up

Item_PropertyChanged

summary: Handles the propertychanged event, and send it up the chain

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

ReadParentDefaults

summary: Read the parent items and set these defaults to match it

Parameters:

ParentDefaults(PactNet.BuildingProject.clsDefaults):

InputXML

summary: Input the XML ShellDefaults node and create it

Parameters:

ShellDefaultsNode(System.Xml.Linq.XElement):

OutputXML

summary: Output the XML node representing this object and sub-objects

Clone

summary: Make a deep clone

returns: Cloned item

Properties

Items

summary: Get the list of all items

IsInherited

summary: Are these defaults inherited from a more global default

Fields

_items

summary: The list of all the items

DeletedIDs

summary: The list of items that have been recently deleted

clsShellTemplate

summary: Shell and Core template (UNUSED currently)

remarks: Parent: clsBuildingFloor clsBuilding Children: clsOccupancyEntry, clsPopulationModel

Methods

MarkDirty

summary: MarkDirty

InputXML

summary: Input the XML Templates Node

Parameters:

FloorNode(System.Xml.Linq.XElement):

OutputXML

summary: Output the XML node representing this Template

Properties

OccupancyModel

summary: The occupancy model used

TemplatePercentage

summary: TemplatePercentage

TemplateName

summary: TemplateName

Entries

summary: Entries

Fields

IsDirty

summary: Is this Dirty

clsEDPInputSimplified

summary: The simplified (linear) EDP inputs for a particular Analysis case and direction

remarks: Parent: ClsRun Children:clsEDPArray

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor

Parameters:

XMLElement(System.Xml.XmlNode): The XML node to create this object from

Constructor

summary: Constructor

Parameters:

XMLElement(System.Xml.Linq.XElement): The XML node to create this object from

Constructor

summary: Another constructor

Parameters:

TempDirection(System.Int32): The direction

TempNumEqs(System.Int32): The number of Earthquake

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}): The array of floor heights

GetDispersion

summary: Get the dispersion for a particular EDPGroup

returns: The Dispersion

Parameters:

EDPGroupName(System.String): The EDP group name

SetDispersion

summary: Set the dispersion for a particular EDP group

exception(cref=T:System.ArgumentException): If EDPGroupName not found

Parameters:

EDPGroupName(System.String): The EDP Group Name

NewDispersion(System.Double): The Dispersion

SetAllEDPNamesUsed

summary: Set all the EDP group names used

Parameters:

AllEDPsUsed(System.Collections.Generic.Dictionary{System.String):

CreateNewArrayGroup

summary: Create a new EDP array of a specified type

Parameters:

GroupName(System.String): The EDP Group name to use

NewEDPType(PactNet.Fragilities.clsEDPType): The EDP type of the group

DoesEDPGroupNameExist

summary: Determine if an EDP group name exists or not

Parameters:

theEDPGroupName(System.String):

NumLevels

summary: Gets the number of levels for a particular EDP group

Parameters:

EDPGroupName(System.String):

SetFloorHeights

summary: Set the list of floor heights

Parameters:

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}):

GetEDPData

summary: Get EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32):

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name, or blank for the floor default

UnitType(SSUtils.SSLocalization.Unit): The measurement unit type to convert the value into

GetAllUsedEDPGroupNames

summary: Get all EDP group names used

SetEDPData

summary: Set EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32): The EQ number

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name

Value(System.Double): The value to set

UnitType(SSUtils.SSLocalization.Unit): The unit type of the value

InputXML

summary: Input an XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an old format XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.XmlNode):

OutputXML

summary: Output an XML representation of this object

Properties

NumEQs

summary: Get or set the number of earthquakes

clsEDPInput

summary: The base type for EDP input for a particular Analysis case and direction

remarks: Parent: ClsRun Children:clsEDPArray Inherited by clsEDPInputNonlinear, clsEDPInputSimplified

RunTypes

summary: The run type, i.e. nonlinear or simplified

Fields

NonLinear

summary: Nonlinear

Simplified

summary: Simplified

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor for Pact 1 data format

Parameters:

XMLElement(System.Xml.XmlNode): XML node used to create this object

Constructor

summary: Constructor for Pact 2 data format

Parameters:

XMLElement(System.Xml.Linq.XElement): XML node used to create this object

Constructor

summary: Another constructor

Parameters:

TempDirection(System.Int32): The direction

TempNumEqs(System.Int32): The number of Earthquake

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}): The array of floor heights

SetAllEDPNamesUsed

summary: Set all the EDP group names used

Parameters:

AllEDPsUsed(System.Collections.Generic.Dictionary{System.String):

CreateNewArrayGroup

summary: Create a new EDP array of a specified type

Parameters:

GroupName(System.String): The EDP Group name to use

NewEDPType(PactNet.Fragilities.clsEDPType): The EDP type of the group

DoesEDPGroupNameExist

summary: Determine if an EDP group name exists or not

Parameters:

theEDPGroupName(System.String):

NumLevels

summary: Gets the number of levels for a particular EDP group

Parameters:

EDPGroupName(System.String):

SetFloorHeights

summary: Set the list of floor heights

Parameters:

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}):

GetEDPData

summary: Get EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32):

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name, or blank for the floor default

UnitType(SSUtils.SSLocalization.Unit): The measurement unit type to convert the value into

GetAllUsedEDPGroupNames

summary: Get all EDP group names used

SetEDPData

summary: Set EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32): The EQ number

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name

Value(System.Double): The value to set

UnitType(SSUtils.SSLocalization.Unit): The unit type of the value

InputXML

summary: Input an XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.Linq.XElement):

InputXMLold

summary: Input an old format XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.XmlNode):

OutputXML

summary: Output an XML representation of this object

GetArrayByEDPGroupName

summary: Gets the array with a particular EDP Group name

Parameters:

EDPGroupName(System.String): The group name

Properties

Direction

summary: The direction of this EDP info

BetaM

summary: The Beta for modeling uncertainty

BetaGM

summary: The Beta for ground motion uncertainty

NumFloors

summary: Gets the number of actual building floors (not levels)

NumEQs

summary: Get or set the number of earthquakes

clsCollapseModes

summary: The collection of possible collapse modes

remarks: Parent: clsBuilding Children: clsCollapseMode

Methods

GetCollapseMode

summary: Get a specific collapse mode

Parameters:

ModeNum(System.Int32):

SetCollapseMode

summary: Set a specific collapse mode

Parameters:

ModeNum(System.Int32):

value(PactNet.BuildingProject.clsCollapseMode):

GetEDPProbabilities

summary: Get the list of the EDP probabilities

FindMode

summary: Find which collapse mode we are in

returns: The collapse mode

Parameters:

CollapseModeRand(System.Double): The random number used to determine mode

InputXML

summary: Inputs the CollapseModes XML node and builds the object and sub-objects

Parameters:

CollapsesNode(System.Xml.Linq.XElement): The "CollapseModes" node

OutputXML

summary: Returns an XML node containing the CollapseModes object and all sub-objects

Properties

UseCollapse

summary: Are we using collapse or not

AllCollapseModes

summary: The list of all collapse modes

Beta

summary: The beta for the chance that collapse occurs

Median

summary: The median for the chance that collapse occurs

NumModes

summary: The number of possible mode for this collapse

NumFloors

summary: Get or set the number of floors (DON'T CHANGE THIS DIRECTLY)

clsProject

summary: The Project object. The root object containing the building info, the runs info, the hazard curve info

remarks: Children: clsBuilding, clsHazardCurve, clsRun

Methods

Constructor

summary: Constructor

CreateRules

summary: Create business rules

DivideAllDriftsByFloorHeight

summary: Divide all the drifts by the floor height

LoadProject

summary: Loads an XML document representing a project, and creates all the objects

Parameters:

FilePath(System.String): The pathname (with trailing \) for the project document

FileName(System.String): The filename for the project document

InputXML

summary: Inputs an XML document containing a project node and subnodes, and creates the object structure

Parameters:

XMLDocument(System.Xml.Linq.XDocument): An XML document that contains a Project node

InputXML

summary: Inputs an XML document containing a project node and subnodes, and creates the object structure

Parameters:

ProjectNode(System.Xml.Linq.XElement): The top level Project node

FragLib(PactNet.Fragilities.clsFragilityCurveLibrary): The fragility library to use

InputXML

summary: (DO NOT USE) Input the XML and create the object and sub-objects

Parameters:

ProjectNode(System.Xml.Linq.XElement):

InputXMLOld

summary: (Depreciated) Reads an XML document containing the Project Node in the old format

Parameters:

XMLDocument(System.Xml.XmlDocument):

OutputXML

summary: Creates an XML representation of the project and all sub nodes

returns: The Project Node containing all subnodes

SaveProjectToFile

summary: Saves the project object (and sub objects) to a file

Parameters:

SaveFilePath(System.String): The FilePath to save to

SaveFileName(System.String): The FileName to save to

iBuilding_NumFloorsChanged

summary: If the number of floors has been changed in the building object, make sure it is changed in the Runs and LagTimes objects as well

Parameters:

sender(System.Object):

e(PactNet.BuildingProject.clsBuilding.BuildingEvent):

Properties

Building

summary: Returns the building object

ProjectID

summary: Returns the project ID string

ProjectDescription

summary: Returns the Project Description string

Client

summary: Returns the client string

Engineer

summary: Returns the Engineer string

FilePath

summary: Returns the file path that this project has last been saved to (or loaded from)

FileName

summary: Returns the file name that this project has last been saved to (or loaded from)

AllRuns

summary: Returns the runs collection

IsIntensity

summary: Is the an Intensity based project, instead of a scenario based one

IsScenario

summary: Is this a Scenario based project, instead of an intensity based one

DateMultiplier

summary: The cost multiplier based on the date, relative to 2009

RegionMultiplier

summary: The Region multiplier based on the cost average for the US

EngineSeedValue

summary: The seed value to use in the evaluation calculations

Fields

_allRuns

summary: All the Linear Analysis Results, and run information

_dateMultiplier

summary: The date multiplier

_regionMultiplier

summary: The region multiplier

_engineSeedValue

summary: The seed value to use in the evaluation calculations

clsCollapseMode

summary: A single collapse mode

remarks: Parent: clsCollapseMode Children: none

Methods

GetFloorPercentage

summary: The percentage of the floor affected by a collapse

Parameters:

FloorNum(System.Int32):

SetFloorPercentages

summary: The percentage of the floor affected by a collapse

Parameters:

FloorNum(System.Int32):

value(System.Double):

GetFloorDeathRates

summary: The death rate for people on a particular floor

Parameters:

FloorNum(System.Int32):

SetFloorDeathRates

summary: The death rate for people on a particular floor

Parameters:

FloorNum(System.Int32):

value(System.Double):

GetFloorDeathRateBeta

summary: The death rate beta for a particular floor

Parameters:

FloorNum(System.Int32):

SetFloorDeathRateBeta

summary: The death rate beta for a particular floor

Parameters:

FloorNum(System.Int32):

value(System.Double):

GetFloorInjuryRates

summary: The injury rate for people on a particular floor

Parameters:

FloorNum(System.Int32):

SetFloorInjuryRates

summary: The injury rate for people on a particular floor

Parameters:

FloorNum(System.Int32):

value(System.Double):

GetFloorInjuryRateBeta

summary: The injury rate beta for a particular floor

Parameters:

FloorNum(System.Int32):

SetFloorInjuryRateBeta

summary: The injury rate beta for a particular floor

Parameters:

FloorNum(System.Int32):

value(System.Double):

InputXML

summary: Inputs the CollapseMode XML node and builds the object and sub-objects

Parameters:

CollapseNode(System.Xml.Linq.XElement): The "CollapseModes" node

OutputXML

summary: Returns an XML node containing the CollapseModes object and all sub-objects

Properties

NumFloors

summary: The number of floors (DON'T SET THIS DIRECTLY)

Probability

summary: The probability of this collapse mode occuring

FloorPercentages

summary: The percentages of each floor collapsing

FloorDeathRates

summary: The salvage values of each floor

FloorInjuryRates

summary: The salvage values of each floor

FloorDeathRateBetas

summary: The salvage values of each floor

FloorInjuryRateBetas

summary: The salvage values of each floor

clsEDPInputNonlinear

summary: The Nonlinear EDP inputs for a particular Analysis case and direction

remarks: Parent: ClsRun Children:clsEDPArray

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor

Parameters:

XMLElement(System.Xml.XmlNode): The (Pact1 style) XML node to create this object from

Constructor

summary: Constructor

Parameters:

XMLElement(System.Xml.Linq.XElement): The XML node to create this object from

Constructor

summary: Constructor

Parameters:

TempDirection(System.Int32): The direction

TempNumEqs(System.Int32): The number of Earthquake

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}): The array of floor heights

SetAllEDPNamesUsed

summary: Set all the EDP group names used

Parameters:

AllEDPsUsed(System.Collections.Generic.Dictionary{System.String):

CreateNewArrayGroup

summary: Create a new EDP array of a specified type

Parameters:

GroupName(System.String): The EDP Group name to use

NewEDPType(PactNet.Fragilities.clsEDPType): The EDP type of the group

DoesEDPGroupNameExist

summary: Determine if an EDP group name exists or not

Parameters:

theEDPGroupName(System.String):

NumLevels

summary: Gets the number of levels for a particular EDP group

Parameters:

EDPGroupName(System.String):

SetFloorHeights

summary: Set the list of floor heights

Parameters:

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}):

GetEDPData

summary: Get EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32):

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name, or blank for the floor default

UnitType(SSUtils.SSLocalization.Unit): The measurement unit type to convert the value into

GetAllUsedEDPGroupNames

summary: Get all EDP group names used

SetEDPData

summary: Set EDP data of a particular type, earthquake, and entry number

Parameters:

EQNum(System.Int32): The EQ number

EntryNum(System.Int32): The entry (level) number

EDPGroupName(System.String): The EDP Group name

Value(System.Double): The value to set

UnitType(SSUtils.SSLocalization.Unit): The unit type of the value

InputXML

summary: Input an XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.Linq.XElement):

InputXMLold

summary: Input an old format XML file repreentation and create the object

Parameters:

EDPNode(System.Xml.XmlNode):

OutputXML

summary: Output an XML representation of this object

Properties

NumEQs

summary: Get or set the number of earthquakes

clsEDPArray

summary: The EDP input for a particular Analysis case, direction, group name, and EDP type (drift, acc, etc.)

remarks: Parent: clsEDPInput, cls Children: clsEDPType

Methods

ReSizeArrays

summary: Resizes the array

Resize2DArray

summary: Copy an old 2D array to a new 2D array, preserving the sizes

Parameters:

OldArray(System.Double[0:): The old array to resize

NewDim1(0:]): The new (0-based) first dimension

NewDim2(System.Double): The new (0-based) second dimension

InitialValue(System.Double): The initial value for new values

GetValue

summary: Return a specific EDP value

Parameters:

EQNum(System.Int32):

LevelNum(System.Int32):

GetValue

summary: Return a specific EDP value

Parameters:

EQNum(System.Int32): The (zero-based) earthquake number

LevelNum(System.Int32): The (zero-based) level (or entry) number

UnitType(SSUtils.SSLocalization.Unit): The unit type to convert the value into

SetValue

summary: Set a specific value

Parameters:

EQNum(System.Int32): The (zero-based) earthquake number

LevelNum(System.Int32): The (zero-based) level (or entry) number

Value(System.Double): The EDP value for this entry

UnitType(SSUtils.SSLocalization.Unit): The unit type to convert the value into

ReadMainArrayCSVString

summary: Reads a comma separated value string and fill it

Parameters:

CSVString(System.String):

ReadCSVString

summary: Read a CSV (Comma seperated value) array and translate it into a double array

Parameters:

CSVString(System.String):

InputXML

summary: Input the XML representation and create this object

Parameters:

EDPArrayNode(System.Xml.Linq.XElement):

OutputXML

summary: Output an XML representation of this object

Properties

DispersionValue

summary: The beta value for the simplified analysis

EDPType

summary: Get or set the EDP type for this array set

NumEntries

summary: Get or set the number of entries (floors, etc) for this array set

NumEQs

summary: get or set the number of earthquakes

EDPGroupName

summary: Get the EDP Group name

MainArray

summary: Returns the main array

clsBuildingFloor

summary: The Building Floor object. Contains all directions, PGs, etc

remarks: Parent: clsBuilding Children: clsBuildingDirection, clsShellTemplate, clsPopulationPercent

OccupancyListChangedHandler

summary: The delegate for OccupancyListChanged

Parameters:

sender():

e():

Methods

Constructor

summary: Constructor

_occupancyTemplatesPercents_ListChanged

summary: If the occupancy list changes

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

FragilityDefaults_PropertyChanged

summary: Handler for when the fragility defaults change

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

ReadParentPopulationDefaults

summary: Read the building population defaults, and update the floor populations to match

Parameters:

ParentList(SSUtils.SortableBindingList{PactNet.BuildingProject.clsPopulationPercent}):

GetAllEDPNamesUsed

summary: Get all the default and special EDP names in every single PG group that match a specified EDP type

returns: A list of names, or a blank list of none

GetDirection

summary: Get the specified direction object

returns: The Direction Object

Parameters:

DirectionNum(System.Int32): The index of the direction (0-2)

CreateRules

summary: Create all the business rules

InputXML

summary: Input the XML Floor node and create the Floor object and all sub-objects

Parameters:

FloorNode(System.Xml.Linq.XElement):

FragLib(PactNet.Fragilities.clsFragilityCurveLibrary): The fragility library to use

InputXML

summary: (DO NOT USE) Input the XML and create the object and sub-objects

Parameters:

Node(System.Xml.Linq.XElement):

InputXMLOld

summary: (Depreciated) Input the old format XML and create the floor object

Parameters:

FloorNode(System.Xml.XmlNode):

OutputXML

summary: Output the XML node representing this floor and sub-objects

GetFragilitiesUsed

summary: Get a list of all fragilties used in this floor, for particular directions

Parameters:

DirectionArray(System.Int32[]): An array of directions to look at, or nothing (or blank) for all

OccupancyTemplateArray(System.String[]): An array of OccupanyTemplateSources to look at, or nothing (or blank) for all

GetQuantity

summary: (NOT USED) Get the quantity for a specified fragility

Parameters:

FragilityID(System.String): The fragility ID to look for

TemplateName(System.String): The originating template name, or "" for all PGs regardless of originating template

EDPGroupName(System.String): The EDP group name, or "" for all PGs regardless of EDPGroupName

Properties

AllDirections

summary: The list of all directions for this floor

PopulationPercents

summary: Gets or sets the list of population templates for this floor

FragilityDefaults

summary: The fragility defaults

FloorNum

summary: Gets or sets the floor number (Be careful with this one)

FloorName

summary: The custom name of the floor

OneBasedFloorNum

summary: Gets (no set) the floor number, with the bottom floor being one

Height

summary: Gets or sets the height of this floor

Area

summary: Gets or sets the area of this floor

LengthDir1

summary: Gets or sets the length of direction 1

LengthDir2

summary: Gets or sets the length of direction 2

IsRoof

summary: Is this floor the roof

HeightFactor

summary: Gets or sets the floor cost factor

HazmatFactor

summary: Multiplier for Hazardous Material adjustment factor

OccupancyFactor

summary: Multiplier for the occupancy adjustment factor

AllEDPsUsed

summary: Get all the default and special EDPs in every single PG group

returns: A list of names, or a blank list of none

Fields

_floorName

summary: The custom name of the floor

_allDirections

summary: The list of all directions for this floor

Events

OccupancyListChanged

summary: The event for when the occupancy (populations) list changes

clsRun

summary: A single intensity or scenario

remarks: Parent: clsRuns Children: ClsEDPInput, clsEDPArray

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor

Parameters:

iRunNum(System.Int32): The Analysis number for this EDP info

iNumEqs(System.Int32): The number of Eearthquake

iNumRealizations(System.Int32): The number of realizations

HeightArray(SSUtils.SortableBindingList{SSUtils.SSLocalization.Measurement}): The number of floors

TheRunType(PactNet.BuildingProject.clsEDPInput.RunTypes): The runtype...nonlinear or simplified

Constructor

summary: Constructor from an XElement

Parameters:

XMLElement(System.Xml.Linq.XElement): The XML element used to create this run

Constructor

summary: Constructor

Parameters:

XMLElement(System.Xml.Linq.XElement): The XML element used to create this run

TheRunType(PactNet.BuildingProject.clsEDPInput.RunTypes): The run type (simplified oer nonlinear)

Constructor

summary: (Depreciated) Constructor from an (old) XML Node

Parameters:

XMLElement(System.Xml.XmlNode):

TheRunType(PactNet.BuildingProject.clsEDPInput.RunTypes): The run type

SetFloorHeights

summary: sets the number of floors (DON'T USE DIRECTLY)

SetAllEDPNamesUsed

summary: Set all the EDP Types and name used (DON'T USE DIRECTLY, use the one in ClsRuns)

Parameters:

AllEDPsUsed(System.Collections.Generic.Dictionary{System.String):

GetDirection

summary: Get the EDP input object for a particular direction This will create the non-directional on the fly

Parameters:

DirectionNum(System.Int32):

CreateNonDirectional

summary: Create a calculated Non-directional Input type baed on the two directional inputs, and return it

SetDirection

summary: Set the EDP input object for a particular direction

Parameters:

DirectionNum(System.Int32):

NewEDP(PactNet.BuildingProject.clsEDPInput):

GetAllUsedEDPTypesAndNames

summary: Get a list of all EDP Types and Names

InputXML

summary: Read an XML node representing this object and fill this object from it

Parameters:

RunNode(System.Xml.Linq.XElement):

InputXMLOld

summary: (Depreciated) Read an old format XML node represetning this object and fill this object from it

Parameters:

RunNode(System.Xml.XmlNode):

OutputXML

summary: Create an XML node representing this object

Properties

NumEQs

summary: Return the number of Earthquakes specified

RunType

summary: The run type (simplified or non-linear)

NumRealizations

summary: Gets or sets the number of realizations

SA

summary: SA (T1) - The acceleration at the base at the period of the building

MAFE

summary: MAFE - The Mean Annual Frequency of Exceedence of this run

SABeta

summary: SA (T1) beta - The acceleration at the base at the period of the building

ModelingDispersion

summary: Modeling Dispersion

NonDirConversionFactor

summary: The conversion factor used to create the non-directional EDP parameters

ResidualDriftEDPs

summary: The residual drifts

RunName

summary: Gets or sets the run name

clsPerformanceGroup

summary: The quantity of a particular floor/direction and fragility from a particular population and EDP Group Name

remarks: Parent: clsBuildingDirection Children: clsFragilityCurve

Methods

CompareTo

summary: Compare this PG to another

Parameters:

obj(System.Object):

CreateRules

summary: The override to determine business rules and IsValid

InputXML

summary: Input the XML representation of this object and create the object

Parameters:

FQNode(System.Xml.Linq.XElement):

FragLib(PactNet.Fragilities.clsFragilityCurveLibrary): The fragility library to use

InputXML

summary: (DO NOT USE) Input the XML and create the object and sub-objects

InputXMLOld

summary: (Depreciated) Input the old format XML representation of this object and create the object

Parameters:

FQNode(System.Xml.XmlNode):

OutputXML

summary: Output the XML representation of this node

Properties

Correlation

summary: The correlation number

EDPGroupName

summary: The EDP Group name

IsUnitEach

summary: Is the unit of this PG "Each"

NoDiceThrowing

summary: UNUSED: Should this be a type of PG where we throw no dice, i.e. uncorrelated and length or area

CorrelationBool

summary: A correlation boolean, is it correlated or not

QuantityUncertainty

summary: The beta for the quantity

PopulationModelName

summary: What Population Model this PG came from

FragilityCurve

summary: Gets or sets the fragility curve

FragilityID

summary: The fragility ID

Quantity

summary: Gets or sets the quantity of this PG

Fields

_fragilityCurve

summary: The fragility curve

_quantity

summary: The quantity (median)

_populationModelName

summary: The name of the population model

_quantityUncertainty

summary: The quantity uncertainty

_eDPGroupName

summary: The EDP group name

_correlation

summary: The correlation

_correlationSet

summary: Has the correlation been manually set. I had a good reason for this, but forgot what it was

clsHazardPoint

summary: NOT USED CURRENTLY: A single Hazard curve point

remarks: Parent: clsHazardCurve

PointTypes

summary: The possible point types

Fields

IMMinPoint

summary: The min point, used indirectly to find slopes

CurvePoint

summary: A non-assigned curve point

AnalysisPoint

summary: A curve point assigned to a particular analysis

IMMaxPoint

summary: The max point, used indirectly to find slopes

Methods

Constructor

summary: Constructor

ReturnPointTypeString

summary: Return a string representation of the point type

CreateRules

summary: Create the list of business rules

InputXML

summary: Inputs the Hazard Point XML node and builds the object

Parameters:

PointNode(System.Xml.Linq.XElement): The Point node

InputXMLOld

summary: (Depreciated) Inputs the old version of the Hazard Point XML node and builds the object

Parameters:

PointNode(System.Xml.XmlNode):

OutputXML

summary: Creates an XML representation of this object

CompareTo

summary: Compares two hazard points. They are sorted bt MAFE

Parameters:

obj(System.Object):

Properties

PointType

summary: Get or set the point type

SA

summary: Get or set Spectral Acceleration

MAFE

summary: Get or set Mean Annual Freq. of Exceedence

AnalysisNum

summary: Get or set the Analysis Number associated with this point

PointTypeName

summary: Return the name of the point (including Analysis number)

clsBuilding

summary: The Building Object. Contains all building info such as directions, floors, and PGs

remarks: Parent: clsProject Children: clsBuildingFloor, clsCollapseModes, clsDefaults

NumFloorsChangedEventHandler

summary: Delegate to call NumFloorsChanged

Parameters:

sender():

e():

EDPsChangedEventHandler

summary: Delegate to call EDPsChanged

Parameters:

sender():

e():

BuildingEvent

summary: Argument for NumFloorsChanged

Fields

FloorHeights

summary: A list of all the floors and their heights

EDPsChangedEvent

summary: Argument for EDPsChanged

Methods

Constructor

summary: Constructor

FragilityDefaults_PropertyChanged

summary: Catch when the building fragility defaults changes, to let the floors update

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

FloorListChanged

summary: Check the floor list for any changes that the project and building need to know about

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

RaiseFloorsChanged

summary: Raise the floors changed event, so the project can be aware that the number of floors has changed

SetNewFloorDefaults

summary: When creating a new floor, set the building defaults for it

Parameters:

NewFloorIndex(System.Int32):

_defaultPopulationModelPercents_ListChanged

summary: Handler for when the Population Model Percents list changes

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

BuildFloorHeights

summary: Build a list of floor heights

CreateRules

summary: Create a list of business rules for the building

GetMissingFrags

summary: A list of any fragilities that have gone missing

MarkClean

summary: Marks the object (and sub-objects) as clean

MarkOld

summary: Mark this object as old, and not in need of saving

GetAllEDPNamesUsed

summary: Get all the default and special EDP names in every single PG group that match a specified EDP type

returns: A list of names, or a blank list of none

GetFloorHeight

summary: Get the height of a floor

returns: The height of the floor

Parameters:

Floor(System.Int32): Floor number to return

SetFloorHeight

summary: Sets the height of a floor

Parameters:

Floor(System.Int32): The floor number to change

FloorHeight(SSUtils.SSLocalization.Measurement): The height of the floor

GetFloorArea

summary: Get the area of a floor

returns: The area of the floor

Parameters:

Floor(System.Int32): The floor number to get

GetFloor

summary: Get the floor object

returns: The floor object

Parameters:

Floor(System.Int32): The floor number to return

InputXML

summary: Inputs the building XML node and builds the object and sub-objects

Parameters:

BuildingNode(System.Xml.Linq.XElement): The building node

FragLib(PactNet.Fragilities.clsFragilityCurveLibrary): The fragility library to use

InputXML

summary: (DO NOT USE) Inputs the building XML node and builds the object and sub-objects

Parameters:

Node(System.Xml.Linq.XElement):

InputXMLOld

summary: (Depreciated) Inputs the building XML node in the old format and builds the object and sub-objects Hell, I don't even know if this works anymore

Parameters:

BuildingNode(System.Xml.XmlNode):

OutputXML

summary: Returns an XML node containing the building object and all sub-objects

GetFragilitiesUsed

summary: Get a list of all the fragilties used in certain floors and directions

returns: The list of all fragilities used

Parameters:

FloorArray(System.Int32[]): An array of all floors to look at, or nothing for all

DirectionArray(System.Int32[]): An array of all directions to look at, or nothing for all

OccupancyTemplateArray(System.String[]): An array of OccupanyTemplateSources to look at, or nothing (or blank) for all

GetQuantity

summary: (NOT USED) Get the quantity for a specified fragility

Parameters:

FragilityID(System.String): The fragility ID to look for

TemplateName(System.String): The originating template name, or "" for all PGs regardless of originating template

EDPGroupName(System.String): EDP Group Name

Properties

CollapseModes

summary: The object containing all the collapse mode information

DefaultPopulationPercents

summary: The defualt population percents

TypicalFloorArea

summary: The typical floor area...used as a shortcut to put in floor area for all floors in the GUI

TypicalFloorHeight

summary: The typical floor height...used as a shortcut to put in floor height for all floors in the GUI

FragilityDefaults

summary: The fragility defaults

ReplacementCost

summary: The total cost to replace the entire building

MaxWorkersPerSF

summary: The maximum workers that can be working per SF

MaxWorkersPerArea

summary: The maximum workers that can be working per SF, returned as a Measurement type

CoreReplacementCost

summary: The total cost to replace the entire building - minus contents

PercentLossThreshold

summary: Percent of total replacement cost that will be reached before the building is considered totaled

ReplacementTime

summary: The total time to replace the entire building

Floors

summary: A list of all floor objects in this building

NumFloors

summary: Gets or sets the number of floors in the building (does not include roof)

UseResidualDrift

summary: A boolean indicating if we are using residual drift

ResidualDriftMedian

summary: The residual drift median value

ResidualDriftDisp

summary: The residual drift dispersion

IsNew

summary: Determines if the object is new and unsaved

IsDirty

summary: Determines if the object (or sub-objects) is Dirty (in need of saving)

IsValid

summary: Determines if the object (or sub-objects) has valid data

AllEDPsUsed

summary: Get all the default and special EDPs in every single PG group

returns: A list of names, or a blank list of none

Events

EDPsChanged

summary: Event raised when the EDPs have been changed, so the project can let clsRuns know

NumFloorsChanged

summary: Event raised when the number of floors changes, so the project can let clsRuns know

clsDefaultItem

summary: The defaults for an item Parent: clsDefaults Children: none

Methods

ReadParentDefaults

summary: Read the parent item and set the defaults to match it (if inherited)

Parameters:

ParentItem(PactNet.BuildingProject.clsDefaultItem):

InputXML

summary: Input the XML ShellDefaultItems node and create it

Parameters:

ShellDefaultItemNode(System.Xml.Linq.XElement):

OutputXML

summary: Output the XML node representing this object and sub-objects

Clone

summary: Deep clone the object

Properties

ID

summary: A unique ID to tie building and floor items together

IsInherited

summary: Is this default inherited from another default (i.e., is this a floor default inherited from a building default)

Dir1Bool

summary: Should this be applied in direction 1

Dir2Bool

summary: Should this be applied in direction 2

PercentageDir1

summary: The percentage of the floor in direction 1 covered by this default

PercentageDir2

summary: The percentage of the floor in direction 2 covered by this default

FragilityID

summary: The fragility ID associated with this default

Population

summary: The population model to use with this default

Fields

_isInherited

summary: Is this item inherited from another. i.e. is this a floor item inherited from a typical building item

clsBuildingDirection

summary: A particular floor in a particular direction

remarks: Parent: clsBuildingFloor Children: clsPerformanceGroup

Methods

Constructor

summary: Constructor

_performanceGroups_ListChanged

summary: The handler for when the performance group list changes.

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

GetAllEDPNamesUsed

summary: Get all the default and special EDP names in every single PG group that match a specified EDP type

returns: A list of names, or a blank list of none

ClearAllFragilities

summary: Clear the entire list of all frags and quantities

ReturnFragilityQuantities

summary: Return the clsPerformanceGroup for a specified Fragility and TemplateName

returns: The quantity for that fragility and population template name, null if not found

Parameters:

FragilityID(System.String): The fragility ID to return

PopModelName(System.String): The originating population model name ("" for all)

EDPGroupName(System.String): The EDP group name ("" for all) WARNING: NOT BEING USED

GetQuantity

summary: Get the quantity for a specified fragility

Parameters:

FragilityID(System.String): The fragility ID to look for

PopModelName(System.String): The originating population model name, or "" for all

EDPGroupName(System.String): The name of the EDP Type, or "" for all

SetQuantity

summary: Set the quantity for a particular Fragility

remarks: Will create a clsFloorFragilityQuantities object if needed

Parameters:

Fragility(PactNet.Fragilities.clsFragilityCurve): The fragility to set

Quantity(System.Double): The quantity of the specified fragility

QuantityUncertainty(System.Double): The quantity uncertainty of the specified fragility

PopModelName(System.String): The name of the poulation model, or "" for none

EDPGroupName(System.String): The name of the EDP group

Coorelated(System.Boolean): The default coorelation of the specified fragility

SetAvailablePopModelNames

summary: If the list of available population names has changed and a particular populationName has been deleted we need to update any PGs that use it

Parameters:

AvailablePopulationNames(System.Collections.Generic.List{System.String}):

CreateRules

summary: Create the rules for this direction....no rules

InputXML

summary: Input the XML and create the object and sub-objects

Parameters:

DirectionNode(System.Xml.Linq.XElement):

FragLib(PactNet.Fragilities.clsFragilityCurveLibrary): The fragility library to use

InputXML

summary: (DO NOT USE) Input the XML and create the object and sub-objects

Parameters:

Node(System.Xml.Linq.XElement):

InputXMLold

summary: (Depreciated) Input the old format XML and create the object and sub-objects

Parameters:

DirNode(System.Xml.XmlNode):

OutputXML

summary: Output an XML node representing this object and sub-objects

GetFragilitiesUsed

summary: Get a list of all the fragilites used

returns: A list of all the fragility curves used here

Parameters:

PopulationModelNamesArray(System.String[]): The list of population model names that we want to select, or null for all

Properties

PerformanceGroups

summary: The list of all performance groups for this direction

AllEDPsUsed

summary: Get all the default and special EDPs in every single PG group

Fields

MissingFrags

summary: A list of any fragilities that have gone missing

clsPopulationPercent

Methods

Constructor

summary: Constructor

Parameters:

TemplateNode(System.Xml.Linq.XElement): XML element used to create this object

Constructor

summary: Constructor

MarkDirty

summary: Overriding the MarkDirty

MarkDirty

summary: Overriding the MarkDirty

Parameters:

PropertyName(System.String):

InputXML

summary: Input the XML Templates Percentage Node

Parameters:

TemplatePercentageNode(System.Xml.Linq.XElement):

OutputXML

summary: Output the XML node representing this Template

Properties

ID

summary: GUID used to tie floor population percents to typical building population percents

FollowingMaster

summary: Is this Population percent following a "master" population percent

PopulationModel

summary: The population model

Percent

summary: The fraction of the floor this population model covers

ControlPrint Assembly

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Default constructor

Constructor

summary: Intialize the component with the selected control

Parameters:

print(System.Windows.Forms.Control): The control to printed

Constructor

summary: Initialize the component with the selected control specifying whether to stretch or not

Parameters:

print(System.Windows.Forms.Control): The control to be printed

iStretch(System.Boolean): Set true to stretch the control to fill a single printed page

Constructor

summary: Initialize the component with the selected control specifying specific width and height

Parameters:

print(System.Windows.Forms.Control): The control to be printed

Width(System.Int32): Printed width

Height(System.Int32): Printed height

Constructor

summary: Intialize the component with a specific container, Insignificant

Parameters:

container(System.ComponentModel.IContainer):

SetControl

summary: Set the control to be printed

Parameters:

print(System.Windows.Forms.Control): The control you wish to print

SetControl

summary: Set the control with a specified height and width

Parameters:

print(System.Windows.Forms.Control): The control to be printed

Width(System.Int32): Control's width

Height(System.Int32): Control's height

GetBitmap

summary: Draw the control fully to a bitmap and return it

CalculateSize

summary: Return the best size that fits the control

ApplyBestSize

summary: Apply the best size that fits the control

Properties

StretchControl

summary: Set true to stretch the control to fill a single printed page

PrintWidth

summary: The width of the control to print. You can change this value if the automatically calculated width does not fit all the elements of the control

PrintHeight

summary: The height of the control to print. You can change this value if the automatically calculated height does not fit all the elements of the control

RepeatArea

summary: The area to be reprinted between pages if there are more than one pages to be printed, to prevent data loss

Fields

components

summary: Required designer variable.

Fragilities Assembly

clsFragTimeConsequence

summary: A single time consequence for a damage state

remarks: Inherits from: clsFragStandardConsequence Parent: clsFragDamageState Children: None

Methods

Constructor

summary: Create a new time consequence object

clsFragStandardConsequence

summary: A single consequence for a damage state

remarks: Parent: clsFragConsequenceGroup Children: None Inherited by: clsFragCostConsequence, clsFragTimeConsequence

CurveTypes

summary: The types of curves

Fields

Normal

summary: Normally distributed

LogNormal

summary: Lognormally distributed

Methods

Validate

summary: Do a rudimentary validation of this consequence info

GetUnitAmount

summary: Get the unit amount (cost/time) given a certain number of total units

Parameters:

NumUnits(System.Double):

GetTotalUnitAmount

summary: Get the total (cost/time) given a certain number of total units

Parameters:

NumUnits(System.Double):

OutputAmountTableCSV

summary: Output the cost table as a comma separated value string

OutputXML

summary: Output the XML node representing this Consequence

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

ConsequenceNode(System.Xml.Linq.XElement):

Properties

CurveType

summary: Is this normally or lognormally distributed

LowerQuantity

summary: The lower quantity for cost discount calculations

MaxAmount

summary: The maximum cost for small quantities

UpperQuantity

summary: The upper quantity for cost discount calculations

MinAmount

summary: The minimum cost for large quantities

Uncertainty

summary: The cost dispersion

IsBlank

summary: Is this consequence completely empty

Fields

_lowerQuantity

summary: The lower quantity for cost discount calculations

_maxAmount

summary: The maximum cost for small quantities

_upperQuantity

summary: The upper quantity for cost discount calculations

_minAmount

summary: The minimum cost for large quantities

_uncertainty

summary: The uncertainty value (dispersion or COV)

_consequenceName

summary: The consequence name

_curveType

summary: Is this normally or lognormally distributed

clsFragilityCurve

summary: A single fragility curve

remarks: Parents:Many, including clsFragilityCurveLibrary Children:clsFragDamageStates, clsFragilityRatings

FragTypes

summary: The 3 basic types of fragilities

Fields

Unknown

summary: Unknown / not yet defined

Sequential

summary: Sequential

Simultaneous

summary: Simultaneous

MutuallyExclusive

summary: Mutually Exclusive

Methods

IsIDFormatValid

summary: Determine if a certain fragility ID is of a valid format

Parameters:

ID(System.String):

ReturnGlobalComponentLevel

summary: Returns just the part of the Unicode component level that you want

Parameters:

LevelNum(System.Int32): The level nunber that you wanted: C1011.009a -> 1=[C]2=[10]3=[1]4=[1]5=[009]6=[a]

FragID(System.String): The unicode ID that you are looking at

GetUpTo(System.Boolean): Do you want to return just the level, or all the above levels as well

DetermineComponentLevelOfString

summary: Given an element ID or a fragility ID, determine what the component level is. (A=1, A10=2, A101=3, A1011=4, A1011.000=5, A1011.000a=6)

Parameters:

ElementOrFragilityID(System.String):

CompareFragilityIDNum

summary: Compare two fragility

Parameters:

Fragility1(PactNet.Fragilities.clsFragilityCurve):

Fragility2(PactNet.Fragilities.clsFragilityCurve):

CombineFragCollections

summary: Combine two collections of fragilities into a single collection

Parameters:

Collection1(SSUtils.SortableBindingList{PactNet.Fragilities.clsFragilityCurve}):

Collection2(SSUtils.SortableBindingList{PactNet.Fragilities.clsFragilityCurve}):

Constructor

summary: Constructor

Constructor

summary: Constructor

ResetFragility

summary: Resets the fragility data (kind of)

_ratings_PropertyChanged

summary: Event handler for the rating object being changed. Pass it up the propertychanged chain.

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

_damageStates_PropertyChanged

summary: Event handler for the damage state group object being changed. Pass it up the propertychanged chain.

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

CompareTo

summary: Compares two fragilities by ID

Parameters:

obj(System.Object):

CompareTo

summary: Compares this fragility to another, by ID

Parameters:

other(PactNet.Fragilities.clsFragilityCurve): The other fragility object

ReturnComponentLevel

summary: Get the component level of this Fragility curve

Parameters:

LevelNum(System.Int32): The level nunber that you wanted: C1011.009a -> 1=[C]2=[10]3=[1]4=[1]5=[009]6=[a]

GetUpTo(System.Boolean):

CreateDefaultFilename

summary: Created the default filename (which is just the ID . replaced with underscore and adding .xml)

Save

summary: Save this fragility to its current location

Save

summary: Save this fragility to a given location

Parameters:

NewParentPath(System.String): The root directory it will be created in

NewProjectName(System.String): The actual fragility directory name

Load

summary: Load this fragility from a specified location

Parameters:

NewRootPath(System.String):

ProjectFolderName(System.String):

Reload

summary: Reloads this fragility from the disc

DeleteSelf

summary: Ask a fragility to delete itself off the hard drive

FixBadImageNames

summary: Verify the damage image filename actually exists, and make it blank if it does not.

OutputXML

summary: Output the XML node representing all this FragilityCurve

InputXMLStr

summary: Read an XML string representing this object and fill this object from it

Parameters:

StringXML(System.String):

InputXML

summary: Read an XML node representing this object and fill this object from it

Parameters:

FragDocument(System.Xml.Linq.XDocument):

InputXML

summary: Input the XML Fragility node and create the Fragility object and all sub-objects

remarks: This will read the fragility versions 1 and 2

Parameters:

FragilityNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Read an (old version) XML document representing this object and fill this object from it

Parameters:

XMLDocument1(System.Xml.XmlDocument):

CreateMD5Hash

summary: Create an MD5 hash of this fragility

Properties

Correlation

summary: The Correlation of the fragility

CorrelationBool

summary: A correlation boolean, is it correlated or not

NonCorrelationBool

summary: A non-correlation boolean, is it non-correlated or not

remarks: Just the opposite of CorrelationBool

Directional

summary: Is this a directional fragility

NonDirectional

summary: Is this a non-directional fragility

remarks: Just the opposite of Directional

EDPType

summary: The parameter (EDP) type of this fragility

Incomplete

summary: Is this fragility incomplete (needs user supplied data)

Notes

summary: The notes for this fragility

Ratings

summary: The ratings given to this fragility

Official

summary: Is this an offial fragility

ForceSaveAsOfficial

summary: Force the save of this fragility as "official" (once), bypassing the automatic marking of a fragility as unofficial when saving

DateCreated

summary: Date fragility created

Approved

summary: Is this fragility approved

UniqueID

summary: A generated unique ID

Author

summary: The author of this fragility

DamageStates

summary: Gets or sets the damage states object

ID

summary: Fragility ID...functionally equivalant to the OfficialID

ProjectName

summary: The name of the project, i.e. the directory name where the fragility XML file and images are stored

FullProjectPath

summary: The full root and project path

ParentPath

summary: The parent path...i.e. the dir that holds the fragility directory

XMLFileName

summary: The XML file name where this was loaded to or saved last

Name

summary: The short name

Description

summary: The long description

BasicUnit

summary: The basic unit of this fragility (if the type is not each)

UseEDPValueOfFloorAbove

summary: Should we use the EDP value of the floor above, instead of the EDP value of this floor

IsUnitEach

summary: Determine if the base unit of this fragility is Each, vs Length or Area

Fields

_uniqueID

summary: The generated unique GUID for the fragility

_iD

summary: The Uniformat ID

_xmlfileName

summary: The filename of the XML file

_parentPath

summary: The parent path...i.e. the dir that holds the fragility directory

_projectName

summary: The fragility directory name

_name

summary: The short name for the fragility

_description

summary: The longer description for the fragility

_basicUnit

summary: The unit the fragility is counted in

_author

summary: The author of the fragility

_correlation

summary: The correlation value

_directional

summary: Is this fragility directional

_EDPType

summary: The EDP type this fragility uses

_ratings

summary: The rating object

_dateCreated

summary: The date this fragility was created

_approved

summary: If this fragility has been approved

_official

summary: If this is an official fragility

_forceSaveAsOfficial

summary: Force the save of this fragility as "official" (once), bypassing the automatic marking of a fragility as unofficial when saving

_useEDPValueOfFloorAbove

summary: Should we use the EDP value from the floor above

_notes

summary: Any other notes

_damageStates

summary: The top level damage state group

_incomplete

summary: Is this fragility incomplete and needs user-entered data

LevelMapping

summary: A list containing all the string widths for each component level Level 1 = "A" Level 2 = "A10" etc.

clsFragSimultDamageState

summary: A single Simultaneous damage state

remarks: Parent: clsFragilityCurve Children: clsFragDamageStates, clsFragConsequenceGroup Inherits clsFragDamageState

Methods

OutputXML

summary: Output the XML node representing this DamageState

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

DamageStateNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an XML representation of this object and create it

Parameters:

DamageStateNode(System.Xml.XmlNode):

Properties

Percent

summary: The probability that the PG is in this damage state

Fields

_percent

summary: The probability that the PG is in this damage state

clsFragDamageState

summary: A single damage state

remarks: Parent: clsFragDamageStates Children: clsFragDamageStates, clsFragConsequenceGroup Inherited by clsFragSeqDamageState, clsSimultDamageState, clsMutExDamageState

Methods

Constructor

summary: Create a new damage state

ConsequenceGroup_PropertyChanged

summary: The event handler for changes in the underlying consequence group. Pass the propertychanged event up the chain.

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

_subStates_PropertyChanged

summary: The event handler for changes in the underlying sub damage state group. Pass the propertychanged event up the chain.

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

FixBadImageNames

summary: Verify the damage image filename actually exists, and make it blank if it does not.

SaveImages

summary: Save the damage image to someplace new (used when saving)

Parameters:

OldProjectPath(System.String): The old path for the images

NewProjectPath(System.String): The new file path for the images

OutputXML

summary: Output the XML node representing this DamageState

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

DamageStateNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an XML representation of this object and create it

Parameters:

DamageStateNode(System.Xml.XmlNode):

Properties

Name

summary: The short name of the damage state

Description

summary: The full description of the damage state

ConsequenceGroup

summary: The consequence group

FragilityType

summary: The fragility type

DamageImageName

summary: The name of the Damage image

DamageImagePath

summary: The relative path of the damage image, if not already saved in the project path

remarks: Images should be in the same dir as the XML file now

SubStates

summary: The sub damage states, if any

Fields

_name

summary: The short (less than one line) name of the damage state

_description

summary: The description of the damage state

_damageImageName

summary: The filename for the damage state

_damageImagePath

summary: The original pathname for the file Only used on new fragilities

_fragilityType

summary: The fragility Type Should be the same as parent clsFragDamageStates...don't set it here

_consequenceGroup

summary: The consequence group, or null if none

_subStates

summary: The sub damage state group, or null if none

clsElementCodes

summary: The master list of element codes

Methods

LoadElementDescriptions

summary: Load all the element descriptions from a file

Parameters:

ElementFileName(System.String):

GetElementCodeDescription

summary: Get an Element code description given the element code

Parameters:

ElementName(System.String):

GetNearestParentElement

summary: Returns the nearest parent element given a LevelID or fragilityID

returns: The XMLNode, or nothing (null)

Parameters:

LevelID(System.String): LevelId or FragilityID

GetElementsSubCodes

summary: Get All elements codes for a particular level

Parameters:

LevelID(System.String): LevelID. "" gets X (A,B,C,etc) , "A" gets all AXX, A10 gets all A10X, A101 gets all A101X, etc

OutputXML

summary: UNUSED

InputXML

summary: UNUSED:

Parameters:

InputNode(System.Xml.Linq.XElement):

Fields

ElementLibraryCodes

summary: The XML document containging all the Uniformat codes

Resource1

summary: A strongly-typed resource class, for looking up localized strings, etc.

Properties

ResourceManager

summary: Returns the cached ResourceManager instance used by this class.

Culture

summary: Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

clsFormulaEntry

summary: UNUSED One lineitem in the Occupancy Template

remarks: Parent: clsOccupancyTemplate, clsShellDefaultItem Children: clsFormulaVariable

Limits

summary: The limits of an entry, should it apply to every floor, only floor1, or only the roof

Fields

None

summary: Apply to none

Floor1

summary: Apply to floor 1

Roof

summary: Apply to roof

Roundings

summary: Should the results be rounded to a whole number

Fields

None

summary: Do not round

RoundUp

summary: Round up

RoundDown

summary: Round Down

Methods

OutputAsRPNText

summary: Output this formula entry as RPN text

Parameters:

IncludeID(System.Boolean):

FillFromRPNText

summary: Fill this formula entry from an RPN text string

Parameters:

OccupancyLine(System.String):

OutputXML

summary: Output the XML node representing this Entry

InputXML

summary: Input the XML Templates Entry Node

Parameters:

TemplateEntryNode(System.Xml.Linq.XElement):

Properties

QuantityUncertainty

summary: The quantity uncertainty

FragilityID

summary: The fragility ID

FragilityName

summary: The name of the fragility

Directional

summary: Does this only apply to directions 0 and 1 (ns and ew), or to direction 2(non-directional)

Limit

summary: The limits of an entry, should it apply to every floor, only floor1, or only the roof

Rounding

summary: Should the results be rounded to a whole number

Correlation

summary: Should it be marked as correlated

Variables

summary: The list of all the variables

RPNText

summary: Get the RPN text

Fields

_fragilityID

summary: The fragility ID

_directional

summary: Does this only apply to directions 0 and 1

_limit

summary: The limits of an entry, should it apply to every floor, only floor1, or only the roof

_rounding

summary: Should the results be rounded to a whole number

_Correlation

summary: Should it be marked as correlated

_variables

summary: The list of all the variables

_quantityUncertainty

summary: The quantity uncertainty

clsOccupancyTemplate

summary: (CURRENTLY UNUSED)An occupancy template, describing the occupancy types, populations, and probable PGs

remarks: Parent: none (used by clsBuilding and clsBuildingFloor) Children: clsFormulaEntry, clsPopulationModel

Methods

ReadAllTemplates

summary: Read all templates from a directory

Parameters:

DirName(System.String):

LoadFile

summary: Load and return an occupancy template file

Parameters:

FullFilename(System.String):

SaveTemplate

summary: Save an Occupancy Template file

Parameters:

FullFileName(System.String):

OccupancyTemplate(PactNet.Fragilities.clsOccupancyTemplate):

GetTemplate

summary: Get a template of a specific name

Parameters:

TemplateName(System.String):

MarkDirty

summary: mark as dirty

ReadRPNFile

summary: (Depreciated) Read the old RPN format and create a new template object from it

Parameters:

Filename(System.String):

Save

summary: Save the template at it's current location

SaveAs

summary: save the template to a new location

Parameters:

FullFileName(System.String):

InputXML

summary: Input the XML Templates Node

Parameters:

TemplateNode(System.Xml.Linq.XElement):

OutputXML

summary: Output the XML node representing this Template

Properties

AllTemplates

summary: The list of all templates

Filename

summary: The filename

PopulationModel

summary: The population model

TemplateName

summary: The name of the template

Entries

summary: The PG item list

Fields

IsDirty

summary: Is this dirty

clsFragMutExDamageState

summary: A single mutually Exclusive damage state

remarks: Parent: clsFragilityCurve Children: clsFragDamageStates, clsFragConsequenceGroup Inherits clsFragDamageState

Methods

OutputXML

summary: Output the XML node representing this DamageState

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

DamageStateNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an XML representation of this object and create it

Parameters:

DamageStateNode(System.Xml.XmlNode):

Properties

Percent

summary: The probability that the PG is in this damage state

Fields

_percent

summary: The probability that the PG is in this damage state

clsFragDamageStates

summary: The damage state info

remarks: Parents:clsFragilityCurve or clsFragDamageState Children:clsFragDamageState

Methods

Constructor

summary: Constructor

_states_ListChanged

summary: Event handler for when the list of damage states has been changed. Pass it up the propertychanged chain.

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

AddState

summary: Add a damage state

ConvertAllStates

summary: Convert all the damage states to another damage state type

Parameters:

DestinationStateType(PactNet.Fragilities.clsFragilityCurve.FragTypes):

ConvertState

summary: Do our best to convert a state of one type to a state of another

Parameters:

SourceState(PactNet.Fragilities.clsFragDamageState):

DestinationStateType(PactNet.Fragilities.clsFragilityCurve.FragTypes):

CreateNewState

summary: Create a new damage state

Parameters:

StateType(PactNet.Fragilities.clsFragilityCurve.FragTypes):

FixBadImageNames

summary: Verify the damage image filename actually exists, and make it blank if it does not.

Count

summary: The number of damage states

Validate

summary: Validate this fragility curve

GetState

summary: Get a specific damage state

remarks: Te damage state, or null if Index not valid

Parameters:

Index(System.Int32):

DeleteState

summary: Delete a specific damage state

Parameters:

Index(System.Int32):

DeleteState

summary: Delete a specific damage state

Parameters:

State(PactNet.Fragilities.clsFragDamageState):

ClearAllStates

summary: Clear all the state information

OutputXML

summary: Output the XML node representing all DamageStates

SaveImages

summary: Save all images, moving them if required

Parameters:

OldProjectPath(System.String): The old path for the images

NewProjectPath(System.String):

InputXMLStr

summary: Read an XML string representing this object and fill this object from it

Parameters:

StringXML(System.String):

InputXML

summary: Fill this object with information from an XML node

Parameters:

DamageStatesNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Read an XML document representing this object and fill this object from it

Parameters:

XMLDocument1(System.Xml.XmlDocument):

Properties

DSGroupType

summary: The fragility type (Sequential, MutEx, etc...)

DamageImagePath

summary: The path to the image of the damage

States

summary: Get or set the list of states

Fields

_DSGroupType

summary: The type of damage state group

_states

summary: The collection of damage states

clsFragConsequenceGroup

summary: A single group of consequences

remarks: Parent: clsFragDamageState Children: clsFragCostConsequence, clsFragTimeConsequence

TagStates

summary: The likely tag state for this damage state

Fields

Green

summary: No red tag consequences

Yellow

summary: (UNUSED) Possible yellow tag consequences

Red

summary: Possible red tag consequences

Methods

Constructor

summary: Constructor

_timeConsequence_PropertyChanged

summary: Event handler for when the underlying time consequences changes, pass it up the chain

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

_costConsequence_PropertyChanged

summary: Event handler for when the underlying cost consequences changes, pass it up the chain

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

OutputXML

summary: Output the XML node representing this Consequence group

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

DamageStateNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an XML representation of this object and create it

Parameters:

DamageStateNode(System.Xml.XmlNode):

Properties

TimeConsequence

summary: The downtime consequences

IsPossibleRedTag

summary: A boolean value indicating if the possible tag state is red, or not

remarks: Treats the tag as either red or not red (i.e. green or yellow)

LongLeadFlag

summary: Is this a long lead item

RedTagMedian

summary: The red tag median value

RedTagBeta

summary: The red tag beta value

UseCasualty

summary: Are we using the casualty rate in this fragility

AffectedFloorArea

summary: The affected floor area to examine for casualties

AffectedDeathRate

summary: The rate of deaths in the affected area

AffectedInjuryRate

summary: The rate of injuries in the affected area

AffectedDeathRateBeta

summary: The death rate dispersion

AffectedInjuryRateBeta

summary: The injury rate dispersion

TagState

summary: The tag this DS will create.

CostConsequence

summary: The cost consequences

RepairMeasures

summary: The description of the repair measures

IsBlank

summary: Is the consequences information completely blank

Fields

_repairMeasures

summary: The description of the repair measures

_tagState

summary: What the possible tag state is for this consequence

_useCasualty

summary: Is there any Casualty consequences

_affectedFloorArea

summary: The affected floor area if casualty consequences

_affectedDeathRate

summary: For casualties, the death rate in the affected area

_affectedInjuryRate

summary: For casualties, the injury rate in the affected area

_affectedDeathRateBeta

summary: For casualties, the death rate beta

_affectedInjuryRateBeta

summary: For casualties, the injury rate beta

_redTagMedian

summary: The red tag median value

_redTagBeta

summary: The red tag beta value

_longLeadFlag

summary: Is there long lead consequences

_costConsequence

summary: The cost consequences

_timeConsequence

summary: The downtime consequences

clsPopulationModel

summary: The Population Model; how many people are in the building at any time

remarks: Parent: clsOccupancyTemplate(unused), or clsPopulationPercent

Methods

ReadDefaultModels

summary: Read all the default population models from a directory

Parameters:

DirName(System.String):

LoadFile

summary: Load and return an occupancy template file

Parameters:

FullFilename(System.String):

SaveDefaultModel

summary: Save a Population Model file

Parameters:

FullFileName(System.String): The full path and filename

PopulationModel(PactNet.Fragilities.clsPopulationModel): The population model to save

GetDefaultModel

summary: Get a population model

Parameters:

ModelName(System.String):

NewDefaultModel

summary: Create a new default population model

GetWeekendOccupantsByMonth
GetWeekdayOccupantsByMonth
GetWeekendOccupantsByHour
GetWeekdayOccupantsByHour
SetWeekendOccupantsByMonth
SetWeekdayOccupantsByMonth
SetWeekendOccupantsByHour
SetWeekdayOccupantsByHour
MarkDirty

summary: Override the mark dirty command

MarkDirty
CalculatePopulationPerSF

summary: Calculate the population per SF at a certain date and time

Parameters:

TheDateTime(System.DateTime):

OutputXML

summary: Output the XML node representing this Model

InputXML

summary: Input the XML Templates Node

Parameters:

ModelNode(System.Xml.Linq.XElement):

Clone

summary: Create a clone of this object

Properties

DefaultPopulationModels

summary: Get the list of default population models

Name

summary: The name of this population model

FilePathAndName

summary: The full filepath and filename of this model

FollowingMaster

summary: If this population model is still following a "master" population model

PeakOccupantsPer1000SF

summary: The peak number of occupants per 1000 square feet

PopulationUncertainty

summary: The population uncertainty

Fields

_DefaultPopulationModels

summary: The list of default population models

_followingMaster

summary: Is this population model a slave of another pop model

_weekendOccupantsByMonth

summary: The population on the weekends, by month

_weekdayOccupantsByMonth

summary: The population on the weekdays, by month

_weekendOccupantsByHour

summary: The population on the weekends, by hour

_weekdayOccupantsByHour

summary: The population on the weekdays, by hour

_peakOccupantsPer1000SF

summary: The peak number of occupants per 1000 square feet

_name

summary: The name of this popmodel

_filePathAndName

summary: The filepath and name of this popmodel

_populationUncertainty

summary: The population dispersion

clsFragilityCurveLibraries

summary: The collection of all the fragility curve libraries

remarks: Child: clsFragilityCurveLibrary

Methods

UseUsersCurrentLibrary

summary: Read the users settings and use that library

ChangeCurrentLibrary

summary: Change the current library

Parameters:

LibraryPath(System.String):

ChangeCurrentLibrary

summary: Change the current library

Parameters:

NewLib(PactNet.Fragilities.clsFragilityCurveLibrary):

CreateLibraryFromDir

summary: Create and fill a library from a directory

Parameters:

Directory(System.String): The directory to load from

ErrorMessages(System.String@): The list of error messages recieved when loading

CreateLibraryFromXML

summary: Create and fill a library from an XML fragment

Parameters:

RootNode(System.Xml.Linq.XElement): The XML node to load from

ErrorMessages(System.String@): The list of error messages recieved when loading

CreateLibraryFromList

summary: Create and fill a library from a list of fragilities

Parameters:

CurvesToLoad(System.Collections.Generic.IList{PactNet.Fragilities.clsFragilityCurve}): The list of fragility curves to load

ErrorMessages(System.String@): The list of error messages recieved when loading

GetDefaultLibraryDirectory

summary: The directory of the default library

OutputXML

summary: NOT USED

InputXML

summary: NOT USED

Parameters:

InputNode(System.Xml.Linq.XElement):

Properties

CurrentLibrary

summary: Get the current library

Fields

_currentLibrary

summary: The currently selected library

clsFragilityCurveLibrary

summary: A single fragility library

remarks: Parent clsFragilityCurveLibraries Child:clsFragilityCurve

Methods

NewFragility

summary: Create a new fragility curve from scratch

NewFragility

summary: Create a fragility curve from an XML node

Parameters:

XMLNode(System.Xml.Linq.XElement):

NewFragility

summary: Create a new fragility curve from a project folder

Parameters:

RootPathName(System.String):

XMLFolderName(System.String):

NewFragility

summary: Create a new fragility curve from copying an existing curve

Parameters:

SourceCurve(PactNet.Fragilities.clsFragilityCurve): The source fragility

NewFragID(System.String): The new fragility ID

GetElementsSubCodes

summary: Get All elements codes for a particular level

Parameters:

LevelID(System.String): LevelID. "" gets X (A,B,C,etc) , "A" gets all AXX, A10 gets all A10X, A101 gets all A101X, etc

AddAnyFragilitiesToList(System.Boolean): Should we add the actual existing fragility IDs to the list of element codes being returned

GetCurvesByLevel

summary: Returns all of the actual fragility curves that are under the Element level (e.g. A11XXX.XX or EXXXX.XX)

Parameters:

LevelID(System.String):

GetCurveByID

summary: Get a curve by the Curve ID

returns: The fragility curve, or null if not found

Parameters:

CurveID(System.String):

GetNonDirectionalOnly

summary: Get a list of all frag curves that are non-directional

LoadAllFragilities

summary: Load all fragility curves from a specific root directory (Deletes whatever was there before)

returns: Error messages if any, blank if everything OK

Parameters:

RootFolderName(System.String):

LoadAllFragilities

summary: Load all fragility curves from a given list (will delete everything there before)

returns: Error messages if any, blank if everything OK

Parameters:

CurvesToLoad(System.Collections.Generic.IList{PactNet.Fragilities.clsFragilityCurve}): The list of curves to load

LoadAllFragilities

summary: Load all fragility curves from an XML root node (Deletes whatever was there before)

returns: Error messages if any, blank if everything OK

Parameters:

RootNode(System.Xml.Linq.XElement): The root node containing all the fragilty nodes

DeleteFragility

summary: Will delete a fragility curve from the internal collection, and off the filesystem too

Parameters:

FragilityID(System.String): The ID of the fragility to delete

DeleteFragility

summary: Will delete a fragility curve from the internal collection, and off the filesystem too

Parameters:

CurveToDelete(PactNet.Fragilities.clsFragilityCurve): The fragility curve object to delete

IsValidFragilityProjectFolder

summary: Determine if a particular project folder is a valid fragility folder or not

Parameters:

NewRootPath(System.String):

ProjectFolderName(System.String):

SortByOrderNum

summary: Sort all fragilities

Duplicate

summary: Create a deep copy of this fragility

OutputXML

summary: NOT USED

InputXML

summary: NOT USED

Parameters:

InputNode(System.Xml.Linq.XElement):

Properties

FragilityCurves

summary: Get or set the list of all fragilities

Fields

BaseDirectory

summary: The base directory of this library

IsDefaultLibrary

summary: Is this the default library

_fragilityCurves

summary: The list of all the fragility curves

clsFragSeqDamageState

summary: A single sequential damage state

remarks: Parent: clsFragilityCurve Children: clsFragDamageStates, clsFragConsequenceGroup Inherits clsFragDamageState

Methods

OutputXML

summary: Output the XML node representing this DamageState

InputXML

summary: Input the XML Damage State and create the DS object and all sub-objects

Parameters:

DamageStateNode(System.Xml.Linq.XElement):

InputXMLOld

summary: Input an XML representation of this object and create it

Parameters:

DamageStateNode(System.Xml.XmlNode):

Properties

Median

summary: The Median value of the fragility curve

Beta

summary: The dispersion of the fragility curve

Fields

_median

summary: The Median value of the fragility curve

_beta

summary: The dispersion value of the fragility curve

Utilities

summary: Miscellaneous utility functions specific to PactNet

Methods

FragilityFillTreeNode

summary: Fill a tree control with fragility curves

Parameters:

TreeviewControl(System.Windows.Forms.TreeView):

CurrentLibrary(PactNet.Fragilities.clsFragilityCurveLibrary): The current frag library being used

FragilityFirstTreeNodeFill

summary: Will Fill a tree control with fragility curves

Parameters:

LevelCode(System.String):

ParentNode(System.Windows.Forms.TreeNode):

CurrentLibrary(PactNet.Fragilities.clsFragilityCurveLibrary): The current fragility library

FragilityFirstTreeNodeFillDirectional

summary: Will Fill a tree control with fragility curves

Parameters:

LevelCode(System.String):

ParentNode(System.Windows.Forms.TreeNode):

Directional(System.Boolean):

CurrentLibrary(PactNet.Fragilities.clsFragilityCurveLibrary): The current frag library being used

FragilityBoldOccupiedNodes

summary: Will bold all the nodes that actually have fragilities

Parameters:

ParentNode(System.Windows.Forms.TreeNode):

FragilityColorNodes

summary: Will color all the nodes matching a series of lists (ParentNode, (ListofNodes1, color1)*)

Parameters:

ParentNode(System.Windows.Forms.TreeNode):

ColorParentNode(System.Boolean):

ListAndColor(System.Object[]):

FindNode

summary: Recursively finds a specific treeview node given the name

Parameters:

ParentNode(System.Windows.Forms.TreeNode):

NodeName(System.String):

clsFragilityRating

summary: The rating of a particular fragility Parent: clsFragilityCurve

QualityRatings

summary: The quality ratings

Fields

NA

summary: N/A

Marginal

summary: Marginal

Average

summary: Average

Superior

summary: Superior

Methods

GetRatingString

summary: Gets the string version of the rating

GetEnumFromString

summary: Given a rating string, get the actual Enum value

Parameters:

RatingString(System.String):

GetEnumList

summary: Get a list of the Enum strings

InputXML

summary: Input the XML rating node and fill the rating object

remarks: This will read the fragility versions 1 and 2

Parameters:

RatingNode(System.Xml.Linq.XElement):

OutputXML

summary: Returns an XML node containing the Rating object and all sub-objects

Properties

DataQuality

summary: The quality of the data

DataRelevance

summary: The relevance of the data

Documentation

summary: The documentation backing the data

Rationality

summary: The rationality of the data

DataQualityNum

summary: The quality of the data, expressed as a number between 0 and 3

DataRelevanceNum

summary: The relevance of the data, expressed as a number between 0 and 3

DocumentationNum

summary: The documentation quality of the data, expressed as a number between 0 and 3

RationalityNum

summary: The rationality of the data, expressed as a number between 0 and 3

DataQualityString

summary: The data quality, expressed as a string

DataRelevanceString

summary: The data relevance, expressed as a string

DocumentationString

summary: The data documentation, expressed as a string

RationalityString

summary: The data rationality, expressed as a string

Fields

_dataQuality

summary: The quality of the data

_dataRelevance

summary: The relevance of the data

_documentation

summary: The documentation backing the data

_rationality

summary: The rationality of the data

clsFragCostConsequence

summary: A single cost consequence for a damage state

remarks: Inherits from: clsFragStandardConsequence Parent: clsFragDamageState Children: None

Methods

Constructor

summary: Create a new cost consequence

clsFormulaVariable

summary: A single variable in an Occupancy Template lineitem

remarks: Parent: clsFormulaEntry Child: None

EntryTypes

summary: The type of entry this variable is

Fields

Unknown

summary: Unknown/Not Defined

Constant

summary: A hardcoded number

Op

summary: An operator, such as + or -

BuildingVar

summary: A building information variable

BuildingVars

summary: All possible building information variables

Fields

Unknown

summary: Unknown / Not Defined

Perimeter

summary: Perimeter length in a particular direction

Height

summary: Height of the Floor

Area

summary: Area of Current Floor

AreaAbove

summary: Area of the Floor Above

TotalFloorArea

summary: Total Area of All Floors

Methods

IsBuildingVar

summary: Is the string fragment a building information type

Parameters:

TestString(System.String):

ParseVar

summary: Parse a string fragment

Parameters:

varString(System.String):

MarkClean

summary: Marks the object (and sub-objects) as clean

MarkDirty

summary: Marks the object as Dirty (in need of saving)

MarkDirty

summary: Marks the project as Dirty (in need of saving)

Parameters:

PropertyName(System.String):

MarkNew

summary: Mark this object as new, and in need of saving

MarkOld

summary: Mark this object as old, and not in need of saving

ToString

summary: Output a string representation of this variable

Properties

EntryType

summary: The type of this entry

BuildingVar

summary: If it is a building var, the type

Variable

summary: The original variable string

ConstantValue

summary: A constant value

IsNew

summary: Determines if the object is new and unsaved

IsDirty

summary: Determines if the object (or sub-objects) is Dirty (in need of saving)

IsValid

summary: Determines if the object (or sub-objects) has valid data

Fields

_entryType

summary: The type of this entry

_buildingVar

summary: If it is a building var, the type

_variable

summary: The original string variable

clsEDPType

summary: The EDP type information

remarks: Parent: clsEDPArray Children:None

DimensionType

summary: How to ascertain the dimension (number of EDPs) needed by this EDP type

Fields

Unknown

summary: Unknown or defined type

NumLevels

summary: Should be equal to the number of levels of the building

NumLevelsPlusOne

summary: Should be equal to the number of levels of the building, plus one for the roof

SingleValue

summary: Should be just on value for the entire building

Methods

NewEDPType

summary: Create a new EDP Type

Parameters:

TheTypeName(System.String):

NewEDPType

summary: Create a new EDP Type, and automatically adds it to the global list

Parameters:

TheTypeName(System.String):

TheDefaultUnits(SSUtils.SSLocalization.Unit):

TheUserDefined(System.Boolean):

TheDimension(DimensionType):

NewEDPType

summary: Create a new EDP type from an XML formatted node

Parameters:

EDPNode(System.Xml.Linq.XElement):

Static Constructor

summary: STATIC constructor

LoadAllXML

summary: Read all EDP Type information from an XML document

Parameters:

XMLDocument1(System.Xml.Linq.XDocument):

GetEDPTypeByName

summary: Get a specific EDP type by name (null if not found)

Parameters:

TheTypeName(System.String):

DoesEDPTypeExist

summary: Determine if an EDP type exists

Parameters:

TheTypeName(System.String):

DeleteEDPType

summary: Delete an EDP type

Parameters:

DeletedEDPType(PactNet.Fragilities.clsEDPType):

GetHashCode

summary: Get a hash code for the EDP type

Equals

summary: Determine if two EDP typenames are equal

Parameters:

obj(System.Object):

op_Equality

summary: Equals override

Parameters:

EDPType1(PactNet.Fragilities.clsEDPType):

EDPType2(PactNet.Fragilities.clsEDPType):

op_Inequality

summary: Not equals override

Parameters:

EDPType1(PactNet.Fragilities.clsEDPType):

EDPType2(PactNet.Fragilities.clsEDPType):

OutputXML

summary: Output an XML representation of this object

InputXML

summary: Input an XML representation of this object

Parameters:

EDPNode(System.Xml.Linq.XElement):

ToString

summary: Returns a string with the Typename

Properties

Types

summary: Get or set this list of all types

TypeName

summary: The English name for the type

DefaultUnits

summary: The default units

UserDefined

summary: Is this type user defined

Dimension

summary: Get the dimension type (how to determine how many there are)

FragilityManager Assembly

frmImportFromExcel

summary: A form and code for importing data from an Excel spreadsheet

Methods

Constructor

summary: Constructor

btnImport_Click

Parameters:

sender(System.Object):

e(System.EventArgs):

ImportATC58Spreadsheet

summary: Import the Excel spreadsheet in the official format

ImportCraigSpreadsheet

summary: Import the spreadsheet format that Craig was using

UniversalImport

summary: A universal import routine

Parameters:

ImportTable(System.Data.DataTable): A table that contains some or all of the fields required to import a fragility

DamageImagePath(System.String): The path to the damage state images (if any)

ParseDSHeirarchy

summary: Parse the DSHeirarchy string and create the requred clsFragDamageState and clsFragDamageStates(groups) objects

Parameters:

TempFragCurve(PactNet.Fragilities.clsFragilityCurve): The fragility curve

TempRow(System.Data.DataRowView): The row from the imported table

ErrorStringHeader(System.String): The header to use for any error strings

DamageImagePath(System.String): The path to the damage images

ParseDSLevel

summary: Parse a particular damage state level

returns: A string containing any error

Parameters:

CurrentActualDS(System.Int32): The actual damage state being looked at

TempFragCurveDSGroup(PactNet.Fragilities.clsFragDamageStates): The damage state group object

TempRow(System.Data.DataRowView): The source imprted row

ErrorStringHeader(System.String): The string to use for any error

DSHeirarchy(System.String): The currently DSHeirarchy fragment to parse

DamageImagePath(System.String): The path containg any damage images

SplitDSHeirarchy

summary: Take a string that may have substates, and returns each substate level in List, taking into account nested parens

returns: A list containing each item (or subitem group) that was separated by commas

Parameters:

HeirarchyString(System.String): The Heirarchy string at this level

ConvertFieldToBoolean

summary: Parse and convert a string field containing a boolean to an actual boolean type.

returns: A boolean, or an ArgumentException if not parseable

Parameters:

InputField(System.String): A field starting with y, n, true, false, 1, or 0

ConvertFieldToCurveType

summary: Parse and convert a string field containing a curvetype to an actual CurveTypes object.

returns: A CurveTypes object, or an ArgumentException if not parseable

Parameters:

InputField(System.String): A field starting with either (l)ognormal or (n)ormal

ConvertFieldToDouble

summary: Convert a field into a double value

returns: The double, or an ArgumentException is not parseable

Parameters:

InputField(System.String): The string holding the double representation

ConvertFieldToQualityRatings

summary: Convert a string field containing a QualityRatings to an actual QualityRatings object.

returns: The quality rating, or QualityRatings.NA is not parseable

Parameters:

InputField(System.String): String field containing a QualityRatings

ConvertFieldToBasicUnit

summary: Convert a string field (or fields) containing a unit type (sf, lf, etc) to a Unit object

returns: The parsed Unit object, or an Exception if not parsable

Parameters:

BasicUnit1(System.String): Either the entire unit type string, or the quantity type (Area, length, each, etc)

BasicUnit2(System.String): The actual unit (Square foot, g, etc)

ConvertFieldToMeasurement

summary: Convert a string field to a measurement

Parameters:

BasicUnit1(System.String):

ConvertFieldToDate

summary: Convert a string field containing a date to an actual DateTime object.

returns: The parsed date, or Exception if not parsable

Parameters:

DateField(System.String): The string field

ConvertFieldToEDPType

summary: Convert a string field containing the EDP type to an actual clsEDPType object.

returns: The EDP Type object, or an exception if the EDP type is not found

Parameters:

EDPTypeField(System.String): The string version of the EDPtype

ParseDamageStateInfo

summary: Parse the data for a single damage state

returns: A list of all the (possible) errors

Parameters:

SpreadsheetDSNum(System.Int32): The damage state number to parse from the spreadsheet that contains the DS values

ActualDSNum(System.Int32): The actual DS Num

TheDamageState(PactNet.Fragilities.clsFragDamageState): The damage state object

TempRow(System.Data.DataRowView): The row from the imported table

ErrorStringHeader(System.String): The header to put on the errorstring if there is an error

ParseConsequenceGroupInfo

summary: Parse the data for a particular consequence group

returns: A list of all the (possible) errors

Parameters:

DSNum(System.Int32): The damage state

TheConsequenceGroup(PactNet.Fragilities.clsFragConsequenceGroup): The Consequencegroup object

TempRow(System.Data.DataRowView): The row on the imported table

ErrorStringHeader(System.String): The header to put on the errorstring if there is an error

ParseConsequenceInfo

summary: Parse out a consequence info object, whether it be time or cost

returns: A list of all (possible) error

Parameters:

DSNum(System.Int32): The damage state number

ConsequenceType(System.String): The type of consequence, either "Time" or "Cost" (case matters)

TheConsequence(PactNet.Fragilities.clsFragStandardConsequence): The Consequence object

TempRow(System.Data.DataRowView): The imported datarow containing the fragility info

ErrorStringHeader(System.String): The header to add to the errorstring

ParseField

summary: Parse a single field

Parameters:

ParseInfo(PactNet.FragilityManager.ParseErrorInfo): The object containing all the field and error information

ExtensiveCheckOfSeqFragility

summary: Does a more extensive check of a sequential fragility to determine if the medians are ordered correctly and there is no overlap of the frag curves.

Parameters:

DamageStateGroupToCheck(PactNet.Fragilities.clsFragDamageStates):

ErrorStringHeader(System.String):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

components

summary: Required designer variable.

ParseException

summary: A parse exception objct, containing the error info

Methods

Constructor

summary: Constructor

Parameters:

TheErrorInfo(PactNet.FragilityManager.ParseErrorInfo): The error info associated with this exception

Fields

ErrorInfo

summary: The errorinfo associated with this exception

ParseErrorInfo

summary: A class to hold information about error parsing the excel file

ErrorLevels

summary: The possible error levels

Fields

Fatal

summary: A fatal error, abort the load process

DataMissing

summary: Some data is missing, flag the fragility

MiscWarning

summary: A misc warning

OverlapWarning

summary: Some Damage states are overlapping.

MinorWarning

summary: Some other minor warning not worth bringing to the users attention

Methods

Constructor

summary: Basic constructor

Constructor

summary: Constructor filling out all the available information when only one column is involved

Parameters:

iBaseObject(System.Object): The base fragility object that this error pertains to

iObjectFieldName(System.String): The field name in the fragility object

iUseDefaultValue(System.Boolean): Should we use the default value if there was an error, or just die

iDefaultValue(System.Object): The default value to use in case of an error

iTableFieldName1(System.String): The field name of the table this pertains to

iTableRow(System.Data.DataRowView): The row this comes from

iErrorLevel(ErrorLevels): What is the severity of the error

iErrorStringHeader(System.String): The header to put on the error message

iPublicErrorString(System.String): The actual error message itself

iResolveString(System.String): The text for the information on how to resolve the error

Constructor

summary: Constructor filling out all the available information when only one column is involved

Parameters:

iBaseObject(System.Object): The base fragility object that this error pertains to

iObjectFieldName(System.String): The field name in the fragility object

iUseDefaultValue(System.Boolean): Should we use the default value if there was an error, or just die

iDefaultValue(System.Object): The default value to use in case of an error

iTableFieldName1(System.String): The 1st field name of the table this pertains to

iTableFieldName2(System.String): The 2nd field name of the table this pertains to

iTableRow(System.Data.DataRowView): The row this comes from

iErrorLevel(ErrorLevels): What is the severity of the error

iErrorStringHeader(System.String): The header to put on the error message

iPublicErrorString(System.String): The actual error message itself

iResolveString(System.String): The text for the information on how to resolve the error

ToString

summary: Return the text description of the error

Properties

WasError

summary: A boolean indicating if there actually was an error for this field

Fields

BaseObject

summary: The base fragility object that this error pertains to

ObjectFieldName

summary: The field name in the fragility object

UseDefaultValue

summary: Should we use the default value if there was an error, or just die

DefaultValue

summary: The default value to use in case of an error

TableFieldName1

summary: The field name of the table this pertains to

TableFieldData1

summary: The value of the database field

TableFieldName2

summary: The field name of the table this pertains to, if the value uses more than one field

TableFieldData2

summary: The value of the database field, if the value uses more than one field

TableRow

summary: The imported datarow containing the information for this fragility

ErrorLevel

summary: The severity of the error

ErrorStringHeader

summary: The header to put on the error message

PublicErrorString

summary: The actual error message itself

ExceptionObject

summary: The Exception object containing the error

ResolveString

summary: The text for the information on how to resolve the error

IsChecked

summary: A boolean indicating this entry has already been checked

_wasError

summary: A boolean indicating if there actually was an error for this field

frmAddNewEDPType

summary: The name is a bit misleading, this will allow you to add OR EDIT an EDPType

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Form Constructor

frmAddNewEDPType_Load

summary: The form load event handler

Parameters:

sender(System.Object):

e(System.EventArgs):

cboBasicUnitType_SelectedIndexChanged

summary: The basic unit type has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

btnOK_Click

summary: The OK button has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCancel_Click

summary: The cancel button has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

Properties

EDPType

summary: The EDP type

Fields

components

summary: Required designer variable.

Mainlog

summary: The logger object

CurrentEDPType

summary: The current EDP type

IsNewEDP

summary: Is this a new EDP, or a previously created one

frmEditFragility

summary: The main form for editing fragility functions

TreeNodePointer

summary: Create a new type of node that holds my misc location data

NodeTypes

summary: What type this node is

Fields

RootNode

summary: A root level node, showing general information NodeObject is null

TopLevelGroup

summary: The top level damage state group NodeObject is of type clsFragDamageStates

DamageStates

summary: A damage state group (other than the top one) NodeObject is of type clsFragDamageStates

DamageState

summary: A damage state NodeObject is of type clsFragDamageState

Consequence

summary: A consequence group NodeObject is of type clsFragConsequenceGroup

AddConsequence

summary: The "Add Consequence" node NodeObject is null

AddDamageState

summary: The "Add Damage State" Node NodeObject is null

AddSubStates

summary: The "Add Subdamage States" Node NodeObject is null

Methods

Constructor

summary: Basic constructor

Constructor

summary: Constructor with a given name

Parameters:

Name(System.String): The node name to use

Properties

IsAddNode

summary: Is this is an "add something" node

Fields

NodeType

summary: The node type

NodeObject

summary: The underlying object associated with this node, depedning on NodeType Will be clsFragDamageStates, clsFragDamageState, clsFragConsequenceGroup, or null

Location

summary: The location of the node, in the form of int:int:int depending on how deep the node and how many nodes are at the same level 1 is always the root node, 1:1 is always the first DSGroup, etc

Methods

Constructor

summary: Form constructor

MyHandler

summary: Catch unexpected exceptions in the code

Parameters:

sender(System.Object):

args(System.UnhandledExceptionEventArgs):

EditFragility_Load

summary: The load event for the form

Parameters:

sender(System.Object):

e(System.EventArgs):

SetDefaultLibrary

summary: Set the library to whatever the user default is

UpdateLibrary

summary: Change the UI to reflect that the Fragility Library changed

SaveAll

summary: Save all the fragilities. Returns a bool indicating complete success (the user did not cancel);

frmEditFragility_FormClosing

summary: Form Closing event

Parameters:

sender(System.Object):

e(System.Windows.Forms.FormClosingEventArgs):

AddUnitConvertParseAndFormat

summary: Add the units measurement conversion routines to a control

Parameters:

dataSource(System.Object):

dataMember(System.String):

ControlObject(System.Windows.Forms.Control):

ParseUnitMeasurements

summary: Parse the unit measurements in whatever the current system is

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

FormatUnitMeasurements

summary: Format the unit measurements into whatever the current system is

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

tvAllFragilities_AfterSelect

summary: When the user has selected a fragility in the global treeview

Parameters:

sender(System.Object):

e(System.Windows.Forms.TreeViewEventArgs):

txtMedianFormatHandler

summary: A handler for the formatting of the median textbox. Used to deal with different measurement systems

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

txtMedianParseHandler

summary: A handler for the parsing of the median textbox. Used to deal with different measurement systems

Parameters:

sender(System.Object):

e(System.Windows.Forms.ConvertEventArgs):

bsClsFragilityCurve_CurrentChanged

summary: If the current fragility we are looking at changes

Parameters:

sender(System.Object):

e(System.EventArgs):

UpdateFragilityStatus

summary: Update the fragility status information (i.e. fragility ID and save status)

Parameters:

CurrentFragility(PactNet.Fragilities.clsFragilityCurve):

SetupCurveTreeView

summary: Do the setup for the tree view of a particular fragility

Parameters:

CurrentFragility(PactNet.Fragilities.clsFragilityCurve):

ChooseTreeViewPlace

summary: Attempts to select the correct node in the fragility treeview

tvSingleCurve_AfterSelect

summary: When they click on a node in the fragility treeview, we need to show the information on the right panel

Parameters:

sender(System.Object):

e(System.Windows.Forms.TreeViewEventArgs):

tvSingleCurve_NodeMouseClick

summary: When they click on the fragility nodes treeview, pop up the context menu

Parameters:

sender(System.Object):

e(System.Windows.Forms.TreeNodeMouseClickEventArgs):

AddConsequence

summary: when the add a new consequence group to a damage state

Parameters:

DamageState(PactNet.Fragilities.clsFragDamageState):

AddSubStates

summary: Trying to add a sub-damage-state-group

Parameters:

DamageState(PactNet.Fragilities.clsFragDamageState):

AddDamageState

summary: Add a new damage state to a damage states group

Parameters:

DamageStates(PactNet.Fragilities.clsFragDamageStates):

SetupSingleCurveTreeNodeFill1

summary: Setup a damage states group node in a fragility's treeview

Parameters:

DamageStates(PactNet.Fragilities.clsFragDamageStates):

CurrentNode(TreeNodePointer):

LevelNum(System.Int32):

SetupSingleCurveTreeNodeFill2

summary: Setup a single damage state node in a fragility's treeview

Parameters:

DamageState(PactNet.Fragilities.clsFragDamageState): The damage state object to change

DSNum(System.Int32): Damage state number

CurrentNode(TreeNodePointer): The current treenode

LevelNum(System.Int32): The Level number

tvSingleCurve_KeyDown

summary: Keydown on the fragility info

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

DeleteNode

summary: Delete a specific node on the fragility info treeview

Parameters:

SelectedTreeNode(TreeNodePointer):

bsClsDamageState_CurrentChanged

summary: When the Damage State is changed

Parameters:

sender(System.Object):

e(System.EventArgs):

ShowCurrentDamageImage

summary: Show the image of the current damage state

ShowMedianLabel

summary: Show the median label

DrawConsequenceInternal

summary: Draw the consequence graph

Parameters:

MinCost(System.Double): The min cost

MaxCost(System.Double): The max cost

LowerQuantity(System.Double): The lower quantity

UpperQuantity(System.Double): The upper quantity

ConsequenceGraph(Nevron.Chart.WinForm.NChartControl): The graph object to draw to

GraphTitle(System.String): The title of the graph

YAxisTitle(System.String): The Y axis title

SetupStatesGraph

summary: Setup the damage states graph

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

ShowDamageStatesGraph

summary: Show the damage states graph

Parameters:

IsTopLevelGroup(System.Boolean):

DrawSeqCurves

summary: Draw the sequential curves

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

BaseFragility(PactNet.Fragilities.clsFragilityCurve):

DamageStates(PactNet.Fragilities.clsFragDamageStates):

DrawSeqCurvesSubtracted

summary: UNTESTED - Draws the seq curves subtracted from one another

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

BaseFragility(PactNet.Fragilities.clsFragilityCurve):

DamageStates(PactNet.Fragilities.clsFragDamageStates):

DrawSimultStates

summary: Draw the simultaneous bar graph

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

DamageStates(PactNet.Fragilities.clsFragDamageStates):

DrawMutExStates

summary: Draw the mutually exclusive pie chart

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

DamageStates(PactNet.Fragilities.clsFragDamageStates):

ClearStatesGraph

summary: Clear the damage states graph

Parameters:

AllSubDSGraph(Nevron.Chart.WinForm.NChartControl):

DrawDSConsequence

summary: Draw a consequence graph

Parameters:

CurrentConsequence(PactNet.Fragilities.clsFragStandardConsequence): The current consequence object

ConsequenceGraph(Nevron.Chart.WinForm.NChartControl): The consequence graph to draw to

GraphTitle(System.String): The title of the graph

YAxisTitle(System.String): The Y Axis title

ClearDSConsequence

summary: Clear the consequences information and graph

Parameters:

ConsequenceGraph(Nevron.Chart.WinForm.NChartControl):

SaveAsToolStripMenuItem_Click

summary: User clicked on the Save As menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveToolStripMenuItem_Click

summary: User clicked on the Save menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

deleteToolStripMenuItem_Click

summary: User clicked on the delete menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCustomAddNew_Click

summary: The user clicked on the add new fragility navbar item

Parameters:

sender(System.Object):

e(System.EventArgs):

OptionsToolStripMenuItem_Click

summary: The user clicked on the Options menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

NewToolStripMenuItem_Click

summary: The user clicked on the New fragility menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

mnuDuplicateFragility_Click

summary: The user clicked on the duplicate fragility menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

ExitToolStripMenuItem_Click

summary: The user clicked on the Exit menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

BindingNavigatorDeleteItem_Click

summary: The user clicked on the Delete Nav bar item

Parameters:

sender(System.Object):

e(System.EventArgs):

ClsFragilityCurveBindingNavigatorSaveItem_Click

summary: The user clicked on the save navbar item

Parameters:

sender(System.Object):

e(System.EventArgs):

CopyToolStripMenuItem_Click

summary: The user clicked on the Copy menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PasteToolStripMenuItem_Click

summary: The user clicked on the Paste menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

CutToolStripMenuItem_Click

summary: The user clicked on the Cut menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SelectAllToolStripMenuItem_Click

summary: The user clicked on the Select All menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

pageSetupToolStripMenuItem_Click

summary: The user clicked on the Page Setup menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PrintToolStripMenuItem_Click

summary: The user clicked on the Print menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

PrintPreviewToolStripMenuItem_Click

summary: The user clicked on the Print Preview menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

AboutToolStripMenuItem_Click

summary: The user clicked on the About menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

FragilitySaveAs

summary: Save the fragility at a specific location

SaveFragility

summary: Save a fragility in it's current location

Parameters:

FragilityToSave(PactNet.Fragilities.clsFragilityCurve):

ShowGeneralInfo

summary: Show the general info (and notes) tab and hide the others

ShowDamageStateGroup

summary: Show the damage state tab and hide the others

Parameters:

IsTopLevelGroup(System.Boolean):

ShowDamageStateInfo

summary: Show the damage state info tab and hide the other

ShowConsequenceInfo

summary: Show the consequence tab and hide the other

AddNewFragility

summary: Add a new fragility curve...ask for the ID

DuplicateFragility

summary: Add a new fragility curve...ask for the ID

bsClsFragilityCurve_AddingNew

summary: Handler for when a new fragility curve has been added

Parameters:

sender(System.Object):

e(System.ComponentModel.AddingNewEventArgs):

btnFragSearch_Click

summary: The user clicked the search button

Parameters:

sender(System.Object):

e(System.EventArgs):

SearchFunction

summary: Find a string in any of the fragility functions

Parameters:

TextToFind(System.String):

txtFragSearch_KeyDown

summary: Handler for when the user hits enter in the search function box

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

bsClsConsequenceGroup_CurrentChanged

summary: When a new consequence group has been chosen

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsFragCostConsequence_CurrentItemChanged

summary: When the current cost consequence has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsFragTimeConsequence_CurrentItemChanged

summary: When the current time consequence has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsFragDamageStates_CurrentChanged

summary: Handler for when the current damage states group changes

Parameters:

sender(System.Object):

e(System.EventArgs):

MeasureSystemChanged

summary: If the user changed from metric to imperial or viceversa

Parameters:

NewMeasureSystem(SSUtils.SSLocalization.StandardSystems):

cboBasicUnitType_SelectedIndexChanged

summary: If they change the unit type (length, area, etc), we need to change the list of actual units (inches, feet, etc)

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCreateNewDP_Click

summary: Create a brand new EDP type

Parameters:

sender(System.Object):

e(System.EventArgs):

btnAddImage_Click

summary: The user hit the button to add a damage state image

Parameters:

sender(System.Object):

e(System.EventArgs):

cboCostCurveType_SelectedValueChanged

summary: If we change the curve type, set it manually

Parameters:

sender(System.Object):

e(System.EventArgs):

cboTimeCurveType_SelectedValueChanged

summary: If we change the curve type, set it manually

Parameters:

sender(System.Object):

e(System.EventArgs):

chkUseCollapse_CheckStateChanged

summary: If we change the Use Collapse box, we need to make some fields enabled or not

Parameters:

sender(System.Object):

e(System.EventArgs):

EnableUseCasualtyFields

summary: If we change the Use Collapse box, we need to make some fields enabled or not

Parameters:

UseFields(System.Boolean): Should we use the collapse fields, or disable them?

EnableRedTagFields

summary: If we change the enable red tag fields, we need to make some fields enabled or not

Parameters:

UseFields(System.Boolean): Should we use the red tag fields, or disable them?

chkDSRedTagPossible_CheckStateChanged

summary: If we change the enable red tag fields, we need to make some fields enabled or not

Parameters:

sender(System.Object):

e(System.EventArgs):

btnEditEDP_Click

summary: Editing one of the existing EDP types

Parameters:

sender(System.Object):

e(System.EventArgs):

DeleteCurrentFragility

summary: Deletes a fragility curve from the internal collection and the database (with confirmation)

bsClsFragilityCurve_CurrentItemChanged

summary: Handler when the current fragility curve has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

bsClsFragilityCurve_ListChanged

summary: Handler for when the list of fragility curves has been changed

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

ChangeLibrary

summary: Change the current fragility library

Parameters:

SelectedPath(System.String):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

CurrentNode

summary: The currently selected node of the fragility function treeview

Mainlog

summary: The logging object

InFormLoad

summary: Are we currently loading the form

components

summary: Required designer variable.

frmAddNewFragility

summary: The form for adding a new fragility. Used to get a uniformat fragility ID

Methods

btnOK_Click

summary: The user clicked the OK button

Parameters:

sender(System.Object):

e(System.EventArgs):

FragilityIDEntered

summary: The fragility ID has been entered. Verify and close the form

btnCancel_Click

summary: The cancel button has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

txtFragilityID_KeyDown

summary: Event handler for the keydown in the textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

Constructor

summary: Form constructor

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

FragID

summary: The fragility ID enetered

components

summary: Required designer variable.

PactEngine Assembly

frmMainEngine

summary: Form used for running the engine

NewResultAvailableEventHandler

summary: Delegate for a new available result

Parameters:

sender():

Methods

BroadcastNewAvailableResult

summary: Go ahead and broadcast that a new event is available

Constructor

summary: Form constructor

frmMainEngine_Load

summary: Called when loading the form

Parameters:

sender(System.Object):

e(System.EventArgs):

FillRecentProjects

summary: Fill the recent project list

RecentBuildingProjects_Click

summary: Called when the user clicks an item in the recent project list

Parameters:

sender(System.Object):

e(System.EventArgs):

LoadProject

summary: Load a project from a filename

Parameters:

ProjectPath(System.String):

ProjectName(System.String):

LoadProjectInternal

summary: The internal version of LoadProject...catches no errors

Parameters:

ProjectPath(System.String):

ProjectName(System.String):

LoadProject

summary: Load a project from a filename

Parameters:

ProjectPathAndName(System.String): The full path and filename to load

mnurdoMainOptionsThreadingYes_CheckedChanged

summary: Called when the user changes Threading Option menu item to Yes

Parameters:

sender(System.Object):

e(System.EventArgs):

mnurdoMainOptionsThreadingNo_CheckedChanged

summary: Called when the user changes Threading Option menu item to No

Parameters:

sender(System.Object):

e(System.EventArgs):

mnurdoMainOptionsExportYes_CheckedChanged

summary: Called when the user changes the Export options to yes

Parameters:

sender(System.Object):

e(System.EventArgs):

mnurdoMainOptionsExportNo_CheckedChanged

summary: Called when the user changes the Export options to no

Parameters:

sender(System.Object):

e(System.EventArgs):

mnutxtMainOptionsSeedValue_Validating

summary: Called when the seed value menu item is validating

Parameters:

sender(System.Object):

e(System.ComponentModel.CancelEventArgs):

OpenProject_Click

summary: User clicked on the Open Project menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveResults_Click

summary: User clicked on the Save Results menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

exitToolStripMenuItem_Click

summary: User clicked on the Exit menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

mnutxtMainOptionsSeedValue_Click

summary: User clicked on the Seed Value menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

aboutToolStripMenuItem_Click

summary: User clicked on the About menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SetupProgressBars

summary: Setup the progress bars

Parameters:

NumRuns(System.Int32):

NumRealizations(System.Int32):

ClearProgressBars

summary: Clear the progress bars

ShowProgress

summary: Show the progress

Parameters:

Description(System.String): The description to put in the text box

AnalysisCase(System.Int32): The intensity or scenario being updated

RealNum(System.Int32): The realization number completed

ShowProgress

summary: Show the progress

Parameters:

Description(System.String): The description to put in the text box

btnCalculate_Click

summary: When the user clicks the calculate (or cancel) button

Parameters:

sender(System.Object):

e(System.EventArgs):

Calculate

summary: Run a full calculation

SaveFile

summary: Save a results file

Parameters:

FilePathAndName(System.String):

SaveProjectToXMLFile

summary: Saves the project object (and sub objects) to a file

Parameters:

SaveFilePathAndName(System.String): The file path and name to save to

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

CurrentProject

summary: The current project

CurrentResults

summary: The current result set

ProgressBarList

summary: The progress bar list

txtProgressInfo

summary: The textbox showig progress information

InFormLoad

summary: Are we currently loading the form

Mainlog

summary: The logger object

RunningEngine

summary: Is the engine currently running

InBatchMode

summary: Are we running in automatic batch mode

components

summary: Required designer variable.

mnutxtMainOptionsSeedValue

summary: The seed value menu

Events

NewResultAvailable

summary: Event raised when a new result is available

PactNetShared Assembly

PopulationManager Assembly

frmPopulationManager

summary: The form for editing the population

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Form constructor

frmPopulationManager_Load

summary: The load event for the form

Parameters:

sender(System.Object):

e(System.EventArgs):

DefaultPopulationModels_ListChanged

summary: Handler for when the default population model list has changed

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

UpdateSaveStatus

summary: Update the save status of all populations

FillAllDefaults

summary: Fill all the default populations

dgvAllDefaultPopulationModels_RowEnter

summary: Handler for when you enter a row in the list of models

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

newToolStripMenuItem_Click

summary: The user clicked on the New menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

saveToolStripMenuItem_Click

summary: The user clicked on the Save menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

saveAsToolStripMenuItem_Click

summary: The user clicked on the Save As menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

exitToolStripMenuItem_Click

summary: The user clicked on the Exit menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

copyToolStripMenuItem_Click

summary: The user clicked on the Copy menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

pasteToolStripMenuItem_Click

summary: The user clicked on the Paste menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

cutToolStripMenuItem_Click

summary: The user clicked on the Cut menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

selectAllToolStripMenuItem_Click

summary: The user clicked on the Select All menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

pageSetupToolStripMenuItem_Click

summary: The user clicked on the Page Setup menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

printToolStripMenuItem_Click

summary: The user clicked on the Print menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

printPreviewToolStripMenuItem_Click

summary: The user clicked on the Print Preview menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

aboutToolStripMenuItem_Click

summary: The user clicked on the About menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

SaveModel

summary: Save a population model to its current location

Parameters:

ModelToSave(PactNet.Fragilities.clsPopulationModel):

SaveAsModel

summary: Save the a Population model, using the SaveFileDialog

frmPopulationManager_FormClosing

summary: Handler for when the form is closing

Parameters:

sender(System.Object):

e(System.Windows.Forms.FormClosingEventArgs):

Fields

components

summary: Required designer variable.

Mainlog

summary: The logger object

PopulationForm

summary: The population model sub-form

CurrentModel

summary: The current model we are looking at

FillingDGV

summary: Are we in the middle of filling the data grid views

frmPopulationModel

summary: A subform for the Population Modeler and Building Manager form to show a single population model

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Form constructor

ShowPopulationModel

summary: Show the preview of the fragilities in this template

Parameters:

CurrentPopulationModel(PactNet.Fragilities.clsPopulationModel): The current population model to show

SetupMonthGrid

summary: Setup the month grid

SetupHourGrid

summary: Setup the hour grid

rdoPopulation_CheckedChanged

summary: If the population radio button has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

DrawPopulationGraph

summary: Draw the population graph

Parameters:

TempPopModel(PactNet.Fragilities.clsPopulationModel):

FillPopulationGrids

summary: Fill all the population grids

Parameters:

TempModel(PactNet.Fragilities.clsPopulationModel):

ClearAllInfo

summary: Clear all the info on the form

ClearPopulationGrids

summary: clear all the population grids

ClearPopulationGraph

summary: Clear the population graphs

DrawPopulationGraphDay

summary: Draw the population graph for all hours in a day

Parameters:

TempModel(PactNet.Fragilities.clsPopulationModel):

GraphControl(Nevron.Chart.WinForm.NChartControl):

Weekend(System.Boolean):

DrawPopulationGraphYear

summary: Draw the population graph for all months and days in a year

Parameters:

TempModel(PactNet.Fragilities.clsPopulationModel):

GraphControl(Nevron.Chart.WinForm.NChartControl):

dgvPopMonth_CellValueChanged

summary: A cell value in the month datagridview has changed. Update the underlying object.

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

dgvPopHour_CellValueChanged

summary: A cell value in the hour datagridview has changed. Update the underlying object.

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

Fields

components

summary: Required designer variable.

Mainlog

summary: The logger

CurrentModel

summary: The current population model

GridFilling

summary: Are we currently a datagridview

ResultManager Assembly

frmResultsOneDSgroup

summary: A form for showing calculations for one damage state group

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Constructor

Parameters:

ResultDSGroup(PactNet.Results.clsResultDSGroup):

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates):

DescriptionString(System.String):

frmResultsOneDSgroup_Load

summary: Called when the form is loading

Parameters:

sender(System.Object):

e(System.EventArgs):

frmResultsOneDSgroup_Shown

summary: Called when the form is shown

Parameters:

sender(System.Object):

e(System.EventArgs):

ShowCalculations

summary: Show the actual calcualtion information

FillDSInfo

summary: Fill the damage state info

FillDSCalculations

summary: Fill the damage state calculations

dgvDSInfo_CellClick

summary: Called when the user clicks on the Damage state datagridview

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellEventArgs):

Fields

components

summary: Required designer variable.

frmResultsTimeBased

summary: The time based results form

Methods

Constructor

summary: Constructor

frmResultsTimeBased_Load

summary: Form load

Parameters:

sender(System.Object):

e(System.EventArgs):

SetResults

summary: A handle to the Results object for an external object

Parameters:

NewResults(PactNet.Results.clsResultSet):

btnRedraw_Click

summary: If the user clicks the redraw button, redraw all the charts

Parameters:

sender(System.Object):

e(System.EventArgs):

DrawAllCharts

summary: Draw all the charts

DrawCostChart

summary: Draw the 2D cost chart

DrawTimeChart

summary: Draw the 2D time chart

DrawDeathChart

summary: Draw the 2D death chart

DrawInjuryChart

summary: Draw the 2D injury chart

DisplayRedTag

summary: Display the red tag probability

DisplayCollapse

summary: Display the collapse probability

DisplayResidualDrift

summary: Display the collapse probability

chrtMainCharts_MouseDown

summary: If the user clicks on the legend, it will highlight certain colors

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

chrtTime_Click

summary: If the use clicks on the time chart, it will switch from serial to parallel

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoGraphType_CheckedChanged

summary: If the Graph type (line, stacked area, reversed area) has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

btnCostArea_Click

summary: Show the cost area

Parameters:

sender(System.Object):

e(System.EventArgs):

btnTimeArea_Click

summary: Show the time area

Parameters:

sender(System.Object):

e(System.EventArgs):

btnDeathArea_Click

summary: Show the deaths area

Parameters:

sender(System.Object):

e(System.EventArgs):

btnInjuryArea_Click

summary: Show the injury area

Parameters:

sender(System.Object):

e(System.EventArgs):

ShowAreaChart

summary: Show an area chart

Parameters:

ChartType(PactNet.Shared.Globals.ResultViewTypes):

rdoArea_CheckedChanged

summary: If the type of area chart (unadjusted vs MAFE-adjusted) has changed

Parameters:

sender(System.Object):

e(System.EventArgs):

exitToolStripMenuItem_Click

summary: Exit

Parameters:

sender(System.Object):

e(System.EventArgs):

pageSetupToolStripMenuItem_Click

summary: Show Page setup dialog

Parameters:

sender(System.Object):

e(System.EventArgs):

printToolStripMenuItem_Click

summary: Show print dialog

Parameters:

sender(System.Object):

e(System.EventArgs):

printPreviewToolStripMenuItem_Click

summary: Show print preview dialog

Parameters:

sender(System.Object):

e(System.EventArgs):

copyToolStripMenuItem_Click

summary: Copy menu item

Parameters:

sender(System.Object):

e(System.EventArgs):

mnuOptionsCurveLogNormal_CheckedChanged

summary: Check to see if the Base Curve mode has been changed to lognormal fit

Parameters:

sender(System.Object):

e(System.EventArgs):

mnuOptionsCurveHistogram_CheckedChanged

summary: Check to see if the Base Curve mode has been changed to histogram

Parameters:

sender(System.Object):

e(System.EventArgs):

toolStripTxtNumBins_Validating

summary: Update value if the number of desired bins has been changed

Parameters:

sender(System.Object):

e(System.ComponentModel.CancelEventArgs):

toolStripTxtNumBins_Validated

summary: If the number of desired bins has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

DrawIntegratedChartLine

summary: Draw the integrated chart 2D line graph

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control

ResultCurveGroup(PactNet.Results.clsResultSetViewCurve): The curve group to draw

ChartTitle(System.String): The title of the chart

XAxisTitle(System.String): The title of the X axis

YAxisTitle(System.String): The title of the Y Axis

DrawIntegratedChartArea

summary: Draw the integrated 2D stacked Area graph

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The control to display to

ResultCurveGroup(PactNet.Results.clsResultSetViewCurve): The result curve group to display

ChartTitle(System.String): The chart title

XAxisTitle(System.String): The X Axis title

YAxisTitle(System.String): The Y Axis title

ReverseIntensity(System.Boolean): Should we reverse the intensities on the chart

DrawAreaChart

summary: Draw the 3D area chart

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart to display to

AllCurves(PactNet.Results.clsResultSetViewCurve): The set of all the curves

ChartTitle(System.String): The chart title

YUnit(System.String): The unit value on the Y Axis

YTitle(System.String): The Y Axis title

MAFEAdjust(System.Boolean): Should we adjust all values for the MAFE

FillData

summary: Do the low level work for DrawAreaChart

Parameters:

surface(Nevron.Chart.NGridSurfaceSeries):

AllCurves(PactNet.Results.clsResultSetViewCurve):

MAFEAdjust(System.Boolean):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

CurrentResults

summary: The current result set

TimeUseSerial

summary: For the time charts, are we currently using serial or parallel

LastAreaButtonPushed

summary: The last Area chart viewed

MainIog

summary: The error logging object

InFormLoad

summary: Are we currently loading the form

components

summary: Required designer variable.

frmResultDrilldown

summary: Allow you to drilldown on all of the results

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Constructor

SetResultSet

summary: Set the result set object

Parameters:

AllResults(PactNet.Results.clsResultSet):

Fields

components

summary: Required designer variable.

frmResults1

summary: The result manager/viewer

VectorWrapper

summary: A class to wrap the Nevron vector point structure

remarks: I need to pass this around by reference instead of value, so I need this to be a class

Fields

InnerData

summary: The actual wrapper for the value

LineChartPaintCallback

summary: The custom paint event for the graphs, so I can draw the point and lines on the main graph when people click around

Methods

Constructor

summary: Constructor

Parameters:

parentUserControl(PactNet.ResultManager.frmResults1): The form

TheChartControl(Nevron.Chart.WinForm.NChartControl): The chart control

TheChartPoint(PactNet.ResultManager.frmResults1.VectorWrapper): The wrapper for a point object

CustomDraw

summary: Intercepts the on chart after paint event and converts point or XYZ coordniates to chart viewport coordinates used to position the custom drawing

Parameters:

panel(Nevron.Chart.NPanel):

eventArgs(Nevron.Chart.NPanelPaintEventArgs):

OnAfterPaint

summary: Called after the rest of the chart has been drawn

Parameters:

panel(Nevron.Chart.NPanel):

eventArgs(Nevron.Chart.NPanelPaintEventArgs):

Methods

Constructor

summary: Constructor

frmResults1_Load

summary: Form loading event

Parameters:

sender(System.Object):

e(System.EventArgs):

ResultsChanged

summary: The results object has been changed, redo everything

FillRecentResults

summary: Fill the recent results list

RecentResults_Click

summary: The user clicked an item in the recent results

Parameters:

sender(System.Object):

e(System.EventArgs):

mnurdoMainOptionsCursormodeCurve_CheckedChanged

summary: The Cursormode Curve option in the main menu has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

mnurdoMainOptionsCursormodeBuckets_CheckedChanged

summary: The Cursormode Buckets option in the main menu has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

openToolStripMenuItem_Click

summary: The Open menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

exitToolStripMenuItem_Click

summary: The Exit menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

copyToolStripMenuItem_Click

summary: The Copy menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

selectAllToolStripMenuItem_Click

summary: The Select All menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

pageSetupToolStripMenuItem_Click

summary: The Page Setup menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

printToolStripMenuItem_Click

summary: The Print menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

printPreviewToolStripMenuItem_Click

summary: The Print Preview menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

saveToolStripMenuItem_Click

summary: The Save menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoSerialOrParallel_CheckedChanged

summary: When they change from serial to parallel in the time based graph

Parameters:

sender(System.Object):

e(System.EventArgs):

aboutToolStripMenuItem_Click

summary: The About menu item has been clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

ShowProgress

summary: Show the progress

Parameters:

Description(System.String):

LoadingCallback

summary: Event handler when the loading event has progress updates

Parameters:

Sender(System.Object):

e(System.ComponentModel.ProgressChangedEventArgs):

CurrentResults_ProgressReport

summary: Show the progress of something

Parameters:

Description(System.String): Description to show

AnalysisCase(System.Int32): Unused

progress(System.Int32): The progress value

ShowCurrentResults

summary: Show the current results

btnCost_Click

summary: Called when the user clicks the realization cost button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnTime_Click

summary: Called when the user clicks the realization time button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnDeaths_Click

summary: Called when the user clicks the realization deaths button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnInjuries_Click

summary: Called when the user clicks the realization injuries button

Parameters:

sender(System.Object):

e(System.EventArgs):

cboAnalysis_SelectedIndexChanged

summary: When the user switches from one intensity/scenario to another

Parameters:

sender(System.Object):

e(System.EventArgs):

OnCostChartMouseClick

summary: Called when the user clicks on the bottom cost chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

OnTimeChartMouseClick

summary: Called when the user clicks on the bottom time chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

OnInjuryChartMouseClick

summary: Called when the user clicks on the bottom injury chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

OnDeathChartMouseClick

summary: Called when the user clicks on the bottom death chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

CostGraphClicked

summary: Find the point of the bottom cost chart the user clicked on, and update the top chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

TimeGraphClicked

summary: Find the point of the bottom time chart the user clicked on, and update the top chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

DeathGraphClicked

summary: Find the point of the bottom death chart the user clicked on, and update the top chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

InjuryGraphClicked

summary: Find the point of the bottom injury chart the user clicked on, and update the top chart

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

SetCursorPoint

summary: Set the cursor at the Xvalue and redraw the bar chart

Parameters:

XValue(System.Double): The X value, in current scale units (0 to use Y value)

YValue(System.Double): The Y value, in percent (0 to use X Value)

ViewType(PactNet.Shared.Globals.ResultViewTypes): The type of curve to show

ChangeGraphCursorLine(System.Boolean): Should we move the axis cursors, or just update the main chart with moving the cursor

GetSelectedAnalysis

summary: Get the currently selected analysis

txtCostPointAmount_KeyDown

summary: Called when the user changes the cost point textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtCostPercent_KeyDown

summary: Called when the user changes the cost percent textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtInjuriesPointAmount_KeyDown

summary: Called when the user changes the injury point textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtInjuriesPointPercent_KeyDown

summary: Called when the user changes the injury percent textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtDeathPointAmount_KeyDown

summary: Called when the user changes the deaths point textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtDeathPointPercent_KeyDown

summary: Called when the user changes the deaths percent textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtDowntimePointAmount_KeyDown

summary: Called when the user changes the time point textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

txtDowntimePointPercent_KeyDown

summary: Called when the user changes the time percent textbox

Parameters:

sender(System.Object):

e(System.Windows.Forms.KeyEventArgs):

mnuitemCopyToClip_Click

summary: Right click from charts

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoLinear_CheckedChanged

summary: The Linear (i.e. no breaks) radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

rdoLinearBreaks_CheckedChanged

summary: The Linear Breaks radio button has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

mnuGroupingLevel_Click

summary: When the Grouping Level has been changed

Parameters:

sender(System.Object):

e(System.EventArgs):

GetFragilityGroupingLevel

summary: Get the currently chosen fragility grouping level

DrawMainCharts

summary: Draw all the main charts

ShowLongLeads

summary: Show the list of long leads

Parameters:

LongLeadTable(System.Windows.Forms.DataGridView): The grid to draw long lead information onto

RunNum(System.Int32):

ClearRealizationChart
RedrawRealizationChart
ClearAllCharts

summary: Clear all the charts

DrawCostMain

summary: Draw the cost graphs

Parameters:

CostChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to draw onto

RunNum(System.Int32):

DrawCostLine

summary: Draw the fitted lognormal cost curve

Parameters:

CostChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to draw onto

CostResultCurve(PactNet.Results.clsResultAnalysisViewFittedCurve):

CurrentCostPoint(VectorWrapper@): The current tracking point on the curve

DrawTimeMain

summary: Draw the main time chart (Bins and lognormal curve)

Parameters:

TimeChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to draw onto

RunNum(System.Int32): The Run to draw

IsSerial(System.Boolean): Is it serial time (vs parallel time)

DrawTimeLine

summary: Draw the time lognormal curve

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to draw onto

TimeResultCurve(PactNet.Results.clsResultAnalysisViewFittedCurve):

CurrentTimePoint(VectorWrapper@):

DrawTimePGsNew

summary: Draw the time PG bar chart, this is very different than the other bars

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to draw to

RunNum(System.Int32): The scenario/intesnity number

XValue(System.Double): The XValue on the bottom chart we are showing

IsSerial(System.Boolean): Is this a serial (or parallel) chart

ShowStackedFloatBar

summary: Draw the horizontal stacked bar chart, used for time by floor

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

AllGroupOrPGNames(System.Collections.Generic.List{System.String}): All PG or group names

FloorList(System.Collections.Generic.List{System.Collections.Generic.Dictionary{System.String): A list of every floor, and each floor is a Dictionary containging fragilityIDs and their values

MobilizationTime(System.Double}}):

Notes(System.Double): The notes to put on the bottom of the chart

FloorNames(System.String): A list containing all the floor names

chrtTimeBar_MouseDown

summary: Highlight certain bars if they click on them in the legend

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

DrawDeathMain

summary: Draw the death graph data

Parameters:

DeathMainChart(Nevron.Chart.WinForm.NChartControl): The chart control to draw onto

RunNum(System.Int32):

DrawInjuryMain

summary: Draw the injury graph data

Parameters:

RunNum(Nevron.Chart.WinForm.NChartControl):

InjuryMainChart(System.Int32): The chart control to draw onto

DrawCurve

summary: Draw the lognormal fitted curve

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

ResultCurve(PactNet.Results.clsResultAnalysisViewFittedCurve):

PointIncrement(System.Double):

SetupGraphs

summary: Setup all the chart controls

SetupLineGraph

summary: Setup one of the (bottom) line graphs

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

DrawCurve

summary: Draw the lognormal fitted curve

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

ResultCurve(PactNet.Results.clsResultAnalysisViewFittedCurve):

SelectedPoint(VectorWrapper@):

PointIncrement(System.Double):

DrawHistogram

summary: Draw the cost Histogram

DrawPGBarsNew

summary: Common routine to draw the PG bars based on a lognormal fitted curve for all PGs

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control

CurveType(PactNet.Shared.Globals.ResultViewTypes): The type of chart we are drawing (cost, time, etc)

RunNum(System.Int32): The scenario/intensity to draw

XValue(System.Double): The XValue of the bottom chart that we are diplsaying on the top chart

chrtBars_MouseDown

summary: Highlight certain bars if they click on them in the legend

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

DrawAllRealizationsCost

summary: Draw the realizations cost chart

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to use

Runnum(System.Int32): The intensity/scenario to show

DrawAllRealizationsDeaths

summary: Draw the realizations deaths chart

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to use

Runnum(System.Int32): The intensity/scenario to show

DrawAllRealizationsInjuries

summary: Draw the realizations injuries chart

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to use

Runnum(System.Int32): The intensity/scenario to show

DrawAllRealizationsTime

summary: Draw the realizations time chart

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to use

Runnum(System.Int32): The intensity/scenario to show

IsSerial(System.Boolean): Serial, or parallel

DrawRealizationChart3

summary: Common routine for all realizations charts

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl): The chart control to use

PGList(System.Collections.Generic.IEnumerable{System.String}): The list of all unique performance groups

ListOfRealAndPG(System.Collections.Generic.List{PactNet.Results.clsRealizationWithPGInfo}): The list of each realization, each one broken down into PGs and cost totals

YAxisTitle(System.String): The Y Axis title

chrtRealizations_MouseDown

summary: Highlight colors when the user clicks on the legend

Parameters:

sender(System.Object):

e(System.Windows.Forms.MouseEventArgs):

ShowRedTags

Parameters:

RunNum(Nevron.Chart.WinForm.NChartControl):

DrawRedTagBar

summary: Draw the red tagging info

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

AllRedTagValues(System.Collections.Generic.List{System.Collections.Generic.KeyValuePair{System.String):

ClearRedTags

summary: Clear the Red Tags chart control

Parameters:

ChartControl(Nevron.Chart.WinForm.NChartControl):

ConfigureAxisCursors

summary: Configure the axis cursors on the line graphs

Parameters:

ChartObject(Nevron.Chart.NChart):

StartPoint(System.Double):

ConfigureRangeSelections

summary: Configure the range selections

Parameters:

ChartObject(Nevron.Chart.NCartesianChart):

spltInjuries_SplitterMoved

summary: The injury splitter has been moved

Parameters:

sender(System.Object):

e(System.Windows.Forms.SplitterEventArgs):

spltDeaths_SplitterMoved

summary: The deaths splitter has been moved

Parameters:

sender(System.Object):

e(System.Windows.Forms.SplitterEventArgs):

spltCasualties_ResizingBegin

summary: Called when the master splitter is resizing

Parameters:

sender(System.Object):

e(System.EventArgs):

spltCasualties_Resize

summary: Called when the master splitter has finished resizing

Parameters:

sender(System.Object):

e(System.EventArgs):

btnDrilldown_Click

summary: The user clicked the drill down button

Parameters:

sender(System.Object):

e(System.EventArgs):

SetResultSet

summary: Set the result set from outside the form

Parameters:

NewResultSet(PactNet.Results.clsResultSet):

btnTimeBased_Click

summary: Load the time based form

Parameters:

sender(System.Object):

e(System.EventArgs):

LoadProjectResults

summary: Load a project results file

Parameters:

FilePathAndName(System.String):

ReportCallBack(PactNet.Results.clsResultSet.IOEventHandler): The event handler to call as it is updating the loading status

SaveAsResults

summary: Save a project results file

SaveResults

summary: Save a project results file

Parameters:

FilePathAndName(System.String):

ExportSetup

summary: Does basic shared setup and checking for all the export commands

returns: True if success

btnExportEDPs_Click

summary: The user clicked the export EDPs button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportEDPs_MaxFiles

summary: Export the EDPs with the maximum number of files (One per Analysis/EDPType/Direction)

ExportEDPs_MinFiles

summary: Export the EDPs with the minimum number of files (One file)

ExportEDPs_Run

summary: Export the EDP info for just one run (/EDPType/Direction/Floor/Real)

Parameters:

NewStream(System.IO.StreamWriter): The output stream

TempAnalysis(PactNet.Results.clsResultAnalysis): The scenario/intensity

IncludeAllColumnHeaders(System.Boolean): Should we include all the column headers

ExportEDPs_EDPType

summary: Export the EDP info for just one EDPType (/Direction/Floor/Real)

Parameters:

NewStream(System.IO.StreamWriter):

TempAnalysis(PactNet.Results.clsResultAnalysis):

TempEDPType(PactNet.Fragilities.clsEDPType):

IncludeAllColumnHeaders(System.Boolean): Should we include all the column headers

ExportEDPs_Direction

summary: Export the EDP info for just one Direction (/Floor/Dir)

Parameters:

NewStream(System.IO.StreamWriter):

TempAnalysis(PactNet.Results.clsResultAnalysis):

TempEDPType(PactNet.Fragilities.clsEDPType):

DirNum(System.Int32):

IncludeAllColumnHeaders(System.Boolean): Should we include the column headers?

btnExportDS_Click

summary: Exports the Damage State information

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportDS_MaxFiles

summary: Exports the Damage State information with the maximum # of files (One file per Analysis/Dir/Floor)

ExportDS_MinFiles

summary: Exports the Damage State information with the minimum # of files (One file)

btnExportCosts_Click

summary: The user clicked the Export Costs button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportCosts_MaxFiles

summary: Exports the cost information with the max number of files (One per Analysis/Direction/Floor)

ExportCosts_MinFiles

summary: Exports the Costs with the minimum number of files (One per analysis)

btnExportTimes_Click

summary: The user clicked the Export Times button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportTime_MaxFiles

summary: Exports the time information with the max number of files (One per Analysis/Direction/Floor)

ExportTime_MinFiles

summary: Exports the time information in one file

btnExportDeaths_Click

summary: The user clicked the Export Deaths button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportDeaths_MaxFiles

summary: Exports the death information with the max number of files (One per Analysis/Direction/Floor)

ExportDeaths_MinFiles

summary: Exports the injury information in one file

btnExportInjuries_Click

summary: The user clicked the Export Injuries button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportInjuries_MaxFiles

summary: Exports the injury information with the max number of files (One per Analysis/Direction/Floor)

ExportInjuries_MinFiles

summary: Exports the injury information in one file

btnExportRedTags_Click

summary: The user clicked the Export Red Tags button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportRedTags_Summary

summary: Exports a red tag summary file

ExportRedTags_MaxFiles

summary: Exports the red tag information with the max number of files (One per Analysis/Direction/Floor)

ExportRedTags_MinFiles

summary: Exports the red tag info in one file

btnExportLongLeads_Click

summary: The user clicked the Export Long Leads button

Parameters:

sender(System.Object):

e(System.EventArgs):

ExportLongLeads_MaxFiles

summary: Exports the long lead information with the max number of files (One per Analysis/Direction/Floor)

ExportLongLeads_MinFiles

summary: Exports the long leads in one file

btnExportCollapse_Click

summary: Exports the collapse information (only "one file" option is available)

btnResidualDrift_Click

summary: Exports the Residual Drift information (only "one file" option is available)

GetSelectedExportFloors

summary: Gets the list of floors the user wants to export

GetSelectedExportDirs

summary: Gets the list of directions the user wants to export

GetSelectedFragilities

summary: Gets the list of fragilities the user wants to export

GetFullExportString

summary: Returns a string representing everything the user is exporting

Parameters:

SelectedFloors(System.Int32[]): The list of selected floors

SelectedDirs(System.Int32[]): The list of selected directions

SelectedFrags(System.String[]): The list of selected fragilities

IncludeSpaces(System.Boolean): Should we include space to make it easier read, or no spaces for file name

GetExportFloorString

summary: Returns a string representing all floors the user is exporting

Parameters:

SelectedFloors(System.Int32[]): The list of selected floors

IncludeSpaces(System.Boolean): Should we include space to make it easier read, or no spaces for file name

GetExportDirString

summary: Returns a string representing all directions the user is exporting

Parameters:

SelectedDirs(System.Int32[]): The list of selected directions

IncludeSpaces(System.Boolean): Should we include space to make it easier read, or no spaces for file name

GetExportFragsString

summary: Returns a string representing all fragilities the user is exporting

Parameters:

SelectedFragilties(System.String[]): The list of selected fragilities

IncludeSpaces(System.Boolean): Should we include space to make it easier read, or no spaces for file name

btnExportRealCosts_Click

summary: The user clicked the Export Realization Costs button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnExportRealTime_Click

summary: The user clicked the Export Realization Times button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnExportRealFatalities_Click

summary: The user clicked the Export Realization Fatalities button

Parameters:

sender(System.Object):

e(System.EventArgs):

btnExportRealInjuries_Click

summary: The user clicked the Export Realization Injuries button

Parameters:

sender(System.Object):

e(System.EventArgs):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

CurrentResults

summary: The current results

LastFileNameUsed

summary: The last filename used

CurrentCostView

summary: The current cost view

CurrentCostDrawPoint

summary: The current cost draw point

CurrentTimeView

summary: The current time view

CurrentTimeDrawPoint

summary: The current time draw point

CurrentDeathView

summary: The current deaths view

CurrentDeathDrawPoint

summary: The current deaths draw point

CurrentInjuryView

summary: The current injuries view

CurrentInjuryDrawPoint

summary: The current injury draw point

NewResultsScreen

summary: The results screen

Mainlog

summary: The logger

CurrentRealizationsGraphDisplay

summary: The current realization chart to draw

SplitterResizing

summary: Is the splitter resizing

components

summary: Required designer variable.

frmResultOneConsequence

summary: Show the consequence calculations

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Constructor

Parameters:

Consequence(PactNet.Results.clsResultConsequence): The current consequence

OriginalConsequence(PactNet.Fragilities.clsFragConsequenceGroup): The original consequence that corresponds

DescriptionString(System.String): The current description of where I am

FillAllInfo

summary: Fill all the information

FillContext

summary: Fill the context information

FillCostConsequences

summary: Fill the cost consequences

FillTimeConsequences

summary: Fill the time consequences

FillOtherConsequences

summary: Fill the misc consequences

frmResultOneConsequence_Shown

summary: When the form is shown

Parameters:

sender(System.Object):

e(System.EventArgs):

Fields

components

summary: Required designer variable.

CurrentConsequence

summary: The current consequence

CurrentDescriptionString

summary: The current description of where I am

CurrentOriginalConsequence

summary: The original consequence that corresponds

frmResultRedTag

summary: Show the red tag result details across a realizations

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Constructor

summary: Basic constructor

Parameters:

CurrentAnalysis(PactNet.Results.clsResultAnalysis): The current analysis we are viewing

RealizationNum(System.Int32): The current realization number

FillRealizationComboBox

summary: Fill the realization combobox with a list of all the realizations

FillAllInfo

summary: Fill in all of the info of a particular realization

cboRealization_SelectedIndexChanged

summary: Changing realizations

Parameters:

sender(System.Object):

e(System.EventArgs):

Fields

components

summary: Required designer variable.

Results Assembly

clsResultAnalysisViewHistogram

summary: A single result set of binned data

Methods

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The original array of cost values

DesiredCurveNumIncrement(System.Int32): The maximum number of bins desired

AdjustmentFactor(System.Double): An adjustment factor (unused)

ForcePowerOf10(System.Boolean): Force the bin value to be a power of 10

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The original array of cost values

NumIncrements(System.Int32): The exact number of bins to be used

IncrementValue(System.Double): The value of each bin

GetCurvePointValue

summary: Get a value of a certain bin

Parameters:

DataPointNum(System.Int32):

GetData

summary: Get the value of a certain bin

Parameters:

BinNum(System.Int32):

SetData

summary: Set the data value of a particular bin

Parameters:

BinNum(System.Int32):

value(System.Double):

GetXValues

summary: Get the X value of a particular bin (i.e., the cost value of the bin)

Parameters:

BinNum(System.Int32):

SetXValues

summary: Set the X value of a particular bin

Parameters:

BinNum(System.Int32):

value(System.Double):

GetCurvePercent

summary: Given a specified X value, return an interpolated Y Value (percent)

Parameters:

Value(System.Double):

GetCurveAmountValue

summary: Given a percent (Y value), return an interpolated value for X

Parameters:

Percent(System.Double): The percent to find

FindClosestXValue

summary: Find the closest X point to the value

returns: The index of the closest point

Parameters:

Value(System.Double):

FindPointBefore(System.Boolean): True to find the point before, false to find the point after

FindClosestYValue

summary: Find the closest point to a given Y Value on the curve

returns: The index of the closest value

Parameters:

Value(System.Double): The value to find

FindPointBefore(System.Boolean): True to find the point before, false to find the point after

GetLongestHorizontalLine

summary: Find the longest horizontal line...i.e. the longest area where I have no actual realizations in my bins

Parameters:

StartIndex(System.Int32@): Out: The start bin

EndIndex(System.Int32@): Out: The end bin

ReturnCumulative

summary: returns a cumulative version of the Histogram percentages object

Clone

summary: Create a deep clone of this object

ReturnPercentages

summary: Return a percentage version of itself

ReturnPercentages

summary: Takes a histogram of (binned) values and returns an array of (binned) percentages

returns: Return arrays (0 to number of Bins, 0 to 1 [data/xvalues])

Parameters:

SourceHistogram(PactNet.Results.clsResultAnalysisViewHistogram):

BinData2

summary: Read an array of source values and Bins

Parameters:

SourceArray(System.Double[]): The source array

Bins(System.Double[]): The Bins to use

BinData

summary: Returns an array of binned values (this figures out an optimal number of bins and values)

Parameters:

MainArray(System.Double[]):

NumBins(System.Int32): The exact number of Bins allowed

BinValue(System.Double): The bin value you want to force it to use

Adjustment(System.Double):

BinData

summary: Returns an array of binned values (this figures out an optimal number of bins and values)

Parameters:

MainArray(System.Double[]):

MaxBins(System.Int32): The maximum number of Bins allowed

ForceMaxValue(System.Double): The maximum number for the histogram, if 0 it will figure it out itself

LimitNumBins(System.Boolean): Limit bins to a power of 10

Adjustment(System.Double):

Properties

IsCumulative

summary: Is this Histogram cumulative (i.e. only rises)

IsPercentData

summary: Is this histogram percent data, (vs actual values)

NumCurveIncrements

summary: The number of bins

Fields

Values

summary: The actual binned values

clsResultAnalysisViewCurve

summary: A transformation into a curve for a single analysis case view Base class for the Log normal fitted curve and histogram

ViewCurveTypes

summary: Is the curve a lognormally fitted curve or a histogram

Fields

LogNormalFittedCurve

summary: A Log normal fitted curve

HistogramCurve

summary: A histogram curve

Methods

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The original array of realization costs

DesiredCurveNumIncrement(System.Int32): The maximum desired number of curve increments

AdjustmentFactor(System.Double): The adjustment factor (1 = none)

ForcePowerOf10(System.Boolean): Do we want to force the incerement to be a power of 10

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The original array of realization costs

NumIncrements(System.Int32): The exact number of curve increments

IncrementValue(System.Double): The value of each curve increment

GetCurvePercent

summary: Get the percent, given an actual value (given X, find Y)

Parameters:

CostValue(System.Double):

GetCurveAmountValue

summary: Get the cost or time, given a percentage (given Y, find X)

Parameters:

Percent(System.Double):

GetCurvePointValue

summary: Get the data at an actual data point

Parameters:

DataPointNum(System.Int32):

Clone

summary: Create a clone of this curve

Properties

TheFloors

summary: The list of floors used for this curve

TheDirections

summary: The list of direction used for this curve

ThePGs

summary: The list of PGs used for this curve

NumCurveIncrements

summary: The number of increments in this curve

CurveIncrement

summary: The increment of the curvearray

CurveMax

summary: The calculated max of the curve, based on the original data

Fields

XAxisTitle

summary: The XAxis Title

YAxisTitle

summary: The YAxis title

CurveTitle

summary: The curve title

AllDataZero

summary: Is all the data actually zero

_theFloors

summary: The list of floors used for this curve

_theDirections

summary: The list of direction used for this curve

_thePGs

summary: The list of PGs used for this curve

_curveIncrement

summary: The increment of the curvearray

_numCurveIncrements

summary: The number of increments in this curve

_curveMax

summary: The calculated max of the curve, based on the original data

clsResultDSGroupSimultCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Sequential Correlated Damage State Group

remarks: Parent: clsResultItemGroup or clsResultDS Children: clsResultDSSimultCor

Methods

Constructor

summary: Constructor for use by InputXML

Parameters:

DSGroupNode(System.Xml.Linq.XElement): The XML node containg info about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

DamageStates(PactNet.Fragilities.clsFragDamageStates): The original damage states object

NewEDP(System.Double): The EDP value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

CalculateDamageState

summary: Calculate the damage state(s) we are in

Parameters:

AllResultDamageStates(System.Collections.Generic.List{PactNet.Results.clsResultDS}):

CalculateFinalValues

summary: Calculate the final consequence values

OutputXML

summary: Create an XML element containing all the data for this object and any sub-objects

InputXML

summary: Input an XML data element and create the object and sub-objects

Parameters:

DSGroup(System.Xml.Linq.XElement):

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates):

Properties

PercentDamaged

summary: Get the fraction of items that are damaged

Fields

AllSDSRands

summary: A list of all the random numbers used to determine damage states

clsResultDSGroup

summary: Base class for one Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group

remarks: Parent: clsResultItemGroup or clsResultDS Children: clsResultDS Inherited by clsResultDSGroupMutExCor, clsResultDSGroupSeqCor, and clsResultDSGroupSimultCor

Methods

Constructor

summary: Constructor for use with InputXML

Parameters:

DSGroupNode(System.Xml.Linq.XElement): The XML node containing the info about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group object

Constructor

summary: Base constructor: Sets the EDP, the random num generation, and PG info

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

DamageStates(PactNet.Fragilities.clsFragDamageStates): The original damage states object

NewEDP(System.Double): The EDP value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

CalculateDamageState

summary: Do the calculations to determine which damage states we are in

Parameters:

AllResultDamageStates(System.Collections.Generic.List{PactNet.Results.clsResultDS}):

CalculateFinalValues

summary: Calculate the final consequence values

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

DSGroup(System.Xml.Linq.XElement): Input Node

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

DSRand

summary: The random number to determine which damage state we are in (for Seq or mutually exclusive)

EDP

summary: The EDP value (unused?)

ActualResultDamageState

summary: The list of actual damage states we are in

NumUnitsDamagedAcrossBuilding

summary: The number of actual units damaged across the entire building Must be set because it is used for the cost discount

FinalCostPerUnit

summary: The final cost per unit

FinalTimePerUnit

summary: The final downtime per unit

UnitDeathArea

summary: The number of square feet where everyone is dead

UnitInjuryArea

summary: The number of square feet where everyone is injured

PercentDamaged

summary: Get the fraction of items damaged

LongLeadFlag

summary: Determine if there is a long lead flag in effect

GetRedTagList

summary: Get the list of all damage states (and substates) that have a red tag state The list is in the format DSNUM/Sub-DSNUM/etc If no red tag, a blank list will be returned If Sequential or Mutex and red tagged, a one item list will be returned If Simult and red tagged, a multiple item list will be returned

GetSubDamageStateList

summary: Get a list of all ACTUAL subdamage states The list is in the format DSNUM/Sub-DSNUM/etc If Sequential or Mutex a one item list will be returned If Simult a multiple item list may be returned

Fields

_actualDamageStates

summary: The list of damage states we are actually in

_dSRand

summary: Random number to determine which DS we are in

RandomObject

summary: The random number generator

_numUnitsDamagedAcrossBuilding

summary: The number of units of this type damaged across the building

_EDP

summary: The calculated EDP value

clsResultAnalysis

summary: One intensity or scenario

remarks: Parent: ClsResultSet Children: clsResultReal, clsResultAnalysisEDPs

CalcCompletedEventHandler

summary: The delegate used to raise the CalcCompleted event

Parameters:

sender():

endTime():

RealCompletedEventHandler

summary: The delegate used to raise the RealCompleted event

Parameters:

sender():

RealEvent():

RealizationCompletedEvent

summary: The information to pass back when a realization has been completed

Fields

AnalysisNum

summary: The analysis number

RealizationNum

summary: The realization number

EndTime

summary: The ending time

Cancel

summary: Was this canceled

Methods

SetProject

summary: Set the project for this intesnity/scenario

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject):

CreateResultTree

summary: Create the whole tree structure of objects to represent every possible Realization/Floor/Dir/PG/DS

CreateResultTree

summary: Create the whole tree structure of objects to represent every possible Realization/Floor/Dir/PG/DS

remarks: I can pass data with the NotUsed object, but why should I?

Parameters:

NotUsed(System.Object): Something required by the thread pool

Calculate

summary: Calculate the final values of costs, time, etc for all realizations

Calculate

summary: Calculate the final values of costs, time, etc for all realizations

remarks: I can pass data with the NotUsed object, but why should I?

Parameters:

NotUsed(System.Object): Something required by the thread pool

ReturnView

summary: Return the curve (and raw data) representing every single realization in this run for a specific data type

Parameters:

AnalysisViewType(PactNet.Shared.Globals.ResultViewTypes): The type of data being looked for (cost, time, deaths, etc...)

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

ReturnView

summary: Return the curve (and raw data) representing every single realization in this run for a specific data type

Parameters:

AnalysisType(PactNet.Shared.Globals.ResultViewTypes): The type of data being looked for (cost, time, deaths, etc...)

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

IsSerial(System.Boolean): For a time curve ONLY, is the data parallel or serial across floors

ReturnCostView

summary: Return the cost curve (and raw data) representing every single realization in this run

Parameters:

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

ReturnDeathView

summary: Return the death view (raw data, fitted curves, etc) representing every single realization in this run

Parameters:

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

ReturnInjuryView

summary: Return the injury view (raw data, fitted curves, etc) representing every single realization in this run

Parameters:

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

ReturnTimeView

summary: Return the time view (raw data, fitted curves, etc) representing every single realization in this run

Parameters:

TheFloors(System.Int32[]): The list of floors to look for, or a blank array for all floors

TheDirections(System.Int32[]): The list of directions to look for, or a blank array for all directions

ThePGs(System.String[]): The list of PGs to look for, or a blank array for all PGs

IsSerial(System.Boolean): Return data serially or parellel across floors

GetAllDistinctFragilities

summary: Get a list of all distinct fragilities used for this analysis run (Uses a cached object if run a second time)

GetLongLeadItems

summary: Get a list of all the long lead items for all realizations, and how many realizations for each

GetRedTagPercent

summary: Get the percentage of realizations that have a red tag

GetCollapsePercent

summary: Get the percentage of realizations that have a collapse

GetResidualDriftPercent

summary: Get the percentage of realizations that have total loss due to residual drift

GetAllRealizationsCost

summary: Get a list of realizations in order of totalcost, with each one having a list of PGs and their totalcost

Parameters:

AllPGs(System.Collections.Generic.IEnumerable{System.String}):

GetAllRealizationsByGroup

summary: Get a list of realizations in order of totalcost, with each one having a list of PGs (or groups) and their totalcost

Parameters:

FragilityLevel(System.Int32): The fragility level to deivide the groups into

ResultType(PactNet.Shared.Globals.ResultViewTypes): The result type to compute

RemovePGsOrGroupsWithNoValues
GetAllDistinctFragilitiesOrGroups

summary: Get a list of all distinct fragilities, or groups, used for this analysis run

GetAllRealizationsDeaths

summary: Get a list of realizations in order of totalcost, with each one having a list of PGs and their totalcost

Parameters:

AllPGs(System.Collections.Generic.IEnumerable{System.String}): A list of all performance groups you want to include

GetAllRealizationsInjuries

summary: Get a list of realizations in order of injuries, with each one having a list of PGs and their total injury rate

Parameters:

AllPGs(System.Collections.Generic.IEnumerable{System.String}):

FindClosestRealizations

summary: Get the list of realizations that bracket a target value

Parameters:

AnalysisType(PactNet.Shared.Globals.ResultViewTypes): The analysis type to look for (cost, time, etc...)

TargetCost(System.Double): The target cost

FindClosestTimeRealizationNums

summary: Get the list of realizations that bracket a target value

Parameters:

TargetTime(System.Double): The target time to look for

IsSerial(System.Boolean): Look for serial or parallel

GetBracketedTimeValue

summary: Get a list of floors, and all the fragilities and time values, given a target total time. Returns a list of floors, each containing a list of fragilities and the total time for each

returns: A list of floors, each containing a list of fragilities and the total time for each

Parameters:

TargetTime(System.Double): The total time value to find the closest realizations for

IsSerial(System.Boolean): Are we expecting serial or parallel information

Notes(System.String@): Output: Text description of the realizations used

GetInterpolatedValues

summary: Given a certain XValue (cost, deaths, etc), return a weighted average result of the two realizations that bracket it

Parameters:

AnalysisType(PactNet.Shared.Globals.ResultViewTypes): Cost, Deaths, Injuries (Not Time)

TargetTotalValue(System.Double):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

Analysis(System.Xml.Linq.XElement): Input Node

OriginalProject(PactNet.BuildingProject.clsProject): The original project

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

RunNum

summary: Get run number of this analysis case

RunName

summary: Get the name of this analysis case

Realizations

summary: Get the list of all realizations

Fields

MasterProject

summary: The Master project file

AbortingViaThreadPool

summary: Should we abort the evaluation

_analysisNum

summary: The analysis number

_analysisName

summary: The name of the analysis (unused?)

_realizations

summary: The list of realizations

RandomObject

summary: The random number generator

CurrentReal

summary: The current realization being evaluated

AnalysisEDPInfo

summary: The calculated EDP value

NumRealizations

summary: The number of realizations

NumFloors

summary: The number of floors of this building

IsSimplifiedAnalysis

summary: Is this a simplified analysis

NonLinearAlgorithm

summary: The non-linear algorithm used (if applicable)

AnalysisException

summary: If there was an error running an analysis, store it here

Events

CalcCompleted

summary: Event raised when all calculations have been completed

RealCompleted

summary: Event raised when a single realization has been calculated

clsRealizationWithPGInfo

summary: A utility class to hold a realization, along with a total for that realization

Methods

Compare1

summary: Compare one clsRealizationWithPGInfo with another, using total

returns: 1 if first is bigger, -1 if second is bigger, 0 if same

Parameters:

x(PactNet.Results.clsRealizationWithPGInfo): First object

y(PactNet.Results.clsRealizationWithPGInfo): second Object

CompareByPG

summary: Assuming the totals are equal, compare them by the fragilityID with the highest value

Parameters:

x(PactNet.Results.clsRealizationWithPGInfo):

y(PactNet.Results.clsRealizationWithPGInfo):

System#Collections#Generic#IComparer{PactNet#Results#clsRealizationWithPGInfo}#Compare

summary: Compare one clsRealizationWithPGInfo with another, using total

returns: 1 if first is bigger, -1 if second is bigger, 0 if same

Parameters:

x(PactNet.Results.clsRealizationWithPGInfo): First object

y(PactNet.Results.clsRealizationWithPGInfo): second Object

Compare

summary: Compare one clsRealizationWithPGInfo with another, using total

returns: 1 if first is bigger, -1 if second is bigger, 0 if same

Parameters:

x(System.Object): First object

y(System.Object): second Object

CompareTo

summary: Compare this clsRealizationWithPGInfo with another

returns: 1 if this is bigger, -1 if the other is bigger, 0 if same

Parameters:

other(PactNet.Results.clsRealizationWithPGInfo): Other object

Fields

PGList

summary: The list of performance groups, and the total

RealizationNum

summary: The realization number

Total

summary: The total for the whole realization

Notes

summary: Any notes

clsResultItemGroup

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/item group

remarks: Parent: clsResultPG Children: clsResultDSGroup

Methods

Constructor

summary: Constructor

Parameters:

ItemGroupNode(System.Xml.Linq.XElement): The XML node containing the info about this object

OriginalPG(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

TheItemGroupNum(System.Int32): The item group number

NewEDP(System.Double): The EDP Value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

Quantity(System.Int32): The number of units in this item group

Calculate

summary: Calculate the final consequences

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

ItemGroupNode(System.Xml.Linq.XElement): Input Node

OriginalPG(PactNet.BuildingProject.clsPerformanceGroup): The original performance group

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

DamageGroup

summary: Get the Damage group results

ItemGroupNum

summary: The item group number

NumUnits

summary: The number of units in this item group

EDP

summary: The EDP value

NumUnitsDamaged

summary: The number of units actually damaged

PercentDamaged

summary: The fraction of items that have been damaged

NumUnitsDamagedAcrossBuilding

summary: The number of units damaged across the entire building

TotalCost

summary: The total cost for this item group

TotalTime

summary: The total downtime for this item group

AllDeathArea

summary: The number of square feet where everyone is dead

AllInjuryArea

summary: The number of square feet where everyone is injured

LongLeadFlag

summary: Is there a long lead flag set by any actual damages states?

GetRedTagList

summary: Get a dictionary of all damage states (and substates) that have a red tag state, and how many. There will be one item per unit/DS that is in a red tag state. The list is in the format DSNUM/Sub-DSNUM/etc. If no red tag, a blank list will be returned.

remarks: This could be deceptive, if there are 20 units, and each is a simultaneous fragility with all of the sub-damage states having red-tag consequences and each unit is in 2 damage states, it will count 40 items

Fields

_numUnitsDamagedAcrossBuilding

summary: The number of units damaged across the entire building

_itemGroupNum

summary: The item group number

_numUnits

summary: The number of units in this item group

RandomObject

summary: The random number generator

_damageStateGroup

summary: The damage state group results

clsResultConsequence

summary: One Result consequence

remarks: Parent: clsResultDS Children: None. It referenced clsFragConsequenceGroup

Methods

Constructor

summary: Constructor only to be used by InputXML

Parameters:

ConsequenceNode(System.Xml.Linq.XElement): The XML node element used to create this object

OriginalConsequence(PactNet.Fragilities.clsFragConsequenceGroup): The original consequence object

Constructor

summary: Constructor

Parameters:

ConsequenceGroup(PactNet.Fragilities.clsFragConsequenceGroup): The original consequence group

NewRandomObject(PactNet.Results.SSRandom): The random number generator

Calculate

summary: Calculate the total consequences

CalculateCost

summary: Calculate the cost

CalculateTime

summary: Calculate the time

InputXML

summary: Input an XML node and the original consequence object and create the object and all sub-objects from it Intended to be smaller

Parameters:

Consequence(System.Xml.Linq.XElement): Input Node

OriginalConsequence(PactNet.Fragilities.clsFragConsequenceGroup): The original consequence object

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

DCRand

summary: The random number used to determine cost uncertainty

DTRand

summary: The random number used to determine downtime uncertainty

MeanCostPerUnit

summary: The mean cost per unit

MedianCostPerUnit

summary: The median cost per unit

MeanTimePerUnit

summary: The mean time per unit

MedianTimePerUnit

summary: The median time per unit

CostDispersion

summary: The cost dispersion

CostLowerQuan

summary: The quantity of items needed to start getting cost discounts

CostUpperQuan

summary: The quantity of items at which you no longer additional cost discounts

CostMax

summary: The maximum cost per unit (No cost discounts)

CostMin

summary: The minimum cost per unit (maximum cost discounts)

TimeDispersion

summary: The time dispersion

TimeLowerQuan

summary: The quantity of items needed to start getting downtime discounts

TimeUpperQuan

summary: The quantity of items at which you no longer additional time discounts

TimeMax

summary: The maximum downtime per unit (No time discounts)

TimeMin

summary: The minimum time per unit (maximum time discounts)

CostCurveType

summary: Is the cost discount curve normal or lognormal

TimeCurveType

summary: Is the time discount curve normal or lognormal

NumUnitsDamagedAcrossBuilding

summary: The number of units damaged across the entire building

FinalCostPerUnit

summary: The final cost per unit

FinalTimePerUnit

summary: The final time per unit

UnitDeathArea

summary: The number of square feet where everyone is dead

UnitInjuryArea

summary: The number of square feet where everyone is injured

IsRedTaggable

summary: Does this consequence indicate a possible red tag

LongLeadFlag

summary: Does this consequence indicate a long lead

clsResultRealDS

summary: Every unique damage state used in this fragility for this realization, used for red tagging

remarks: Parent: clsResultRealFrag Children: None

Methods

Constructor

summary: Constructor

Parameters:

RealDSNode(System.Xml.Linq.XElement): The XML element node used to create this object

Constructor

summary: Constructor

Parameters:

TheConsequenceGroup(PactNet.Fragilities.clsFragConsequenceGroup): The original consequence group

NewRandomObject(PactNet.Results.SSRandom): The random number generator

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

RealDamageStateNode(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

GetFragID

summary: Get the fragility ID part of the Damage State

GetTopLevelDS

summary: Gets the top level damage state

Fields

PathName

summary: The class path to the damage state, the format is fragiltyid|dsnum/subsdnum etc...

QuantityInThisDamageState

summary: The quantity of items in this damage state

RedTaggablePercentNeeded

summary: The fraction of items required to mark this item as red tagged

RedTaggablePercentRand

summary: The random number generated to calculate the RedTaggablePercentNeeded

clsResultSetViewHistogram

summary: A class that holds all the fitted curves view, and the adjusted total, for all analysis cases

Methods

Constructor

summary: Constructor

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

AllAnalysisViews(System.Collections.Generic.List{PactNet.Results.clsResultAnalysisView}): The list of individual analysis results

NumBinsDesired(System.Int32): The maximum number of bins desired

clsResultSetViewCurve

summary: The collection of all results in a set...i.e. all analysis runs Inherited by clsResultSetViewFittedCurve and clsResultSetViewHistogram

Methods

FillHazardInfo

summary: Fill the hazard curve information

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

DetermineMaxOfAllCurves

summary: Get a maximum value for all the curves

Parameters:

OriginalViews(System.Collections.Generic.List{PactNet.Results.clsResultAnalysisView}):

GetReasonableXMaxPointForAllCurves

summary: Get a resonable max x point for all the curves

Parameters:

UseAdjustedCurves(System.Boolean):

GetReasonableYOfAllCurves

summary: Try and determine a reasonable minimum y value for the curves...will be 1% of the max

GetMaxYOfAnyCurve

summary: Try and get a reasonable max y value of the curves

Parameters:

UseAdjustedCurves(System.Boolean):

CalculateGrandTotal

summary: Calculate a single grand total for this curve

CalculateIntensityAdjustmentTotals

summary: Calculate the intesnity adjustments based on the MAFE values

GetCurvePointValue

summary: Get a curve point value at a particular data point

Parameters:

DataPointNum(System.Int32):

Properties

IMs

summary: The IMs for the intensities or scenarios

AllCurvesArray

summary: The 2D array of fully adjusted curves, [RunNum,PointNum]

AllAnalysisViewCurves

summary: All the original curves for all analysis

IntensityAdjustmentTotals

summary: The adjustments for intensity based on the IMs

NumCurveIncrements

summary: The number of increments

CurveMax

summary: The max value for the curve

ReasonableTotalXMaxPoint

summary: Try and determine a reasonable maximum x point for the curve

ReasonableTotalXMaxValue

summary: Try and determine a reasonable maximum x value for the curves

ReasonableTotalYMin

summary: Try and determine a reasonable minimum y value for the curves...will be 1% of the max

CurveIncrement

summary: The curve increment value

MaxTotalYValue

summary: The maximum y value of any curves

Fields

_curveArray

summary: The final curve array

_curveIncrement

summary: The increment value for each point

_curveMax

summary: The max value for the curve

_curveMin

summary: The min value for the curve

_numCurveIncrements

summary: The number of increments

_maxTotalYValue

summary: The maximum value for the "total", i.e. _curveArray

_allAnalysisViewCurves

summary: The original curves

_allCurvesArray

summary: The collection of all the fully-adjusted individual curves [runnum,point num]

_intensityAdjustmentTotals

summary: The adjustments for intensity based on the IMs

NumRuns

summary: The number of intensities or scenarios

_iMs

summary: The IMs for the intensities or scenarios

clsResultDSGroupSeqCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Sequential Correlated Damage State Group

remarks: Parent: clsResultItemGroup or clsResultDS Children: clsResultDSSeqCor

Methods

Constructor

summary: Constructor for use with InputXML

Parameters:

DSGroupNode(System.Xml.Linq.XElement): The XML node containing the info about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group object

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

DamageStates(PactNet.Fragilities.clsFragDamageStates): The original damage states object

NewEDP(System.Double): The EDP value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

CalculateDamageState

summary: Calculate which damage state we are in

Parameters:

AllResultDamageStates(System.Collections.Generic.List{PactNet.Results.clsResultDS}): The list of all possible damage states

CalculateFinalValues

summary: Calculate the final consequence values

Properties

PercentDamaged

summary: Calculate the fraction of items damaged

clsResultAnalysisViewFittedCurve

summary: A log normally fitted curve for a single analysis case and a specified set of floors, dirs, and PGs

Methods

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The array of calculated realization values

DesiredCurveNumIncrement(System.Int32): The maximum number of increments

AdjustmentFactor(System.Double): The adjustment factor if any (unused?)

ForcePowerOf10(System.Boolean): Force the increment value to be a power of 10

Constructor

summary: Constructor

Parameters:

MainArray(System.Double[]): The array of calculated realization values

NumIncrements(System.Int32): The exact number of incerements

IncrementValue(System.Double): The value of each incerement

DetermineMean

summary: Determine the mean and stddev of the input array

Parameters:

MainArray(System.Double[]):

Mean(System.Double@):

StdDev(System.Double@):

FillCurveArray

summary: Fill the curve array with values derived from the lognormal curve

Parameters:

NumIncrements(System.Int32):

TheCurveIncrement(System.Double):

CurveDistributions(MathNet.Numerics.Distributions.LognormalDistribution):

DetermineMaxValues

summary: Get a rounded up maximum value for this curve for graphing

Parameters:

ActualMaxValue(System.Double): The actual max value of the underlying data

AdjustmentFactor(System.Double): The adjustment factor, if any

NumIncrements(System.Int32): The maximum desired number of increments

NewMax(System.Double@): Output: The new maximum value calculated

NewIncrement(System.Double@): Output: The increment value calculated

ForcePowerOf10(System.Boolean): Do we want to force the increment value to be a power of 10

Create

summary: Create the curvearray with a set number of increments and increment values

Parameters:

MainArray(System.Double[]):

NumIncrements(System.Int32):

IncrementValue(System.Double):

MarkAllDataAsZero

summary: Mark this curve as being all zeros

Create

summary: Determine all values and create the curvearray

Parameters:

MainArray(System.Double[]):

NumIncrements(System.Int32):

AdjustmentFactor(System.Double):

ForcePowerOf10(System.Boolean): Should we force the increment value to be a power of 10

GetCurvePointValue

summary: Get the data at an actual data point

Parameters:

DataPointNum(System.Int32):

GetCurvePercent

summary: Get the percent, given an actual value (given X, find Y)

Parameters:

CostValue(System.Double):

GetCurveAmountValue

summary: Get the cost or time, given a fraction probability (given Y, find X)

Parameters:

Probability(System.Double): The probability of occurance

FitCurveToValues

summary: Tries to fit a lognormal curve onto an array of values, and returns the median of the population logs and the stddev of the population logs(dispersion)

Parameters:

MainArray(System.Double[]): An array of values

ReturnedLogMedian(System.Double@): Output: The median of the population logs

ReturnedLogStdDev(System.Double@): Output: The stddev of the population logs

FitCurveToValues2

summary: Alternative curve fit from Craig...not as good it seems

Parameters:

MainArray(System.Double[]): An array of values

ReturnedLogMedian(System.Double@): Output: The median of the population logs

ReturnedLogStdDev(System.Double@): Output: The stddev of the population logs

Properties

LogNorm

summary: The LognormalDistribution object that describes this curve

Mean

summary: The calculated mean of this curve

StdDev

summary: The calculated standard deviation of this curve

Fields

_curveArray

summary: An array of lognormal fitted curve values

_logNorm

summary: The lognormal object that has been fitted

clsResultAnalysisEDPCommon

summary: The base class for a collection of all Cholesky or Frazin calculated EDP values for this analysis case

remarks: Parent: clsResultAnalysis Children: clsResultDirectionEDPCommon

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The scenario or intensity number

RandObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

CalculateValues

summary: Calculate all values from the project

GetEDPValue

summary: Get a single EDP value

returns: The EDP Value

Parameters:

DirNum(System.Int32): The direction number

ValueNum(System.Int32): The value (floor) of the EDP value

EDPGroupName(System.String): The EDP Group name

Realization(System.Int32): The realization number

GetRealizationEDPs

summary: Get all EDPs for a realization, grouped by EDP group name

returns: A dictionary containing EDPGroup names and an EDP collection for each

Parameters:

RealizationNum(System.Int32): The realization number

MasterProject(PactNet.BuildingProject.clsProject): The original project

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Fields

AllDirections

summary: The list of all the EDPs for the individual directions

clsResultAnalysisEDPs

summary: The base class for a collection of all calculated EDP values for this analysis case

remarks: Parent: clsResultAnalysis Inherited by:

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The scenario or intensity number

NewRandomObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

FillAllEDPTypesUsed

summary: Get a list of all the EDP types used for this analysis

GetEDPValue

summary: Get a single EDP value

returns: The EDP Value

Parameters:

DirNum(System.Int32): The direction number

ValueNum(System.Int32): The value (floor) of the EDP value

EDPGroupName(System.String): The EDP Group name

Realization(System.Int32): The realization number

GetRealizationEDPs

summary: Get all EDPs for a realization, grouped by EDP group name

returns: A dictionary containing EDPGroup names and an EDP collection for each

Parameters:

RealizationNum(System.Int32): The realization number

MasterProject(PactNet.BuildingProject.clsProject): The master project

InputXML

summary: Fill this object with info from an XML node

Parameters:

ThisNode(System.Xml.Linq.XElement):

OutputXML

summary: Create an XML element with the information about this object

Fields

AllEDPTypesAndNames

summary: A list of all EDP Group Names and their underlying EDP types

AnalysisNum

summary: The scenario or intensity number

RandomObject

summary: The random number generator

clsResultSetView

summary: A class that holds the result view for all analysis cases

Methods

Constructor

summary: Constructor

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project

OriginalAnalysisViews(System.Collections.Generic.List{PactNet.Results.clsResultAnalysisView}): The list of scenario or intesity views

Constructor

summary: Constructor

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project

OriginalAnalysisViews(PactNet.Results.clsResultAnalysisView[]): One or more scenario or intesity views

GetFittedCurves

summary: Get a FittedCurve view for this set

Parameters:

NumBinsDesired(System.Int32): The maximum number of bins desired

GetHistogram

summary: Get a histogram view for this set

Parameters:

NumBinsDesired(System.Int32): The maximum number of bins desired

Properties

AllAnalysisViews

summary: The original curves

NumRuns

summary: The number of intesnities or scenarios

clsResultDS

summary: A MustInherit Base class One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Damage State

remarks: Parent: clsResultDSGroup Children: clsResultDSGroup or clsResultConsequence Inherited by clsResultDSSeqCor, clsResultDSSimCor, clsResultDSMutexCor,

Methods

Constructor

summary: Constructor

Parameters:

DSNode(System.Xml.Linq.XElement): The XML node containing information about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

Constructor

summary: Base Constructor, Sets the EDP and Random seed. Creates the Substates or consequence groups as well

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original PG object

DamageState(PactNet.Fragilities.clsFragDamageState): The original damage state object

NewEDP(System.Double): The EDP value for this item

NewRandomObject(PactNet.Results.SSRandom): The random number generator

InputXML

summary: Input an xml node and fill this object with the information

Parameters:

DamageStateNode(System.Xml.Linq.XElement): The xml node

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group object

OutputXML

summary: Output this object (and sub-objects) as an xml object

CalculateDamageState

summary: Calculate the damage state

CalculateFinalValues

summary: Do the actual calculations

Properties

DSNum

summary: The damage state number

SDSRand

summary: SDSRand is only used by the simultaneous damage states

HasSubstates

summary: Does this damage state has sub damage states

HasConsequence

summary: Does this damage state have consequences

Consequence

summary: The result consequence object, if available

SubState

summary: The substate object, if available

EDP

summary: The EDP value

FinalCostPerUnit

summary: The final cost per unit stub

FinalTimePerUnit

summary: The final time per unit stub

NumUnitsDamagedAcrossBuilding

summary: The number of units of this type damaged across the building Must be set, required for determing cost discount

LongLeadFlag

summary: Does this object have a long lead flag

UnitDeathArea

summary: The number of square feet where everyone is dead

UnitInjuryArea

summary: The number of square feet where everyone is injured

GetRedTagList

summary: Get a list of all actual consequences (or subdamage states) that show a red tag event The list is in the format DSNUM/Sub-DSNUM/etc If no red tag, a blank list will be returned If Sequential or Mutex and red tagged, a one item list will be returned If Simult and red tagged, a multiple item list will be returned

GetSubDamageStateList

summary: Get a list of all ACTUAL subdamage states The list is in the format DSNUM/Sub-DSNUM/etc If Sequential or Mutex a one item list will be returned If Simult a multiple item list may be returned

Fields

_EDP

summary: The EDP value

_numUnitsDamagedAcrossBuilding

summary: The number of units damaged across the entire building

_dSNum

summary: The damage state number

_SDSRand

summary: The random number used to determine if we are in this damage state (Unused?)

_consequence

summary: The consequences, if applicable

_subStates

summary: The sub damage state group, if applicable

RandomObject

summary: The random number generator

DSDescription

summary: The description of this damage state (Unused?)

clsResultAnalysisEDPCholesky

summary: The class for a collection of all Cholesky calculated EDP values for this analysis case NOT USED as of 9-2012

remarks: Base class: clsResultAnalysisEDPCommon Parent: clsResultAnalysis Children: clsResultDirectionEDPCholesky

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The scenario or intensity number

RandObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

CalculateCholeskyValues

summary: Calculate all Cholesky values from the project

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

CholeskyAnalysis(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

clsResultRealFrag

summary: Every single unique fragility used in this realization Needed to help with RedTags and quantity discounts

remarks: Parent: clsResultReal Children: clsresultRealDS

Methods

Constructor

summary: Constructor

Parameters:

RealFragilityNode(System.Xml.Linq.XElement): The XML node containing info about this object

Constructor

summary: Constructor

Parameters:

OriginalCurve(PactNet.Fragilities.clsFragilityCurve): The original fragility curve

NewRandomObject(PactNet.Results.SSRandom): The random number generator

IncrementDamageState

summary: For red Tagging, get a damage state path and increment the appropriate ResultRealDS

Parameters:

FragID(System.String): The fragility ID

DSPath(System.String): The DS path, in the format of DSNUM/Sub-DSNUM/etc

NumToIncrement(System.Int32): The number of items to increment

GetAllStates

summary: Find all the possible damage states (and substates) of a fragility curve

Parameters:

OriginalCurve(PactNet.Fragilities.clsFragilityCurve):

GetSubStates

summary: Find all the possible sub damage states (and sub-substates) of a damage state

Parameters:

ParentState(PactNet.Fragilities.clsFragDamageState):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

RealFragility(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

FragDamageTotals

summary: The total quantity of items of this fragility type that are in any damage state

Totals

summary: The total quantity of items of this fragility type that are in the building

FragilityID

summary: The fragility ID

IsRedTagged

summary: Is this realization Red tagged

Fields

_fragDamageTotals

summary: The total number damaged

_totals

summary: The total quantity of items of this fragility type that are in the building

_fragilityID

summary: The fragility ID

AllDamageStates

summary: A list of all possible damage states

RandomObject

summary: The random number generator

clsResultDirectionEDPCholesky

summary: All the generated Cholesky EDP values for a particular analysis and direction NOT USED as of 9-2012

remarks: Base class: clsResultDirectionEDPCommon Parent: clsResultAnalysisEDPCholesky Children: clsEDPType, clsEDPArray (CombinedMatrixKeys)

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The project this is based on

TheAnalysisNum(System.Int32): The analysis number

TheDirectionNum(System.Int32): The direction

TheEDPTypesAndNames(System.Collections.Generic.Dictionary{System.String): The list of EDP types that need to be calculated

NewRandomObject(PactNet.Fragilities.clsEDPType}): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

ThisNode(System.Xml.Linq.XElement): Input XML Node

FindMeansOfMatrix

summary: Returns a matrix of means of the source matrix

Parameters:

SourceMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

FindStdDevsOfMatrix

summary: Returns a matrix of stand deviations of the source matrix

Parameters:

SourceMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

MeanMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

GetCorrelationCoefficientMatrix

summary: Returns the correlation coefficient matrix

Parameters:

SourceMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

MeanMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

GetFinalMatrix

summary: Takes all the matrixes, does a bunch of stuff I don't really have a clue about, and creates a final matrix

Parameters:

MeanMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

StdDevMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

NormRNMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

CholeskyMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

GetYMatrix

summary: Returns the "Y" matrix, which I have no idea what is

Parameters:

StdDevMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

NormRNMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

CholeskyMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

ResultDirectionEDP(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Fields

EDPILog

summary: The original matrix, log of all values

Means

summary: The means of all the (log) original values

StdDevs

summary: The StdDevs of all the values

CorrCoef

summary: The coorelation coefficient matrix

CholeskyResults

summary: The Cholesky Results

clsResultDirectionEDPCommon

summary: All the generated values for all the EDPs for a particular Analysis and Direction The base class for clsResultDirectionEDPCholesky and clsResultDirectionEDPEigenDecomp

remarks: Parent: clsResultAnalysisEDPCommon Children: clsEDPType, clsEDPArray (CombinedMatrixKeys)

CombinedMatrixKeyList

summary: A class to help keep track of the mapping between the PGs and the combined matrix for a particular realization/direction

Methods

GetRow

summary: Get the row number for a particular PG

returns: The row number in the combined matrix

Parameters:

GroupName(System.String): The group name (Drift, Acceleration, etc) for the PG

FloorNum(System.Int32): The floor number

AddRow

summary: Add a row to the combined matrix

Parameters:

Row(System.Int32): The row number in the combined matrix

GroupName(System.String): The PG group name (Drift, Acceleration, etc)

FloorNum(System.Int32): The floor number the PG is on

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

CombinedMatrixKeysNode(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Fields

CombinedMatrixKeys

summary: The actual list of mappings

CombinedMatrixKeyValue

summary: For the combined matrix (i.e., all PGs for a particular realization/direction), a single entry

Methods

Constructor

summary: Constructor for use with InputXML

Parameters:

KeyValueNode(System.Xml.Linq.XElement): The XML node containing the data

Constructor

summary: Basic constructor

Parameters:

TheRow(System.Int32): The row in the combined matrix

TheGroupName(System.String): The PG GroupName associated with the row

TheFloorNum(System.Int32): The floor number associated with the row

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

KeyValue(System.Xml.Linq.XElement): Input Node

Fields

Row

summary: The row in the combined matrix

GroupName

summary: The PG GroupName associated with the row

FloorNum

summary: The floor number associated with the row

Methods

Constructor

summary: Basic constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project

TheAnalysisNum(System.Int32): The number of the analysis

TheDirectionNum(System.Int32): The direction number

TheEDPTypesAndNames(System.Collections.Generic.Dictionary{System.String): The list of all EDP types and names

NewRandomObject(PactNet.Fragilities.clsEDPType}): The random number generator object

Constructor

summary: Constructor for use with InputXML

Parameters:

ThisNode(System.Xml.Linq.XElement): Input XML Node

FillAllOriginalEDPInfo

summary: Get the original EDP info for all EDP groups

CreateOriginalCombinedMatrix

summary: Create the combined matrix of all EDP groups and floors

CreateOriginalCombinedLogMatrix

summary: Gets a clone of the original matrix, and then logs each item

Parameters:

OriginalEDPI(MathNet.Numerics.LinearAlgebra.Matrix):

GetNormalRandomMatrix

summary: Returns a matrix consisting of random normally distributed numbers

Parameters:

NumCols(System.Int32):

NumRows(System.Int32):

GetFinalValue

summary: Get the final value of a specific PG, Floor, and Realization.

Parameters:

GroupName(System.String):

FloorNum(System.Int32):

Realization(System.Int32):

MatrixToXML

summary: Convert a matrix to XML

returns: An XML node with the information

Parameters:

MatrixObject(MathNet.Numerics.LinearAlgebra.Matrix):

XMLToMatrix

summary: Will create a matrix object from an XML node

returns: A matrix

Parameters:

MatrixNode(System.Xml.Linq.XElement): An XML Node

CheckMatrix

summary: Check a matrix for NaN and infinite

Parameters:

MatrixToCheck(MathNet.Numerics.LinearAlgebra.Matrix):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

ResultDirectionEDP(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

NumEQs

summary: The number of earthquakes

NumCombinedValues

summary: The number of combined values for all fragilty types (usually NumFloors * 2 +1)

Fields

AllEDPTypesAndNames

summary: A list of all EDP Group Names and their underlying EDP types

AnalysisNum

summary: the scenario or intensity number

DirectionNum

summary: The direction number

NumRealizations

summary: The number of realizations

_numEqs

summary: The number of earthquake samples

EDPI

summary: The orginal matrix of all combined EDP values

OriginalEDPValues

summary: The list of original EDP values

UseBuildingResidualDrift

summary: Should we use the building residual drift

BuildingResidualDriftValues

summary: The collection of residual drift values

EDPO

summary: The final result matrix

RandObject

summary: The ranom number generator

CombinedMatrixKeys

summary: The list of combined matrix keys

NormRN

summary: The array of normal random numbers

GlobalDispersionValues

summary: All the dispersion values

clsResultDSSeqCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Seq Damage State

remarks: Parent: clsResultDSGroup Children: clsResultDSGroup or clsResultConsequence Inherits clsResultDS

Methods

Constructor

summary: Constructor

Parameters:

DSNode(System.Xml.Linq.XElement): The XML node containing information about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original PG object

DamageState(PactNet.Fragilities.clsFragSeqDamageState): The original damage state object

NewEDP(System.Double): The EDP value for this item

NewRandomObject(PactNet.Results.SSRandom): The random number generator

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

CalculateEDPValue

summary: The calculated probability we are in this damage state given the EDP value

Parameters:

EDP(System.Double): The actual EDP value

Median(System.Double): The median demand

Beta(System.Double): The demand dispersion

Properties

Beta

summary: The dispersion for this damage state

EDPPercent

summary: the calculated fraction of the time we are in this damage state given the EDP value

Median

summary: The median value for this damage state

FinalCostPerUnit

summary: The final cost per unit

FinalTimePerUnit

summary: The final time per unit

clsResultDirection

summary: One Result/Analysis Run/Realization/Floor/Direction

remarks: Parent: ClsResultFloor Child: clsResultPG

Methods

CreateResultTree

summary: Create the result tree structure for this direction...set up all objects

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project

EDPInfo(System.Collections.Generic.Dictionary{System.String): All of the EDP information

NewRandomObject(PactNet.Results.clsResultRealizationEDPCollection}): The random number generator

GetTotalCost

summary: Get the total cost for this direction

Parameters:

ThePGs(System.String[]): The fragilities to include, null for all

GetTotalTime

summary: Gets the total time (in person days)

Parameters:

ThePGs(System.Collections.Generic.IEnumerable{System.String}): The fragilities to include, null for all

GetTotalDeathAreas

summary: The number of square feet where everyone is dead Result is a dictionary of population group names and areas

Parameters:

ThePGs(System.Collections.Generic.IEnumerable{System.String}): The fragilities to include, null for all

GetTotalInjuryAreas

summary: The number of square feet where everyone is injured Result is a dictionary of population group names and areas

Parameters:

ThePGs(System.Collections.Generic.IEnumerable{System.String}): The fragilities to include, null for all

Calculate

summary: Calculate the consequences

GetPG

summary: Gets a particular PG result, or null if not found

Parameters:

FragilityID(System.String):

PopulationName(System.String):

EDPGroupName(System.String):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

DirectionNode(System.Xml.Linq.XElement): Input Node

OriginalFloor(PactNet.BuildingProject.clsBuildingFloor): The original floor object

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

DirectionNum

summary: The direction number

PGs

summary: The list of performance group results

GetLongLeadItems

summary: Get a list of long lead items

GetRedTagList

summary: Get the list of all damage states (and substates) that have a red tag state. There will be one entry per fragid/DS that is in a red tag state. The list is in the format value=DSNUM/Sub-DSNUM/etc.

remarks: This could be deceptive, if there are 20 units, and each is a simultaneous fragility with all of the sub-damage states having red-tag consequences and each unit is in 2 damage states, there will be 2 entries, each with the count of 20

Fields

FloorNum

summary: The floor number

_PGs

summary: The list of performance group results

clsResultAnalysisEDPNormal

summary: The class for a collection of all (normally distributed) calculated EDP values for this analysis case NOT USED as of 9-2012

remarks: Parent: clsResultAnalysis Children: clsResultRealizationEDPCollection

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The intesnity or scenario number

RandObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

GetEDPValue

summary: Get a single EDP value

returns: The EDP Value

Parameters:

DirNum(System.Int32): The direction number

ValueNum(System.Int32): The value (floor) of the EDP value

EDPGroupName(System.String): The EDP Group name

Realization(System.Int32): The realization number

GetRealizationEDPs

summary: Get all EDPs for a realization, grouped by EDP group name

returns: A dictionary containing EDPGroup names and an EDP collection for each

Parameters:

RealizationNum(System.Int32): The realization number

MasterProject(PactNet.BuildingProject.clsProject): The original project

CalculateNormallyDistributedData

summary: Calculate EDP values for all realizations

Parameters:

MasterProject(PactNet.BuildingProject.clsProject):

CalculateNormalRealizationEDPs

summary: Create all EDPs for this realization, using a normally distributed curve

GetNormalEDP

summary: Create a normally distributed EDPs by getting a mean and std dev from all existing EDP data, and using that to produce a new random EDP

returns: The EDP value

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

AnalysisNum(System.Int32): The intensity or scenario number

FloorNum(System.Int32): The floor number

DirNum(System.Int32): The direction number

EDPGroupName(System.String): The EDP group name

RandomNum(System.Double): The random number used to create the new EDP value

Median(System.Double@): Out: The median EDP value calculated

Dispersion(System.Double@): Out: The EDP dispersion calculated

GetNormalEDP2

summary: Create normally distributed EDPs by getting a mean and std dev from all existing EDP data, and using that to produce a new random EDP

returns: The calculated EDP Value

Parameters:

TempEDPClsArray(PactNet.BuildingProject.clsEDPArray): The array used to find the median and dispersion

FloorNum(System.Int32): The floor number

RandomNumber(System.Double): The random number used to create the new EDP value

Median(System.Double@): Out: The median EDP value calculated

Dispersion(System.Double@): Out: The EDP dispersion calculated

ExportAllEDPData

summary: Exports all data to the file system (if requested)

InputXML

summary: Fill this object from an XML node

Parameters:

ThisNode(System.Xml.Linq.XElement):

OutputXML

summary: Create an XML node containing all the information from this object

Fields

AllNormalEDPValuesList

summary: Yes, this is a hack, it is a list of every single realization, and within that a dictionary of EDP types and All other data like floors, directions, and EDPs

SSRandom

summary: A wrapper for the built in Random class

Methods

Constructor

summary: Constructor, with a time based seed value

Constructor

summary: Initializes a new random number with a given seed value

Parameters:

SeedValue(System.Int32): The seed value

Next

summary: Returns a non-negative random number

NextDouble

summary: Returns a number greater than or equal to 0 and less than 1

AllPoint5Random

summary: A class to make all random numbers be 0.5 instead

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor

Parameters:

SeedValue(System.Int32): Ignored

Next

summary: Return 50

NextDouble

summary: Return 0.5

returns: 0.5

clsResultAnalysisEDPEigenDecomp

summary: The class for a collection of all calculated EDP values for this analysis case using Farzin's alternative decomposition method

remarks: Base class: clsResultAnalysisEDPCommon Parent: clsResultAnalysis Children: clsResultDirectionEDPEigenDecomp

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The scenario or intensity number

RandObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

CalculateEigenDecompValues

summary: Calculate all Cholesky values from the project

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

CholeskyAnalysis(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

clsResultDSGroupMutExCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Mutually Exclusive Correlated Damage State Group

remarks: Parent: clsResultItemGroup or clsResultDS Children: clsResultDSMutExCor

Methods

Constructor

summary: Constructor for use by InputXML

Parameters:

DSGroupNode(System.Xml.Linq.XElement): The XML node containg info about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original performance group object

DamageStates(PactNet.Fragilities.clsFragDamageStates): The original damage states object

NewEDP(System.Double): The EDP value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

CalculateDamageState

summary: Calculate the damage state(s) we are in

Parameters:

AllResultDamageStates(System.Collections.Generic.List{PactNet.Results.clsResultDS}):

CalculateFinalValues

summary: Calculate the final consequence values

Properties

PercentDamaged

summary: Calculate the fraction of items damaged

clsResultAnalysisEDPSimplified

summary: The class for a collection of all (normally distributed) calculated EDP values for this analysis case

remarks: Parent: clsResultAnalysis Children: clsResultRealizationEDPCollection

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The original project file

TheAnalysisNum(System.Int32): The intensity or scenario number

RandObject(PactNet.Results.SSRandom): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

EDPNode(System.Xml.Linq.XElement): The XML element used to create this object

OriginalProject(PactNet.BuildingProject.clsProject): The original project object

GetEDPValue

summary: Get a single EDP value

returns: The EDP Value

Parameters:

DirNum(System.Int32): The direction number

ValueNum(System.Int32): The value (floor) of the EDP value

EDPGroupName(System.String): The EDP Group name

Realization(System.Int32): The realization number

GetRealizationEDPs

summary: Get all EDPs for a realization, grouped by EDP group name

returns: A dictionary containing EDPGroup names and an EDP collection for each

Parameters:

RealizationNum(System.Int32): The realization number

MasterProject(PactNet.BuildingProject.clsProject): The original project

GetNormalRealizationEDPs

summary: Get all EDPs for a realization, by EDP group name

Parameters:

RealizationNum(System.Int32): The realization number

CalculateSimplifiedDistributedData

summary: Calculate the EDP values

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

CalculateSimplifiedRealizationEDPs

summary: Create all EDPs for this realization, using the simplified method

GetSimplifiedEDP

summary: Create simplified EDPs by using the mean and std dev

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

AnalysisNum(System.Int32): The intensity number

FloorNum(System.Int32): The floor number

DirNum(System.Int32): The direction number

EDPGroupName(System.String): The EDP group name

RandomNumber(System.Double): The random number to use to calculate the EDP value

ExportAllEDPData

summary: Exports all data to the file system (if requested)

InputXML

summary: Fill this object from an XML node containg info about this object

Parameters:

ThisNode(System.Xml.Linq.XElement):

OutputXML

summary: Create an XML node containing the info about this object

Fields

AllSimplifiedEDPValuesList

summary: Yes, this is a hack, it is a list of every single realization, and within that a dictionary of EDP types and All other data like floors, directions, and EDPs

clsResultRealizationEDPCollection

summary: A class holding all the original info for a particular EDP type (Drift, accel, etc) for a particular analysis and realization (all directions, and all floors)

remarks: Parent: clsResultAnalysisEDPNormal and clsResultAnalysisEDPSimplified Children: None

Methods

Constructor

summary: Constructor

Constructor

summary: Constructor

Parameters:

EDPCollectionNode(System.Xml.Linq.XElement): The XML node containing all the information about this object

Constructor

summary: Constructor

Parameters:

TheEDPType(PactNet.Fragilities.clsEDPType): The EDP type

TheNumDirections(System.Int32): The number of directions (3)

TheNumValues(System.Int32): The number of values (typically # of floors or # of floors+1)

SetRand

summary: Set the random number used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

value(System.Double):

GetRand

summary: Get the random number used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

SetMedian

summary: Set the median used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

value(System.Double):

GetMedian

summary: Get the median used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

SetBeta

summary: Set the beta used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

value(System.Double):

GetBeta

summary: Get the beta used to create the EDP value

Parameters:

DirNum(System.Int32):

FloorNum(System.Int32):

ChangeNumDirections

summary: Change the number of directions (which should never really happen)

Parameters:

NewNumDirections(System.Int32):

ChangeNumValues

summary: Change the number of values (floors)

Parameters:

NewNumValues(System.Int32):

InputXML

summary: Fill this object from information from an XML node

Parameters:

EDPCollectionNode(System.Xml.Linq.XElement): The XML Node

OutputXML

summary: Output an XML node containging all the info about this object

Properties

Item

summary: Get the EDP value for a specific direction and floor

Parameters:

DirectionNum(System.Int32): The direction

ValueNum(System.Int32): The value (floor)

NumDirections

summary: The number of directions

NumValues

summary: The number of values (floors)

EDPType

summary: The EDP type

Fields

_numDirections

summary: The number of directions (silly I know, it should always be 3)

_numValues

summary: The number of values (which is either # of floors, or # of floors +1)

_EDPType

summary: The EDP type

InternalRandArray

summary: The array of random numbers used to calculate EDP values

InternalMedianArray

summary: The array of EDP median values

InternalBetaArray

summary: The array of EDP dispersions

InternalEDPArray

summary: The array of calculated EDP values

clsResultSetViewFittedCurve

summary: A class that holds all the fitted curves view, and the adjusted total, for all analysis cases

Methods

Constructor

summary: Constructor

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

AllAnalysisViews(System.Collections.Generic.List{PactNet.Results.clsResultAnalysisView}): The list of individual analysis results

NumBinsDesired(System.Int32): The maximum number of bins desired

clsResultPG

summary: One Result/Analysis Run/Direction/Realization/Floor/PG

remarks: Parent: clsResultDirection Children: clsResultItemGroup

Methods

Constructor

summary: Constructor

Parameters:

PGNode(System.Xml.Linq.XElement): The XML node containing the information about this object

OriginalDirection(PactNet.BuildingProject.clsBuildingDirection): The original clsBuildingDirection object

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original Performance Group object

NewEDP(System.Double): The EDP value

NewRandomObject(PactNet.Results.SSRandom): The random number generator

CalculateQuantity

summary: Calculate the quantity based on the quantity uncertainty

Calculate

summary: Do the main calculations

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

PGNode(System.Xml.Linq.XElement): Input Node

OriginalDirection(PactNet.BuildingProject.clsBuildingDirection): The original direction object

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

EDP

summary: The EDP value for this item

FragilityID

summary: The Fragility ID associated with this item

Correlation

summary: The correlation of this item

CorrelationBool

summary: A correlation boolean, is it correlated or not

ItemGroups

summary: The list of all the item groups

NumUnits

summary: The number of units calculated after uncertainty

OriginalNumUnits

summary: The original number of units from the original PG

QuantityUncertaintyRand

summary: The random number created for calculating the quantity uncertainty

QuantityUncertainty

summary: The quantity uncertainty dispersion

NumUnitsDamaged

summary: The number of units actually damaged

NumUnitsDamagedAcrossBuilding

summary: The number of units of this type damages across the building

TotalCost

summary: Get the total cost for this PG

TotalTime

summary: Get the total downtime for this PG

AllDeathArea

summary: The number of square feet where everyone is dead, seperated by population group Return a keyvaluepair of populationgroupnames and amounts

AllInjuryArea

summary: The number of square feet where everyone is injured, seperated by population group Return a keyvaluepair of populationgroupnames and amounts

LongLeadFlag

summary: Does this PG have any actual damage states that indicate a long lead

GetRedTagList

summary: Get a dictionary of all damage states (and substates) that have a red tag state, and the count. There will be one item per fragid/DS that is in a red tag state. The list is in the format value=DSNUM/Sub-DSNUM/etc.

remarks: This could be deceptive, if there are 20 units, and each is a simultaneous fragility with all of the sub-damage states having red-tag consequences and each unit is in 2 damage states, there will be 2 entries, each with the count of 20

Fields

_numUnits

summary: The number of items in this PG (after quantity uncertainty)

_originalNumUnits

summary: The number of utems in this PG (before quantity uncertainty)

_quantityUncertaintyRand

summary: The random number used for quantity uncertainty

_quantityUncertainty

summary: The quantity uncertainty (dispersion)

_fragilityID

summary: The fragility id

FragilityName

summary: The name of the fragility

Population

summary: The population associated with this PG

_correlation

summary: The correlation value for this PG, 0 = correlated, 1 = non-correlated

_itemGroups

summary: The list of items groups (which will always be a 1 item list if correlated)

RandomObject

summary: The random number generator

EDPGroupName

summary: The EDP Group name this PG is associated with

_edp

summary: The EDP value

_numUnitsDamagedAcrossBuilding

summary: The number of units damaged across the entire building

clsResultDSSimultCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Simultaneous Damage State

remarks: Parent: clsResultDSGroup Children: clsResultDSGroup or clsResultConsequence Inherits clsResultDS

Methods

Constructor

summary: Constructor only for use with InputXML

Parameters:

DSNode(System.Xml.Linq.XElement): The XML node containing info about this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates):

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original PG object

DamageState(PactNet.Fragilities.clsFragSimultDamageState): The original damage state object

NewEDP(System.Double): The EDP value for this item

NewRandomObject(PactNet.Results.SSRandom): The random number generator

InputXML

summary: Fill this object with information from an XML node element

Parameters:

DamageStateNode(System.Xml.Linq.XElement): The XML node

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

Percent

summary: The fraction of the time this damage state will be encountered

FinalCostPerUnit

summary: Get the final cost per unit

FinalTimePerUnit

summary: Get the final downtime per unit

clsResultFloor

summary: One Result/Analysis Run/Realization/Floor

remarks: Parent: clsResultReal Children: clsResultDirection

Methods

Calculate

summary: Do the final calculation of consequences

CreateResultTree

summary: Create the result tree and all subobjects

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The original project file

EDPInfo(System.Collections.Generic.Dictionary{System.String): The collection of EDP information

NewRandomObject(PactNet.Results.clsResultRealizationEDPCollection}): The random number generator

GetTotalCost

summary: Gets the total repair cost for the floor

Parameters:

TheDirections(System.Int32[]):

ThePGs(System.String[]):

GetTotalTime

summary: Gets the total downtime for the floor (in total days)

Parameters:

TheDirections(System.Collections.Generic.IEnumerable{System.Int32}):

ThePGs(System.Collections.Generic.IEnumerable{System.String}):

GetTotalDeaths

summary: Gets the total for all deaths across all population models

Parameters:

TheDirections(System.Collections.Generic.IEnumerable{System.Int32}):

ThePGs(System.Collections.Generic.IEnumerable{System.String}):

GetTotalInjuries

summary: Gets the total of all the injuries for all population models

Parameters:

TheDirections(System.Collections.Generic.IEnumerable{System.Int32}):

ThePGs(System.Collections.Generic.IEnumerable{System.String}):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

FloorNode(System.Xml.Linq.XElement): Input Node

OriginalBuilding(PactNet.BuildingProject.clsBuilding): The original building object

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

Directions

summary: The list of direction results

FloorPopulation

summary: The total population for the floor

CollapseProb

summary: The percent of the floor collapsed

CollapseDeaths

summary: The deaths resulting from collapse

CollapseInjuries

summary: The injuries resulting from collapse

CollapseDeathRateBetaRand

summary: The random number used to calc the deaths resulting from collapse

CollapseInjuryRateBetaRand

summary: The random number used to calc the injuries resulting from collapse

PMax

summary: The maximum workers per square foot

FloorAreaFeet

summary: The floor area in feet

CostFactor

summary: The multiplication cost factor

FloorNum

summary: The floor num

DeathsAndInjuriesAdjusted

summary: Determine if the deaths and injuries had to be adjusted to make sure they didn't exceed the maximum of the building

GetLongLeadItems

summary: Get a list of all the long lead items

GetRedTagList

summary: Get the list of all damage states (and substates) that have a red tag state. There will be one entry per fragid/DS that is in a red tag state. The list is in the format value=DSNUM/Sub-DSNUM/etc.

remarks: This could be deceptive, if there are 20 units, and each is a simultaneous fragility with all of the sub-damage states having red-tag consequences and each unit is in 2 damage states, there will be 2 entries, each with the count of 20

Fields

_floorNum

summary: The floor number

_directions

summary: The directions

RandomObject

summary: The random number generator

_floorAreaFeet

summary: The floor area, in square feet

PopulationAreas

summary: The list of all populations models and the areas of each

PopulationPeople

summary: The list of all population models, and the number of people in each

IsCollapsed

summary: Is this building collapsed

_collapseDeathRateBetaRand

summary: Death rate beta rand

_collapseInjuryRateBetaRand

summary: Injury rate beta rand

_collapseDeaths

summary: The number of deaths from collapse

_collapseInjuries

summary: The number of injuries from collapse

_collapseProb

summary: The fraction of the floor collapsed

_costFactor

summary: The total cost multiplication factor for this floor

_pMax

summary: The maximum workers per square foot

clsResultDSMutExCor

summary: One Result/Analysis Run/Direction/Realization/Floor/PG/(dsgroup/DS)*/Damage State Group/Correlated Mutually Exclusive Damage State

remarks: Parent: clsResultDSMutExGroup Children: clsResultDSGroup or clsResultConsequence Inherits clsResultDS

Methods

Constructor

summary: Constructor only for use with InputXML

Parameters:

DSNode(System.Xml.Linq.XElement): The XML node containing the information used to fill this object

OriginalDSGroup(PactNet.Fragilities.clsFragDamageStates): The original damage state group

Constructor

summary: Constructor

Parameters:

PGGroupInfo(PactNet.BuildingProject.clsPerformanceGroup): The original PG object

DamageState(PactNet.Fragilities.clsFragMutExDamageState): The original damage state object

NewEDP(System.Double): The EDP value for this item

NewRandomObject(PactNet.Results.SSRandom): The random number generator

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

Percent

summary: The fraction of the time this damage state will be encountered

FinalCostPerUnit

summary: The final cost per unit

FinalTimePerUnit

summary: The final time per unit

clsResultRedTagItem

summary: A class representing a single fragility and damage state, to help determine quantity for red tagging Parent: clsResultReal, clsResultFloor, ClsResultDirection, clsResultPG

Methods

Equals

summary: Compare two objects to see if the are equal

Parameters:

other(PactNet.Results.clsResultRedTagItem):

Equals

summary: Compare two objects to see if the are equal

Parameters:

x(PactNet.Results.clsResultRedTagItem):

y(PactNet.Results.clsResultRedTagItem):

GetHashCode

summary: Get a hashcode

Parameters:

obj(PactNet.Results.clsResultRedTagItem):

Equals

summary: Compare two objects to see if the are equal

Parameters:

obj(System.Object):

GetHashCode

summary: Get a hashcode (this is the one that does the actual comparison I think)

CompareTo

summary: Compare two objects to see if the are equal

Parameters:

obj(System.Object):

Properties

DamageStateID

summary: The Damage state ID, in the format of DSNUM/Sub-DSNUM/etc.

Fields

FragID

summary: The fragility ID

_damageStateID

summary: The format is DSNUM/Sub-DSNUM/etc.

clsResultDirectionEDPEigenDecomp

summary: All the generated values for all the EDPs for a particular Analysis and Direction for Farzin's eigen value decomposition method

remarks: Base class: clsResultDirectionEDPCommon Parent: clsResultAnalysisEDPCommon Children: clsEDPType, clsEDPArray (CombinedMatrixKeys)

Methods

Constructor

summary: Constructor

Parameters:

TheMasterProject(PactNet.BuildingProject.clsProject): The project this is based on

TheAnalysisNum(System.Int32): The analysis number

TheDirectionNum(System.Int32): The direction

TheEDPTypesAndNames(System.Collections.Generic.Dictionary{System.String): The list of EDP types that need to be calculated

NewRandomObject(PactNet.Fragilities.clsEDPType}): The random number generator

Constructor

summary: Constructor for use with InputXML

Parameters:

ThisNode(System.Xml.Linq.XElement): Input XML Node

ConvertToDiag

summary: Convert a 1D matrix (with the long dimension being rows) into a diagonal matrix

Parameters:

OriginalMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

FindMeansOfMatrix

summary: Returns a matrix of means of the source matrix

Parameters:

SourceMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

FindVariance

summary: Returns the variance of an array

Parameters:

OneDArray(System.Double[]):

FindMean

summary: Finds the mean of an array

Parameters:

OneDArray(System.Double[]):

FindVariance

summary: Finds the variance of an array

Parameters:

Mean(System.Double):

OneDArray(System.Double[]):

FindCovariance

summary: Finds the covariance between two 1D arrays

Parameters:

Array1(System.Double[]):

Array2(System.Double[]):

FindCovarianceMatrix

summary: Get a full covariance matrix for a 2D array

Parameters:

TempMatrix(MathNet.Numerics.LinearAlgebra.Matrix):

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

ResultDirectionEDP(System.Xml.Linq.XElement): Input Node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Fields

lnEDPs

summary: The original matrix, log of all values

lnEDPs_Means

summary: The means of all the (log) original values

lnEDPs_cov

summary: The cov of all the values

clsResultSet

summary: One result set...top level object

remarks: Parent: None Child: clsResultAnalysis

ProgressReportEventHandler

summary: The delegate for the progress report

Parameters:

Description(): The description of the current progress

AnalysisCase(): The intensity or scenario this applies to

progress(): The fraction of completion

IOEventHandler

summary: The delegate for the progress report

Parameters:

Description(): The description of the current progress

AnalysisCase(): The intensity or scenario this applies to

progress(): The fraction of completion

Methods

Constructor

summary: Constructor

UpdateProgressReport

summary: Call the Progress Report event

Parameters:

Description(System.String):

AnalysisCase(System.Int32):

progress(System.Int32):

UpdateIOReport

summary: Call the IO Report event

Parameters:

Description(System.String):

AnalysisCase(System.Int32):

progress(System.Int32):

Calculate

summary: Calculate all Analysis cases

returns: String indicating # of milliseconds taken

Parameters:

UseThreads(System.Boolean): Do you wish to use the thread pool, or just run them sequentially

RandomSeed(System.Int32): The random seed to use for the evaluation

CreateResultTree

summary: Create the entire result tree structure

CreateResultTreeWithThreadPool

summary: Create the entire result tree structure

SubCalculate

summary: Actually go through the tree of pre-calculated damage states and determine costs (no threads)

SubCalculateWithThreadPool

summary: Actually go through the tree of pre-calculated damage states and determine costs (using threads)

ThreadCompleted

summary: Keeps a count of the threads completed

Parameters:

sender(System.Object):

EndTime(System.DateTime):

GetIntegratedResults

summary: Get the integrated cost results for all analysis cases

Parameters:

ViewType(PactNet.Shared.Globals.ResultViewTypes): The view type (Cost, Time, etc) that you wish to get

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

GetIntegratedResults

summary: Get the integrated cost results for all analysis cases

Parameters:

ViewType(PactNet.Shared.Globals.ResultViewTypes): The view type (Cost, Time, etc) that you wish to get

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

IfTimeUseSerial(System.Boolean): If the ViewType is Time, do you want to use serial or parallel

GetIntegratedCostResults

summary: Get the integrated cost results for all analysis cases

Parameters:

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

GetIntegratedTimeResults

summary: Get the integrated time results for all analysis cases

Parameters:

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

UseSerial(System.Boolean):

GetIntegratedDeathResults

summary: Get the integrated death results for all analysis cases

Parameters:

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

GetIntegratedInjuryResults

summary: Get the integrated injury results for all analysis cases

Parameters:

TheFloors(System.Int32[]): The list of floors to get

TheDirections(System.Int32[]): The list of directions to get

ThePGs(System.String[]): The list of performance groups to get

GetIntegratedRedTag

summary: Get the integrated red tag result for all analysis cases

GetIntegratedCollapse

summary: Get the integrated collapse result for all analysis cases

GetIntegratedResidualDrift

summary: Get the integrated residual drift result for all analysis cases

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

AnalysisSet(System.Xml.Linq.XElement): Input Node

InputXML

summary: Inputs an XML document containing a Analysis Set node and subnodes, and creates the object structure

Parameters:

XMLDocument(System.Xml.Linq.XDocument): An XML document that contains a Project node

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

LoadXMLProject

summary: Loads an XML document representing a Analysis run, and creates all the objects

Parameters:

FilePath(System.String): The pathname (with trailing \) for the project document

FileName(System.String): The filename for the project document

LoadXMLProject

summary: Loads an XML document representing a Analysis run, and creates all the objects

Parameters:

FilePathAndName(System.String): The path and file name for the project document

SaveProjectToXMLFile

summary: Saves the project object (and sub objects) to a file

Parameters:

SaveFilePath(System.String): The FilePath to save to

SaveFileName(System.String): The FileName to save to

SaveProjectToXMLFile

summary: Saves the project object (and sub objects) to a file

Parameters:

SaveFilePathAndName(System.String): The file path and name to save to

Constructor

summary: A specific constructor for the serializer to use It manually adds the master project and some other info

Parameters:

si(System.Runtime.Serialization.SerializationInfo):

ctx(System.Runtime.Serialization.StreamingContext):

GetObjectData

summary: Handler for when I am serializing, I add all of the result set data manually

Parameters:

info(System.Runtime.Serialization.SerializationInfo):

context(System.Runtime.Serialization.StreamingContext):

Properties

RunDate

summary: The date this was run on

AnalysisRuns

summary: The list of all analysis runs

NumRuns

summary: The number of runs

AllUniqueFragilities

summary: A collection of all the unique fragilities used for all runs

MasterProject

summary: The project this is tied to

IsTimeBased

summary: Can this Result set show time-based info i.e. does it contain all the MAFEs

Fields

AttemptingCancel

summary: Are we attempting to cancel the evaluation

_runDate

summary: When was this run

_analysisRuns

summary: The list of scenarios or intensities

_numRuns

summary: The number of intensities

MasterRandomObject

summary: The master random number generator

AnalysisRandomObjects

summary: The random number generators for each scenario or intensity

StageName

summary: The description of the stage of evaluation we are currently on

_allUniqueFragilities

summary: A list of all unique fragilities

_masterProject

summary: The master project file

RunningThreads

summary: The number of currently running threads

Events

ProgressReport

summary: An event that runs during evaluation to update the UI

IOReport

summary: An event that runs during evaluation to update the UI

clsResultReal

summary: One Result/Analysis Run/Realization

remarks: Parent: clsResultAnalysis Children: clsResultFloor, clsResultRealFrag

Methods

CreateResultTree

summary: Create the whole tree structure of objects to represent every possible Floor/Dir/PG/DS

Parameters:

MasterProject(PactNet.BuildingProject.clsProject): The ClsProject object

RandObject(PactNet.Results.SSRandom): The random number generator

AllEDPs(PactNet.Results.clsResultAnalysisEDPs): A list of all EDP groups and the EDP info that goes with them

RealNum(System.Int32): The current realization number

SetupAllFrags

summary: Setup the fragility list

Parameters:

AllUniqueFragilities(SSUtils.SortableBindingList{PactNet.Fragilities.clsFragilityCurve}): A list of all the unique fragilities

FindActualQuantitiesDamaged

summary: Go through the PGS and find the actual quantities that have been damaged, after quantity uncertainty adjustment

PushActualQuantitiesDamaged

summary: Push the Actual Quantities Damaged information to every PG

Calculate

summary: Do the final calculation

FillPopulation

summary: Fill all the population variables for the building

GetFinalFloorPopulation

summary: Gets the final floor population

CalculateCollapse

summary: Calculate if this realization has collapsed

CalculateCollapseCost

summary: Assuming we have collapsed, calculate the total cost

CalculateCollapseCasualties

summary: Assuming we have collapsed, calculate the casualties

CalculateResidualDrift

summary: Calculate if this realization has a residual drift failure

GetTotalCost

summary: Get the total cost for this realization

Parameters:

TheFloors(System.Int32[]): A list of floors, or empty (or null) for all floord

TheDirections(System.Int32[]): A list of directions, or empty (or null) for all directions

ThePGs(System.String[]): A list of PGs, or empty (or null) for all PGs

GetTotalTime

summary: Get the total repair time for this realization

Parameters:

TheFloors(System.Int32[]): A list of floors, or empty (or null) for all floors

TheDirections(System.Int32[]): A list of directions, or empty (or null) for all directions

ThePGs(System.String[]): A list of PGs, or empty (or null) for all PGs

IsSerial(System.Boolean): Get the time as serial, or parallel

GetTotalDeaths

summary: Get the total deaths for this realization

Parameters:

TheFloors(System.Int32[]): A list of floors, or empty (or null) for all floors

TheDirections(System.Int32[]): A list of directions, or empty (or null) for all directions

ThePGs(System.String[]): A list of PGs, or empty (or null) for all PGs

GetTotalInjuries

summary: Get the total injuries for this realization

Parameters:

TheFloors(System.Int32[]): A list of floors, or empty (or null) for all floors

TheDirections(System.Int32[]): A list of directions, or empty (or null) for all directions

ThePGs(System.String[]): A list of PGs, or empty (or null) for all PGs

FindQuantitiesInEachDamageState

summary: For Red Tagging, get a count of how many of each fragility/DS have occured

IsRedTagged

summary: Is this realization red tagged (or collapsed)

GetRedTaggedItems

summary: Get a list of all red tagged items for this realization

GetSAT1Final

summary: Get the final SAT1

InputXML

summary: Input an XML node and create the object and all sub-objects from it

Parameters:

Realization(System.Xml.Linq.XElement): Input Node

OriginalProject(PactNet.BuildingProject.clsProject): The original project

OutputXML

summary: Output an XML node containing info about this object and all sub-objects from it

Properties

AllFrags

summary: A list of all fragilities, and the total amount of items damaged for each

IsCollapsed

summary: Has this realization collapsed

AnalysisNum

summary: Which run number this realization is in

RealizationNum

summary: The realization number

Floors

summary: The list of all clsResultFloor objects

RealizationTime

summary: The simulated date and time of day this realization happened on

GetLongLeadItems

summary: Get a list of long lead items for this realization

ResidualDriftFail

summary: Did this realization fail it's residual drift test

Fields

_analysisNum

summary: The intensity or scenario number

_realizationNum

summary: The realization number

_floors

summary: The collection of floor results

_realizationTime

summary: The date and time of day the realization was calculated to have occured

_allFrags

summary: A list of all fragilities, and the total amount of items damaged for each

NewRandomObject

summary: The random number generator

_collapseModes

summary: A copy of the original collapse modes object from the building

DateMultiplier

summary: The date cost multiplier

RegionMultiplier

summary: The region cost multiplier

_isCollapsed

summary: Is the building collapsed

CollapseMode

summary: If collapsed, what collapse mode is it in

BuildingReplacementTime

summary: The original building replacement time

BuildingReplacementCost

summary: The original building replacement cost

BuildingCoreAndShellReplacementCost

summary: The original building core and shell replacement cost

PercentLossThreshold

summary: The fraction of the total cost required to consider it a total loss

NumFloors

summary: The number of floors

PopulationModelRands

summary: A list of all Population models, and the random numbers used to generate their population

PopulationModel_PopPerSF

summary: A list of all Population models, and the calculated population per SF

CollapseDeathRateBetaRand

summary: The random number used with the Death Rate Beta to get the final Collapse Death Rate

CollapseInjuryRateBetaRand

summary: The random number used with the Injury Rate Beta to get the final Collapse Injury Rate

clsResultAnalysisView

summary: A single result set for a single analysis case and a specified set of floors, dirs, and PGs

Methods

Constructor

summary: Constructor

Parameters:

TheFloors(System.Int32[]): The floors to include, null for all

TheDirections(System.Int32[]): The directions to include, null for all

ThePGs(System.String[]): The PGs to include, null for all

GetMaxValue

summary: Get the maximum actual calculated value

GetFittedCurve

summary: Gets a log-normally fitted curve, letting it determine increments

Parameters:

DesiredCurveNumIncrement(System.Int32): The max desired number of increments to create the internal array (note: the result may be different)

ForcePowerof10(System.Boolean): Force the increment value to be a power of 10

AdjustmentFactor(System.Double): A muliplication factor to use to give a nice buffer on the right side of the graph

GetFittedCurve

summary: Gets a log-normally fitted curve

Parameters:

NumIncrements(System.Int32): The number of increments to create the internal array

IncrementValue(System.Double): The value of each increment

GetHistogram

summary: Return the binned data for this curve, letting it determine increments

returns: The binned data for this curve

Parameters:

NumBins(System.Int32): The max number of bins desired

ForcePowerOf10(System.Boolean): Do we want to force the incerement value to be a power of 10

Adjustment(System.Double): The adjustment factor (1 for none)

GetHistogram

summary: Return the binned data for this curve, specifying the increments

Parameters:

NumBins(System.Int32): The number of bins to use

BinAmount(System.Double): The amount of each bin

Properties

NumRealizations

summary: The number of realizations

TheFloors

summary: The floors to include, null for all

TheDirections

summary: The directions to include, null for all

ThePGs

summary: The PGs to include, null for all

MainArray

summary: The original array of realization costs

Fields

_mainArray

summary: the original array of values that all other data is based on

_theFloors

summary: The list of floors used for this curve

_theDirections

summary: The list of direction used for this curve

_thePGs

summary: The list of PGs used for this curve

_numRealizations

summary: The number of realizations

ResultCurveType

summary: The type of curve (cost, time, etc)

SSUtils Assembly

ObjectCopier

summary: Reference Article http://www.codeproject.com/KB/tips/SerializedObjectCloner.aspx Provides a method for performing a deep copy of an object. Binary Serialization is used to perform the copy.

Methods

Clone2``1

summary: Perform a deep Copy of the object.

typeparam(name=T): The type of object being copied.

returns: The copied object.

Parameters:

source(``0): The object instance to copy.

SSMisc

summary: Stuff that fits nowhere else

MiscEnum

summary: Enumeration help stuff

Methods

ReturnEnumNames

summary: Get a string list of the enumeration names

Parameters:

EnumType(System.Type):

DoesStringBelong

summary: Does a particular string match one of the enumeration names

Parameters:

EnumType(System.Type):

EnumString(System.String):

ReturnEnumValue

summary: Get a value of an enumeration if given the enum and string

Parameters:

EnumType(System.Type):

EnumString(System.String):

TriState

summary: A class that is true, false, or neither

Methods

Constructor

summary: Constructor, defaults to unknown

Constructor

summary: Create a new tristate with a nullable boolean value

Parameters:

Value(System.Nullable{System.Boolean}):

Constructor

summary: Create a new tristate with a set boolean value

Parameters:

Value(System.Boolean):

Equals

summary: Determine if this tristate is equal to another

Parameters:

obj(System.Object):

IsEqual

summary: Determine if this tristate is equal to another

Parameters:

T1(SSUtils.SSMisc.TriState):

IsEqual

summary: Determine if this tristate is equal to another

Parameters:

B1(System.Boolean):

GetHashCode

summary: Get a hash code

op_LogicalNot

summary: Logical not. True=false, false=true, Unknown = unknown

Parameters:

T1(SSUtils.SSMisc.TriState):

op_BitwiseAnd

summary: Logical And. True and True=True, False and False=False, Unknown and Anything=Unknown

Parameters:

T1(SSUtils.SSMisc.TriState):

T2(SSUtils.SSMisc.TriState):

op_BitwiseOr

summary: Logical Or. unknown|unknown=unknown, true|anything=true, false|unknown = false, false|false = false

Parameters:

T1(SSUtils.SSMisc.TriState):

T2(SSUtils.SSMisc.TriState):

op_True

summary: Is this tristate true

Parameters:

T1(SSUtils.SSMisc.TriState):

op_False

summary: Is this tristate false

Parameters:

T1(SSUtils.SSMisc.TriState):

op_Equality

summary: Are two tristate equal

Parameters:

T1(SSUtils.SSMisc.TriState):

T2(SSUtils.SSMisc.TriState):

op_Inequality

summary: Are two tristates not equal

Parameters:

T1(SSUtils.SSMisc.TriState):

T2(SSUtils.SSMisc.TriState):

op_Equality

summary: Is a tristate equal to a true boolean

Parameters:

T1(SSUtils.SSMisc.TriState):

B2(System.Boolean):

op_Inequality

summary: Is a tristate not equal to a boolean

Parameters:

T1(SSUtils.SSMisc.TriState):

B2(System.Boolean):

op_Equality

summary: Is a tristate equal to a true boolean

Parameters:

T1(System.Boolean):

B2(SSUtils.SSMisc.TriState):

op_Inequality

summary: Is a tristate not equal to a boolean

Parameters:

T1(System.Boolean):

B2(SSUtils.SSMisc.TriState):

op_Implicit

summary: Implicit cast

Parameters:

B1(System.Boolean):

op_Implicit

summary: Implicit cast

Parameters:

B1(System.Nullable{System.Boolean}):

ToString

summary: Return text describing the string

Properties

IsTrue

summary: Get or set the truth state

IsFalse

summary: Get or set the false state

IsUnknown

summary: Get or set the third state

FlagStack

summary: A class to only change an underlying boolean value if the number of times it has been set to true is the same as the number set to false

Methods

Constructor

summary: Keep track of the changes in a boolean variable

Parameters:

InitialValue(System.Boolean): Whether the initial value is true or false...important, the first set should be the opposite value

Constructor

summary: Keep track of the changes in a boolean variable

Parameters:

InitialValue(System.Boolean): Whether the initial value is true or false

AutoIncludeFirstSet(System.Boolean): Should it include the first set...if true the first set should be the same value as the inital value

Set

summary: Sets a boolean value, and returns the actual boolean value to use

Parameters:

Value(System.Boolean):

Properties

LastSetChanged

summary: Did the last set change the actual underlying value

Value

summary: Gets the current actual underlying value

StackCount

summary: Gets the current state of the stack (0 means equal pushes and pops)

Fields

stackCount

summary: The size of the stack

initialValue

summary: The initial value of the stack

lastSetChanged

summary: Did the last set change the actual underlying value

Profiler

summary: A very simple profiler so we can see how long some things take

Methods

StartClock

summary: Start a new timer clock

Parameters:

ProfileName(System.String): A key to differentiate this from another timer

GetCurrentTick

summary: Get the number of ticks since the clock started

Parameters:

ProfileName(System.String): The key for this timer

GetDiff

summary: Get the number of ticks since the last time we called GetDiff or started the clock

Parameters:

ProfileName(System.String): The key for this timer

PrintTick

summary: Print the number of ticks since the clock started to Debug.Print

Parameters:

ProfileName(System.String): The key for this timer

Label(System.String): A label to use for this particular tick

PrintTick

summary: Print the number of ticks since the clock started to Debug.Print

Parameters:

ProfileName(System.String): The key for this timer

Label(System.String): A label to use for this particular tick

DividedBy(System.Int64): A number to divide the ticks by (10,000,000 will be seconds)

PrintDiff

summary: Print the number of ticks since the last time we called GetDiff or started the clock

Parameters:

ProfileName(System.String): The key for this timer

Label(System.String): A label to use for this particular tick

PrintDiff

summary: Print the number of ticks since the last time we called GetDiff or started the clock

Parameters:

ProfileName(System.String): The key for this timer

Label(System.String): A label to use for this particular tick

DividedBy(System.Int64): A number to divide the ticks by (10,000,000 will be seconds)

EnumHelper

Methods

GetDescription

summary: Retrieve the description on the enum, e.g. [Description("Bright Pink")] BrightPink = 2, Then when you pass in the enum, it will retrieve the description

returns: A string representing the friendly name

Parameters:

en(System.Enum): The Enumeration

GetList

summary: Get the text version of the entire Enum list

Parameters:

EnumType(System.Type):

GetEnumfromDescription

summary: Get an enum if given the type and the description string (or name, just to hedge my bets)

Parameters:

Description(System.String):

EnumType(System.Type):

Methods

ArrayJoin

summary: Returns a single string from an array of string, seperated by a specified string

Parameters:

Seperator(System.String):

ArrayObject(System.String[]):

isStringValue(System.Boolean):

ArrayJoin

summary: Returns a single string from an ArrayList of string, seperated by a specified string

Parameters:

Seperator(System.String):

ArrayObject(System.Collections.ArrayList):

isStringValue(System.Boolean):

ArrayJoin

summary: Returns a single string from a DataTable Column seperated by a specified string

Parameters:

Seperator(System.String):

tempDataTable(System.Data.DataTable):

ColumnName(System.String):

isStringValue(System.Boolean):

ArrayJoin

summary: Returns a single string from a particular column in a DataRow array, seperated by a specified string

Parameters:

Seperator(System.String):

tempDataTable(System.Data.DataRow[]):

ColumnName(System.String):

isStringValue(System.Boolean):

ArrayJoin

summary: Returns a single string from a strongly typed list of strings, seperated by a specified string

Parameters:

Seperator(System.String):

ListObject(System.Collections.Generic.List{System.String}):

isStringValue(System.Boolean): Should there be a ' around the values

SSGraphing

summary: Functions to help build graphs

Methods

GetRandomColorList

summary: Returns a list of random colors that will be as far apart as possible Will return different lists every time

Parameters:

NumColors(System.Int32): Number of colors desired

GetRandomColorList

summary: Returns a list of colors that will be as far apart as possible

Parameters:

NumColors(System.Int32): Number of colors desired

CloseColors(System.Collections.Generic.List{System.Drawing.Color}@): A two item list with the two closest colors (for testing)

GetRandomColorList

summary: Runs a single attempt to get a color list Not recomended because of the high probability of failure, use the other version that will retry several times in a row

Parameters:

NumColors(System.Int32):

CloseColors(System.Collections.Generic.List{System.Drawing.Color}@):

EscapeHatchUsed(System.Int32@):

GetStaticColorListSimple

summary: Returns a non-shuffled static list of colors from across the palette

Parameters:

NumColors(System.Int32):

GetStaticColorList

summary: Returns a static list of colors from across the palette Will return the same palette for the same NumColors

Parameters:

NumColors(System.Int32):

GetColorDiff

summary: Gets the difference between two colors in 3D space

Parameters:

Color1(System.Drawing.Color):

Color2(System.Drawing.Color):

ColorListTest1

summary: Runs a test of the color list routine to determine how often it fails which therefore gets our constants

Parameters:

NumColors(System.Int32):

MinDifference(System.Int32):

MaxReached(System.Int32@):

ColorListTest3

summary: Runs ColorListTest1 a bunch of times and reports the results

FindRoundedValues

summary: Find Rounded Min and Max Values for Graphs

Parameters:

MinNumber(System.Double): The minimum number in your dataset

MaxNumber(System.Double): The maximum number in your dataset, 0 will means it will figure it out

MaxSteps(System.Int32): The maximum number of steps allowed

ReturnedMinNumber(System.Double@): The actual calculated min number

ReturnedMaxNumber(System.Double@): The actual calculated max number

ReturnedStep(System.Double@): The actual step value returned

FindRoundedValues

summary: Find Rounded Min and Max Values for Graphs

Parameters:

MinNumber(System.Double): The minimum number in your dataset

MaxNumber(System.Double): The maximum number in your dataset, 0 will means it will figure it out

MaxSteps(System.Int32): The maximum number of steps allowed

ReturnedMinNumber(System.Double@): The actual calculated min number

ReturnedMaxNumber(System.Double@): The actual calculated max number

ReturnedStep(System.Double@): The actual step value returned

ForcePower10(System.Boolean): Force a power of 10 value

SimpleInterpolate

summary: Do a simple linear interpolation of two points given a given X to find the Y for

Parameters:

Point1(System.Drawing.PointF):

Point2(System.Drawing.PointF):

XCoord(System.Single):

SimpleInterpolate

summary: Do a simple linear interpolation of two coordinates given a given X to find the Y for

Parameters:

X1(System.Double):

Y1(System.Double):

X2(System.Double):

Y2(System.Double):

XCoord(System.Double):

SplineInterpolate

summary: Interpolates a spline curve

Parameters:

wavetable(System.Double[]):

Location(System.Double):

ExampleSection

summary: An example configuration section class.

Methods

Static Constructor

summary: Predefines the valid properties and prepares the property collection.

Properties

StringValue

summary: Gets the StringValue setting.

BooleanValue

summary: Gets the BooleanValue setting.

TimeSpanValue

summary: Gets the TimeSpanValue setting.

Properties

summary: Override the Properties collection and return our custom one.

SSDate

summary: Date functions

Methods

IsDate

summary: Is this a date

Parameters:

DateObject(System.Object):

ReturnLaterDate

summary: return the later of two dates

Parameters:

DateObject1(System.Object):

DateObject2(System.Object):

SSDataGridView

summary: A DataGirdView mod that allows copying and pasting

remarks: Based on a disussion at http://www.xtremedotnettalk.com/showthread.php?t=96905

Methods

GetDataGridViewCells

summary: Copy the contents of the selected cells

CopyDataGridViewCellsToClipboard

summary: Copy data from the selected cells

CutDataGridViewCellsToClipboard

summary: Cut data from the selected cells

PasteDataGridViewCellsFromClipboard

summary: Paste data into the selected cells

PasteDataGridViewCells

summary: Internal routine to paste cells

Parameters:

cells(System.Windows.Forms.DataGridViewCell[0:):

maintainColumns(0:]):

TryAndConvertData

summary: Try and convert the data from some random format (probably text) to the format required for the cell

Parameters:

DestinationCell(System.Windows.Forms.DataGridViewCell):

SourceData(System.Object):

SortCells

summary: Organize the contents so that they are from Left to Right And Top to Bottom, as needed

Parameters:

selectedCells(System.Windows.Forms.DataGridViewSelectedCellCollection):

CopyToClipboard

summary: copy the cell information to the clipboard

Parameters:

cells(System.Windows.Forms.DataGridViewCell[0:):

AsciiTableToDataGridViewCellsArray

summary: Convert clipboard ASCII table to an array of DataGridViewCells

Parameters:

table(System.String):

OnMouseDown

summary: Override the mouse down event. Why?

Parameters:

e(System.Windows.Forms.MouseEventArgs):

OnMouseClick

summary: Override the mouse click event to catch when I should pop up the context menu

Parameters:

e(System.Windows.Forms.MouseEventArgs):

OnMenuCut

summary: Cut was clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

OnMenuCopy

summary: Copy was clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

OnMenuPaste

summary: Paste was clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

OnMenuSelectAll

summary: Select All was clicked

Parameters:

sender(System.Object):

e(System.EventArgs):

Properties

AllowCopying

summary: Can we copy things from this grid

AllowCutting

summary: Can we cut things from this grid

AllowPasting

summary: Can we paste things to this grid

Fields

NewContextMenu

summary: The context menu

Copy_OnClickEventHandler

summary: Event handler for when the copy is selected

Cut_OnClickEventHandler

summary: Event handler for when the cut is selected

Paste_OnClickEventHandler

summary: Event handler for when the paste is selected

SelectAll_OnClickEventHandler

summary: Event handler for when the select all is selected

SSArrayConversion

summary: Library class to deal with different array conversions

Methods

String1DToArray

summary: Will convert a CSV string into a 1 double dimensional array

returns: A 1 dimensional double array

Parameters:

InputString(System.String): The input CSV string

String2DToArray

summary: Will convert a CSV and CRLF string into a two dimensional double array

returns: A two dimensional double array

Parameters:

InputString(System.String): A CSV string

Array1DToString

summary: Will convert a one dimensional double array into a CSV string

returns: A CSV string

Parameters:

InputArray(System.Double[]): A one dimensional double array

Array1DToString

summary: Will convert a one dimensional double array into a CSV string

Parameters:

InputArray(System.Double[]): A one dimensional double array

NumTabs(System.Int32): The number of tabs to put before every row

Array1DToString

summary: Will convert a one dimensional double array into a CSV string

returns: A CSV string

Parameters:

InputArray(System.Double[]): A one dimensional double array

NumTabs(System.Int32): The number of tabs to put before every row

IncludeEndingReturn(System.Boolean): Should the ending carriage return be included?

Array1DToString

summary: Will convert a one dimensional int array into a CSV string

returns: A CSV string

Parameters:

InputArray(System.Int32[]): A one dimensional int array

Array1DToString

summary: Will convert a one dimensional int array into a CSV string

Parameters:

InputArray(System.Int32[]): A one dimensional int array

NumTabs(System.Int32): The number of tabs to put before every row

Array1DToString

summary: Will convert a one dimensional int array into a CSV string

returns: A CSV string

Parameters:

InputArray(System.Int32[]): A one dimensional int array

NumTabs(System.Int32): The number of tabs to put before every row

IncludeEndingReturn(System.Boolean): Should the ending carriage return be included?

Array2DToString

summary: Will convert a 2 dimensional double array into a CSV and CRLF string

returns: A CSV string

Parameters:

InputArray(System.Double[0:): A 2 dimensional double array

NumTabs(0:]): The number of tabs to put before every row

Array2DToString

summary: Will convert a 2 dimensional double array into a CSV and CRLF string, with headers

returns: A CSV string

Parameters:

InputArray(System.Double[0:): A 2 dimensional double array

RowHeaders(0:]): The headers for the rows

ColHeaders(System.String[]): The headers for the columns

Measures

summary: A strongly-typed resource class, for looking up localized strings, etc.

Properties

ResourceManager

summary: Returns the cached ResourceManager instance used by this class.

Culture

summary: Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

MeasuresList

summary: Looks up a localized string similar to <?xml version="1.0" encoding="utf-8" ?> <Measurements> <Measurement> <Name>Each</Name> <DefaultUnit>Each</DefaultUnit> <Unit> <Name>Each</Name> <UnitType>Universal</UnitType> <Abbr>Ea.</Abbr> <Multiple>Each</Multiple> <Conversion>1</Conversion> <DefaultConversion>Each</DefaultConversion> </Unit> </Measurement> <Measurement> <Name>Length</Name> <DefaultUnit>Meter</DefaultUnit> <Unit> <Name>Mil</Name> <UnitTyp [rest of string was truncated]";.

NumericEvalException

summary: Custom exception for evaluation errors

Methods

Constructor

summary: Constructor

Parameters:

message(System.String): Message that describes this exception

position(System.Int32): Position within expression where exception occurred

Properties

Column

summary: Zero-base position in expression where exception occurred

Message

summary: Gets the message associated with this exception

NumericEval

summary: Expression evaluator class

Methods

Execute

summary: Evaluates the given expression and returns the result

Parameters:

expression(System.String): The expression to evaluate

TokenizeExpression

summary: Converts a standard infix expression to list of tokens in postfix order.

Parameters:

expression(System.String): Expression to evaluate

ParseNumberToken

summary: Parses and extracts a numeric value at the current position

Parameters:

parser(SSUtils.TextParser): TextParser object

ParseSymbolToken

summary: Parses and extracts a symbol at the current position

Parameters:

parser(SSUtils.TextParser): TextParser object

EvaluateFunction

summary: Evaluates a function and returns its value. It is assumed the current position is at the opening parenthesis of the argument list.

Parameters:

parser(SSUtils.TextParser): TextParser object

name(System.String): Name of function

pos(System.Int32): Position at start of function

ParseParameters

summary: Evaluates each parameter of a function's parameter list and returns a list of those values. An empty list is returned if no parameters were found. It is assumed the current position is at the opening parenthesis of the argument list.

Parameters:

parser(SSUtils.TextParser): TextParser object

EvaluateParameter

summary: Extracts and evaluates a function parameter and returns its value. If an exception occurs, it is caught and the column is adjusted to reflect the position in original string, and the exception is rethrown.

Parameters:

parser(SSUtils.TextParser): TextParser object

paramStart(System.Int32): Column where this parameter started

EvaluateSymbol

summary: This method evaluates a symbol name and returns its value.

Parameters:

name(System.String): Name of symbol

pos(System.Int32): Position at start of symbol

ExecuteTokens

summary: Evaluates the given list of tokens and returns the result. Tokens must appear in postfix order.

Parameters:

tokens(System.Collections.Generic.List{System.String}): List of tokens to evaluate.

GetPrecedence

summary: Returns a value that indicates the relative precedence of the specified operator

Parameters:

s(System.String): Operator to be tested

SerializableDictionary`2

summary: A dictionary object that is serializable

Methods

ReadXml

summary: Read some XML and fill the dictionary

Parameters:

reader(System.Xml.XmlReader):

WriteXml

summary: Write XML containing all the info about the dictionary

Parameters:

writer(System.Xml.XmlWriter):

SSLocalization

summary: My version of a measurement localization system.

remarks: I am not really satisified with it, if you have a better methodology I would love to hear it. Email me at hagie@hagie.net

StandardSystems

summary: The standard measurement systems

Fields

Unknown

summary: Unknown/undefined measurement system

Metric

summary: Metric system

Imperial

summary: Imperial system

Universal

summary: System independant

MeasurementType

summary: A single measurement type (Length, Area, etc)

Methods

ImportMeasurementTypeXML

summary: Import a measurement type (and all sub-units) from an XML node

Parameters:

MeasurementNode(System.Xml.Linq.XElement):

GetMeasurementOfType

summary: Get a measurement of a specific type, with the default unit (throws exception if not found)

returns: The measurement

exception(cref=T:System.ArgumentException): If MeasurementTypeName is not found

Parameters:

MeasurementTypeName(System.String): The name of the measurement

GetMeasurementType

summary: Get a measurement type by name

returns: The measurement

exception(cref=T:System.ArgumentException): If MeasureTypeName is not found

Parameters:

MeasurementTypeName(System.String): The name of the MeasurementType to get

GetFirstUnit

summary: Gets a unit by the name from any measurement type, null if does not exist

returns: The unit, or null if not found

Parameters:

UnitName(System.String):

TryParseUnitLoose

summary: Try and parse a unit, using any method possible

Parameters:

UnitName(System.String):

MakeMeasurementType

summary: Create a new measurement type and return it

Parameters:

TheName(System.String):

TheDefaultUnitName(System.String):

Constructor

summary: Private constructor, use MakeMeasurementType externally

Parameters:

TheName(System.String):

TheDefaultUnitName(System.String):

ToString

summary: Print the name of the MeasurementType

GetUnit

summary: Returns the unit object, null if not found. Will find the singular and plural forms

returns: The unit if found, null if not found

Parameters:

UnitName(System.String): The name of the unit

GetMeasurement

summary: Get a measurement of a specified type

Parameters:

UnitType(SSUtils.SSLocalization.Unit):

DoesContainUnit

summary: Does this measuretype contain this unit

Parameters:

UnitType(SSUtils.SSLocalization.Unit):

Fields

AllMeasurementTypes

summary: The list of all measurement types

Name

summary: The name of the measurement type

DefaultUnitName

summary: The default unit name for this measurement type

DefaultUnit

summary: The default unit for this measurement type

AllUnits

summary: All units for this measurement type

Unit

summary: A measurement unit

Methods

getAs

summary: Get this unit in a different measurement system

Parameters:

NewStandardType(SSUtils.SSLocalization.StandardSystems):

ToString

summary: Return the unit name

InputXML

summary: Input an XML node and create the node type

Parameters:

UnitNode(System.Xml.Linq.XElement):

GetHashCode

summary: Get a hashcode for the unit

Equals

summary: Determine if a unit is the same type as this one

Parameters:

obj(System.Object):

op_Equality

summary: Determine if two units are the same type

Parameters:

M1(SSUtils.SSLocalization.Unit):

M2(SSUtils.SSLocalization.Unit):

op_Inequality

summary: Determine if two units are not the same type

Parameters:

M1(SSUtils.SSLocalization.Unit):

M2(SSUtils.SSLocalization.Unit):

Properties

MeasurementType

summary: The text description of the measurement type (Each, Area, etc)

MeasurementSystem

summary: The standard of the unit (Metric, Imperial, etc)

Name

summary: The name of the unit (Foot, Ohm, Liter, etc)

Abbr

summary: The short label for the unit

AlternateNames

summary: Any alternate names or abbreviations for this unit

PluralName

summary: The plural form of the name (feet vs foot)

ConversionFactor

summary: The conversion factor to the standard unit

MetricImperialEquivalent

summary: The metric equivalant unit

Fields

_measurementType

summary: The text description of the measurement type (Each, Area, etc)

_measurementSystem

summary: The system of the unit (Metric, Imperial, etc)

_Name

summary: The name of the unit (Foot, Ohm, Liter, etc)

_Abbr

summary: The short label for the unit

_alternateNames

summary: Any alternate names or labels for this unit

_PluralName

summary: The plural form of the name (feet vs foot)

_ConversionFactor

summary: The conversion factor to the standard unit

_MetricImperialEquivalent

summary: The metric equivalant unit

Measurement

summary: A single measurement, containing a unit and a value (3 feet, 30 volts, etc)

Methods

Constructor

summary: Simple constructor

Constructor

summary: Create a measurement

Parameters:

value(System.Double): The value of the new measurement

unit(SSUtils.SSLocalization.Unit): The unit of the new measurement

Constructor

summary: Create a measurement

Parameters:

value(System.Double): The value of the new measurement

UnitName(System.String): The unit name of the new measurement

getAbbr

summary: Language sensitive short label

getMultiple

summary: Language sensitive long label

getAs

summary: Get the existing Unit in terms of another unit type

Parameters:

targetUnit(SSUtils.SSLocalization.Unit):

getAs

summary: Get the existing Unit in terms of another standard type (Metric, Imperial, etc)

Parameters:

StandardType(SSUtils.SSLocalization.StandardSystems):

getAs

summary: Get the existing Unit in terms of another unit type

Parameters:

UnitTypeName(System.String):

getValueAs

summary: Get the existing Value in terms of another unit type

Parameters:

TargetUnit(SSUtils.SSLocalization.Unit):

getValueAs

summary: Get the existing Value in terms of another unit type

Parameters:

UnitTypeName(System.String):

setValueAs

summary: Sets the unit, but does a unit type conversion if needed

Parameters:

Value(System.Double):

SourceUnit(SSUtils.SSLocalization.Unit):

InputXMLConvert

summary: A little helper function to help me convert my old data files to use this new class

Parameters:

ParentElement(System.Xml.Linq.XElement):

MeasurementType(System.String):

InputXMLConvert

summary: A little helper function to help me convert my old data files to use this new class

Parameters:

ParentOfParentElement(System.Xml.Linq.XElement):

ParentElementName(System.String):

MeasurementType(System.String):

InputXMLInternal

summary: Create a measurement from an XML node

Parameters:

UnitNode(System.Xml.Linq.XElement):

DefaultUnit(SSUtils.SSLocalization.Unit):

OutputXML

summary: Output an XML node containing the info for this measurement

GetHashCode

summary: Get a unique hashcode for comparison

Equals

summary: Is this measurement equal to another one

Parameters:

obj(System.Object):

ToString

summary: Convert the measurement to a string

ToFullstring

summary: Returns the string representation of the value and unit. Uses the default format string value, but can end with a M for unit name (and plural if value is more than one), and A for abbreviated name.

ToFullstring

summary: Returns the string representation along with the name of the unit (or plural units if more than one)

op_Equality

summary: Are two measurements equal

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_Inequality

summary: Are two measurement not equal

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_LessThan

summary: Is one measurement less than another

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_GreaterThan

summary: Is one measurement greater than another

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_LessThanOrEqual

summary: Is one measurement less than or equal to another

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_GreaterThanOrEqual

summary: Is one measurement greater than or equal to another

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_Addition

summary: Add two measurements together

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_Subtraction

summary: Subtract one measurement from another

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

M2(SSUtils.SSLocalization.Measurement): Measurement2

op_Multiply

summary: Multiply a measurement by a constant value

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

ConstantValue(System.Double): The constant value

op_Division

summary: Divide a measurement by a constant value

Parameters:

M1(SSUtils.SSLocalization.Measurement): Measurement1

ConstantValue(System.Double): The constant value

Parse

summary: Parse a string containing a value and a measurement type.

Parameters:

ValueAndMeasurementName(System.String):

TryParse

summary: Converts the string representation of a measurement to its Measurement number equivalent. A return value indicates whether the conversion succeeded or failed.

Parameters:

ValueAndMeasurementName(System.String):

MeasurementValue(SSUtils.SSLocalization.Measurement@):

Properties

Value

summary: The value of the measurement

MUnit

summary: The unit of the measurement

Fields

_value

summary: Place holder for the numerical value

_unit

summary: Corresponding unit object

Methods

Static Constructor

summary: Static constructor. Will load all the measurement xml file

LoadAllMeasurements

summary: Load all the measurements

QuickConvert

summary: Do a quick convert without creating measurement units.

Parameters:

SourceUnit(Unit): The unit the value is assumed to be in

TargetUnit(Unit): The unit you want to convert it to

SourceValue(System.Double): The actual value to convert

InputXMLMeasurements

summary: Input all measurement types and units from XML

Parameters:

MeasurementsNode(System.Xml.Linq.XElement):

GetUnit

summary: (DEPRECIATED: You should be asking the MeasurementType instead.) Get a unit by name

Parameters:

UnitName(System.String):

GetEquivUnit

summary: Get the equivalant unit in the desired standard type (if possible)

returns: A converted unit

Parameters:

OriginalUnit(Unit): The original unit

DesiredSystem(StandardSystems): The desired measurement system to convert it to

StaticHack

summary: A hack to make sure the static constructor gets called

ParseUnitMeasurements

summary: Take a bound control and parse out the numeric value the user entered, converting it to the measurement expected

Parameters:

sender(System.Object): The bindingsource object, from the Binding.Parse event handler

e(System.Windows.Forms.ConvertEventArgs): The ConvertEventArgs from the Binding.Parse event handler

TheChosenMeasureSystem(StandardSystems): The measurement system the user is currently working in

ParseUnitMeasurements

summary: Take a bound control and parse out the numeric value the user entered, converting it to the measurement expected.

returns: A measurement object suitable for putiing back into the bound object

Parameters:

DataSource(System.Windows.Forms.BindingSource): The original data source used

PropertyName(System.String): The property name in the data source

NewValue(System.String): The new numeric value the user has entered

TheChosenMeasureSystem(StandardSystems): The measurement system the user is currently working in, and we may need to need to convert it from

FormatUnitMeasurements

summary: Format a measurement object into a bound control

Parameters:

e(System.Windows.Forms.ConvertEventArgs): The ConvertEventArgs from the Binding.Format event handler

TheChosenMeasureSystem(StandardSystems): The measurement system the user is currently working in, and we may need to need to convert it to

FormatString(System.String): The format string the bindingsource says it wants

FormatUnitMeasurements

summary: Format a measurement object into a bound control

returns: A string value suitable for putting in the control

Parameters:

OriginalMeasurement(Measurement): The original measurement object to convert

TheChosenMeasureSystem(StandardSystems): The measurement system the user is currently working in, and we may need to need to convert it to

FormatString(System.String): The format string the bindingsource says it wants

UnitTest1

summary: Some simplistic unit testing

FolderBrowser

summary: A class to allow functionality kind of like the openFileDialog, but allow someone to choose a directory instead of a file Kind of like the New Project form inside Visual Studio

Methods

SelectStartupFolder

summary: Take the startup folder, expand the tree all the way out to it, and then select it

Parameters:

FullPathName(System.String):

ExpandFolder

summary: Expand a folder (and sub folders)

Parameters:

TempNode(System.Windows.Forms.TreeNode):

DirNames(System.Collections.Generic.List{System.String}):

FullDirName(System.String):

SetupDrives

summary: Set up all the drives and the first level of directories

treeFolders_BeforeExpand

summary: Each time we expand the node, fill in all the subnodes before you do

Parameters:

sender(System.Object):

e(System.Windows.Forms.TreeViewCancelEventArgs):

FillSubNodes

summary: Fill all the immediate subnodes of the specified node (sub folders of the specified folder)

Parameters:

TempRootNode(System.Windows.Forms.TreeNode):

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

components

summary: Required designer variable.

StringExtensions

summary: Extensions to string objects

Methods

RemoveLastChars

summary: Removes the last X characters from a string

Parameters:

str(System.String):

NumCharsToRemove(System.Int32):

RemoveFirstChars

summary: Removes the first X characters from a string

Parameters:

str(System.String):

NumCharsToRemove(System.Int32):

UpperFirstLetter

summary: Make the first letter after whitespace be uppercase

Parameters:

str(System.String):

WrapString

summary: Wrap a string to a certain maximum characters by inserting CRLFs

Parameters:

InputString(System.String): The input string

MaxChars(System.Int32): The maximum characters per line

ForceWordSplit(System.Boolean): The program will try and avoid splitting words and allow long words to be more than Max, set this to true to force maxchars.

SSString

summary: Misc String functions

Methods

Format

summary: Silly, but it makes the conversions from VB easier

Parameters:

Number(System.Double):

FormatString(System.String):

UpperFirstLetter

summary: Make the first letter after whitespace be uppercase

Parameters:

TextString(System.String):

FindAllIndexes

summary: Returns a list of all substring match locations...demands whitespace, ignores . and ,

Parameters:

OriginalString(System.String):

MatchString(System.String):

SplitOnAllWhiteSpace

summary: Split a string into a substring list on space, tab, period, comma, or line break

Parameters:

OriginalString(System.String):

CombineStrings

summary: Combine a list of string into one string, seperating them with a specified string

Parameters:

Separator(System.String):

StringToJoin(System.String[]):

RemoveAnyChars

summary: Removes any and all specified characters from a string

Parameters:

SourceString(System.String):

ReplaceChars(System.Char[]):

SplitPathFileName

summary: Split a Path And File name string into component parts

Parameters:

PathAndFileName(System.String):

PathName(System.String@):

FileName(System.String@):

SplitPathFileName

summary: Split a Path And File name string into component parts

Parameters:

PathAndFileName(System.String):

RelativePathName(System.String@):

FileName(System.String@):

SplitFileNameAndExtension

summary: Splits a file into into its "main" filename and extension. If no extension, a blank extension is returned

Parameters:

Filename(System.String):

MainFileName(System.String@):

Extension(System.String@):

LowerFirstLetter

summary: Make the first letter after whitespace be lowercase

Parameters:

TextString(System.String):

InsertSpacesBeforeUppercaseChars

summary: Insert a space before every upper case character (e.g. ThisIsATest -> This Is A Test)

Parameters:

SourceString(System.String):

WrapString

summary: Wrap a string to a certain maximum characters

Parameters:

InputString(System.String):

MaxChars(System.Int32):

ForceWordSplit(System.Boolean): The program will try and avoid splitting words and allow it long words to be more than Max, set this to true to force maxchars.

SSMath

summary: Misc Math functions

RoundingTypes

summary: The list of possible ways to round a double

Fields

MidPointHalfEven

summary: When a number is halfway between two others, it is rounded toward the nearest even number.

MidPointAwayFromZero

summary: When a number is halfway between two others, it is rounded away from zero.

MidPointTowardsZero

summary: When a number is halfway between two others, it is rounded toward zero.

AlwaysUp

summary: Always round a number up to the next largest whole number

AlwaysDown

summary: Always round a number down to the next smallest whole number

AlwaysAwayFromZero

summary: Always round a number away from zero to the next whole number

AlwaysTowardsZero

summary: Always round a number towards zero to the next whole number

Methods

ProbabilityOR

summary: Returns the probability of either of 2 independent events happening

Parameters:

Prob1(System.Double):

Prob2(System.Double):

IsNumeric

summary: Is a string a numeric number

Parameters:

TextString(System.String):

IsNumeric

summary: Is an object a numeric number

Parameters:

TestObject(System.Object):

ObjectToDouble

summary: Converts an object to a double, if it can be...else returns 0

Parameters:

TestObject(System.Object):

Max

summary: Get the maximum value of a series of numbers

Parameters:

Numbers(System.Single[]):

Max

summary: Get the maximum value of a series of numbers

Parameters:

Numbers(System.Double[]):

Min

summary: Get the minimum value of a series of numbers

Parameters:

Numbers(System.Double[]):

IsOdd

summary: Determines if a number is odd or even

Parameters:

Number(System.Int32):

Min

summary: Get the minimum value of a series of numbers

Parameters:

Numbers(System.Single):

Min

summary: Get the minimum value of a series of numbers

Parameters:

Numbers(System.Double):

FindSigDigits

summary: Find the first significant digit

Parameters:

theNumber(System.Double):

IsBetween

summary: Is a number between two other values

Parameters:

TestValue(System.Double):

Value1(System.Double):

Value2(System.Double):

GetScaleValue

summary: Find a scale value for a number that is way too big or way too low

Parameters:

MaxValue(System.Double): The maximum value of the scale

Swap

summary: Easily swap to variable

Parameters:

a(System.Double@):

b(System.Double@):

IsClose

summary: Find if two numbers are "close" to each other (within 0.1% of the larger number)

Parameters:

Number1(System.Double):

Number2(System.Double):

IsClose

summary: Find if two numbers are "close" to each other (within a Difference value)

Parameters:

Number1(System.Double):

Number2(System.Double):

Difference(System.Double):

IsInteger

summary: Determines if a string is an integer

Parameters:

NumberString(System.String):

SafeDivide

summary: Divides two numbers, returns null if they can't

Parameters:

Dividend(System.Object):

Divisor(System.Object):

Round

summary: Round a number to the nearest integer given any number of rules

Parameters:

Number(System.Double): The number to round

RoundingType(RoundingTypes): The Rounding type

Round

summary: Round a number given any number of rules (Warning, double has precision errors)

Parameters:

Number(System.Double): The number to round

Precision(System.Double): The precision to round, 1 is to a whole number, 0.1 is to one decimal place

RoundingType(RoundingTypes): The Rounding type

Round

summary: Round a number to the nearest integer given any number of rules

Parameters:

Number(System.Decimal): The number to round

RoundingType(RoundingTypes): The Rounding type

Round

summary: Round a number given any number of rules

Parameters:

Number(System.Decimal): The number to round

Precision(System.Decimal): The precision to round to, 1 is to a whole number, 0.1 is to one decimal place, 10 is the tens place, etc

RoundingType(RoundingTypes): The Rounding type

RoundAwayFromZero

summary: Round a number Away from zero

Parameters:

theNumber(System.Decimal):

RoundTowardZero

summary: Round a number towards zero

Parameters:

theNumber(System.Decimal):

RoundMidPointTowardsZero

summary: Round a number, with the midpoint always going towards zero

Parameters:

theNumber(System.Decimal):

RoundAwayFromZero

summary: Round a number Away from zero

Parameters:

theNumber(System.Double):

RoundTowardZero

summary: Round a number towards zero

Parameters:

theNumber(System.Double):

RoundMidPointTowardsZero

summary: Round a number, with the midpoint always going towards zero

Parameters:

theNumber(System.Double):

RoundingTestNew

summary: Some test cases

remarks - unknown

DegToRad

summary: Convert Degrees To Radians

Parameters:

Degrees(System.Double):

RadToDeg

summary: Convert Radians To Degrees

Parameters:

Radians(System.Double):

ArrayMedian

summary: Resturns the median of a set of doubles

Parameters:

Values(System.Double[]):

ArrayStandardDeviation

summary: Returns the standard deviation for a set of numbers

Parameters:

Values(System.Double[]):

ListStandardDeviation

summary: Returns the standard deviation for a set of numbers

Parameters:

Values(System.Collections.Generic.IEnumerable{System.Int32}):

ListMean

summary: Returns the mean for a set of numbers

Parameters:

Values(System.Collections.Generic.IEnumerable{System.Int32}):

ListMean

summary: Returns the mean for a set of numbers

Parameters:

Values(System.Collections.Generic.IEnumerable{System.Double}):

ArrayMean

summary: Returns the mean for a set of numbers

Parameters:

Values(System.Double[]):

SSDataset

summary: Dataset and database Tools

Methods

GetAllSqlDataSources

summary: Gets a datatable containing every Sql dataource (server) on the network...in theory Rows -> ServerName

GetAllDatabases

summary: Returns a datatable containing every database on a particular server Rows -> database_name

Parameters:

SqlServerName(System.String):

IntegratedLogin(System.Boolean):

UserName(System.String):

Password(System.String):

GetAllTables

summary: Gets all tables in a particular Database Rows -> TABLE_NAME

Parameters:

SqlServerName(System.String):

Database(System.String):

IntegratedLogin(System.Boolean):

UserName(System.String):

Password(System.String):

GetAllFields

summary: Gets all fields in a specified table Good Rows -> TABLE_NAME,COLUMN_NAME,COLUMN_DEFAULT,IS_NULLABLE,DATA_TYPE,CHARACTER_MAXIMUM_LENGTH

Parameters:

SqlServerName(System.String):

Database(System.String):

TableName(System.String):

IntegratedLogin(System.Boolean):

UserName(System.String):

Password(System.String):

FilterDataSet

summary: Changes a datatable by running a SQL filter on it

Parameters:

theDataSet(System.Data.DataSet@):

TableName(System.String):

Filter(System.String):

ColumnEqual

summary: Determine if two columns are the same

Parameters:

A(System.Object):

B(System.Object):

ReturnGreater

summary: Determine if one object is greater

Parameters:

A(System.Object):

B(System.Object):

GetSQLStringFromColumn

summary: Return a SQLstring based on the values of the rows from a particular column

Parameters:

theDataTable(System.Data.DataTable@):

theColumnName(System.String):

isString(System.Boolean):

GetSqlStringFromColumn

summary: Return a SQLstring based on the values of the rows from a particular column

Parameters:

theDataView(System.Data.DataView@):

theColumnName(System.String):

isString(System.Boolean):

BooleanToDB

summary: Converts a boolean to a DB 1/0 value

Parameters:

Value(System.Boolean):

DBToBoolean

summary: Converts a 1/0 value to a boolean

Parameters:

value(System.Int32):

SSTextBox

summary: A textbox that includes cut, copy, paste, select all, undo, and redo

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

components

summary: Required designer variable.

SSControlMods

summary: A class introducing various changes to various controls

DataGridViewComboBoxItemColumn

summary: An upgrade to the Combox Box Column to allow databinding to a list of objects

remarks: This comes from
http://www.devnewsgroups.net/group/microsoft.public.dotnet.framework.windowsforms/topic59949.aspx

DataGridViewComboBoxItemCell

summary: An upgrade to the Combox Box Column to allow databinding to a list of objects

remarks: This comes from
http://www.devnewsgroups.net/group/microsoft.public.dotnet.framework.windowsforms/topic59949.aspx

Tab

summary: A class to add some "Hide" and "Show" functionality to tabpages Code got from
http://dotnetrix.co.uk/controls.htm

Methods

HideTabPage

summary: "Hide" a tabpage by removing it (if it exists)

Parameters:

theTabControl(System.Windows.Forms.TabControl@):

tp(System.Windows.Forms.TabPage@):

ShowTabPage

summary: "Show" a tabpage by adding it

Parameters:

theTabControl(System.Windows.Forms.TabControl):

tp(System.Windows.Forms.TabPage):

ShowTabPage

summary: "Show" a tabpage by adding it at a certain location

Parameters:

theTabControl(System.Windows.Forms.TabControl):

tp(System.Windows.Forms.TabPage):

index(System.Int32):

InsertTabPage

summary: Internal insert command

Parameters:

theTabControl(System.Windows.Forms.TabControl):

tabpage(System.Windows.Forms.TabPage):

index(System.Int32):

SwapTabPages

summary: Internal swap command

Parameters:

theTabControl(System.Windows.Forms.TabControl):

tp1(System.Windows.Forms.TabPage):

tp2(System.Windows.Forms.TabPage):

SSToolStripTextBox

summary: An attempt to make the ToolStripTextBox work something approaching sanely. Validating, validated, TextChanged; all of its events are a mess.

Methods

OnEnter

summary: OnEnter never fires

Parameters:

e(System.EventArgs):

OnLeave

summary: OnLeave never fires

Parameters:

e(System.EventArgs):

Fields

_internalTextChanged

summary: Keep a check of the internal text status

OldValue

summary: If we abort a validation, we need to put the old value back, so keep it

ToolStripRadioButtonMenuItem

summary: A radio button control for a tool strip menu

remarks: Originally from http://msdn.microsoft.com/en-us/library/ms404318.aspx But heavily modified by Scott Hagie

Methods

Constructor

summary: Basic constructor

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

Constructor

summary: Constructor

Parameters:

image(System.Drawing.Image): The menu image

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

image(System.Drawing.Image): The menu image

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

image(System.Drawing.Image): The menu image

onClick(System.EventHandler): The event handler if it is clicked

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

image(System.Drawing.Image): The menu image

onClick(System.EventHandler): The event handler if it is clicked

name(System.String): The name of the menu item

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

image(System.Drawing.Image): The menu image

dropDownItems(System.Windows.Forms.ToolStripItem[]): The list of subitems that will drop down

Constructor

summary: Constructor

Parameters:

text(System.String): The menu text

image(System.Drawing.Image): The menu image

onClick(System.EventHandler): The event handler if it is clicked

shortcutKeys(System.Windows.Forms.Keys): The shortcut keys associated with the menu item

Initialize

summary: Called by all constructors to initialize CheckOnClick.

OnCheckedChanged

summary: Override for the check changed event

Parameters:

e(System.EventArgs):

OnClick

summary: If the menu item has been clicked

Parameters:

e(System.EventArgs):

OnPaint

summary: Let the item paint itself, and then paint the RadioButton where the check mark is normally displayed.

Parameters:

e(System.Windows.Forms.PaintEventArgs):

OnMouseEnter

summary: Event handler for when the mouse enters the control

Parameters:

e(System.EventArgs):

OnMouseLeave

summary: Event handler for when the mouse leaves the control

Parameters:

e(System.EventArgs):

OnMouseDown

summary: Event handler for when the mouse button is down

Parameters:

e(System.Windows.Forms.MouseEventArgs):

OnMouseUp

summary: Event handler for when the mouse button is back up again

Parameters:

e(System.Windows.Forms.MouseEventArgs):

OnOwnerChanged

summary: When OwnerItem becomes available, if it is a ToolStripMenuItem with a CheckOnClick property value of true, subscribe to its CheckedChanged event.

Parameters:

e(System.EventArgs):

OwnerMenuItem_CheckedChanged

summary: When the checked state of the parent item changes, repaint the item so that the new Enabled state is displayed.

Parameters:

sender(System.Object):

e(System.EventArgs):

Properties

Enabled

summary: Enable the item only if its parent item is in the checked state and its Enabled property has not been explicitly set to false.

Fields

InternalScrewingWithTheCheckState

summary: If we are screwing with the check state internally, we will set a flag

mouseHoverState

summary: Is the mouse on top of the control

mouseDownState

summary: Is the mouse button currently down

Methods

DataGridViewAdvancedParsing

summary: Will try and parse a DataGridView with a sub-object column. Should be wired to a CellParsing event

remarks: May be incomplete for all data types, needs some further testing Use as follows: ColumnObject = new DataGridViewTextBoxColumn() ColumnObject.DataPropertyName = "Property_That_Returns_A_SubObject"; ColumnObject.Name = "Name_Of_Property_In_SubObject"; DataGridViewObject.Columns.Add(ColumnObject);

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellParsingEventArgs):

DataGridViewAdvancedFormatting

summary: Will try and format a DataGridView with a sub-object column. Should be wired to a CellFormatting event

remarks: May be incomplete for all data types, needs some further testing Use as follows: ColumnObject = new DataGridViewTextBoxColumn() ColumnObject.DataPropertyName = "Property_That_Returns_A_SubObject"; ColumnObject.Name = "Name_Of_Property_In_SubObject"; DataGridViewObject.Columns.Add(ColumnObject);

Parameters:

sender(System.Object):

e(System.Windows.Forms.DataGridViewCellFormattingEventArgs):

BindingArrayList`1

summary: An arraylist with a fixed size, that will make sure the objects inside keep the same numbering and order Make sure and use both ChangeSize and InternalIndexProperty

Methods

Constructor

summary: Create a new BindingArrayList

Constructor
Constructor

summary: Create a new BindingArrayList and tell it what the name of the underlying object's index property

Parameters:

TheInternalIndexProperty(System.String): The name of the property in the underlying object that should be kept updated if its index changes

Constructor

summary: Create a new BindingArrayList and tell it what the name of the underlying object's index property and the default size

Parameters:

DefaultSize(System.Int32): The default (1-based) size

TheInternalIndexProperty(System.String): The name of the property in the underlying object that should be kept updated if its index changes

Count

summary: A real count

ChangeSize

summary: Change the array size

Parameters:

NewSize(System.Int32): The 1-based new size

CreateNewItem

summary: Creates a new item and assigns it's indexed property automatically

Parameters:

Index(System.Int32):

SetSize

summary: Change the array size

Parameters:

NewSize(System.Int32):

InsertItem

summary: Insert should never work

Parameters:

index(System.Int32):

item(`0):

RemoveItem

summary: Removing an item is actually just replacing it with a blank item

Parameters:

index(System.Int32):

SetItem

summary: Set an item

Parameters:

index(System.Int32):

item(`0):

BindingArrayList_PropertyChanged

summary: Event handler for when one of the objects in the list has changed

Parameters:

sender(System.Object):

e(System.ComponentModel.PropertyChangedEventArgs):

FindPropertyDescriptor

summary: Find the property descriptor for the named index property

Parameters:

property(System.String):

GetPropertyValue

summary: Get the value of a property via reflection

Parameters:

MyObject(System.Object):

property(System.String):

SetPropertyValue

summary: Set the value of a property via reflection

Parameters:

value(System.Int32):

property(System.String):

MyObject(System.Object):

Properties

InternalIndexProperty

summary: Optional: If you want the BindingingArrayList to tell the objects inside what their current index is, set the name of the objects index property here

Fields

_numItems

summary: The number of items it should have

_internalIndexProperty

summary: Optional: If you want the BindingingArrayList to tell the objects inside what their current index is, set the name of the objects index property here

_addingNewItem

summary: Are we adding a new item

SmartDateTimePicker

summary: A supposedly smarter datetime picker. I am not happy with this, I am going to have to make my own

Methods

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Fields

components

summary: Required designer variable.

SSException

summary: A class to log exceptions to database or file

Methods

Log

summary: Log exception to database or file

Parameters:

Ex(System.Exception): Root exception

Log

summary: Log exception to database or file

Parameters:

Ex(System.Exception): Root exception

InformUser(System.Boolean): Inform user of an error?

Log

summary: Log exception to database or file

Parameters:

ErrorMessage(System.String): Error message name

Log

summary: Log exception to database or file

Parameters:

ErrorMessage(System.String): Error message name

InformUser(System.Boolean): Inform user of an error?

Log

summary: Log exception to database or file

Parameters:

ErrorMessage(System.String): Error message name

Ex(System.Exception): Root exception

Log

summary: Log exception to database or file

Parameters:

ErrorMessage(System.String): Error message name

Ex(System.Exception): Root exception

InformUser(System.Boolean): Inform user of an error?

CreateXmlErrorDocument

summary: Internal function to create an error document with embedded information

Parameters:

ErrorMessage(System.String):

Ex(System.Exception):

LogExceptionEasy

Parameters:

Ex(System.Exception):

LogExceptionRecurse

summary: Unused Internal function to recursively log exceptions and inner exceptions

remarks: UNUSED

Parameters:

ex(System.Exception):

sw(System.IO.StreamWriter):

level(System.Int32):

CreateXmlExceptionNode

summary: Internal function to recursively log exceptions and inner exceptions

Parameters:

ex(System.Exception):

RootDoc(System.Xml.XmlDocument):

InnerException(System.Boolean):

SaveErrorToFile

summary: Save the error to a file in the default directory

Parameters:

ErrorDoc(System.Xml.XmlDocument):

UpgradeException

summary: An example custom exception from http://codebetter.com/blogs/karlseguin/archive/2008/05/29/foundations-of-programming-pt-8-back-to-basics-exceptions.aspx

BindableListBox

summary: A listbox that will contain a list of objects, and bind that list to a datasource of type IList, so that when you change the value of a specific property, it will keep them in sync (Honestly, I have no idea why this is useful at this point)

BindingList

summary: The binding list used for this listbox

Methods

Constructor

summary: Create a new blank binding list

Constructor

summary: Create a new binding list with all the info needed

Parameters:

dataSource(System.Collections.IList): The underlying list of objects used to fill the list

childType(System.Type): The type of data the list contains

dataMember(System.String): The name of the data member in the ChildType object that will be used to fill the list

Fields

DataSource

summary: The underlying list of objects used to fill the list

ChildType

summary: The type of data the list contains

DataMember

summary: The name of the data member in the ChildType object that will be used to fill the list

Methods

Constructor

summary: Create a new BindableListBox

BindableListBox_ListChanged

summary: If the list has been changed, we need to update the listbox

Parameters:

sender(System.Object):

e(System.ComponentModel.ListChangedEventArgs):

OnValidated

summary: After the data has been validated, we need to transfer the data back to the underlying object

Parameters:

e(System.EventArgs):

TransferFromObjectToListBox

summary: Transfer the info from the list of objects to the listbox

TransferFromListBoxToObject

summary: Transfer the information from the list box to the IList of objects

Dispose

summary: Clean up any resources being used.

Parameters:

disposing(System.Boolean): true if managed resources should be disposed; otherwise, false.

InitializeComponent

summary: Required method for Designer support - do not modify the contents of this method with the code editor.

Properties

DataBindingList

summary: The Bindinglist containing all the items used to fill the list

SelectedItems

summary: Hide the base SelectedItems method

Fields

CurrentlyTransfering

summary: We are currently transfering information between the bound source and the listbox

CurrentDataBindingList

summary: The current list of items

components

summary: Required designer variable.

TextProgressBar

summary: A version of the progress bar that allows text to be written on top of it

Methods

Constructor

summary: Basic constructor

Properties

Alignment

summary: Gets or sets text alignment information on the vertical plane.

FormatFlags

summary: Gets or sets a System.Drawing.StringFormatFlags enumeration that contains formatting information.

LineAlignment

summary: Gets or sets the line alignment on the horizontal plane.

Trimming

summary: Gets or sets the System.Drawing.StringTrimming enumeration for this System.Drawing.StringFormat object.

Text

summary: The text to display

StringFormat

summary: Encapsulates text layout information (such as alignment, orientation and tab stops) display manipulations (such as ellipsis insertion and national digit substitution) and OpenType features.

Font

summary: The font for the text

SortableBindingList`1

summary: A sortable binding list

remarks - unknown

PropertyComparer`1

summary: Comparing properties

remarks: The following code contains code implemented by Rockford Lhotka:
msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet01272004.asp"
href="http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/dnadvnet/html/vbnet01272004.asp">http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/dnadvnet/html/vbnet01272004.asp

Methods

Constructor

summary: Create a new PropertyComparer

Parameters:

property(System.ComponentModel.PropertyDescriptor):

direction(System.ComponentModel.ListSortDirection):

Compare

summary: Compare two objects based on a property

Parameters:

xVal(`1):

yVal(`1):

Equals

summary: Are two objects equal

Parameters:

xVal(`1):

yVal(`1):

GetHashCode

summary: Get a unique hashcode for an object

Parameters:

obj(`1):

CompareAscending

summary: Compare two property values of any type

Parameters:

xValue(System.Object):

yValue(System.Object):

CompareDescending

summary: Compare two items

Parameters:

xValue(System.Object):

yValue(System.Object):

GetPropertyValue

summary: Get the value of a property

Parameters:

value(`1):

property(System.String):

PropComparePair

summary: A tuple containg the property and direction for a sort

Methods

Constructor

summary: Constructor

Parameters:

property(System.ComponentModel.PropertyDescriptor):

direction(System.ComponentModel.ListSortDirection):

Properties

TheProperty

summary: The property

Direction

summary: The direction

MultiPropertyComparer`1

summary: Compare two objects based on multiple properties as keys

remarks: The following code contains code implemented by Rockford Lhotka:
msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet01272004.asp"
href="http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/dnadvnet/html/vbnet01272004.asp">http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/dnadvnet/html/vbnet01272004.asp

Methods

Constructor

summary: Constructor

Parameters:

PropPairs(System.Collections.Generic.List{SSUtils.SortableBindingList{`0}.PropComparePair}):

Compare

summary: Compare two objects

Parameters:

xVal(`1):

yVal(`1):

Equals

summary: Determines if two objects are equal

Parameters:

xVal(`1):

yVal(`1):

GetHashCode

summary: Get a unique hashcode

Parameters:

obj(`1):

CompareAscending

summary: Compare two property values of any type

Parameters:

xValue(System.Object):

yValue(System.Object):

CompareDescending

summary: Compare if two objects are in descending order

Parameters:

xValue(System.Object):

yValue(System.Object):

GetPropertyValue

summary: Get a property value

Parameters:

value(`1):

property(System.String):

Methods

Sort

summary: Do a sort on the default property

Sort

summary: Do a sort on a named property

Parameters:

property(System.String):

Sort

summary: Do a sort on a named property in a named direction

Parameters:

property(System.String):

direction(System.ComponentModel.ListSortDirection):

Sort

summary: Sort items based on a list of properties

Parameters:

PropSortList(System.Collections.Generic.List{SSUtils.SortableBindingList{`0}.PropComparePair}):

ApplySortCore

summary: Do a sort on a named property in a named direction

Parameters:

property(System.ComponentModel.PropertyDescriptor):

direction(System.ComponentModel.ListSortDirection):

ApplySortCore

summary: Sort items based on a list of properties

Parameters:

PropSortList(System.Collections.Generic.List{SSUtils.SortableBindingList{`0}.PropComparePair}):

Properties

IsSortedCore

summary: Are we currently sorted

Fields

_isSorted

summary: Are we currently sorted

_dir

summary: What direction are we sorting

_sort

summary: What is the property we are sorting on

SSThreading

summary: Some threading code

Safe

summary: A safe way to lock more than object with a timeout

remarks: From http://blog.decarufel.net/2009/06/how-to-implement-lock-with-timeout.html

SSFilePermissions

summary: Permission info

PermissionInfo

summary: Permission info for a file

Methods

ReadPermissionRules

summary: Read permission rules for this classes file

Constructor

summary: Creates a new permission entry for a file

Constructor

summary: Creates a new permission entry for a file

returns: Bool

Parameters:

TheFilePath(System.String):

TheFileName(System.String):

CanRead

summary: Does the specified user have Read Permission

returns: Bool

Parameters:

UserName(System.String):

CanWrite

summary: Does the specified user have Write Permission

returns: Bool

Parameters:

UserName(System.String):

CanExecute

summary: Does the specified user have execute Permission

returns: Bool

Parameters:

UserName(System.String):

CanModify

summary: Does the specified user have Modify Permission

returns: Bool

Parameters:

UserName(System.String):

IsAllowed

summary: Finds a certain permission by a certain user

returns: Bool

Parameters:

UserName(System.String):

Eval

summary: A class to "Eval" an expression string. Will have full access to local variable Limitations: No nested members, no extension methods Works: Property; ObjectInstance1.Field; ObjectInstance1.SubObject2.Method("test1"); StaticMethod(1,5,"test"); DateTime.Now().ToString("F"); Does not work: method1(method2());

Methods

Evaluate

Parameters:

CallingObject(System.Object):

ExpressionString(System.String):

GetParsedValue

summary: Parse a value string, and return the parsed value (or exception if failed) Works: Property; ObjectInstance1.Field; ObjectInstance1.SubObject2.Method("test1"); StaticMethod(1,5,"test"); DateTime.Now().ToString("F"); Does not work: method1(method2()); Method("()");

Parameters:

CallingObject(System.Object):

InputString(System.String):

ProcessMember

summary: Process a single member (method, field, or property)

Parameters:

CurrentObject(System.Object):

MemberName(System.String):

ParamValues(System.String[]):

ProcessMemberAsStaticReferencedType

summary: Assume the Member is a referenced static class, return the Type ref if so (Exception if not)

Parameters:

ClassName(System.String):

ParamValues(System.String[]):

ProcessOperandAsExtensionMethod

summary: Assume the operand is an extension method, find a matching entry (or exception if none)

Parameters:

CurrentObject(System.Object):

Operand(System.String):

ParamValues(System.String[]):

GetAllAssemblies

summary: (Recursive) Get all assemblies referenced (or referenced by reference) across the entire project

Parameters:

StartingAssembly(System.Reflection.Assembly):

AssemblyList(System.Collections.Generic.List{System.Reflection.Assembly}@):

IsMethodParamMatch

summary: Does this method's parameters match a given set of parameters

Parameters:

MethodToMatch(System.Reflection.MethodInfo):

ParamValues(System.Object[]):

NewParamValues(System.Object[]@):

GetExtensionMethods

summary: UNUSED: Get all extension methods that might match a type

Parameters:

assembly(System.Reflection.Assembly):

extendedType(System.Type):

GetExtensionMethods

summary: UNUSED: Get all extension methods that might match a certain type with a certain name

Parameters:

assembly(System.Reflection.Assembly):

extendedType(System.Type):

Name(System.String):

Array2D

summary: A wrapper for a 2D double array that will preserve its contents when you resize in either direction

Methods

Constructor

summary: Create a new Array2D with no information

Constructor

summary: Create a new Array2D by giving the size

Parameters:

Dim1(System.Int32): The zero-based dimension 1

Dim2(System.Int32): The zero-based dimension 2

Constructor

summary: Create a new Array2D by giving the size and default value

Parameters:

Dim1(System.Int32): The zero-based dimension 1

Dim2(System.Int32): The zero-based dimension 2

TheDefaultValue(System.Double):

Constructor

summary: Create a new Array2D by giving it another array

Parameters:

OriginalArray(System.Double[0:):

Constructor

summary: Create a new Array2D from a comma(and \r\n)-delimited CSV string

Parameters:

CSVString(System.String):

SetValue

summary: Set a value

Parameters:

Dim1(System.Int32): The zero-based dimension 1

Dim2(System.Int32): The zero-based dimension 2

value(System.Double):

GetValue

summary: Get a value

Parameters:

Dim1(System.Int32): The zero-based dimension 1

Dim2(System.Int32): The zero-based dimension 2

ReSizeArrays

summary: Do a resize of the array

Resize2DArray

summary: Copy an old 2D array to a new 2D array, preserving the sizes

Parameters:

OldArray(System.Double[0:):

NewDim1(0:]@): Zero based dimension 1

NewDim2(System.Int32): Zero based dimension 2

GetUpperBound

summary: Get the upperbound of the array

Parameters:

Dimension(System.Int32):

ToString

summary: Output a CSV string of the array

TestArrayResize

summary: Test the array sizing and the Default values

TestCSV

summary: Test the CSV stuff

Properties

InternalArray

summary: The internal array. Be careful with messing with it, you may destabilize the internal data if you do

Item

summary: Indexer...Get the array value

Parameters:

Dim1(System.Int32): The zero-based dimension 1

Dim2(System.Int32): The zero-based dimension 1

DefaultValue

summary: The default value for new cells

Dimension1

summary: 0-based count

Dimension2

summary: 0-based count

Fields

_internalArray

summary: The internal array

_defaultValue

summary: The default value to use for new elements

DisplayType

summary: How to display information

SSUI

summary: Class to help with common user input and output scenarios

Methods

CenterControl

summary: Return the value to use for left or top so that the control is centered in the parent object

Parameters:

ParentTopOrLeft(System.Int32):

ParentWidthOrHeight(System.Int32):

ChildWidthOrHeight(System.Int32):

GetFocusedControl

summary: Go through all the sub-controls and sub-sub-controls of a control to find if anything has focus

returns: The control, or null if no control has focus

Parameters:

Controls(System.Windows.Forms.Control.ControlCollection):

CopyFromFormControlToClipboard

summary: Find the focused control on the form and copy it to the clipboard

Parameters:

MainForm(System.Windows.Forms.Form):

CopyFromFormControlToClipboard

summary: copy a control into the clipboard

Parameters:

FocusedControl(System.Windows.Forms.Control):

PasteToFormControlFromClipboard

summary: Find the focused control on the form and paste data into it

Parameters:

MainForm(System.Windows.Forms.Form):

PasteToFormControlFromClipboard

summary: Paste data from the clipboard into a control

Parameters:

FocusedControl(System.Windows.Forms.Control):

CutFromFormControlToClipboard

summary: Find the focused control on the form and cut it to the clipboard

Parameters:

MainForm(System.Windows.Forms.Form):

CutFromFormControlToClipboard

summary: Cut a control and put it to the clipboard

Parameters:

FocusedControl(System.Windows.Forms.Control):

SelectAllInFormControl

summary: Finds the currently selected control and selects all text in it

Parameters:

MainForm(System.Windows.Forms.Form):

SelectAllInFormControl

summary: Select all text in a control

Parameters:

FocusedControl(System.Windows.Forms.Control):

ParseArea

summary: Parse an input string that is in the format of an area of square feet

Parameters:

OrigNumber(System.String): The string to parse

ParseArea

summary: Parse an input string that is in the format of an area of square feet

Parameters:

OrigNumber(System.String): The string to parse

WasError(System.Boolean@): Out: Was there an error in parsing

ParseCurrency

summary: Parse a number that is in a (US) currency format

Parameters:

OrigNumber(System.String): The string to parse

ParseCurrency

summary: Parse a number that is in a (US) currency format

Parameters:

OrigNumber(System.String): The string to parse

WasError(System.Boolean@): Out: Was there an error in parsing

OutputCurrency

summary: Output a number that is in a specified currency format

Parameters:

OrigNumber(System.Double): The value to format

theDisplayType(SSUtils.DisplayType): How to display the data

theSignificantDigits(System.Int32): The number of significant digits to show

ParseNumber

summary: Parse a number

Parameters:

OrigNumber(System.String):

ParseNumber

summary: Parse a number that could have a larger quantity indicator, Such as thousand, or million, or mill

Parameters:

OrigNumber(System.String):

OutputNumberSimple

summary: Create a string that includes commas and a specified number of sig digits

Parameters:

OrigNumber(System.Double):

SigDigits(System.Int32):

OutputNumber

summary: Create a string that includes commas and a specified number of sig digits

Parameters:

OrigNumber(System.Double):

theDisplayType(SSUtils.DisplayType):

SigDigits(System.Int32):

OutputNumber

summary: Create a string that includes commas and a specified number of sig digits And will also add in thousand, million, etc if required

Parameters:

OrigNumber(System.Double):

theDisplayType(SSUtils.DisplayType):

SanityCheck

summary: Does a sanity check on imported strings, makes sure it is a number, and not higher or lower than expected

returns: a string error message if error, "" if not

Parameters:

NumberString(System.String):

LowValue(System.Double):

HighValue(System.Double):

SanityRangeCheck

summary: Not all that sure why this is here :)

Parameters:

LowValue(System.Double):

HighValue(System.Double):

ParsePercent

summary: Parse a string that is a perrcent value

Parameters:

PercentNumber(System.String):

SSGraphics

summary: Graphics routines

Methods

ReturnScaledAndRotatedBitmap

summary: Return a scaled bitmap that has been rotated by a certain number of degrees

Parameters:

SourceBitmap(System.Drawing.Bitmap):

RotationDeg(System.Double):

BackgroundColor(System.Single):

FindRectThatWillFitWithinRotatedRect

summary: Find the largest rectangle that will fit within a larger, but rotated, rectangle

Parameters:

MasterRectangle(System.Drawing.RectangleF):

RotationDeg(System.Single):

SlaveAspectRatio(System.Single):

Translate

summary: Do a translate of a rectangle

Parameters:

SrcSideWidth(System.Double):

SrcSideHeight(System.Double):

DestSideHeight(System.Double):

DestSideWidth(System.Double):

RotationRad(System.Double):

RescalePercentage

summary: Scale a rectangle by a percentage

Parameters:

OriginalSize(System.Drawing.SizeF):

MaxBounds(System.Drawing.SizeF):

OriginalScalePercentage(System.Single):

DoesMaintainAspectRatio(System.Boolean):

UseMaxSize(System.Boolean):

WidthScalePercent(System.Single@):

HeightScalePercent(System.Single@):

TextParser

summary: A text parser, allowing you easy access to a parsed string

Methods

Constructor

summary: constructor

Parameters:

TextString(System.String):

GetBaseStringCopy

summary: Returns a copy of the base string

Peek

summary: Look at the current position

MoveNext

summary: Move ahead one space

Extract

summary: Return a substring from any position, without moving the index

Parameters:

start(System.Int32):

end(System.Int32):

MovePastWhitespace

summary: Move the position to the next whitespace character (or end)

CharIsWhiteSpace

summary: See if a character is whitespace

Parameters:

ValueToTest(System.Char):

GetNext

summary: Gets the next character, and increments the position (Peek + MoveNext)

GetNext

summary: Gets the next X characters, and increments the position

Parameters:

NumChars(System.Int32):

Properties

EndOfText

summary: Are we at the end of the text

Position

summary: The current position

Fields

_baseString

summary: The base string

_position

summary: The current position

SSParsing

summary: Classes for helping with various parsing jobs

InfixToPostfix

summary: A converter to convert Infix notation to postfix(and run it)

remarks: All of this is from http://community.bartdesmet.net/

MathOp

summary: The math operators

Fields

Add

summary: +

Sub

summary: -

Mul

summary: *

Div

summary: /

Mod

summary: Mod

MathOpOrVal

summary: An object that is a math operation or a value

remarks: In C/C++ one would typically use a union to represent this token, but we stick with two nullable fields wrapped in a class.

Methods

Constructor

summary: Create a new object, using an op

Parameters:

op(SSUtils.SSParsing.InfixToPostfix.MathOp):

Constructor

summary: Create a new object, using a value

Parameters:

value(System.Double):

ToString

summary: The text representation

Fields

Op

summary: If it is an Op, this is the op

Value

summary: If it is a value, this is the value

ParseException

summary: A have a trivial parser exception object:

TreeNode

summary: A node in the parsed tree

Methods

TestPostFix

summary: Some test code

Execute

summary: Execute the result of the tree node (INT ONLY, no doubles)

Parameters:

root(TreeNode):

Methods

ParseNamesAndPhones

summary: Try and parse a string that is assumed to be a name and phone number

Parameters:

StringToParse(System.String):

ParsePhones

summary: Try to parse a string that is assumed to be a phone number

Parameters:

StringToParse(System.String):

ParseNames

summary: Try to parse a string assumed to be a name

Parameters:

StringToParse(System.String):

ParseCityStateAndCountry

summary: Try and parse a string assumed to be a city, state, and country

Parameters:

OriginalString(System.String):

SSSplitContainer

summary: An override of the SplitContainer class that has one extra event to tell when a SplitContainer is in the middle of resizing

OnResizingBeginHandler

summary: The delete to handle the ResizingBegin event

Parameters:

sender():

e():

Methods

OnResize

summary: When the OnResize is called, pass it up as the ResizingBegin event

Parameters:

e(System.EventArgs):

Events

ResizingBegin

summary: The container is currently resizing

SSMultiListBox

summary: A list box that bounds both the data for the list, as well the data for the list choice.

Properties

SelectedListDataSource

summary: The bound Select List

Appendix B – Environmental Loss Computation in PACT

Environmental Loss Computation-In PACT

R. Hamburger
17 February 2017

For each Fragility Specification "j" and each Damage State "k", the Environmental Products team provides:

1. A median value of the Embodied Carbon $\widehat{EC}_{j,k}$ in kg of CO_2
2. A median value of Embodied Energy $\widehat{EE}_{j,k}$ in Megajoule (MJ)

per fragility Unit.

As an example, for Concrete Walls, up to 12-in wide, a fragility unit is an 8' x 8' 2all panel. Damage states are:

Damage State	Description	$\widehat{EC}_{j,k}$ kg, CO_2	$\widehat{EE}_{j,k}$ MJ
DS1	Cracking requiring epoxy injection	15	25
DS2	Cracking and spalling, requiring injection, chipping of damaged concrete and recasting portion of wall with repair concrete	30	50
DS3	Extensive cracking, spalling and yielding/buckling or reinforcing. Demolish wall and reconstruct	1000	250

(note numbers are fictitious for illustration only)

The median EC and EE are evaluated as:

1. Derive the non-labor and non-energy cost of the repair (Material cost only)
2. Distribute this cost into each of the several EIO economic sectors
3. Apply the EIO rate for each sector per $ times the $ in each sector in the repair cost
4. Sum the resulting EC and EE values to obtain the medians

Dispersion is not directly discernible from the EIO database, therefore, dispersion is taken as the dispersion in repair cost, enriched to account for additional uncertainty in the database. The enrichment is denoted here as β_{EP} for environmental products dispersion.

The method for implementing calculation of these impacts in PACT is presented below, as are screen shots of the accompanying out put screens.

Calculation Procedure For PACT Implementation

In this procedure, the fragility database is modified to include the following quantities for each fragility specification and each damage state:

- EE_k – the median embodied energy in MJ for one fragility unit.

- β_{EEk} – dispersion in embodied energy consequence taken as the repair cost dispersion for damage state k, enriched slightly to account for uncertainty in computation of EE as a function of repair cost. Note enrichment is not large because it is recognized that the repair cost dispersion includes some labor efficiency function which is not correlated with

EE. Thus in reality the repair cost dispersion is de-rated to remove the labor uncertainty than inflated to recognize the EE uncertainty relative to material cost. It may be reasonable just to use the β_{RC}.

- EC_k - the median embodied energy in kg for one fragility unit.

- β_{ECk} – the dispersion in embodied carbon consequence taken as the repair cost dispersion for damage state k, enriched slightly to account for uncertainty in computation of EC as a function of repair cost. Seem discussion for β_{EE} above.

1. For each realization “i”, fragility group “j” and damage state “k” do the following:

    a. Select random numbers n_{rc}, n_{rt} (in addition to other random numbers used in the process to determine damage state, hour of day, etc), where n_{rc} is used to determine the number of standard deviations above (or below) the median of repair cost, and n_{rt} is the number of standard deviations above (or below) the median repair time. Note that PACT already does this.

    b. Compute the repair cost, $RC_{i,j},k$; repair time $RT_{i,j,k}$, casualties, Red Tag and other consequences, as before.

    c. Compute the $EE_{i,j,k}$ and ECi,j,k using the same routine as fro RC and RT, except that n_{rc} is used with EE_k, β_{EEk}, EC_k, and $\beta_{ECk.}$

2. Sum the $EE_{i,j,k}$ and $EC_{i,j,k}$ over all damage states “k” and all fragility groups “j” to obtain the realization EE_i and EC_i.

3. Plot the EE and EC outputs for all fragilities, just as is done for RC, RT, etc.

PACT Output

The main output page for environmental impacts will be a dual screen format, similar to that for Casualties, which shows Fatalities and Injuries, with the exception that this page will show Embodied Carbon and Embodied Energy, respectively on the left and right sides (see Figure 1). Secondary output, showing the distribution of either impact by fragility specification and realization, is shown in Figure 2 for Embodied Carbon. Embodied energy will have similar output. Data Drill Down and Exports will be similar to the other impacts. Modify the drill down page as shown on Figure 3.

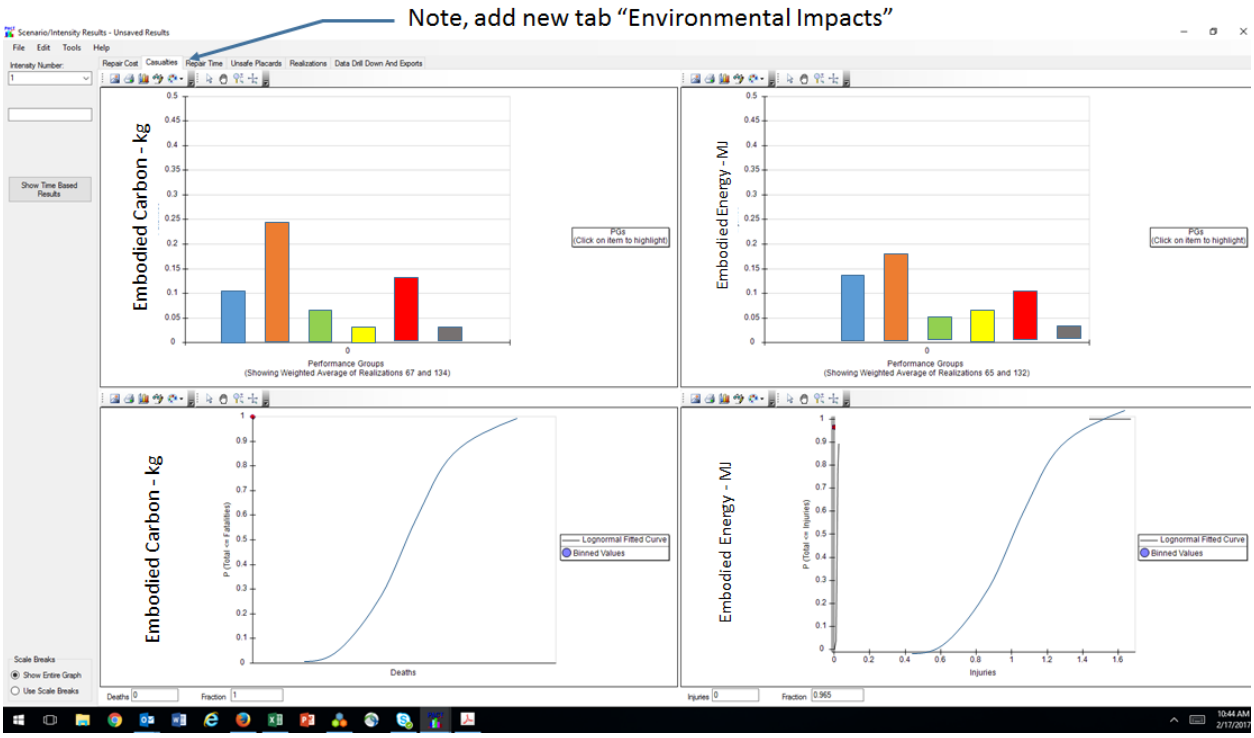


Figure 1 – Primary Output for Environmental Products

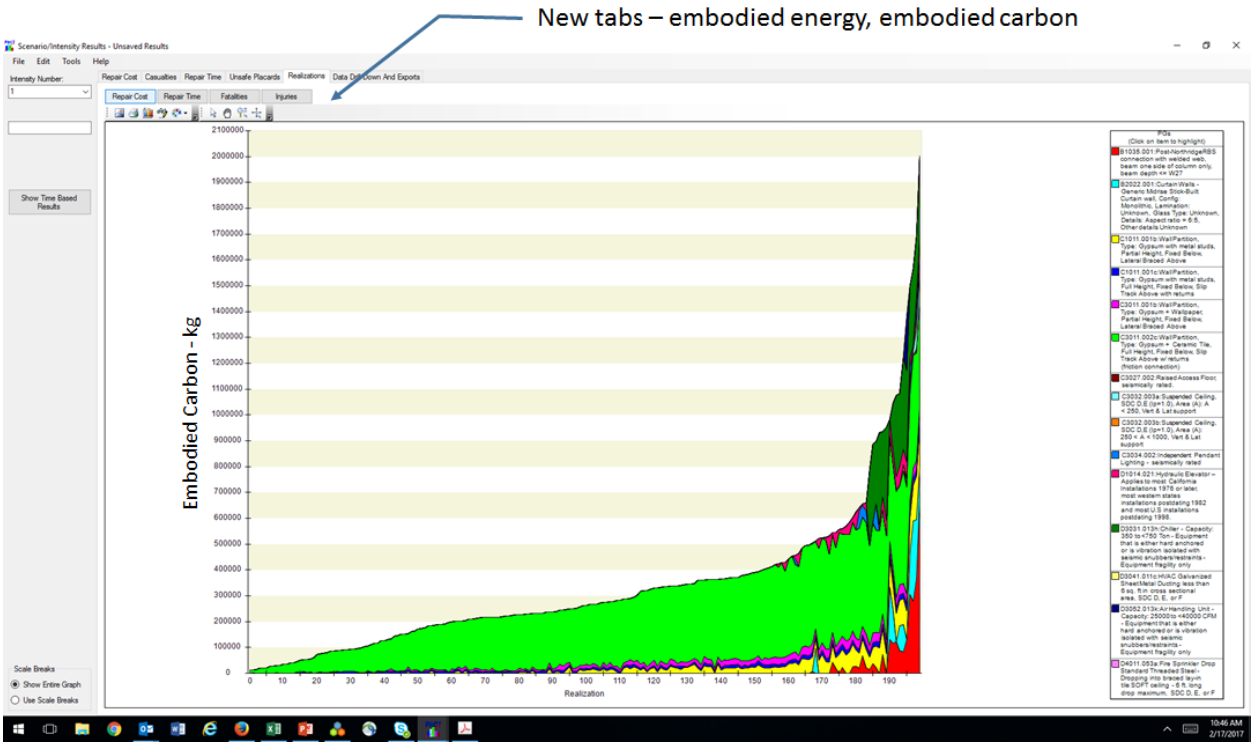


Figure 2 – Realization-based Embodied Carbon (Embodied Energy similar)

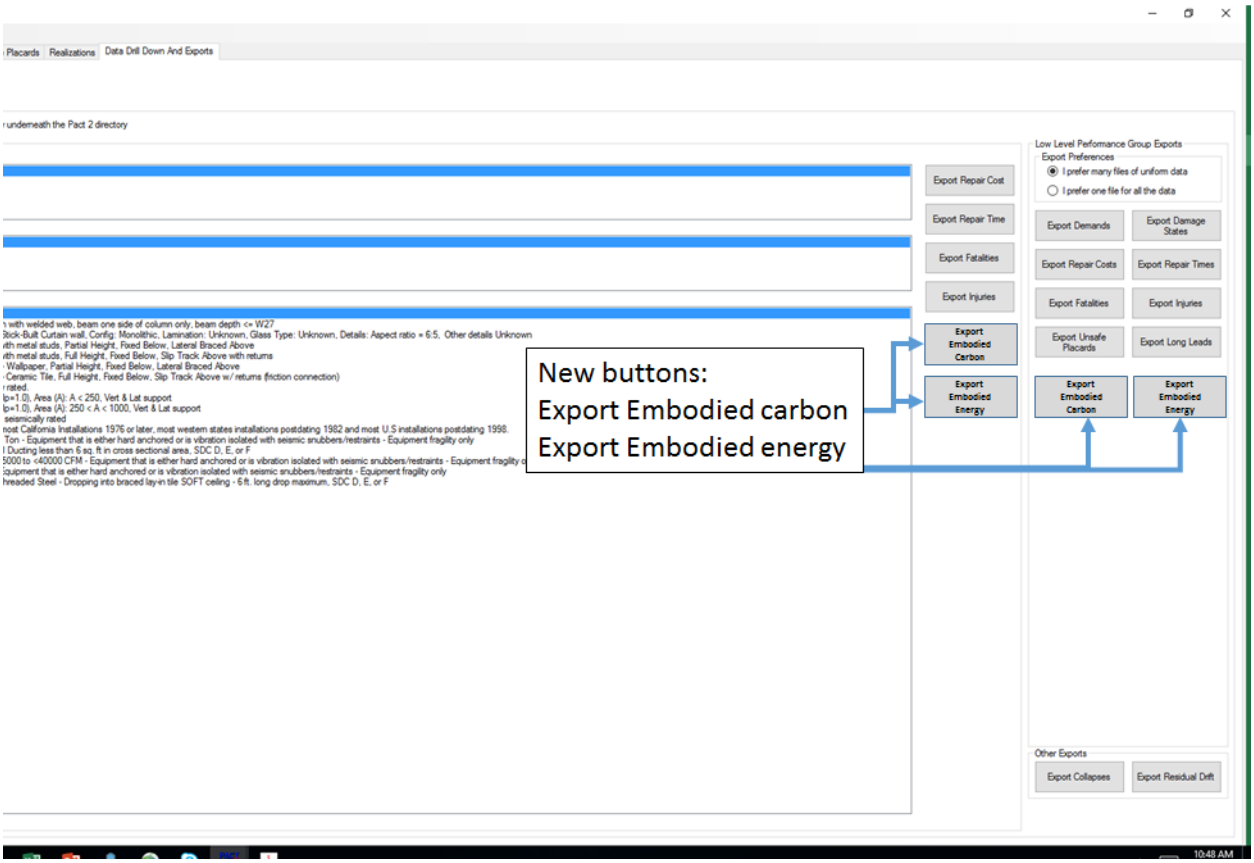


Figure 3 – Export and drill down page