

IWR Planning Suite II

Technical Documentation

Prepared by
CDM Smith
Carbondale, IL

November 2016



“Microsoft, Windows, Excel, Visual Studio, Visual C++ and Visual C# are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.”

© 2014 Microsoft Corporation. All rights reserved. This case study is for informational purposes only.

MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Microsoft, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Contents

List of Figures.....	vi
List of Tables.....	vii
Section 1 Introduction.....	1
1.1 Purpose of the Model	1
1.2.1 Plan Editor	2
1.2.2 Plan Generator Module	2
1.2.3 Uncertainty Module	2
1.2.4 CE/ICA.....	3
1.2.5 MCDA Module	4
1.2.6 Annualizer.....	4
1.3 Contribution to the Planning Effort	4
1.4 Description of Input Data.....	5
1.5 Description of Output Data.....	8
1.6 Model Capabilities and Limitations	9
Section 2 Technical Quality.....	11
2.1 Theory	11
2.2 System Represented by the Model.....	11
2.3 Analytical Requirements.....	11
2.4 Assumptions	12
2.5 Conformance with USACE Policies.....	13
2.6 CE/ICA Formulas	13
2.6.1 Average Cost.....	13
2.6.2 Incremental Cost	14
2.6.3 Incremental Output.....	14
2.6.4 Incremental Cost per Unit	15
2.6.5 Data Envelopment Analysis	Error! Bookmark not defined.
2.7 Data Envelopment Analysis.....	15
2.8 MCDA Formulas	16
2.8.1 Analytical Hierarchy Process	17
2.8.2 Weighted Scoring	18
2.8.3 Efficient Frontier.....	24
2.8.4 Compromise Programming.....	27
2.8.5 Outranking.....	29
2.9 Annualization Formulas	32

2.9.1	Capital Recovery Factor	32
2.9.2	Present Value Factor	32
2.9.3	Equivalent Annual Cost.....	33
2.10	Uncertainty Computation	33
2.10.1	Uncertainty Distributions	33
2.10.2	Uncertainty Correlation Matrix.....	34
2.10.3	Uncertainty Variable Convergence	35
Section 3 System Quality.....		37
3.1	Programming Language and Supporting Software	37
3.2	Program Installation	37
3.2.1	Installation Prerequisites.....	37
3.3	Availability of Hardware and Software Required	37
3.4	Testing and Model Validation Process Description	38
Section 4 Usability		39
4.1	Availability of Input Data	39
4.2	Output Format	39
4.3	Usefulness of Analytical Results	39
Section 5 References.....		41
Appendix A: Proposed Approach for Including Correlation in IWR Planning Suite		43
A.1	Summary.....	43
A.2	IWR Plan-RE Distributions.....	44
A.3	Prior Work.....	44
A.4	Correlation	44
A.5	Valid Correlation Matrices.....	45
A.6	Technical Approach.....	45
A.7	Implementation Notes.....	50
A.8	References	50
Appendix B: Generating Correlated Uniform Variates.....		51
B.1	Overview	51
B.2	The Problem.....	51
B.3	Solution	51
B.4	A Matlab Implementation.....	53
B.5	An R Implementation.....	53
B.6	Internal Links.....	54
B.7	External Links	54
Appendix C: R Worked Example.....		55
Appendix D: Database Design.....		61

D.1	Study Manager Database.....	61
D.2	Study Manager DDL	61
D.3	Planning Study Database	62
D.3.1	Plan Editor Diagram.....	62
D.3.2	Plan Generator Diagram.....	64
D.3.3	MCDA Diagram	65
D.3.4	Uncertainty Diagram	66
D.3.5	Annualizer Diagram	67
D.3.6	Watershed Diagram.....	68
D.3.7	Planning Study Database DDL	68
D.4	Uncertainty Database for Child Sets	96
D.5	Uncertainty Database DDL.....	97

List of Figures

Figure 1 IWR Planning Suite Description.....	3
Figure 2 Levels of Input.....	6
Figure 3 Variable Sensitivity Input Table.....	7
Figure 4 Solution	8
Figure 5 Distance Measure	Error! Bookmark not defined.
Figure 6 The Efficient Frontier of Dominant Plans.....	25
Figure 7 Dominated and Non-Dominated Plans	26
Figure 8 Running Mean and Standard Deviation During Simulation	35
Figure 9 Running CDF During Simulation	36
Figure 10 Study Manager Database Diagram	61
Figure 11 Plan Editor Diagram (Core Tables)	63
Figure 12 Generator Diagram	64
Figure 13 MCDA Diagram.....	65
Figure 14 Uncertainty Diagram.....	66
Figure 15 Annualizer Diagram.....	67
Figure 16 Watershed Diagram	68
Figure 17 Uncertainty Child Database Diagram.....	97

List of Tables

Table 1. Testing Average Cost Computation	14
Table 2. Testing Incremental Cost Computation.....	14
Table 3. Testing Incremental Output Computation	14
Table 4. Testing Incremental Output Computation	15
Table 5. Initial Decision Matrix.....	18
Table 6. Negate Minimized Criteria Values.....	19
Table 7. Criteria Values Normalized by Range	21
Table 8. Ranked Plan Alternatives	21
Table 9. Normalized Criteria Values.....	22
Table 10. Negate Minimized Criteria Values.....	22
Table 11. Normalize Criteria Values.....	23
Table 12. Apply Criteria Weights to Normalized Matrix Values.....	23
Table 13. Ranking of Weighted Criteria	24
Table 14. Two Variable (Input/Output) Example	26

Section 1

Introduction

In order to ensure the validity of the methods and tools employed by the U.S. Army Corps of Engineers (USACE) to guide their investment decisions concerning water resources and the natural environment, the USACE Planning Models Improvement Program (PMIP) was established in 2003. The PMIP was designed to develop procedures “to review, improve, and validate analytical tools and models for USACE Civil Works business programs” (USACE, 2011).

PMIP categorizes models into four groups depending on the developing entity; the categorization of the model dictates the level of review performed by the certification team. The categories are: corporate models (developed by USACE and applicable nationwide), regional/local models (developed by USACE to deal with unique situations that corporate models cannot be cost effectively modified to address), commercial off-the-shelf models (developed by private corporations), and models developed by others (developed by other Federal agencies, states, counties, etc.). Only corporate and regional/local models need to be fully certified by the Planning Center of Expertise (PCX). The other two categories only need to be submitted for approval by PCX (USACE, 2011). The Institute for Water Resources (IWR) developed IWR Planning Suite, and therefore this model is classified as a corporate model and must go through the full certification process.

The PMIP determined three criteria areas that must be satisfied in order for a model to be certified or approved for use on USACE projects. These criteria are also the foundation of the necessary documentation provided to PCX when certification is requested. The criteria are: technical quality, system quality, and usability. If one of the criteria is not met to the satisfaction of PCX, the model is not certified or approved for use (USACE, 2011).

The purpose of this document is to describe software and hardware requirements, intended uses, critical or underlying assumptions that might affect validity of module applications, recognized limitations, as well as the supported distribution types, mathematical formulas associated with the Monte Carlo computations, important programming procedures and functions, and other necessary information to support satisfaction of the Corps of Engineers Model Certification process.

This section describes the model, its purpose, its contribution to the planning effort, input and output data, model limitations, and the development process. Section 2 provides a discussion of technical quality, Section 3 discusses the model’s system quality, Section 4 describes the usability of the IWR Planning Suite, and Section 5 provides a list of references.

1.1 Purpose of the Model

The purpose of IWR Planning Suite is to assist users in selecting cost-effective ecosystem restoration or mitigation plans. The original model, IWR Plan, was expanded into IWR Planning Suite to meet the changing needs of USACE planners. The need for this expansion was determined based on feedback from the community of IWR Plan users. The IWR Planning Suite provides a mechanism to perform the cost effectiveness and incremental cost analysis (CE/ICA) required for USACE planning projects. The goal of the

software is to allow planners to make more informed decisions; however, it cannot select the best alternative plan for the user (USACE, 2004, p. 1-1).

1.2.1 Plan Editor

Plan Editor is the interface for the fundamental components needed to manage and create planning studies and display their individual planning sets. It allows users to define the variables and attributes that will be shared across all planning sets within a study (including variables that are derived from a formula), and it provides users with a way to access and edit planning sets.

1.2.2 Plan Generator Module

Plan Generator module is where the user enters information concerning each solution (an action that could be applied to achieve the planning goals), including the solution's description, benefits, and costs. This module is also where relationships (e.g., two solutions are incompatible) and sensitivities are entered into the analysis. Sensitivities can be placed on specific variables (e.g., benefits or costs) of a given solution to allow the user to compute both high and low values for a given variable. The specifics of solutions, relationships, and variable sensitivities are addressed in further detail in Section 1.4.

Information entered in the Plan Editor is utilized by the Plan Generator module to generate all possible alternative plans (combinations of solutions). This module applies all the relationships entered in the Plan Editor to automatically eliminate plans that, in reality, cannot be applied. In this module, the user can also enter constraints to filter the planning set (a group of all possible plans generate by the Plan Generator) to include only plans that meet the desired criteria (minimum or maximum values for the benefits and costs of the solutions).

The user can also adjust the solution outputs using the Automated Edits feature of Plan Generator. This feature relaxes the software's assumption that all solution outputs are additive by allowing the user to enter a mathematical function to describe complex solution effects. In addition, the Plan Generator has a solution sensitivity feature that is similar to the variable sensitivity feature found in the Plan Editor. Solution sensitivity allows the user to compute high and low values for a given variable and solution combination.

1.2.3 Uncertainty Module

The Uncertainty Module allows users to specify a distribution for each variable of each solution/scale. Available distribution types are:

- Fixed
- Normal
- Uniform
- Triangular
- Truncated Normal
- Cumulative Distribution Function (CDF)

A Monte Carlo simulation process can then be used to generate numerous sets of plans. Within each iteration of the simulation, the distribution would be sampled for each solution/scale variable, producing a single set of values for that iteration. A final “parent” set is produced containing the averaged results.

1.2.4 CE/ICA

Ability to perform CE/ICA is crucial to the IWR Planning Suite as the primary purpose of the model to assist planner in selecting cost-effective plans. The plans produced by the Plan Generator (or user-entered into the Editor) are evaluated by the CE/ICA methods to determine the cost effectiveness of each plan. Plans are classified as not cost-effective, cost-effective, or as a best buy.¹ In order to classify each plan’s cost effectiveness, the model calculates the Average Cost, Incremental Cost, Incremental Output, and Incremental Output per Unit for each plan.

Basic steps followed in the IWR Planning Suite for non-uncertainty planning sets are depicted below (Figure 1).

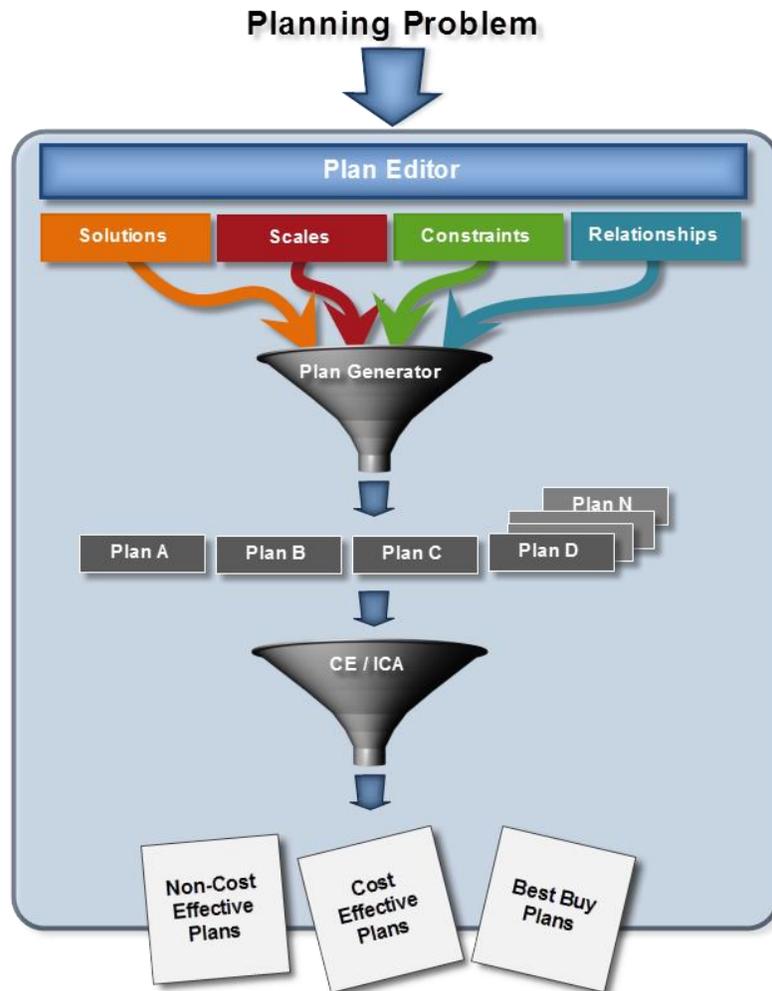


Figure 1
IWR Planning Suite Description

¹ A best buy plan is the least expensive plan at a given level of output.

CE/ICA is performed on uncertainty planning sets by evaluating child set and then computing totals for each plan alternative for the number of times the alternative was determined to be cost effective or best buy. The CE/ICA results are used to generate reports, graphs, and ultimately guide planning decisions.

While the software categorizes the plans by cost effectiveness, this software is *not* designed to tell the user the best plan to implement. The mathematics underlying CE/ICA are not complicated. However, the model can be complex; for example, the more solutions included in the analyses, the more complex the analyses become to account for all of the possible alternatives.

1.2.5 MCDA Module

IWR Planning Suite is limited in its capacity to compare plans effectively because it can only evaluate one type of benefit to produce the incremental cost analysis. The Multi Criteria Decision Analysis (MCDA) Module provides Corps planners with a toolbox for conducting multi criteria decision analysis and exploring multiple dimensions of a planning decision. MCDA techniques are tools the Corps of Engineers can use to improve the transparency of the decision making process. MCDA provides a proven mathematical means for comparing criteria with differing units such as habitat units, cultural resources, public sentiment and total cost (USACE, 2014).

1.2.6 Annualizer

The Annualizer is a planning tool available through the IWR Planning Suite that allows users to interpolate NED and NER benefits and costs over the period of analysis. It also estimates average annual equivalent NED costs and benefits and net present values, and estimates the average annual NER outputs. The Annualizer has no direct impact on planning sets contained within a study, and is only intended to serve as an aid to planners and economists.

1.3 Contribution to the Planning Effort

Planners are often faced with solving critical and complex environmental problems with limited funding. Therefore, planners are forced to make tradeoffs between the benefits and costs of alternative plans. The IWR Planning Suite is designed to assist users in making these tradeoffs, since benefit-cost analysis is typically not applicable for ecosystem restoration and mitigation projects. Traditional benefit-cost analysis often cannot provide answers to environmental planning problems because there is no industry standard to measure environmental benefits.

Generally speaking, economists are comfortable using CE/ICA, but other disciplines involved in the decision process may not be. Ecologists, planners, biologists, geologists, and even politicians may be some of the parties involved in making final decisions. The IWR Planning Suite and accompanying User Guide are designed to assist all disciplines involved in the decision process in understanding how their respective inputs are used in CE/ICA, and how they affect the output at various stages of the planning process. This characteristic allows all disciplines to be more involved and potentially more effective team members in the overall planning effort (USACE, 2014).

As outlined in the User Guide, there are six basic steps in the planning process, which can be repeated depending on the specific needs of the project. The basic steps are:

1. Identify problems and opportunities;
2. Inventory and forecast without-project conditions;

3. Formulate alternative plans;
4. Evaluate effects of alternative plans;
5. Compare alternative plans; and
6. Select a plan (USACE, 2014).

IWR Planning Suite can assist decision makers with steps 3, 4, and 5 listed above. Without step 1, there would be no need to use the software because there would be no planning problem to address with analyses. The model cannot assist with step 2, as all project costs and benefits must be determined prior to using the software. While IWR Planning Suite can reduce the number of alternative plans being considered, the software is not designed to select a specific plan to implement (step 6).

For a more detailed description of IWR Planning Suite’s contribution to the Planning Effort, refer to the User Guide, Section 1.

1.4 Description of Input Data

Solutions are the primary data input in the IWR Planning Suite. In a broader sense, solutions could include management measures, alternative plans, and programs that are designed to meet some portion of the planning objective. Management measures are features or activities that are employed to achieve desired effects, and are the “building blocks of alternative plans.” Management measures can be entered into the software individually (i.e., as solutions), or combined in an alternative plan predetermined by the user. A program is a set of alternative plans, and is typically spread over a large geographic area. To avoid confusion, throughout this document, the term “solution” refers to a single management measure, “plan alternative” refers to a combination of solutions, and “planning set” refers to a combination of plan alternatives. Figure 2 illustrates the different levels of data.

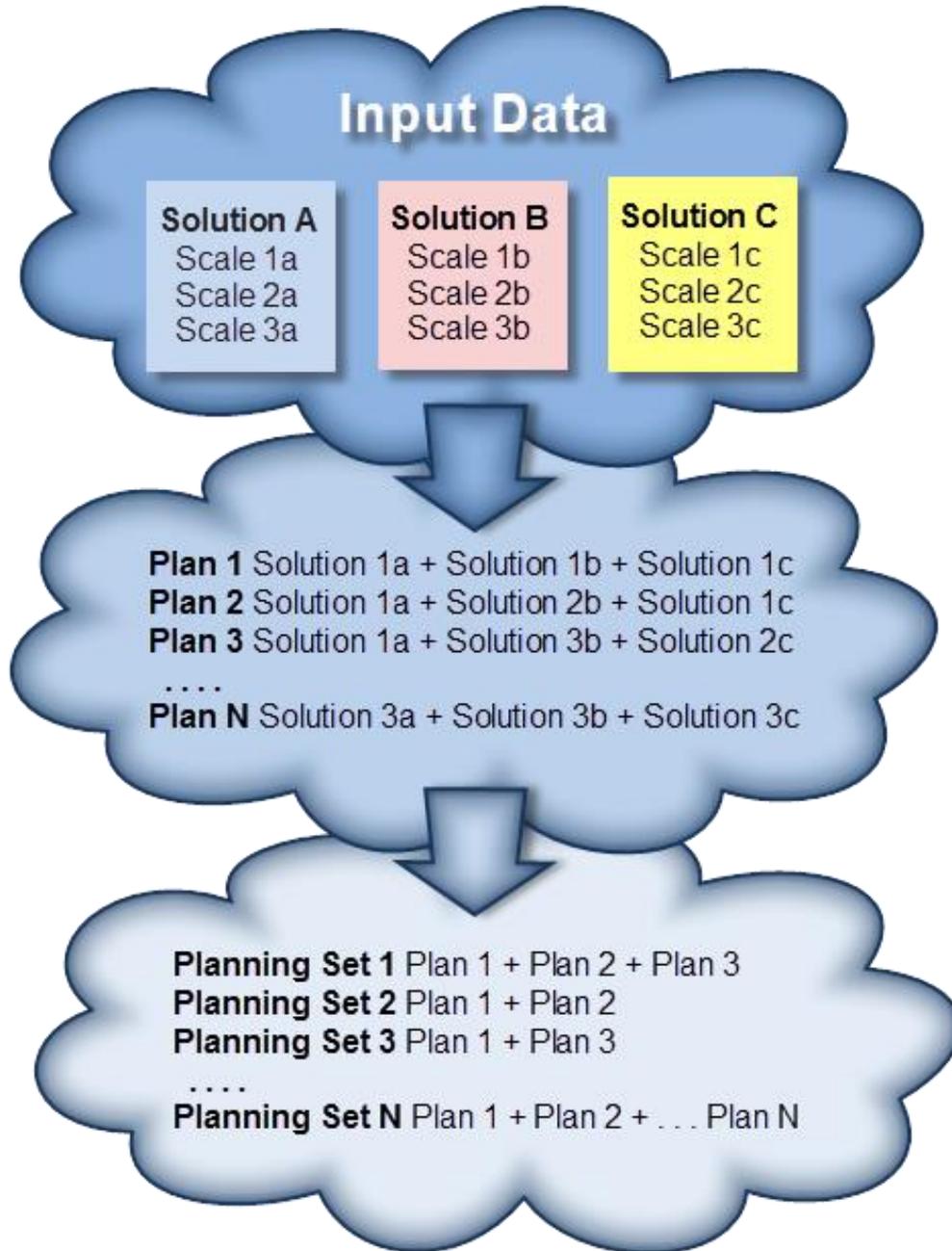


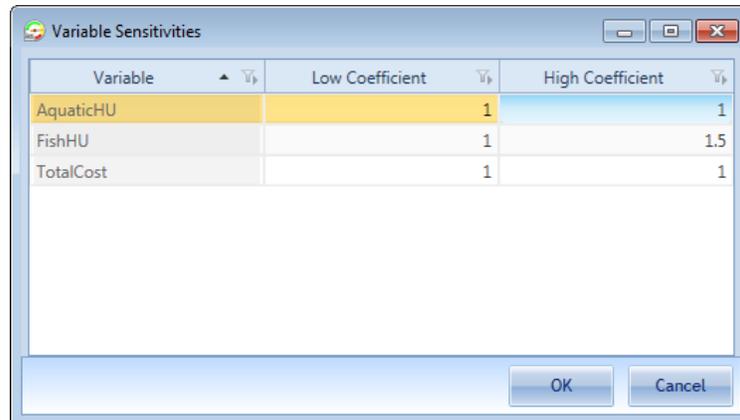
Figure 2
Levels of Input

Solutions for a given planning set entered into IWR Planning Suite consist of the same set of variables. Variables can include habitat units (for birds, fish, and vegetation), increases in species population, acreage protected, total cost of the solution, and numerous other options as dictated by the specific needs of the project. The possible variables and data sources for IWR Planning Suite inputs are abundant and typically very project specific. Planners can collect data first hand, or consult previously published studies and expert elicitations.

Users must enter scales for each solution. Solution scales can be physical properties (size, amount), composition (materials, methods), locations, timing, or duration. The scales are mutually exclusive, and a plan can only select one scale option for each solution. In order to optimize the plan, the user must determine the number of scales used in the analysis. It is important for the number of scales to be meaningful, practical, revealing, reasonable; a “change in scale should result in changes in output, cost, or both” (USACE, 2014.)

Relationships between solutions must be considered by the user and entered into the program to avoid generating plans that cannot be realistically implemented. This will limit the number of plans generated by the software. Solutions can be combined, exclude one another, or depend upon each other. Relationships can be created using “and” or “or” statements, and there is an option to prohibit combination of any solutions.

Sensitivity (producing low and high results) of both the variables and solutions can be analyzed within the IWR Planning Suite. Variables include both the benefits (habitat units produced, increase in species population) and costs associated with each solution. Variable sensitivity adjusts the low and high values for the given variable across all solution scales. For example, if the high coefficient for FishHU (habitat unit for fish) is 1.5, across all solution possibilities the high value output is multiplied by 1.5 for FishHU. Figure 3 shows a screen shot of the variable sensitivity input table. Low and high coefficients are entered to allow the computer to generate low and high values for the selected variable. The low coefficients must always be a real number less than or equal to one (negative numbers are permitted), and the high coefficients must always be greater than or equal to one. Typically, the low and high values calculated for each variable will be hidden unless the user chooses to display them.



Variable	Low Coefficient	High Coefficient
AquaticHU	1	1
FishHU	1	1.5
TotalCost	1	1

Figure 3
Variable Sensitivity Input Table

Solution sensitivities are similar to variable sensitivities, except the low and high coefficients are applied to a variable for only one specific solution. This means that high and low coefficients adjustments are not uniformly applied to variables across the board. For example, the high coefficient for FishHU can be adjusted to 1.5 for only the Mercury Reduction solution and the FishHU for all other solutions will remain the same (USACE, 2014). Figure 4 shows a screen shot of the solution sensitivity input table.

Code	Solution	Variable	Low Coefficient	High Coefficient
S	Streambed stabilization	TotalCost	1	1
S	Streambed stabilization	AquaticHU	1	1
S	Streambed stabilization	FishHU	1	1
M	Mercury reduction	TotalCost	1	1
M	Mercury reduction	AquaticHU	1	1
M	Mercury reduction	FishHU	1	1.5
T	Temperature control	TotalCost	1	1
T	Temperature control	AquaticHU	1	1
T	Temperature control	FishHU	1	1

Figure 4
Solution

For a complete description of the input data for the IWR Planning Suite, refer to the User Guide, Section 5 and the Test Plan.

1.5 Description of Output Data

In relation to the IWR Planning Suite, an output is defined as an “intended, beneficial, nonmonetary effect” (USACE, 2014.) Outputs do not have standard metrics, but common ones include physical dimensions (acres, days), population counts, and habitat units; however, the user must determine the most appropriate metrics to use to meet the project objectives.

As described in Section 1.2.3, each plan is categorized as not cost-effective, cost-effective, or a best buy. These results are presented initially in the software in a basic table similar in appearance to a Microsoft Excel spreadsheet. The user can generate reports or graphs to assist with final planning decisions. When users generate a report or graph, they can select certain criteria dictating what is or is not displayed. For example, the user can graph all plans, “cost-effective only” plans, or “best buy only” plans. Depending on the number of plans generated, the ability to filter results easily is extremely helpful, and can reduce the amount of information included in the final planning decision process.

Reports and graphs can be generated on a single planning set (any group of plan alternatives), or a selection from all generated planning sets in the software. Reports can focus on “Total and Average Costs,” with the ability to filter by best buy or cost-effective plans, Incremental Cost, Is it worth it? (Cost-effective only plans), or All Variables (with the ability to filter by best buy or cost-effective plans). Graph options include Cartesian graphs (which also allow the user to graph all plans, cost-effective only plans, or best buy only plans), box graphs with an optional duel-axis overlay for either Total Cost or Average Cost (which can only be displayed using Incremental Costs), 3-D surface plots, and 3-D scatter plots. The results can be displayed uniformly, or differentiated by not cost-effective, cost-effective, and best buy. The variety of report and graph options available allows the user to generate tables and graphs as necessary for their final planning reports without requiring separate software packages.

For a complete description of the Output Data for the IWR Planning Suite, refer to Section 5 of the User Guide.

1.6 Model Capabilities and Limitations

IWR Planning Suite can assist the user in making more informed decisions, but may not lead to a single solution as in a typical benefit-cost analysis. The model was designed for restoration and mitigation planning, but is not limited to these areas, and is applicable for large- and small-scale projects.

The model cannot measure or forecast environmental outputs, monetize outputs, reduce environmental requirements, or identify the optimal solution. Also, each planning problem requires a unique approach to conducting CE/ICA. The basic procedures are the same, but each problem requires the procedures to be adjusted to meet the project circumstances and objectives (USACE, 2014).

IWR Planning Suite cannot convert values into average annual benefits and costs directly, allow the Annualizer module can assist with providing average annual costs and benefits. The software assumes the values are already converted and conducts the CE/ICA accordingly. All users need to be aware of this limitation to ensure the appropriate values are entered into the software.

Multiple Criteria Decision Analysis helps individuals understand what factors are influencing decisions by using evaluation criteria and weights. An MCDA will present a preferred plan; however that plan may vary depending on the algorithm used and the sensitivity of the solution set. There are two parts in the development of MCDA. There is the selection of the analysis technique that includes criteria and weights; and the actual computation of the MCDA. It is the responsibility of the user to be transparent in their selection of the criteria and weights.

IWR Planning Suite has the ability to perform risk and uncertainty analyses. Variable and solution sensitivities can be adjusted (as previously discussed) to calculate a range of values. It should be noted however, that while the software allows variables or solutions to be weighted differently, it cannot provide justification for the weights applied by the user. Users must know what the weights mean and be able to defend any weights employed in their reports.

Additionally, uncertainty analyses can be performed by using the Uncertainty Module to specify a distribution for each variable of each solution/scale and then using the Monte Carlo simulation to generate numerous sets of plans. During each iteration, the distribution for every solution/scale variable is sampled, producing a single set of values for that iteration. A final parent set is then calculated as an average. CE/ICA can be performed for each set of plans with total cost effective and best buy counts being displayed in the parent set for each plan alternative. Currently, the software does not include the ability to perform a Data Envelopment Analysis. All other risk and uncertainty analyses must be calculated outside of the IWR Planning Suite.

For the results of CE/ICA analysis to be interpretable, the units for solution scales must be cardinal. If the scales are different for the given solution, the generated results will be meaningless. The user needs to be aware of this issue, because if different scales are used, the results will still be generated without producing a warning or error message.

Planners sometimes choose to include options such as educational or aesthetic benefits in their alternatives. Depending on the circumstances, including these types of benefits in the IWR Planning Suite

may not be appropriate because of USACE policy restrictions. The user should always be aware of applicable policy restrictions when using the IWR Planning Suite. The outputs and costs of such benefits could be included in the overall planning decision external to the software, generally with little extra effort on the part of the user. However, no matter what benefit types the user includes in his or her IWR Planning Suite analyses, benefits must be reasonable and well formulated to produce reasonable and well-formulated plans.

It is also important to note that the results from the IWR Planning Suite are not comparable across projects unless the identical methodology and measurements are used in both projects. This should not be confused with the ability to compare information from different *planning sets*, which are different sets of alternative plans generated for the same project.

For a basic description of the Model Capabilities and Limitations for the IWR Planning Suite, refer to the User Guide, pages 3 to 7.

Section 2

Technical Quality

Technical quality as the name suggests, focuses on the technical aspects of the model. The model must use sound theory as a foundation for all analyses, provide a realistic representation of the system being modeled, identify and include necessary analytical properties, and use correct formulas for all model computations (USACE, 2011). The following sections address each of these criteria respectively.

2.1 Theory

The theory behind the IWR Planning Suite is based on cost effectiveness.² While traditional benefit-cost analysis requires that all benefits be monetized, because benefits and costs must be expressed in the same units in order to calculate the benefit-cost ratio for each alternative plan, many environmental benefits cannot be monetized; therefore benefit-cost analysis is typically not applicable to environmental planning problems. However, since IWR Planning Suite uses cost effectiveness, benefits do not need to be monetized, or even all be in the same units (e.g., one unit could be habitat units, and another population counts), making it ideal to use in environmental planning.

Cost effectiveness determines the least expensive plan with the maximum level of output produced. After the IWR Planning Suite determines cost effectiveness, it conducts an incremental cost analysis, which determines changes in cost as output levels increase. This allows planners to decide whether producing output at the next level is worth the additional expense. In effect, it provides planners with the ability to make more informed decisions about the trade-offs between environmental output and project expense. This is helpful when faced with limited funding (USACE, 2014).

For a complete description of the theory underlying the IWR Planning Suite, please refer to the User Guide, Section 1.

2.2 System Represented by the Model

Unlike other models utilized by USACE or other governmental agencies, the IWR Planning Suite does not represent a specific system. It simply provides a platform for building alternative plans based on a pre-determined set of criteria entered by the user.

2.3 Analytical Requirements

IWR Planning Suite requires that users be specific in what they are, and are not, trying to do with the analyses. The purpose of IWR Planning Suite is to provide a platform for planners to evaluate and compare alternative plans for ecosystem restoration and mitigation. The model accomplishes this by offering several analytical options to the user, allowing the program to meet the specific needs of a given project.

² Identifies the least cost plan alternative for each possible level of environmental output and that for any level of investment, the maximum level of output is identified (USACE, 2014).

2.4 Assumptions

The IWR Planning Suite applies several assumptions as it conducts the various stages of CE/ICA analysis. While these assumptions are sound, the user needs to be aware of them to ensure the analysis is performed correctly.

The most important assumption the IWR Planning Suite employs is that the user is proficient with the software. The user must know what can and cannot be done with the software, and understand all of the assumptions applied by the program. Lack of user understanding can easily result in invalid output, and planning decisions based on inaccurate information. Also, some of the IWR Planning Suite's assumptions can be relaxed or adjusted for some features built into the program. However, the user has to know the features are there, and how to use them correctly.

The software assumes that the user correctly inputs cost and output levels for all solution scales or pre-determined plans. This is related to the assumption that the user has already examined and determined the benefits of the solutions he or she wishes to consider, including the alternative plans. If the benefits and costs for solutions have not been pre-determined, then the IWR Planning Suite cannot be correctly used in the planning process. The software also assumes that the user has already calculated the annual average values for all variables.

Another set of assumptions employed by the IWR Planning Suite concerns classifying certain plans as best buys, even if selecting the plans to implement would result in poor planning decisions. The first plan that will always be a best buy is the "No Action" plan, which produces zero output and has zero cost. By definition it is a best buy, because at an output of zero it is the least expensive option. It can be argued that "No Action" resulting in zero output and zero cost is another assumption made by the software. For the analysis, it is necessary to have a zero cost, zero output plan, and the "No Action" plan serves this purpose.

The other plan always assumed to be a best buy is the plan with the highest output, and typically an extremely high cost. While this plan may produce the highest level of output, it may not be the most appropriate plan to implement when compared to other best buy options. The IWR Planning Suite will not tell the user which plan to implement, but only inform the planning decision.

The IWR Planning Suite assumes that outputs and total costs of solutions are additive. However, in many circumstances solution outputs and costs do not have a linear relationship. Implementing one solution may make another solution more efficient, less expensive, or less time consuming. While the software assumes a linear relationship, this assumption can be relaxed using the Automated Edits feature in Section 5 of the User Guide.

Nearly every software package used to model systems, predict outcomes, or involving the environment is forced to make the same assumption. They must assume that outputs and costs entered into the program can be predicted to a precise value. The IWR Planning Suite is no exception. In order to perform the CE/ICA analysis, the software must assume the output and cost values entered reflect the actual values. This assumption can be relaxed slightly by using the Variable Sensitivities feature, but even then the software assumes the values will fall strictly within the specified range. This assumption will always be present to some degree in environmental planning decisions, and users need to be aware of this and account for it in their final analyses.

2.5 Conformance with USACE Policies

The primary user audience for the IWR Planning Suite is USACE planners. Therefore, the model needs to address USACE policies that must be included in analyses and planning decisions. The specific USACE policies that need to be applied will vary by project, as each project has a unique set of circumstances and planning objectives. This report will briefly discuss USACE policies that will likely apply to most projects using the IWR Planning Suite.

It is important for users to distinguish the habitat types involved in their planning project. They must distinguish between upland and wetland habitats, because funding priorities are considerably higher for wetland habitats. Also, projects occurring in upland habitats typically require a larger cost share from the local sponsor than wetland projects. The IWR Planning Suite is neutral on the habitat types involved in solutions or plans entered into the software. Users can make the distinction in the software between the importance placed on upland and wetland habitats, but they must be aware that the software will not make the distinction automatically.

It is USACE policy for benefits and costs presented in project proposals to be calculated as net values from without-project conditions.³ If calculations are performed from gross values,⁴ the IWR Planning Suite will produce the same results as when calculations are performed from net values. While the mathematics employed by the software are indifferent to the distinction between gross and net values, USACE policy is not. The user needs to be aware of this policy restriction to avoid potential problems in the later stages of the planning process.

As mentioned in Section 1.6, benefit and cost values need to be entered into the software in average annual units. The IWR Planning Suite assumes this input as it performs CE/ICA analysis, and it is USACE policy to have values calculated in average annual units. All users of the program need to be aware of this policy requirement to ensure the appropriate figures are entered into the program for the analyses.

2.6 CE/ICA Formulas

The sections below outline the CE/ICA formulas used within the IWR Planning Suite software, and contain an example that demonstrates the software performs the computations correctly.

2.6.1 Average Cost

$$\text{Average Cost} = [\text{Total Cost of Alternative A}] / [\text{Total Output of Alternative A}]$$

The testing results for average cost computations from one example used by the model review team are displayed below in Table 1. The calculated average cost values computed by the IWR Planning Suite and match those calculated in Microsoft Excel.

³ Net values reflect the difference between the output level under without-project conditions and the output level produced if the solution is implemented.

⁴ Gross value is the total output that would occur if a given solution is implemented without subtracting the output that would occur if no project occurred.

Table 1.
Testing Average Cost Computation

Plan	Output	Cost (\$1000)	Average Cost – IWR Planning Suite	Average Cost - Excel	Difference in Average Cost Values
No Action	0	0	0	n/a	n/a
Plan 1	41.63	19.7	0.473	0.473	0
Plan 2	49.5	29.7	0.6	0.6	0
Plan 3	83.75	260.2	3.107	3.107	0
Plan 4	84.5	313.8	3.714	3.714	0

2.6.2 Incremental Cost

$$\text{Incremental Cost of Alternative B} = [\text{Total Cost of Alternative B}] - [\text{Total Cost of Alternative A}]$$

The IWR Planning Suite only reports incremental cost values for best buy plans, which is why results can only be tested and compared for these plans. The calculated incremental cost values in the IWR Planning Suite match those calculated in Microsoft Excel.

Table 2.
Testing Incremental Cost Computation

Plan	Output	Cost (\$1000)	Incremental Cost – IWR Planning Suite	Incremental Cost - Excel	Difference in Incremental Cost Values
No Action Plan	0	0	0	n/a	n/a
Plan 1	41.63	19.7	19.7	19.7	0
Plan 2	49.5	29.7	10	10	0
Plan 3	83.75	260.2	230.5	230.5	0
Plan 4	84.5	313.8	53.6	53.6	0

2.6.3 Incremental Output

$$\text{Incremental Output of Alternative B} = [\text{Total Output of Alternative B}] - [\text{Total Output of Alternative A}]$$

The testing results for incremental output computations are displayed below in Table 3. Similar to reports for incremental costs, the IWR Planning Suite only reports incremental output values for best buy plans. The calculated incremental output values in the IWR Planning Suite match those calculated in Microsoft Excel.

Table 3.
Testing Incremental Output Computation

Plan	Output	Cost (\$1000)	Incremental Output – IWR Planning Suite	Incremental Output - Excel	Difference in Incremental Output Values
No Action Plan	0	0	0	n/a	n/a
Plan 1	41.63	19.7	41.63	41.63	0
Plan 2	49.5	29.7	7.87	7.87	0
Plan 3	83.75	260.2	34.25	34.25	0
Plan 4	84.5	313.8	0.75	0.75	0

2.6.4 Incremental Cost per Unit

$$\text{Incremental Cost per Unit of Alternative B} = \frac{[\text{Incremental Cost of Alternative B}]}{[\text{Incremental Output of Alternative B}]}$$

The testing results for incremental cost per output computations are displayed below in Table 4. Similar to reports for incremental costs, the IWR Planning Suite only reports incremental cost per output values for best buy plans. The calculated incremental cost per output values in the IWR Planning Suite match those calculated in Microsoft Excel.

Table 4.
Testing Incremental Output Computation

Plan	Output	Cost (\$1000)	Incremental Cost per Output – IWR Planning Suite	Incremental Cost per Output - Excel	Difference in Incremental Cost per Output Values
No Action Plan	0	0	0	n/a	n/a
Plan 1	41.63	19.7	0.473	0.473	0
Plan 2	49.5	29.7	1.271	1.271	0
Plan 3	83.75	260.2	6.73	6.73	0
Plan 4	84.5	313.8	71.467	71.467	0

2.7 Data Envelopment Analysis

When performing CE/ICA in the Uncertainty module, an additional option is available to perform Data Envelopment Analysis. While CE/ICA is a useful statistic, it still maintains the “knife-edge” approach, where a plan is either efficient or not, as opposed to the thick/fuzzy frontier, where a plan can be close to efficient. This analysis is performed for each iteration for each plan, relative to the frontier established in that particular iteration. It is assumed that the input and output variables are incommensurate (dollars, HU), otherwise traditional benefit-cost ratio (BCR) analysis would be possible.

After the cost effective alternatives for an iteration have been determined, the Pareto frontier is constructed by connecting all cost effective and best buy alternatives. Each alternative is then evaluated to determine appropriate distance measures for both input and output.

2.7.1 Distance Measures

The distance of a point to a line is typically taken to be the perpendicular distance from the point to line, as shown in Figure 5. This is acceptable as an efficiency measure if both the x and y axes are in the same units, but this is not typically the case in IWRPS. The same perpendicular distance can be obtained by large input distance and small output distance, or small input distance and large output distance. Any consistent measure that combines the input and output distance implies a relative weighting between the measures. It was decided that the most practical solution to this problem was to maintain the separate distances, measured on the input and output axes.

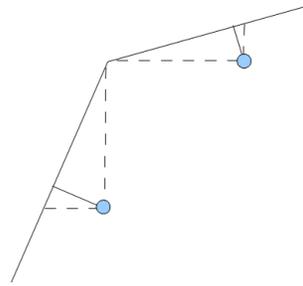


Figure 5
Distance Measure

2.8.2 Efficiency Measures

Assuming that any point on one of the frontier lines is an efficient point, metrics can be calculated for the points not on the line. An “input-oriented” efficiency measure uses the distances along the input axis, while an “output-oriented” measure uses distance along the output axis. In the example of Figure 6, points A and N are the efficient points on the frontier line, and B is the point for which we wish to calculate the efficiency. The ratio OA/OB will be 1.0 if B is on the frontier and will be less than 1 for interior points. The greater the distance AB, the smaller the efficiency measure. Similarly, the ratio BM/NM gives an output-oriented measure ranging from 0 to 1.

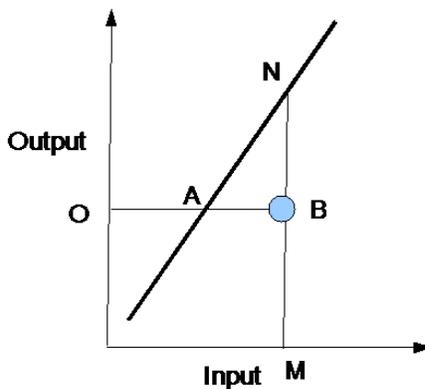


Figure 6
Efficiency Measures

The input and output oriented measures are dimensionless ratios and thus can be combined without the implicit weighting issues of direct measures.

2.8 MCDA Formulas

All MCDA work relies on the use of a decision matrix comprised of alternative and criteria combinations. Specifically, the matrix corresponds to an IWR Planning Suite planning set, where each plan alternative is a row in the matrix and each criterion is a column (USACE, 2014). Criterion weights are used to determine the relative importance of each criterion. Users are allowed to either manually assign weights to each criterion or use the Analytic Hierarchy Process (AHP). The software allows for ranking from 1 of 4 methods: Weighted Scoring, Efficient Frontier, Compromise Programming and Outranking.

2.8.1 Analytical Hierarchy Process

The fundamental approach behind AHP is to place all n criteria in an $n \times n$ matrix, and then place in each cell of the matrix one of the following values (9, 8, 7, 6, 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9). Thus, for two criteria i and j , cell $[i,j]$ of the matrix has one of the values from the above scale. The scale is defined such that a value of 1 indicates that the criteria are equally important, while a value of 9 indicates that i is extremely important or preferred to j . The inverse (1/9) indicates that j is preferred to i . According to the rules and logic, all entries on the matrix diagonal are 1, because a criterion is equivalent to itself. As well, the matrix is symmetrical, i.e., if $[i,j]$ equals 5, then $[j,i]$ equals 1/5. Thus, a decision-maker must enter $n(n-1)/2$ pairwise preferences. For 10 criteria, this requires entry of 45 different preference values, which is seen to rapidly get unwieldy and highly demanding of the decision-maker.

This specific scale 1 to 9 is designed by Saaty, developer and chief proponent of AHP. The more detailed description of the scale is given in Triantaphyllou, *Multi-Criteria Decision Making Methods: A Comparative Study*, Kluwer Academic Publishers, 2000, as follows:

Intensity of Importance (value of $a_{[i,j]}$)	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
3	Weak importance of One over another	Experience and judgment slightly favor one activity over another.
5	Essential or strong importance	Experience and judgment strongly favor one activity over another.
7	Demonstrated Importance	An activity is strongly favored and its dominance demonstrated in practice.
9	Absolute Importance	The evidence favoring one activity over another is of the highest possible order of affirmation.
2,4,6,8	Intermediate values between the two adjacent judgments	When compromise is needed
Reciprocals of Above non-zero		

Within AHP, the criterion preference matrix is then massaged mathematically to determine the implied weights of each criterion, $w[i]$. [In theoretical terms, the decision-maker preferences in each cell are taken to be approximations of the true preferences based on the weights, which, if known, would be $w[i]/w[j]$ for cell $[i,j]$. In fact, the process is run in reverse, to get the weights from the preferences. The methodology involves calculation of eigenvalues and eigenvectors.

Once the information has been entered into the matrix by the user, it is submitted to a consistency check. For any three entries in the matrix, if consistency in preferences holds, then:

$$a_{ij} * a_{jk} = a_{ik}$$

AHP does not presume that decision-maker preferences as expressed in the matrix have this quality. Rather, AHP explicitly assumes inconsistency, and measures it. Once a user has entered the values in the matrix, the calculated value of a_{ik} can be compared directly to the entered value, to provide immediate feedback on consistency. AHP uses the concept of a coefficient of inconsistency, CI, for the matrix as a

whole. CI is a function of the number of criteria, n , and the calculated dominant eigenvalue of the preference matrix, δ_{max} (used in calculating the weights from the matrix) as follows:

$$CI = (\delta_{max} - n) / (n - 1)$$

This is compared to a coefficient of random inconsistency (CRI), established in a lookup table and available from AHP texts, to get the inconsistency ratio:

$$IR = CI / CRI$$

When $IR < 10\%$, the inconsistency in preferences is acceptable.

Note that, in the full AHP, it is also possible to create ratings of the decision matrix in this same fashion, i.e., by pairwise comparisons of alternatives against criteria. IWRPS MCDA module does NOT do this; rather, it uses the ratings in the matrix, and apply AHP-determined weights.

2.8.2 Weighted Scoring

The IWR Planning Suite MCDA module has among its various ranking methodologies a weighted scoring capability that may be used to rank plan alternatives based on criteria, and to apply weights to them during the ranking process as well. We will proceed with a discussion of the weighted scoring methodology applied to the decision matrix used in the previous example.

The matrix may be entered into the manually, it may be the result of another analysis, or it may be imported from an Excel spreadsheet as has been done in example in Table 5.

Table 5.
Initial Decision Matrix

	Flood Risk	HUs	Recreation Days	Property Value Effects	Cost
Plan A	\$10.00	46	700	2	\$5.00
Plan B	\$800.00	0	0	3	\$15.00
Plan C	\$60.00	50	850	2	\$23.00
Plan D	\$80.00	90	2000	1	\$33.00
Plan E	\$25.00	85	1000	3	\$20.00
Plan F	\$60.00	75	800	1	\$27.00
Plan G	\$100.00	116	3000	2	\$53.00
Plan H	\$900.00	116	3000	3	\$68.00

Equal Weighting of Criteria

In general terms, given m plan alternatives and n criteria, a decision matrix D may be represented by the matrix notation

$$D = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

Where each row represents a plan alternative and each column represents a criterion.

After importing the decision matrix, the Weighted Scoring with Normalization by Range ranking method is chosen. For this initial example, all weights are entered as 1, so that everything will be weighted equally. We also want to Minimize our Cost criterion values while Maximizing our outputs.

Normalization by Range guarantees that each normalized criterion value, whether maximized or minimized, will be in the range of 0 to 1. The MCDA module performs a negation of any minimized criteria by multiplying those values by negative one before the normalization occurs.

$$D = \left(\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix} \times (-1) \right)$$

Because in this example all values are to be maximized except for the Cost value, the following matrix is produced, with the Cost values negated.

Table 6.
Negate Minimized Criteria Values

	Flood Risk	HUs	Recreation Days	Property Value Effects	Cost
Plan A	\$10.00	46	700	2	-\$5.00
Plan B	\$800.00	0	0	3	-\$15.00
Plan C	\$60.00	50	850	2	-\$23.00
Plan D	\$80.00	90	2000	1	-\$33.00
Plan E	\$25.00	85	1000	3	-\$20.00
Plan F	\$60.00	75	800	1	-\$27.00
Plan G	\$100.00	116	3000	2	-\$53.00
Plan H	\$900.00	116	3000	3	-\$68.00

Next, the matrix is normalized by range, with each criteria varying from 0 to 1, including the negated criteria. Note that the highest negated cost, - \$ 5.00, corresponds to the maximum normalized cost of 1, and the smallest negated cost, - \$ 68.00, corresponds to the minimum normalized cost of 0.

Given that a set of criterion values from the original decision matrix is represented as $D_i = [y_1, y_2, \dots, y_n]$ for n criteria and a column of criterion values from the normalized decision matrix

$$D' = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

is represented as $D'_i = [v_1, v_2, \dots, v_m]$, then the summation

$$v_i = \sum_{i=1}^m [(a_i - a_{\min}) \div (a_{\max} - a_{\min})]$$

represents the normalization of a criterion value. The application of this equation to the original example matrix results in the following normalized matrix.

Table 7.
Criteria Values Normalized by Range

	Flood Risk	HUs	Recreation Days	Property Value Effects	Cost
Plan A	0	0.396551724	0.233333333	0.5	1
Plan B	0.887640449	0	0	1	0.841269841
Plan C	0.056179775	0.431034483	0.283333333	0.5	0.714285714
Plan D	0.078651685	0.775862069	0.666666667	0	0.555555556
Plan E	0.016853933	0.732758621	0.333333333	1	0.761904762
Plan F	0.056179775	0.646551724	0.266666667	0	0.650793651
Plan G	0.101123596	1	1	0.5	0.238095238
Plan H	1	1	1	1	0

Any criteria weights are then applied to the matrix, and all of the criteria for each plan are summed. The sum of criteria values for each plan produces a composite score s_n for plan n such that $s_n = \sum_{i=1}^m D'_{n,i}$,

where $D'_{n,i}$ is the set of normalized criterion scores for the plan. The composite score calculated in this fashion is used to rank the plan.

Within the IWR Planning Suite MCDA module, ranking is performed by sorting the composite scores into descending (highest-to-lowest) order and assigning ascending positive integers to them as ranks. To elaborate, plan alternatives are ranked such that the plan with the highest score is assigned a value of one, representing the optimal alternative and the highest ranking.

The plan with the next-to-highest score is given rank two, and so on until the alternative with the lowest score receives the lowest ranking, m , where m is the number of plans. This produces a ranking of plans for our sample data set as shown in Table 8.

Table 8.
Ranked Plan Alternatives

Plan	Rank
Plan A	5
Plan B	4
Plan C	7
Plan D	6
Plan E	2
Plan F	8
Plan G	3
Plan H	1

Application of Varied Criteria Weights

Using the same decision matrix as earlier, different weights can now be applied to the criteria values to explore how the rank outcome may change. Like before, the MCDA Weighted Scoring with Normalization by Range ranking method is chosen.

In this example, we want our all the benefits to be weighted the same as cost. Because there are 4 output variables, The Cost criteria is given a weight of one, and all other criteria are given a weight of 0.25. The weights are then normalized by range. The set of normalized weights may be represented as W , where $W = [\omega_1, \omega_2, \dots, \omega_n]$. The resultant normalized weights to be used in our ranking process is shown in Table 9.

Table 9.
Normalized Criteria Values

Criterion	Weight	Normalized Weight
Cost	1.00	0.5000
Flood Risk	0.25	0.1250
HUs	0.25	0.1250
Recreation Days	0.25	0.1250
Property Value Effects	0.25	0.1250

Again, selecting Minimize for the Cost criterion, and leaving the other criteria at the default Maximize optimization will optimize our matrix correctly. Normalization by Range guarantees that each normalized criterion value, whether maximized or minimized, will be in the range of 0 to 1. The application performs a negation of any minimums before the first normalization. Because in this example all values are to be maximized except for the Cost value, the decision matrix shown in Table 10 is produced, with the Cost values negated.

Table 10.
Negate Minimized Criteria Values

	Flood Risk	HU's	Recreation Days	Property Value Effects	Cost
Plan A	\$10.00	46	700	2	-\$5.00
Plan B	\$800.00	0	0	3	-\$15.00
Plan C	\$60.00	50	850	2	-\$23.00
Plan D	\$80.00	90	2000	1	-\$33.00
Plan E	\$25.00	85	1000	3	-\$20.00
Plan F	\$60.00	75	800	1	-\$27.00
Plan G	\$100.00	116	3000	2	-\$53.00
Plan H	\$900.00	116	3000	3	-\$68.00

Next, the matrix is normalized by range, with each criteria varying from 0 to 1, including the negated criteria. The resultant matrix is shown in Table 11. Note that the highest negated cost, - \$ 5.00, corresponds to the maximum normalized cost of 1, and the smallest negated cost, - \$ 68.00, corresponds to the minimum normalized cost of 0.

Table 11.
Normalize Criteria Values

	Flood Risk	HUs	Recreation Days	Property Value Effects	Cost
Plan A	0	0.396551724	0.233333333	0.5	1
Plan B	0.887640449	0	0	1	0.841269841
Plan C	0.056179775	0.431034483	0.283333333	0.5	0.714285714
Plan D	0.078651685	0.775862069	0.666666667	0	0.555555556
Plan E	0.016853933	0.732758621	0.333333333	1	0.761904762
Plan F	0.056179775	0.646551724	0.266666667	0	0.650793651
Plan G	0.101123596	1	1	0.5	0.238095238
Plan H	1	1	1	1	0

The normalized criteria weights, $W = [\omega_1, \omega_2, \dots, \omega_n]$ are then applied to the matrix D' so that given a set of normalized criterion values $D'_i = [w_1, w_2, \dots, w_m]$, a weighted criterion value is calculated as the sum of each criterion value multiplied by the normalized criterion weight ω_j , as represented by $w_i = \sum_{j=1}^m (v_i \times \omega_j)$.

This application of weights to each criteria produces a weighted matrix.

$$D'' = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \vdots & \vdots & & \vdots \\ W_{m1} & W_{m2} & \cdots & W_{mn} \end{bmatrix}$$

Following the application of weights, the criterion values are changed accordingly as shown in Table 12.

Table 12.
Apply Criteria Weights to Normalized Matrix Values

	Flood Risk	HUs	Recreation Days	Property Value Effects	Cost
Plan A	0	0.049568966	0.029166667	0.0625	0.5
Plan B	0.110955056	0	0	0.125	0.420634921
Plan C	0.007022472	0.05387931	0.035416667	0.0625	0.357142857
Plan D	0.009831461	0.096982759	0.083333333	0	0.277777778
Plan E	0.002106742	0.091594828	0.041666667	0.125	0.380952381
Plan F	0.007022472	0.080818966	0.033333333	0	0.325396825
Plan G	0.012640449	0.125	0.125	0.0625	0.119047619
Plan H	0.125	0.125	0.125	0.125	0

Due to the higher weighting of the Cost criteria (as compared to the individual output categories), the cost values are all increased compared to the other individual criteria values, and so the effect of cost on the ranking outcome will be greater than the effect of any single output.

Now, when composite scores are calculated, they will be the sum of the weighted criterion values for the plan alternative. Markedly different values will be computed for the composite scores due to the

application of the criterion weight values, and these altered scores will directly affect the order in which the plans are ranked.

This time, when ranking is performed, Table 13 displays that the plan ranks have changed, with plans that have lower costs receiving better ranks than in the previous example. Examine the resultant ranking of plans in Table 13, with the rankings from the former example (all criteria weighted equally) appended for comparison.

Table 13.
Ranking of Weighted Criteria

Plan	Benefits = Cost	All Criteria Equal
Plan A	3	5
Plan B	1	4
Plan C	4	7
Plan D	6	6
Plan E	2	2
Plan F	7	8
Plan G	8	3
Plan H	5	1

For example, Plan B, the lowest cost alternative, is now the 1st ranked plan, where formerly it was ranked 4th. And Plan H, the most costly alternative, was formerly ranked number 1, but with a heavier weight applied to cost, it now ranks a lowly 5! Several other plan rankings have also changed in similar ways, their rank improved by a lower cost or worsened by a higher one. The effects of applying weights to criteria on plan ranking is clearly represented in this example and the reader is cautioned to use a defensible, repeatable methodology when determining criterion weights.

2.8.3 Efficient Frontier

The “Efficient Frontier” method is so named because it identifies a multi-dimensional frontier of cost-efficient points. The method finds those plan alternatives that are termed non-dominated, meaning that there is no plan that provides more of all outputs for less of all inputs. The method produces a result that indicates only the order in which plans are preferred, and as such is well-suited to the analysis of qualitative (ordinal) criteria.

Within the framework of the IWR Planning Suite CEICA Module, which handles a single cost and a single input variable, the selected non-dominated plans are termed “cost-effective”. The set of points that are non-dominated is often referred to as the “Efficient Frontier” as shown in Figure 5.

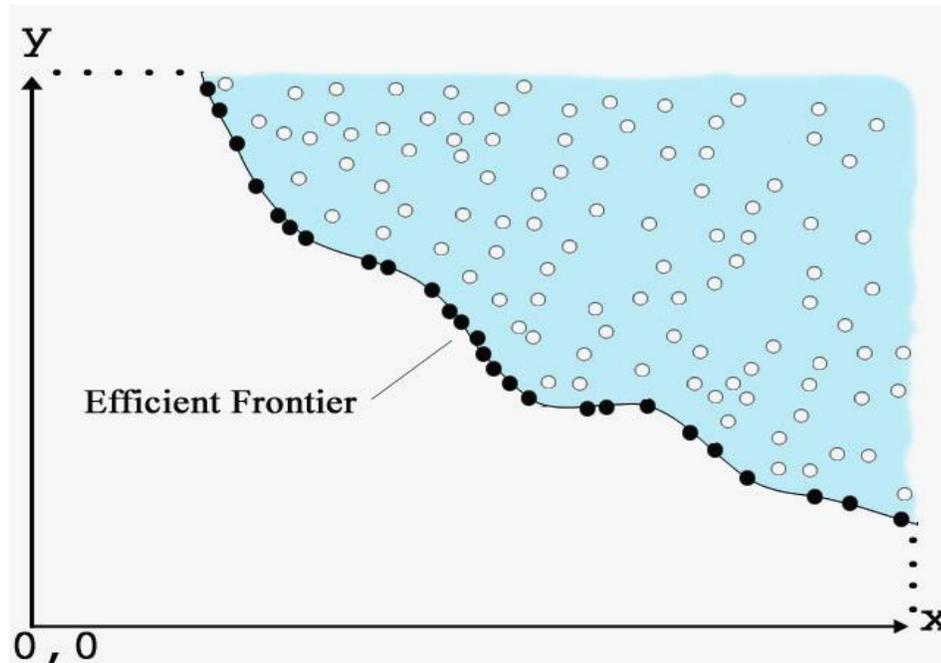


Figure 7
The Efficient Frontier of Dominant Plans

Each plan is measured on a set of attributes or criteria. Some of those criteria can be considered to be inputs (costs), while others are outputs (benefits). The problem is usually expressed as that of finding the “non-dominated” plan alternatives. Dominated plan alternatives are those for which there is at least one plan alternative that is “better”, defined as at least as much output for fewer inputs. The set of plans that are not dominated by other plans results in the cost-effective set.

Note that the concept of dominance does not imply any valuation of the inputs or outputs, only an attempt to find those plans that are unambiguously “better” than other candidate plans from the point of view of giving more output for fewer (or the same) inputs, or the same output for less input. The relative value of input and output is not considered, nor does the concept of incremental cost analysis come into play at this step.

The problem is easily understood in two dimensions, with a single input (cost) and output variable, as is handled in the IWR-Planning Suite. Consider a set of plan alternatives defined with two criteria, cost, and an environmental quality output indicator, as shown in Table 14:

Table 14.
Two Variable (Input/Output) Example

Plan	Cost	Environmental Quality Output
A	100	30
B	200	25
C	145	40
D	150	60
E	250	65
F	350	75
G	300	10
H	175	53

Dominated and Non-Dominated Plans

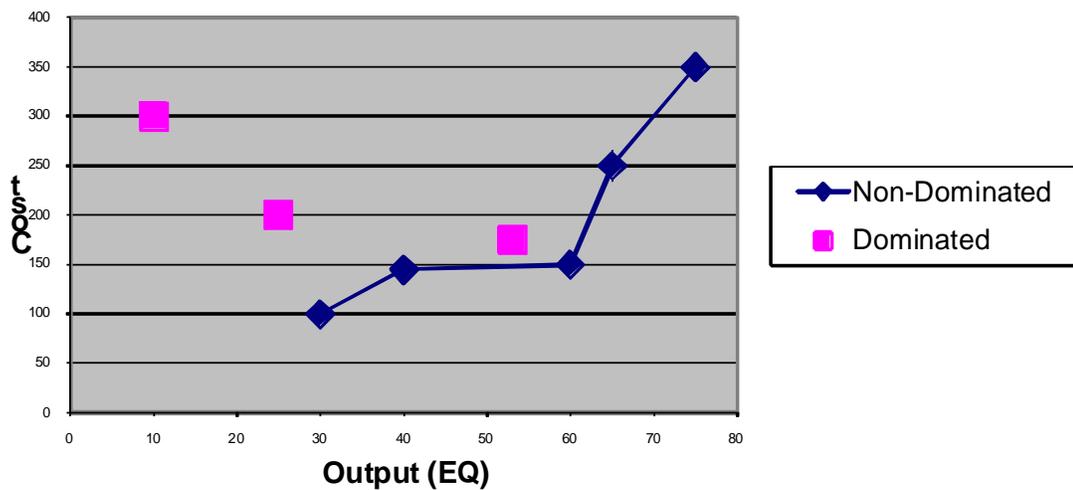


Figure 8
Dominated and Non-Dominated Plans

Plan B is dominated by Plan A – it costs more, but results in less output. There is no dominance relationship between plans A and C. Plan C costs more, but gives more output, so it is not dominated. If all plans are plotted, then the dominant plans are those shown below: Plans A, C, D, E, and F.

For multiple variables, the general concept can be extrapolated from the single-variable example outlined. That is, a pairwise comparison of plan alternatives is performed, determining which plans are dominated, and the resultant non-dominated plans comprise the efficient frontier.

2.8.4 Compromise Programming

Imagine an analytical process that results in the following decision matrix:

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H
Flood risk management (\$1000s EAD)	\$10	\$800	\$60	\$80	\$25	\$50	\$100	\$900
Annual HU's	46	0	50	90	85	75	116	116
Annual user days of urban recreation	700	0	850	2000	1000	800	3000	3000
Property value effects	2	3	2	1	3	1	2	3
Cost (\$Millions)	\$15	\$15	\$23	\$33	\$20	\$27	\$53	\$68

Sample Decision Matrix

Flood damages are measured in thousands of expected annual dollars. Habitat units and recreation user days are annual estimates. Property value effects are measured qualitatively. A 1 is a negative effect, 2 is a neutral effect and 3 is a positive effect. First costs of construction are measured in millions of dollars. All dollar values are at the same price level.

Compromise Programming Approach

The first task is to normalize the values in the decision matrix. This first example is the only one that will provide a detailed description of the calculations. The normalized values are shown in the table below:

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H
Flood risk management (\$1000s EAD)	0	0.88764	0.05617978	0.0786517	0.016854	0.044944	0.101124	1
Annual HU's	0.396552	0	0.43103448	0.7758621	0.732759	0.646552	1	1
Annual user days of urban recreation	0.233333	0	0.28333333	0.6666667	0.333333	0.266667	1	1
Property value effects	0.5	1	0.5	0	1	0	0.5	1
Cost (\$Millions)	1	1	0.8490566	0.6603774	0.90566	0.773585	0.283019	0
Distance	1.483889	1.41867	1.41453646	1.4579352	1.221204	1.620524	1.25381	1

A Normalized Decision Matrix

The methodology is a simple one and only one of many that could be used. It is based on the assumption that the contributions of the formulated plans to the decision criteria are nontrivial.

Step 1: Assign the poorest performance for a criterion a zero and the best performance a one.

Consider flood risk management in the tables. In the example, Plan A has the lowest FDR benefits and it is arbitrarily scored as a 0 to indicate it is the least desirable performance for this criterion in the choice set. Plan H with a maximum value of 900 is assigned a 1 to indicate it is the most desirable performance available in your choice set.

Step 2. Normalize the scores of each plan.

The calculations can be done in any number of ways; the one presented here is but one example. The first step is to calculate the plans' deviations from their minimum values. This is done in the table below.

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H
Flood risk management (\$1000s EAD)	0	790	50	70	15	40	90	890
Annual HU's	46	0	50	90	85	75	116	116
Annual user days of urban recreation	700	0	850	2000	1000	800	3000	3000
Property value effects	1	2	1	0	2	0	1	2
Cost (\$Millions)	0	0	8	18	5	12	38	53

Deviations from the Minimum Value

Note that Plan A is \$0 above its minimum, it is the minimum. Plan H, on the other hand, is \$890 above the minimum. The minimum, maximum and range for each criterion are identified.

	FDR	HU	UD	PV	Cost
Minimum	10	0	0	1	15
Maximum	900	116	3000	3	68
Range	890	116	3000	2	53

Selected Statistics

The basic idea now is simply to identify how much of the range in values for a criterion is covered by an individual plan. For example, consider Plan B for FDR above. From the previous tables, we see that the value for Plan B after normalization is \$790 and we see the range is \$890. Consequently, Plan B is 790/890 = 0.88764 of the way between the poorest and best performing plans. Thus, this plan is rated an 0.88764 for the FDR criterion. This method works for any criterion for which the idea is to maximize the criterion value.

The cost variable warrants special attention because it is a value that we want to minimize. The cost data from the tables above are reproduced below. Notice that the normalized values reflect the proper rankings. Plan A with the lowest cost is the plan rated best, i.e., one. Plan H with the highest cost is identified as the poorest performing plan with a zero.

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H
Cost (\$Millions)	15	15	23	33	20	27	53	68
Cost (\$Millions) Deviations from Minimum	0	0	8	18	5	12	38	53
Cost (\$Millions) Normalized Ranking	1	1	0.849057	0.660377	0.90566	0.773585	0.283019	0

The formula used to transform these values is:

$$\text{Absolute Value } [(Deviation/Range) - 1]$$

Consider Plan C. Its deviation is 8 above the minimum, a pretty good performance. The range is 53 so we obtain:

$$\text{Absolute Value } [(8/53) - 1] = 0.849057$$

Step 3: Assign weights to the criteria.

The most vexing step is likely to be coming up with the weights to reflect the relative importance of the criteria. There is no magic wand for this step. For this illustration the weights are considered given.

Step 4: Use weights and normalized decision matrix to rank the plans.

Assuming all decision criteria are equally important and equally weighted the distance of each plan from the idealized plan is given below.

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H
Flood risk management (\$1000s EAD)	0	0.88764	0.05617978	0.0786517	0.016854	0.044944	0.101124	1
Annual HU's	0.396552	0	0.43103448	0.7758621	0.732759	0.646552	1	1
Annual user days of urban recreation	0.233333	0	0.28333333	0.6666667	0.333333	0.266667	1	1
Property value effects	0.5	1	0.5	0	1	0	0.5	1
Cost (\$Millions)	1	1	0.8490566	0.6603774	0.90566	0.773585	0.283019	0
Distance	1.483889	1.41867	1.41453646	1.4579352	1.221204	1.620524	1.25381	1

Normalized Value and Distance from Idealized Plan

Bear in mind the idealized plan is a conceptual ideal. It is the plan with the highest observed performance for each criterion. In the present example it would be a plan with the following effects:

	FRM	HU	UD	PV	Cost
Ideal Plan	900	116	3000	3	15

Idealized Plan Performance

The lower the distance score the closer the actual plan is to the conceptual ideal.

Consider Plan H. The calculation of the distance is given by the following:

$$\text{Square root } [(1-1)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2 + (1-0)^2] = 1$$

Because all criteria are equally weighted the weight of one is implicit. Now imagine that it has been decided that costs are equal in importance to benefits. Assume the first four categories are considered benefits. A simple method is to choose one of the criteria to serve as a numeraire, set it equal to one. Then compare all other criteria to it.

	Plan A	Plan B	Plan C	Plan D	Plan E	Plan F	Plan G	Plan H	Weights
Flood risk management (\$1000s EAD)	0	0.88764	0.05617978	0.0786517	0.016854	0.044944	0.101124	1	0.25
Annual HUs	0.396552	0	0.43103448	0.7758621	0.732759	0.646552	1	1	0.25
Annual user days of urban recreation	0.233333	0	0.28333333	0.6666667	0.333333	0.266667	1	1	0.25
Property value effects	0.5	1	0.5	0	1	0	0.5	1	0.25
Cost (\$Millions)	1	1	0.8490566	0.6603774	0.90566	0.773585	0.283019	0	1
Distance	0.741945	0.709335	0.71924702	0.7860671	0.616044	0.83365	0.882359	1	

Weighted Distance Scores

In this example the four benefit categories would have a weight equal to the cost, which has a weight of one. Hence, the four benefit weights must sum to one. For simplicity let the four benefit categories be equally important. Note each has a weight of 0.25. The revised calculation is now:

$$\text{Square root } [.25(1-1)^2 + .25(1-1)^2 + .25(1-1)^2 + .25(1-1)^2 + 1(1-0)^2] = 1$$

By coincidence the plan that was closest to the ideal with equal weights retains the same score but it now flips from the best performing plan to the worst. Plan E is now closest to the idealized plan. It has the second lowest costs but it has more benefits than the least costly plans, A and B.

2.8.5 Outranking

The Outranking method used by the MCDA Module is PROMETHEE II. II. Promethee stands for Preference Ranking Organization METHOD for Enrichment Evaluations II. Outranking methods compare variables pairwise, developing a numerical matrix showing the degree to which each alternative is preferred to each other alternative. The numbers in this matrix can then be used to develop an ordering that shows strength of preference between alternatives. The ordering can also display incomparability between alternatives, i.e., situations in which it is not unambiguously possible to determine which alternative is to be preferred.

Within Promethee, it is necessary to define explicitly how one alternative is preferred to another on a given criterion, and then to aggregate this for all criteria. For this, the concept of indifference, and a preference index, is useful. A preference function is a number between 0 and 1, defining the degree of preference between two alternatives A and B measured on a particular criterion. If the rating of alternative A on a given criterion is equal to the rating of Alternative B on that criterion, then there is indifference, and the preference value is 0. According to traditional Corps planning approaches, an alternative with net benefits (NED) of \$1000 is strictly preferred to one whose benefits are \$999, and thus would have a value of 1.0. Such a preference function can be represented as follows:

$$\text{Let } d = \text{abs}(V(A) - V(B))$$

where $V(i)$ = criterion rating for alternative i

Then preference = 0 if $d = 0$

1 if $d > 0$

This particular preference function states that there is a strict preference for alternative A over alternative B at any level by which the rating of A exceeds the rating of B (Promethee Type I)

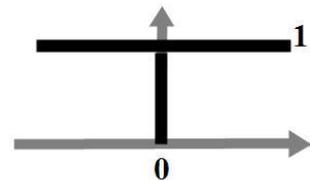
This concept can be extended by adding an indifference value, q . Then, if $d < q$, the preference is 0, otherwise it is 1. This says that A must exceed B by q units to be preferred over B (Promethee Type II). Other functions can be defined, such as a linear function, stating that as the difference between A and B moves from a value q to a value p , the preference function increases linearly from 0 to 1 over that range of differences. That is, the greater the difference past the threshold value of q , the more that A is preferred to B.

Each criteria can be assigned its own Preference Function, from the choices of Strict, Threshold, Linear Over Range, Stair-Step, and Linear with Threshold. Some of the preference functions require the user define an Indifference Value (q) and an Absolute Preference Value (p) as indicated in the detailed descriptions below.

Preference Functions

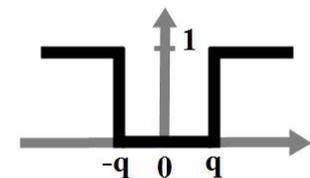
Strict

There is a strict preference for alternative A over alternative B at any level by which the rating of absolute value of A exceeds the rating of absolute value of B. Neither the Absolute Preference nor Indifference Value need be identified for this preference function. When they are equal there is no preference and the value is zero.



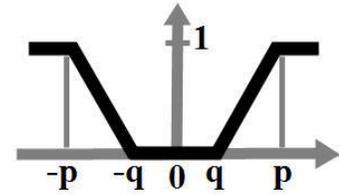
Threshold

For this method an Indifference Value must be identified. Alternative A must exceed Alternative B by an amount q greater than or equal to the indifference value (q) to be preferred over B. The user defines the indifference threshold.



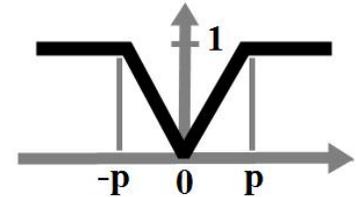
Linear With Threshold

For this method an Absolute Preference value, p , and an Indifference Value, q , must be identified. This says that any two alternatives that differ by the Indifference value or less cannot be distinguished among one another. As the difference grows closer to the Absolute Preference one alternative can be said to be better than the other alternative. Once the difference reaches the Absolute Preference, one alternative can be said to be absolutely better than another alternative.



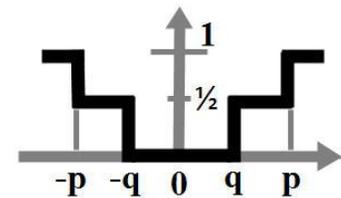
Linear Over Range

For this method an Indifference Value must be identified. This indicates that as the difference between A and B moves from a value 0 to a value p , the preference function increases linearly from zero to one over that range of differences. That is, the greater the difference past the threshold value of 0, the more that A is preferred to B.



Stairstep

For this method an Absolute Preference value, q , and an Indifference Value, p , must be identified. This says that a difference less than q gives a preference of zero, a difference between q and p gives a preference of one half, and a difference greater than p gives a preference of one.



In summary, the five types of preference supported by IWRPS MCDA are as follows:

Type		q	p
I	strict	0	0
II	threshold	indifference value	0
III	linear over range	0	value at 1.0
IV	stair-step	value at .5	value at 1.0
V	linear with threshold	indifference value	value at 1.0

Note that 0's are used as placeholders when the criterion type does not use a particular parameter.

Promethee uses the concept of flows to determine rankings between alternatives. Three kinds of multicriteria preference flows are used. They provide three ways to rate the actions and are the basis of the PROMETHEE rankings. The following description is taken from the help file of Decision Lab software:

“The positive flow ($F+$) of an action measures to which extent that action is preferred to the other ones. It is defined in the $[0,1]$ interval: a value equal to 0 indicates that the action is not preferred at all to any other one, while a value equal to 1 indicates that the action is completely preferred to all the other ones. These are of course two extreme situations that won't appear generally. The larger $F+$ the better the action.

The negative flow ($F-$) of an action measures to which extent the other actions are preferred to that action. It is also defined in the $[0,1]$ interval: a value equal to 0 indicates that no other action is preferred to the action, while a value equal to 1 indicates that all

the other actions are completely preferred to that action. These are again two extreme situations that won't appear generally. The smaller F- the better the action.

The net flow (F) of an action combines the values of F+ and F- into a single rating. It is simply defined as the difference between F+ and F- and is thus defined in the [-1, +1] interval: the best actions have positive values while the worst ones have negative values. The larger F the better the action."

In the IWRPS MCDA module, the net flow (total flow) is reported as the score, and ranking is based on that score.

2.9 Annualization Formulas

Annualizing ecosystem costs and outputs is required by U.S. Army Corps of Engineers (Corps) planning guidance. While annualizing costs is typically performed by economists, annualizing ecosystem outputs requires knowledge on the part of biologists or ecologists in terms of ecosystem response rates for various project alternatives. Responses over the period of analysis are compared to future without project conditions to estimate the "lift" or "benefit" provided by project alternatives (USACE, 2014).

The Annualizer utility allows users to interpolate NED and NER benefits and costs over the period of analysis. The utility also estimates average annual equivalent NED costs and benefits and net present values, and estimates the average annual NER outputs. The Annualizer uses well known accounting and finance formulas to produce present value costs associated with a project.

2.9.1 Capital Recovery Factor

The capital recovery factor is the ratio of a constant annuity to the present value of receiving that annuity for a given length of time. Using an interest rate i , the capital recovery factor is:

$$CRF = i(1 + i)^n / (1 + i)^n - 1$$

where:

n = the number of annuities received

http://en.wikipedia.org/wiki/Capital_recovery_factor

2.9.2 Present Value Factor

The present value (PV) Factor is used to calculate the present value per dollar that is received in the future.

$$PV\ Factor = 1 / (1 + r)^n$$

where:

P = the present value factor

r = the interest rate

n = the number of periods over which payments are made

http://en.wikipedia.org/wiki/Present_value

2.9.3 Equivalent Annual Cost

The equivalent annual cost (EAC) is the annual cost of owning an asset over its entire lifespan and is often used for capital budgeting decisions when comparing projects of unequal lifespans.

$$EAC = (\text{asset price} * d) / 1 - (1 + d)^{-n}$$

where:

d = the discount rate

n = the number of periods over which payments are made

http://en.wikipedia.org/wiki/Equivalent_annual_cost

2.10 Uncertainty Computation

The IWR Planning Suite incorporates modeling uncertainty through use of Monte Carlo simulation. Often referred to as 'Risk Analysis', the use of simulation facilitates the comparison of multiple scenarios over multiple performance measures and allows for the prediction of a distribution of results as opposed to a single predicted number or outcome.

The methodology used by the IWR Planning Suite to perform Monte Carlo analysis (including variable correlation) was provided by Dr. Richard Males of RMM Technical Services. Dr. Males has been closely involved in the implementation of Monte Carlo simulation in other planning models developed for IWR including HarborSym and Beach-fx.

The basic process for producing uncertainty planning sets is as follows:

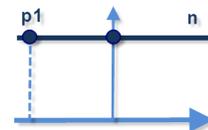
- User enters number of iterations N
- Simulation produces N-number of potential "realities" that might be realized based on user-defined distributions for variables related to:
 - Cost
 - Benefit
 - Variables that produce costs or benefits

2.10.1 Uncertainty Distributions

The IWR Planning Suite Uncertainty module supports six distribution types:

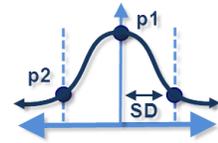
Fixed

An unchanging constant value. A single value with no variability.



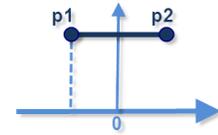
Normal

The normal distribution is the standard statistical bell curve distribution. P1 is the Mean value and P2 is the Standard Deviation.



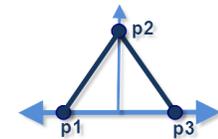
Uniform

The Continuous uniform distribution or rectangular distribution: all values between P1, the Minimum, and P2, the Maximum, are equally probable.



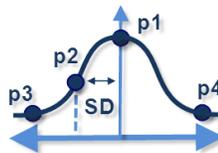
Triangular

The triangular distribution is defined by P1, a minimum returned value, P2, the most likely value, and P3, the maximum returned value.



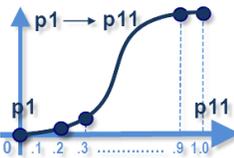
Truncated Normal

The Truncated Normal distribution is a normal distribution with P1 as the Mean and P2 as the standard deviation, but with possible values bounded by P3, a minimum, and P4, a maximum value.



Cumulative Distribution Function (CDF)

The Cumulative Distribution Function (CDF) parameters P1 through P11 describe the probability of a returned value less than the parameter. P1 corresponds to zero and P11 to 1. Other parameters have equally spaced correspondences between 0 and 1, namely, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.



2.10.2 Uncertainty Correlation Matrix

It is simplistic to assume uniform, independent distributions for uncertainty modeling as in the real world relationships (measured as correlations) oftentimes exist between variables. In fact, models that do not adequately account and allow for relationships tend to underestimate the variance of the model and therefore underestimate risk.

The IWR Planning Suite produces correlated uniform variables based on a user input correlation matrix, which is transformed based on the use of a Cholesky Decomposition, and then uses inverse distributions to obtain the needed uncertainty variables. This is consistent with the simplified approaches to perfect correlation that have been used in previous studies. The use of Cholesky Decomposition is well-established as an appropriate technique for this purpose (R Males, personal communication, January 21, 2011).

In order for Cholesky Decomposition to work, the matrix must be well-formed in that it must be symmetric and positive definite. IWR Planning Suite uses the 3rd party library Math.Net

(<http://www.mathdotnet.com/>) to perform the necessary matrix algebra. Appendix A includes an email from January 2011 from Dr. Richard Males to Dr. Dave Moser detailing the proposed (and now implemented) process.

2.10.3 Uncertainty Variable Convergence

Given that users specify their own variable distributions, then if the model is performing its random sampling correctly, it is reasonable to expect that after some unknown number of iterations the sampled data should conform to the specified distribution. For example, given a variable with a defined normal distribution with a mean of 20 and a standard deviation of 18, it should be expected that, given enough time (iterations), model samples would eventually stabilize to produce the desired curve. Figure 7 shows the running mean and standard deviation of the distribution for 1000 iterations.

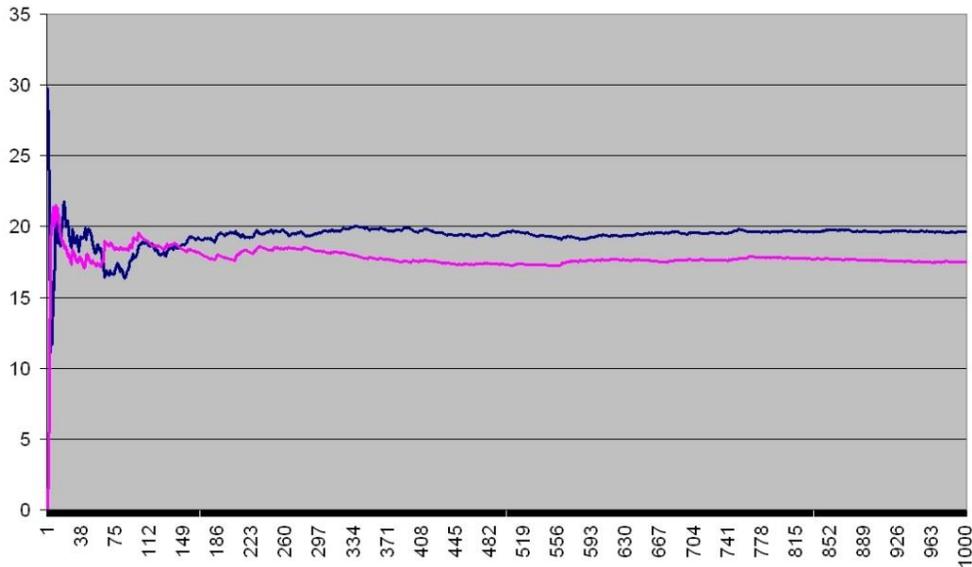


Figure 9
Running Mean and Standard Deviation During Simulation

Initially the mean and standard deviation fluctuate wildly, but eventually settle down to closely match the defined input values. The sequence has approached its limit, and the variable has reached convergence. In other words, as the number of samples increases the next sample in the sequence will become better and better modeled by a certain probability distribution (Figure 8).

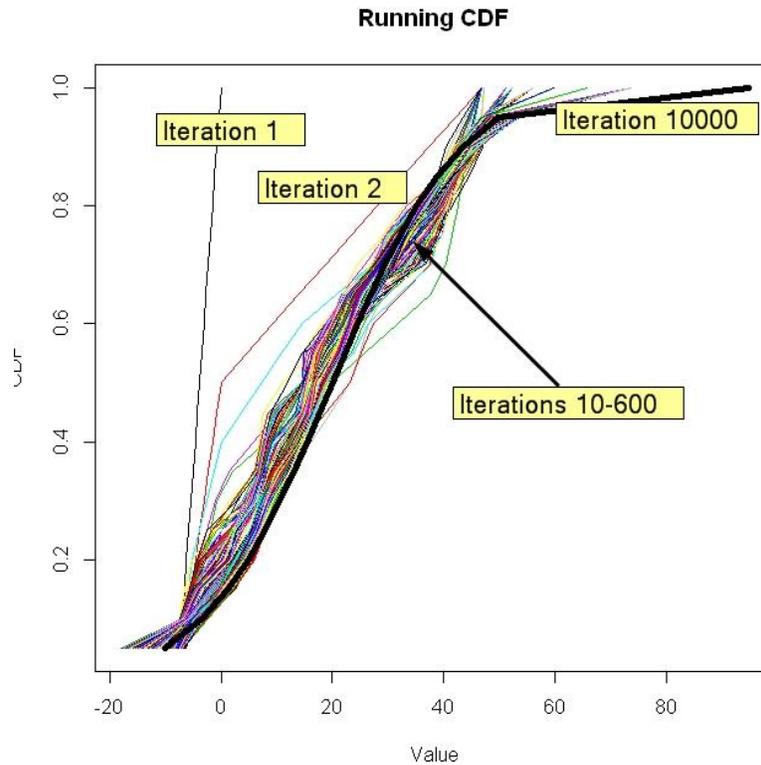


Figure 10
Running CDF During Simulation

The IWR Planning Suite allows users to check for variable convergence. Users can specify which variables they want to monitor and provide either a numeric or percentage threshold prior to running a Monte Carlo simulation.

During simulation, if a variable has been flagged for convergence, a running cumulative distribution function (CDF) is generated at every iteration based on the values sampled for the variable thus far. The running CDF is compared against the previous iteration's CDF to determine if the user-specified convergence threshold has been met. Only when all flagged variables have met their convergence thresholds will the simulation be terminated. If all flagged variables have not reached convergence, then the simulation will continue to run for the user-specified maximum number of iterations. Detailed output is generated for each variable in the form of CSV files written to a 'ConvergenceLogs' subdirectory located within the planning study's home directory.

Section 3

System Quality

System quality refers to the entire system used for model development, use, and support, including software and hardware requirements, and data interoperability or compatibility with other systems. The supporting software, programming language underlying the software, hardware and software requirements, and testing process description are discussed in this section of the report (USACE, 2011).

3.1 Programming Language and Supporting Software

The IWR Planning Suite project is a WPF application written in C# utilizing the Visual Studio 2013 Integrated Development Environment (IDE) and targets the Microsoft .NET Framework 4.5.1. It uses SQLite for all project databases, and leverages Entity Framework 6 for its object-relational mapping (ORM) tool. The 3rd party library Math.NET is used for the more complex mathematical functions including much of the matrix math used in the Uncertainty correlation routines. All non-Excel reports are displayed using Crystal Reports, and all graphs are rendered using either Gigasoft ProEssentials or Infragistics charting controls. For database diagrams and data definition language (DDL) see Appendix D.

3.2 Program Installation

The IWR Planning Suite installer is built using InstallShield®, a commercial, off-the-shelf (COTS) product which relieves the programming team of developing and supporting this part of the program. The use of a COTS product also means the installation process uses a standard and familiar interface and runs successfully on multiple versions of Microsoft Windows®. Upon completion of the installation, a new application entry will be added to the Programs menu.

3.2.1 Installation Prerequisites

The IWR Planning Suite has several prerequisites that are required to be installed in order for the application to run properly. The provided installation package will attempt to install all prerequisites before installing the model software. However, for documentation purposes each prerequisite, along with its download URL, is listed in this section.

The IWR Planning Suite has the following prerequisites:

- Microsoft .NET Framework 4.5.1 (<http://www.microsoft.com/en-us/download/details.aspx?id=40779>)
- Visual C++ Redistributables for Visual Studio 2013 (<http://www.microsoft.com/en-us/download/details.aspx?id=40784>)
- SAP Crystal Reports Runtime Engine v.13.10.1385 (<http://scn.sap.com/docs/DOC-7824>)

3.3 Availability of Hardware and Software Required

The only hardware required to run the IWR Planning Suite is a computer with an operating system of Microsoft Windows® 7 or later. It is recommended the computer have at least a 2.8 GHz processor and at

least 4 GB of RAM (although 8 GB is recommended). The software should be able to be easily downloaded from the Internet and installed using the standard COTS installation program (i.e., Windows Installer® version 4.0 or greater) present on any computer with Microsoft Windows® 7 and later versions.

3.4 Testing and Model Validation Process Description

Previous versions of IWR Planning Suite have been available and put to widespread use since early 2007. A certified version (2.0.6.0) was used for comparative analysis testing during development for previously certified modules including MCDA and the Annualizer.

Section 4

Usability

Usability refers to the ability to access the model, receive training to run the model, secure input data required for the model, run the model, obtain outputs from the model as well as receive documentation to guide the process and technical support if problems occur (USACE, 2011). It refers to the overall ease and efficiency with which users are able to operate the program and obtain the relevant information required to assist in the planning decision process. This section discusses the availability of input data, output formatting, usefulness of analytical results, exportability, training availability, user documentation, availability of technical support, availability of software/hardware platforms, accessibility of the model, and model transparency.

4.1 Availability of Input Data

The availability of input data varies based on the details of the solutions the user is evaluating. There are no special data requirements for the IWR Planning Suite, and the information included and excluded from the analysis is at the discretion of the user.

4.2 Output Format

There are several options available to display and report the output generated in the IWR Planning Suite. Tables, line graphs, scatter-plot graphs, and 3-D graphs are some of the options users have to display results. The user can reduce the number of plans displayed in CE/ICA outputs by either selecting “plans of interest” or by displaying only the cost-effective plans. Report information can be exported using the export menu function into Microsoft Word, Excel, and other software packages depending on the needs of the user.

4.3 Usefulness of Analytical Results

While the output of the IWR Planning Suite is useful to decision-makers, the model’s analyses cannot dictate the best solution to the specific planning problem. As reiterated throughout this document as well as the User Guide, the purpose of the software is to provide additional data to decision-makers, allowing them to make more informed decisions. That being stated, the IWR Planning Suite can be extremely useful in narrowing the alternative plans to a manageable number to be examined more closely by the planning team. The ability of the software to display analytical results in a variety of ways should prove helpful to users as well.

Section 5

References

U.S. Army Corps of Engineers. 2004. *IWR-Plan Planning Suite Concept Document*. Institute for Water Resources. January.

U.S. Army Corps of Engineers. 2007. *Protocols for Certification of Planning Models: Planning Models Improvement Program*. July.

U.S. Army Corps of Engineers. 2011. *Assuring Quality of Planning Models*. Engineer Circular No. 1105-2-412.

U.S. Army Corps of Engineers. 2014. *IWR Planning Suite User's Guide*. Institute for Water Resources. October.

Appendix A:

Proposed Approach for Including Correlation in IWR Planning Suite

To: Dave Moser

cc: Cory Rogers

From: R. Males

Date: 1/21/2011

Re: Correlation in IWRPlan-Risk Edition

A.1 Summary

IWRPlan-RiskEdition (IWRPlan-RE) is an extension of IWRPlan, attempting to account for variability through use of Monte Carlo simulation. Multiple iterations are run, with costs and outputs for each variable generated from a distribution. At each iteration, the cost-effective and best-buy plans are determined for that iteration, based on the values of the variables for that iteration.

During development of IWRPlan-RE, the issue of correlation between variables was brought up, but a decision was made to assume complete independence of variables. The issue of correlation has re-surfaced in the review process for IWRPlan-RE as a needed capability, and approaches and methods are being investigated. The most promising appears to be the generation of correlated uniform variables based on a user input correlation matrix, which is transformed based on use of a Cholesky Decomposition, and then the use of inverse distributions to obtain the needed IWRPlan-RE variables. This is consistent with the simplified approaches to perfect correlation that we have used in previous studies. The use of Cholesky Decomposition is well-established as an appropriate technique for this purpose.

There are some technical and data issues that need to be addressed to come up with a workable approach. Before implementing in IWRPLAN-RE, we would like to have:

1. Your review of the technical approach described below and in Appendix A;
2. Your thoughts on what we can realistically expect users to provide in terms of a correlation matrix, other than rough estimates. Is it more realistic to allow only perfect or no correlation (which greatly simplifies the generation problem)? Where would real correlation numbers come from? Since a correlation matrix needs to be well-formed, i.e., satisfy certain rules of linear algebra to be a valid/consistent correlation matrix to which a Cholesky Decomposition can be applied, we must insure that the user-provided correlation matrix is reasonable. It is easy to foresee a situation in which users would just assume correlation values and insert them into a matrix. What is the appropriate process when a user provides a badly-formed matrix? [Note that there are algorithmic methods of creating well-formed matrices from invalid matrices].

A.2 IWR Plan-RE Distributions

IWRPlan-RE currently supports the following distributions for individual variables:

- Fixed (single value, no variability)
- Normal (Mean, SD)
- Uniform (within range)
- Triangular
- User-defined Cumulative Distribution Function (CDF), described as a piece-wise linear curve given by 11 values at probabilities of 0.0, 0.1, 0.2, ... 0.9, 1.0

At present, a maximum of 10 variables can be used.

Currently unsupported, but high priority, distributions, are:

- Beta (two shape parameters)
- Truncated Normal (mean, SD, upper and lower limits)

A.3 Prior Work

We have used correlation of random variables in previous Monte Carlo Simulation efforts, under the assumption of perfect correlation. For example, damages in BeachFx are correlated between structure and contents damages. Correlation is assumed to be 1.0 between the structure and contents damages. Both structure and contents damages are represented by individual triangular distributions. A single random number on (0,1) is generated and applied to the appropriate inverse triangular distribution, in essence “looking up” the contents and structure damage at the same level of probability. I think I recall another approach of correlating two triangular distributions, where we sample from the first distribution, determine if we have drawn from the “high” or “low” side of the distribution, and then force the sampling from the second (correlated) distribution from the same side, but without the requirement that it have the same probability.

I don’t recall any situation where we have attempted to capture correlations of less than 1.0, or multiple correlations.

A.4 Correlation

The correlation matrix of n random variables X_1, \dots, X_n is the $n \times n$ matrix whose i,j entry is $\text{corr}(X_i, X_j)$, where the i,j entry is on the range $(-1,1)$, with 0 meaning no correlation (independent variables), 1 showing perfect positive correlation, and -1 showing perfect negative correlation.

http://en.wikipedia.org/wiki/Correlation_matrix#Correlation_matrices

The matrix is symmetrical with 1’s on the diagonal.

Accordingly, for a situation of 4 variables, a sample correlation matrix would appear as:

$$C = \begin{bmatrix} 1.0 & 0.7 & 0.6 & 0.6 \\ 0.7 & 1.0 & 0.6 & 0.6 \\ 0.6 & 0.6 & 1.0 & 0.8 \\ 0.6 & 0.6 & 0.8 & 1.0 \end{bmatrix}$$

Since the matrix will always be symmetrical with 1's on the diagonal, only the upper (or lower) half of the matrix is required to completely describe it. Thus, for N variables, the number of required known values is $(N*N-1)/2$, i.e., for 4 variables we need 6 values.

Such a matrix can always be calculated after the fact, i.e., given the randomly generated values of our N variables, the correlation can be determined. We are interested in the inverse problem: Given N random variables with known distributions, and a valid NxN correlation matrix, generate random values that preserve the correlation and the original distribution.

A.5 Valid Correlation Matrices

The input correlation matrix needs to be well-formed, i.e., satisfy certain rules of linear algebra to be a valid/consistent correlation matrix. The matrix must be symmetric and positive definite. Only under such conditions does the proposed approach using Cholesky Decomposition work. If a user is simply “making up” correlation estimates, we will need to have methods of testing the made up matrix for validity. It has been a long time since I took linear algebra, but there appear to be simple methods of testing a given matrix for validity (calculate the determinant, which must be non-negative, or calculate the eigenvalues which must be positive), and of converting an ill-formed matrix into a valid matrix (<http://www.risklatte.com/features/quantKnow070224.php>). The problem of starting out with an invalid correlation matrix appears to be common, in particular when a user sets values in the matrix through manual adjustment. Some further research is required, but this is not expected to be a critical issue. The remainder of this discussion will assume that the correlation matrix is well-formed.

A.6 Technical Approach

Based on internet research, this is a challenging problem in the general case. The most basic solution is to use Cholesky Decomposition, which should be suitable for our purposes. [Other solutions involve “Gibbs Sampling” or “Copulas”, which appear to be more complex than we can handle, or than I can understand]. The Cholesky approach involves transforming the given correlation matrix through a standard procedure for which many implementations are available, and then post-multiplying a matrix of uncorrelated variables by the transformed correlation matrix, to yield a set of correlated random variables. As noted above, the matrix needs to be symmetric and positive semi-definite. Code for checking this condition is also readily available, and there are also numerous examples of methodology for turning an invalid correlation matrix into a valid matrix.

Cholesky decomposition works easily and simply for normally distributed variables. Many examples are available, using R or Matlab. (<http://comisef.wikidot.com/tutorial:correlation>).

The problem is somewhat different when working with random uniform variables. This requires two additional steps. Initially, normal random variables are used. The correlation matrix, however, is adjusted

for uniform variables prior to the calculation of the Cholesky matrix. This matrix is then applied to the normal variables, which are then transformed to uniform variables using the normal distribution function.

The approach is given in Appendix B, copied from <http://comisef.wikidot.com/tutorial:correlateduniformvariates> (Schumann). I have successfully implemented the example tests provided there, in both R and Matlab, and done extensions to see how this would work with triangular distributions.

The basic steps, as I understand them, are as follows:

1. We have a correlation matrix C for N variables.
2. We have the definition of the distribution for each of the N input variables, as distribution type and parameter. We also have an inverse distribution corresponding to each such distribution (e.g., given a value between 0 and 1, return the numeric value from the parameterized distribution. This is available in various algorithms, or through the cumulative distribution function (CDF).
3. We wish to do K iterations.
4. Generate N separate vectors of length K, containing randomly generated normal variables (mean 0, SD=1). Combine these into a matrix X of K rows, N columns
5. We wish to convert this matrix X into a matrix Y' of correlated uniform random variables.
6. Once we have Y', we will back convert this to the values reflective of the true distributions of our original input variables, to obtain Z, our desired output matrix.
7. To get to Y', we only need to know about the input correlation matrix C, the number of variables N, and the number of iterations K. We do not need to know anything about the input distributions.
8. Adjust the correlations in the matrix C as follows for all non-diagonal elements:

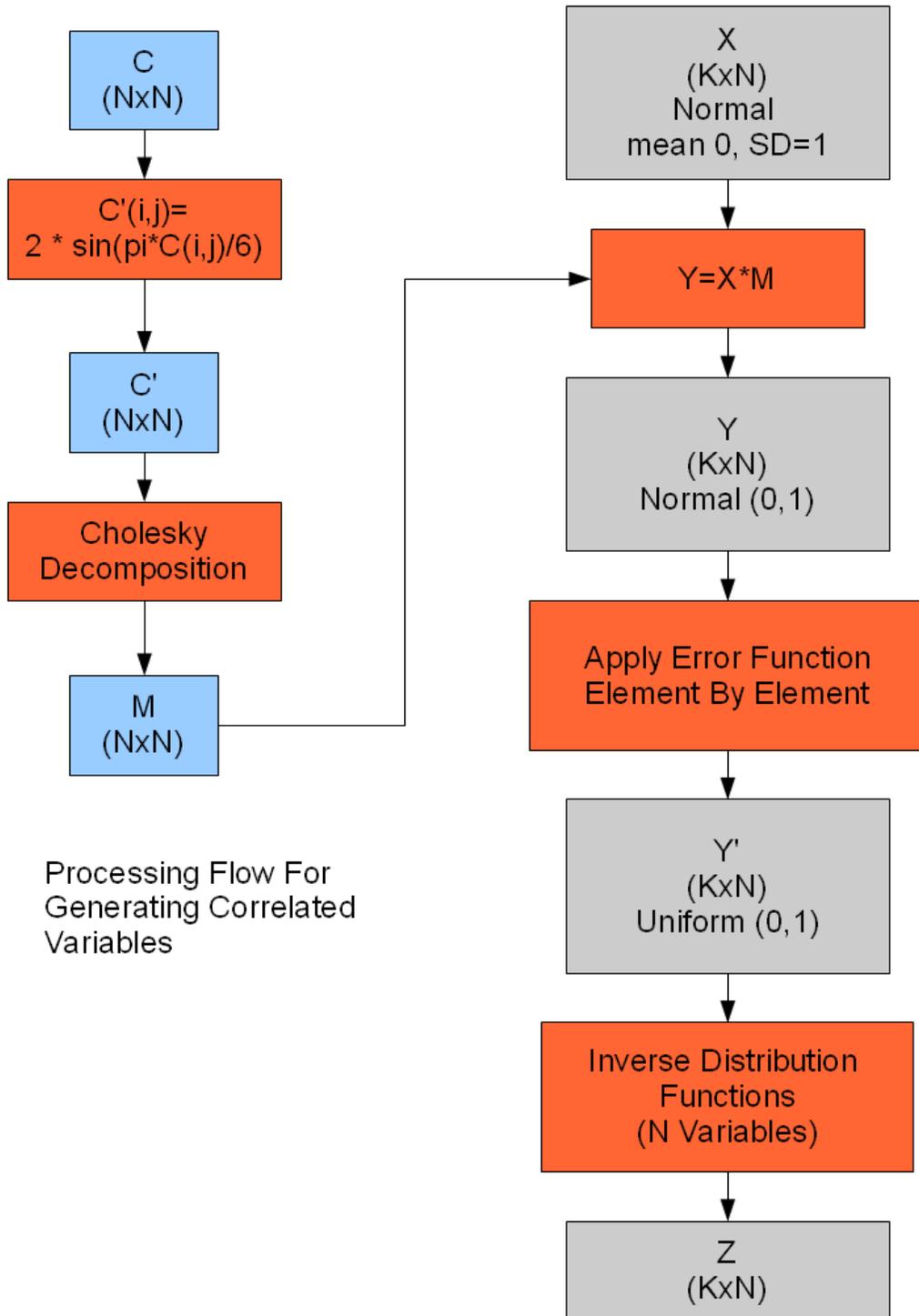
$$C(i,j) = 2 * \sin(\pi * C(i,j) / 6)$$

Per Schumann, this transformation converts the Spearman correlations in the original matrix into “Bravais-Pearson” correlations.

9. This gives a revised correlation matrix C', where all off-diagonal elements have been adjusted per the above equation.
10. Determine the Cholesky matrix M associated with C'. This can be done using calculations in R and MATLAB® by use of native functions, or C# (see the implementation section, below).
11. Transform matrix X by post-multiplying by the Cholesky matrix M to obtain matrix Y.
12. For each element in matrix Y, use the probability integral transform for a normal variable to back-calculate a probability value. This is called the probability integral transform, and, I believe, the error function. References to code for this given below, testing needed. Apparently there is a version in Math.net. That is, enter the normal (mean 0, SD 1) density function at the value in Y[i,j], which will give a probability value between 0 and 1. Store that value in the results matrix Y'.

13. Y' can be tested column-wise for each variable to check that the variables are uniformly distributed on (0,1), and the correlation matrix of Y' can be determined to see that it reproduces the original correlation matrix C .
14. At this point, we can use the values of $Y'[i,j]$ with inverse distributions for the actual distributions of our variables. Recall that these can be fixed, normal, uniform, triangular, and custom CDF.
 - a. Fixed values do not require any processing, and should not have been in the original correlation matrix in the first place.
 - b. Normal and Triangular distributions can be handled through the inverse normal and inverse triangular distributions, given the input value.
 - c. The CDF value can also be directly calculated given the uniform variable
 - d. For distributions such as a truncated normal, it can be pre-processed to obtain the associated CDF, and use that CDF as input at this step, since I am not sure we have a closed-form solution for an inverse truncated normal.

This results in the final matrix Z , with values that should reflect both the correlation and the distributions.



I have gone through this approach, testing in R with triangular distributions, and getting back what I expected. The results of my testing, using 10000 iterations and 4 variables, transforming to 4 triangular distributions, and following the approach above, are shown in Appendix B. The comparison of the original correlation matrix, and the final correlation matrix, after all the transformations have been accomplished, is:

Original Input Correlation Matrix C:

```

      [,1] [,2] [,3] [,4]
[1,] 1.0  0.7  0.6  0.6
[2,] 0.7  1.0  0.6  0.6
[3,] 0.6  0.6  1.0  0.8
[4,] 0.6  0.6  0.8  1.0

```

Correlation Matrix of final transformed variables (Mtri):

```

cor(Mtri)
      atri      btri      ctri      dtri
atri 1.0000000 0.7148652 0.6192860 0.6188518
btri 0.7148652 1.0000000 0.6171815 0.6203829
ctri 0.6192860 0.6171815 1.0000000 0.8109147
dtri 0.6188518 0.6203829 0.8109147 1.0000000

```

Residuals – difference between Input Correlation Matrix and Correlation Matrix of Transformed Variables:

```

cor(Mtri) - C
      atri      btri      ctri      dtri
atri 0.00000000 0.01683479 0.02740248 0.01832946
btri 0.01683479 0.00000000 0.02097035 0.02081596
ctri 0.02740248 0.02097035 0.00000000 0.01608053
dtri 0.01832946 0.02081596 0.01608053 0.00000000

```

This seems to be reasonably close.

This process may seem somewhat complicated, but in fact should be fairly straightforward to implement. Further testing, of course, will be required.

A.7 Implementation Notes

These are implementation notes primarily for Cory and CDM, but I wanted to set these references down here, since there is little point in pursuing this path unless we can get it working in the chosen programming environment.

IWRPlan-RE is coded in C# using the .Net framework. Libraries are available that implement the needed matrix algebra for Cholesky Decomposition, including:

- NMath Library – Commercial (<http://www.centerspace.net/products/nmath/>)
- Math.Net – Open Source (<http://www.mathdotnet.com/>) (utilized in Rubble Mound Breakwater Simulation)
- CodeProject Code:
 - CMSL – Freely available source: (<http://www.codeproject.com/KB/cs/CSML.aspx>)
- Ports of the Java General Matrix Library (JAMA)
 - <http://www.codeproject.com/KB/recipes/ManetMatrix.aspx>
 - <http://www.codeproject.com/KB/recipes/psdotnetmatrix.aspx>

The specification application of these libraries for Cholesky Decomposition has not yet been tested.

Calculation of correlation coefficients may be a feature of the above code as well. The following statistics class can calculate correlation coefficients for paired vectors of observations:

<http://www.codeproject.com/KB/cs/csstatistics.aspx>. Other such classes undoubtedly exist.

I have coded (for use with Beach-Fx) an inverse triangular class in C++, and have found C# code for the inverse normal distribution at <http://stackoverflow.com/questions/1662943/standard-normal-distribution-z-value-function-in-c>

A.8 References

Generation of Correlated Variables:

<http://comisef.wikidot.com/tutorial:correlateduniformvariates>

http://www.sitmo.com/doc/Generating_Correlated_Random_Numbers

<http://www.stat.uiuc.edu/stat428/cndata.html>

http://www.columbia.edu/~mh2078/MCS04/MCS_framework_FEegs.pdf

References for Error Function in C#:

<http://numerics.mathdotnet.com/special-functions/>

<http://www.codeproject.com/KB/cs/SpecialFunction.aspx>

http://www.johndcook.com/csharp_erf.html

Reference for Quantile Function for Normal Distribution:

<http://stackoverflow.com/questions/1662943/standard-normal-distribution-z-value-function-in-c>

Appendix B:

Generating Correlated Uniform Variates

<http://comisef.wikidot.com/tutorial:correlateduniformvariates>

Author

Enrico Schumann⁵

Keywords

covariance matrices, Matlab, R, random numbers, random variables

Review Status

Reviewed; revised 2009-02-12.

B.1 Overview

The tutorial describes a method to generate uniformly distributed random variates that exhibit a pre-specified linear correlation.

B.2 The Problem

Assume one wants to create a vector U of random variates where each element is distributed uniformly over the interval $[0, 1]$, and the elements of U are linearly correlated as specified in a matrix Σ .

B.3 Solution

Assume a multivariate random variable X is defined as:

$$(1) \quad X = F^{-1}(U)$$

where F is some distribution function⁶ and $U \sim \mathcal{U}[0, 1]$. A well-known fact (sometimes used for generating random variates with non-uniform distributions) is that X will be a random variable with distribution F . Reversing this expression, one can create uniformly distributed variates from variates following other distributions by inserting the latter into their respective distribution function. Such transformations, however, will typically affect the dependence between the original variables.

Bravais-Pearson (or linear) correlation is not necessarily invariant to transformations (only to certain linear ones), thus if the original non-uniform variates X have been linearly correlated, there is no guaranty that this linear correlation will be preserved after the transformation. However, other measures of dependence are more 'robust' to transformations of the variables. One of those measures is Spearman correlation (also called rank correlation, or fractile correlation) which is invariant to any strictly increasing transformation.

⁵ Alper Odabasioglu provided helpful comments.

⁶ Of course, F or F^{-1} may not be available in closed form and hence may need to be approximated numerically.

Hence, as long as F is strictly monotonically increasing, Spearman correlation will be preserved. A useful result relating Bravais-Pearson and Spearman correlation is that if $X \sim F$ and $Y \sim G$ (where G is another distribution function), then:

$$(2) \quad \rho_{X,Y}^S = \rho_{F(X),G(Y)}^B.$$

Here ρ^B is the Bravais-Pearson correlation, the ρ^S is the Spearman correlation. So the linear correlation between the uniforms obtained from transforming the original variates equals the Spearman correlation between the original variates. (For details, see [2].) So what is needed are random variables (following some distribution) for which we can easily generate realisations with a given Spearman correlation.

Assume F is the normal distribution function. Thus, let the multivariate random variable Z be distributed as:

$$(3) \quad Z \sim \mathcal{N}(0, \Sigma).$$

Assume further that the marginals of Z are standardised, that is Σ is also the Bravais-Pearson correlation matrix of Z . (See [here](#) for how to create linearly correlated normal variates.)

For the normal distribution, the exact relationship between Spearman correlation and Bravais-Pearson correlation is known (see [3]) and given by:

$$(4) \quad \rho^B = 2 \sin\left(\frac{\pi}{6} \rho^S\right).$$

Equations (2) and (4) suggest a simple way to obtain a desired Bravais-Pearson correlation for uniforms: Set up the desired Bravais-Pearson correlation matrix Σ (these are actually Spearman correlations for the normals, but will be Bravais-Pearson correlations for the uniforms). Then, find the linear correlation matrix Σ^{adj} corresponding to the Spearman matrix Σ (i.e., adjust Σ according to Equation (4)). Create normal variates with the adjusted correlations Σ^{adj} , and transform these normals into uniforms.

Summarised in pseudocode:

$$(5) \quad \begin{array}{l} 1: \text{ set } \Sigma \\ 2: \text{ compute } \Sigma^{\text{adj}} = 2 \cdot \sin(\pi/6 \cdot \Sigma) \\ 3: \text{ create } Z \sim \mathcal{N}(0, \Sigma^{\text{adj}}) \\ 4: \text{ compute } U = F(Z) \end{array}$$

(The computations in step 2 need to be done element wise on Σ .)

In fact, even without the adjustment (step 2), the method works very well for the case normal-to-uniform, since the maximum absolute difference between ρ^B and ρ^S after the transformation is less than 0.02. For a more detailed discussion, see for instance [1].

B.4 A Matlab Implementation

The sample code creates 1,000 realisations of four correlated random variates, where the first two variates have a normal distribution and the other two are uniformly distributed.

```
% generate normals, check correlations
X = randn(1000,4);
corrcoef(X)

% desired correlation
M =[1.0 0.7 0.6 0.6;
    0.7 1.0 0.6 0.6;
    0.6 0.6 1.0 0.8;
    0.6 0.6 0.8 1.0];

% adjust correlations for uniforms
for i = 1:4
    for j = max(3,i):4
        if i ~= j
            M(i, j) = 2 * sin(pi * M(i, j) / 6);
            M(j, i) = 2 * sin(pi * M(j, i) / 6);
        end
    end
end

% induce correlation, check correlations
C = chol(M);
Y = X * C;
corrcoef(Y)

% create uniforms, check correlations
Y(:,3:4) = normcdf(Y(:,3:4));
corrcoef(Y)

% plot results (marginals)
for i=1:4
    subplot(2,2,i);
    hist(Y(:,i))
    title(['Y ', int2str(i)])
end
```

B.5 An R Implementation

The sample code creates 1,000 realisations of four correlated random variates, where the first two variates have a normal distribution and the other two are uniformly distributed.

```
# generate normals, check correlations
X <- array(rnorm(4000), dim = c(1000, 4))
cor(X)

# desired correlation
M <- c(1.0, 0.7, 0.6, 0.6, 0.7, 1.0, 0.6, 0.6, 0.6, 0.6, 1.0, 0.8, 0.6, 0.6,
0.8, 1.0)
dim(M) <- c(4, 4)

# adjust correlations for uniforms
for (i in 1:4){
```

```

    for (j in max(i, 3):4){
      if (i != j){
        M[i, j] <- 2 * sin(pi * M[i, j] / 6)
        M[j, i] <- 2 * sin(pi * M[j, i] / 6)
      }
    }
  }

# induce correlation, check correlations
C <- chol(M)
Y <- X %*% C
cor(Y)

# create uniforms, check correlations
Y[, 3:4] <- pnorm(Y[, 3:4])
cor(Y)

# plot results (marginals)
par(mfrow = c(2, 2))
for (i in 1:4){
  hist(Y[, i], main = paste("Y", i), xlab = "")
}

```

B.6 Internal Links

Concepts:

[Generating correlated normal variates](#)

B.7 External Links

References:

1. Dias, C.T.d.S., A. Samaranayaka and B. Manly (2008). On the use of correlated beta random variables with animal population modelling. *Ecological Modelling* 215, 293-300.
2. Embrechts, P., F. Lindskog and A. J. McNeil (2001). [Modelling Dependence with Copulas and Applications to Risk Management](#), in: S. T. Rachev (ed). *Handbook of Heavy Tailed Distributions in Finance*. Elsevier.
3. Hotelling, H. and M.R. Pabst (1936). Rank Correlation and Tests of Significance Involving No Assumption of Normality. *Annals of Mathematical Statistics* 7, 29-43.

Weblinks:

[E. Schumann. 'Creating rank-correlated triangular variates'](#)

Appendix C:

R Worked Example

Using the sample code from Appendix B, somewhat revised.

10000 iterations of 4 variables.

1. Original Correlation Matrix

C

```
      [,1] [,2] [,3] [,4]
[1,]  1.0  0.7  0.6  0.6
[2,]  0.7  1.0  0.6  0.6
[3,]  0.6  0.6  1.0  0.8
[4,]  0.6  0.6  0.8  1.0
```

2. Transformed Correlation Matrix CPrime

```
      [,1]      [,2]      [,3]      [,4]
[1,] 1.0000000 0.7167359 0.6180340 0.6180340
[2,] 0.7167359 1.0000000 0.6180340 0.6180340
[3,] 0.6180340 0.6180340 1.0000000 0.8134733
[4,] 0.6180340 0.6180340 0.8134733 1.0000000
```

3. Cholesky Matrix based on CPrime

M

```
      [,1]      [,2]      [,3]      [,4]
[1,]  1 0.7167359 0.6180340 0.6180340
[2,]  0 0.6973447 0.2510478 0.2510478
[3,]  0 0.0000000 0.7449893 0.4946142
[4,]  0 0.0000000 0.0000000 0.5571048
```

4. 1st 10 rows of untransformed matrix X of independent normal variables

X[1:10,]

	[,1]	[,2]	[,3]	[,4]
[1,]	-0.76536521	1.039120319	-0.52110800	-0.83879742
[2,]	0.90118204	-1.650483054	-0.52247343	-1.26281490
[3,]	-0.24657905	1.359854456	-0.25062497	1.50235860
[4,]	0.41066382	2.456016554	0.37067625	1.83273187
[5,]	0.95407258	1.549632500	0.96251686	-0.01342548
[6,]	0.50068039	2.383269986	0.04881254	-2.33460764
[7,]	0.31762701	-1.028775690	0.52674865	-0.67105187
[8,]	-1.06851384	-0.190567635	1.50619370	-0.74332249
[9,]	-0.08359724	-0.753149803	0.70768103	0.19267471
[10,]	-1.21389479	0.009902063	-0.88619478	0.16080740

5. 1st 10 rows of X Post-multiplied by Cholesky Matrix = Y

Y[1:10,]

	[,1]	[,2]	[,3]	[,4]
[1,]	-0.76536521	0.1760603	-0.600372730	-0.9371984
[2,]	0.90118204	-0.5050461	-0.246626068	-0.8193320
[3,]	-0.24657905	0.7715553	0.002281295	0.9020027
[4,]	0.41066382	2.0070277	1.146531528	2.0747472
[5,]	0.95407258	1.7644461	1.695745801	1.4472762
[6,]	0.50068039	2.0208163	0.944116951	-0.3687256
[7,]	0.31762701	-0.4897566	0.330454519	-0.1752764
[8,]	-1.06851384	-0.8987336	0.413878681	-0.3773432
[9,]	-0.08359724	-0.5851222	0.286472248	0.2166266
[10,]	-1.21389479	-0.8631368	-1.407947945	-1.0964803

6. Inverse transform using normal distribution function to get uniform distribution

```
YPrime[1:10,]
```

	[,1]	[,2]	[,3]	[,4]
[1,]	0.2220271	0.5698767	0.27412893	0.1743283
[2,]	0.8162542	0.3067632	0.40259881	0.2062985
[3,]	0.4026170	0.7798111	0.50091010	0.8164723
[4,]	0.6593405	0.9776266	0.87421236	0.9809950
[5,]	0.8299765	0.9611715	0.95503298	0.9260902
[6,]	0.6917020	0.9783506	0.82744506	0.3561661
[7,]	0.6246161	0.3121531	0.62947172	0.4304312
[8,]	0.1426444	0.1843973	0.66051852	0.3529593
[9,]	0.4666883	0.2792328	0.61274178	0.5857503
[10,]	0.1123940	0.1940311	0.07957324	0.1364343

7. Correlation matrix on all 10000 rows of YPrime

```
cor(YPrime)
```

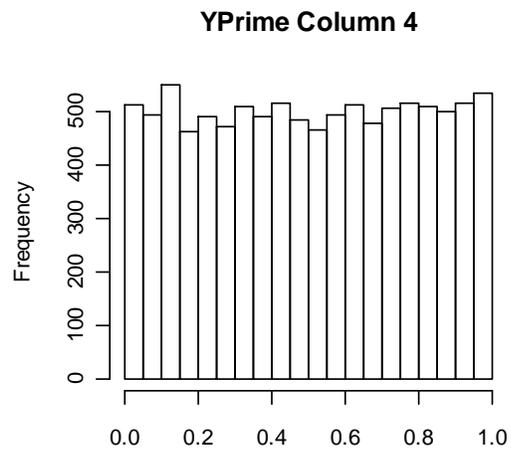
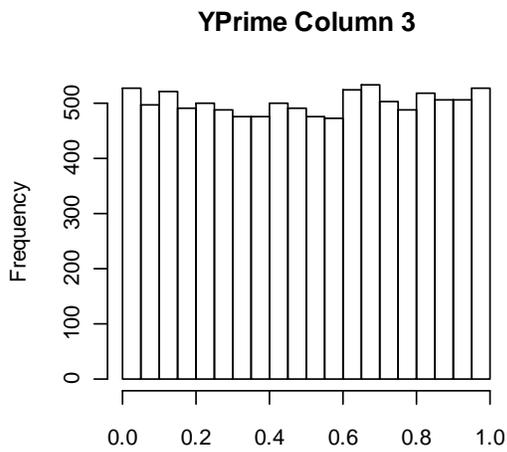
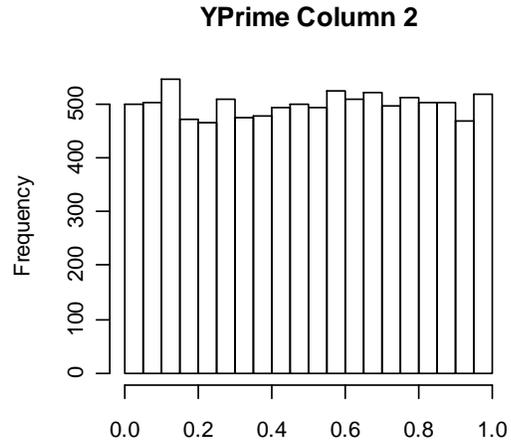
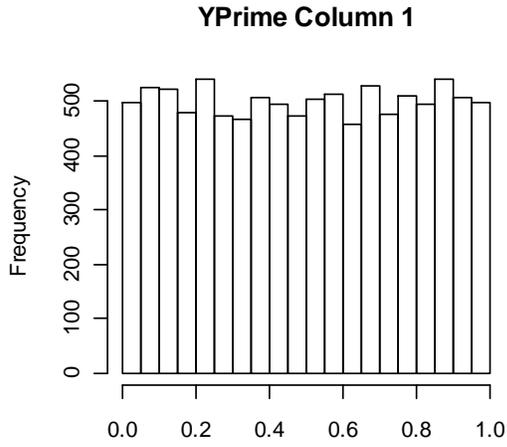
	[,1]	[,2]	[,3]	[,4]
[1,]	1.0000000	0.6910619	0.6073196	0.6082253
[2,]	0.6910619	1.0000000	0.6095116	0.6072309
[3,]	0.6073196	0.6095116	1.0000000	0.7998597
[4,]	0.6082253	0.6072309	0.7998597	1.0000000

8. Residuals (difference between original correlation matrix and correlation matrix for YPrime)

```
cor(YPrime)-C
```

	[,1]	[,2]	[,3]	[,4]
[1,]	0.000000000	-0.008938083	0.0073195776	0.0082253080
[2,]	-0.008938083	0.000000000	0.0095115803	0.0072308920
[3,]	0.007319578	0.009511580	0.0000000000	-0.0001402890
[4,]	0.008225308	0.007230892	-0.0001402890	0.0000000000

9. Distributions of Each column of YPrime



10. Transformation into 4 triangular distributions

Column	Name	Minimum	Most Likely	Maximum
1	atri	1	2	3
2	btri	2	3	4
3	ctri	6	8	10
4	dtri	100	150	200

11. 1st 10 rows of transformed matrix (transforming YPrime, step 6 above), using inverse triangular distribution function:

Mtri[1:10,]

	atri	btri	ctri	dtri
[1,]	1.600186	3.082959	8.207274	149.0160
[2,]	1.557319	2.828422	6.609534	118.6422
[3,]	1.889074	3.046369	7.786915	135.4779
[4,]	1.337878	2.410147	7.345885	108.5749
[5,]	2.543510	3.529213	9.217656	189.8622
[6,]	1.941639	3.175627	9.022913	163.6024
[7,]	1.490518	2.172649	6.996267	138.9084
[8,]	2.395466	3.514874	9.235987	162.1566
[9,]	1.505528	3.348422	7.735875	134.9158
[10,]	1.755423	3.000385	8.087079	148.9552

12. Correlation of transformed matrix

cor(Mtri)

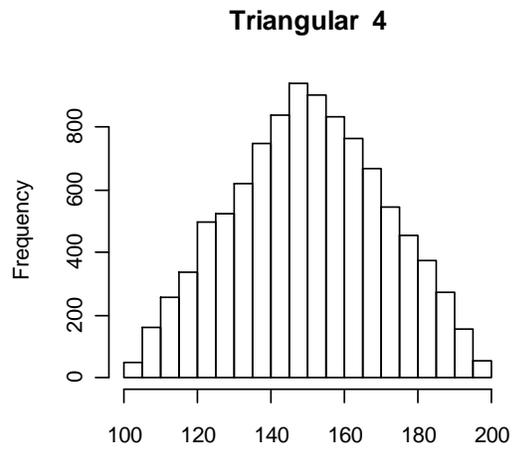
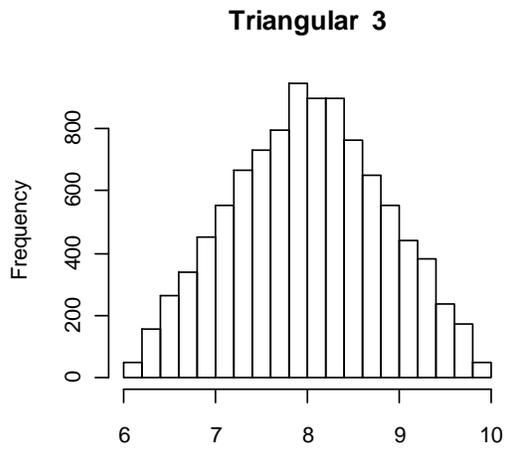
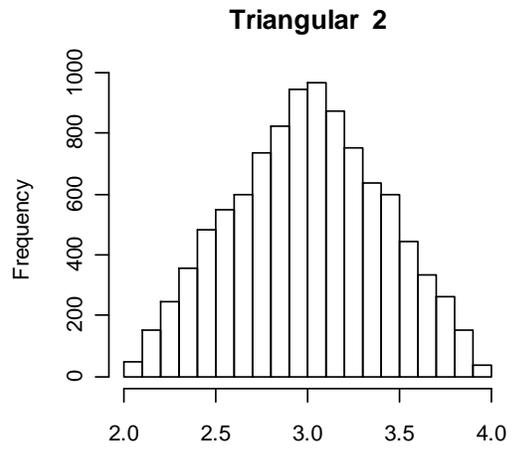
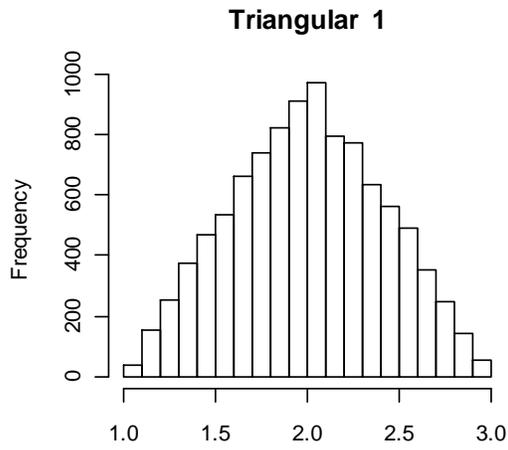
	atri	btri	ctri	dtri
atri	1.0000000	0.7168348	0.6274025	0.6183295
btri	0.7168348	1.0000000	0.6209704	0.6208160
ctri	0.6274025	0.6209704	1.0000000	0.8160805
dtri	0.6183295	0.6208160	0.8160805	1.0000000

13. Residuals, original vs. transformed

cor(Mtri) - C

	atri	btri	ctri	dtri
atri	0.00000000	0.01683479	0.02740248	0.01832946
btri	0.01683479	0.00000000	0.02097035	0.02081596
ctri	0.02740248	0.02097035	0.00000000	0.01608053
dtri	0.01832946	0.02081596	0.01608053	0.00000000

14. Distributions for each variable



Appendix D:

Database Design

D.1 Study Manager Database

Previous versions of IWR Planning Suite did not allow users to track multiple planning studies. In order to allow users to better manage their various projects as well as allow them to quickly open recent projects, a global, managing database needed to be used.

On application startup, IWR Planning Suite looks for this database in the Common Application Data folder (on Windows® 7 C:\ProgramData). If no database is found, one is created. This database contains a single table intended to track planning study databases created and accessed on the user's machine.

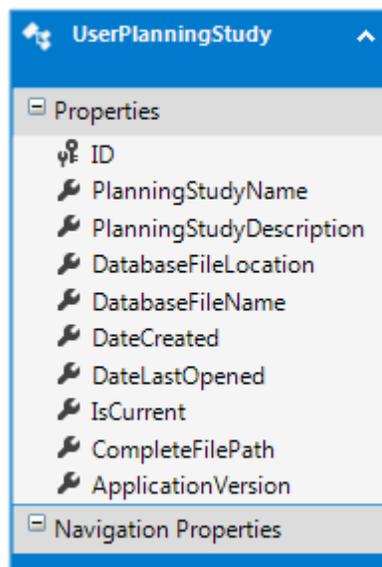


Figure 11
Study Manager Database Diagram

D.2 Study Manager DDL

The DDL for the managing database is included below:

```
CREATE TABLE [UserPlanningStudies] (  
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    [PlanningStudyName] TEXT(500),  
    [PlanningStudyDescription] TEXT(500),  
    [DatabaseFileLocation] TEXT NOT NULL,  
    [DatabaseFileName] TEXT(200) NOT NULL,  
    [DateCreated] DATETIME NOT NULL,  
    [DateLastOpened] DATETIME,  
    [IsCurrent] BOOLEAN NOT NULL DEFAULT 0,
```

```
[CompleteFilePath] TEXT(500) NOT NULL,  
[ApplicationVersion] TEXT(255));  
  
CREATE INDEX [IDX_UserPlanningStudies_DateLastOpened] ON  
[UserPlanningStudies] ([DateLastOpened]);  
  
CREATE INDEX [IDX_UserPlanningStudies_IsCurrent] ON [UserPlanningStudies]  
([IsCurrent]);  
  
CREATE UNIQUE INDEX [UIDX_UserPlanningStudies_CompleteFilePath] ON  
[UserPlanningStudies] ([CompleteFilePath]);
```

D.3 Planning Study Database

As with previous versions of IWR Planning Suite, a separate database is created for each planning study. This database contains information specific to that planning study, and is fairly complex given the number of tables needed to support the various modules. In order to improve the readability of the associated entity relationships, the diagram for the database has been split into multiple selections.

D.3.1 Plan Editor Diagram

The database diagram in Figure 10 contains all the core data structures shared across all modules of the application such as variables, attributes, and planning sets.

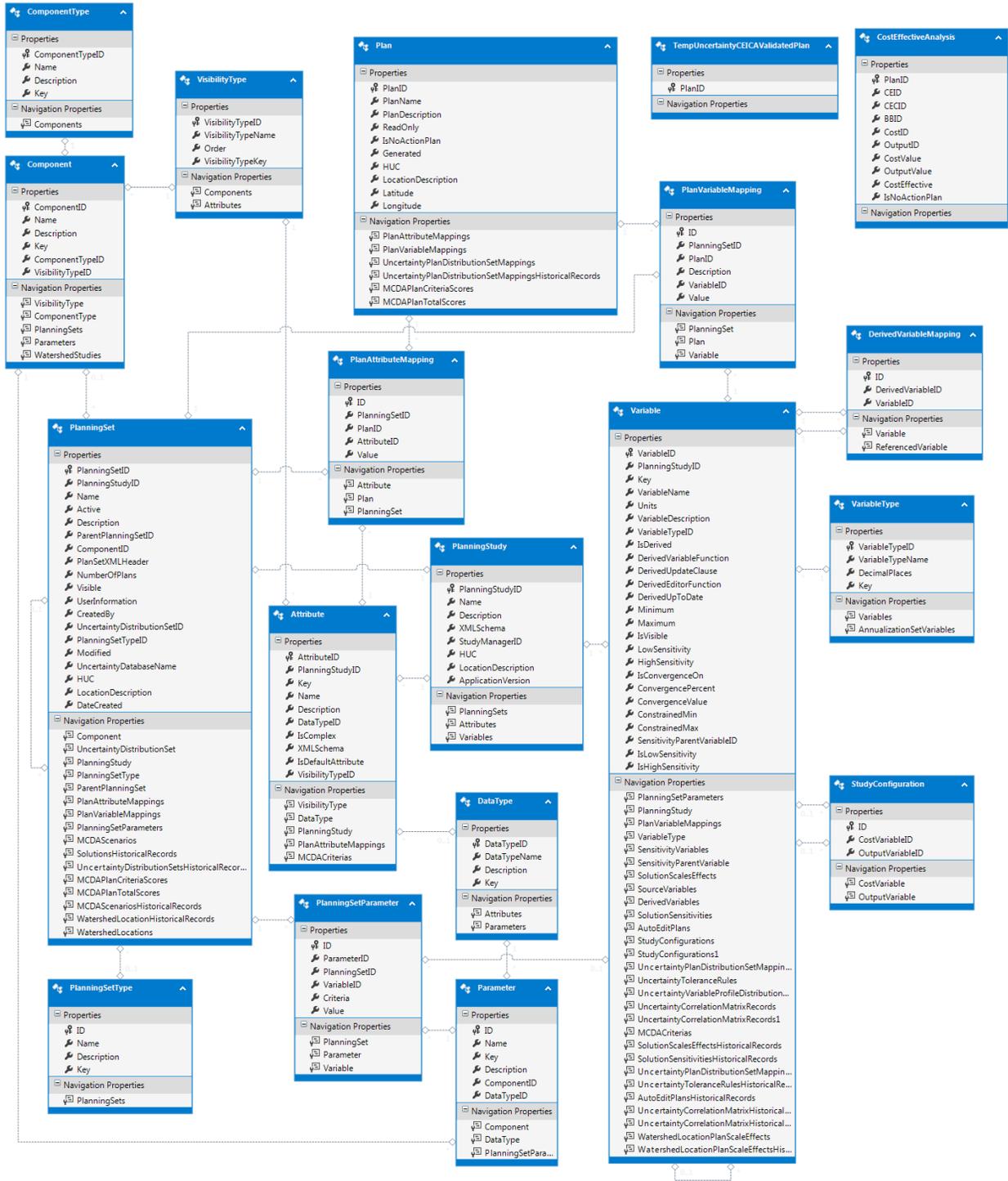


Figure 12
Plan Editor Diagram (Core Tables)

D.3.2 Plan Generator Diagram

The database diagram in Figure 11 contains all the data structures required by the Generator Module.

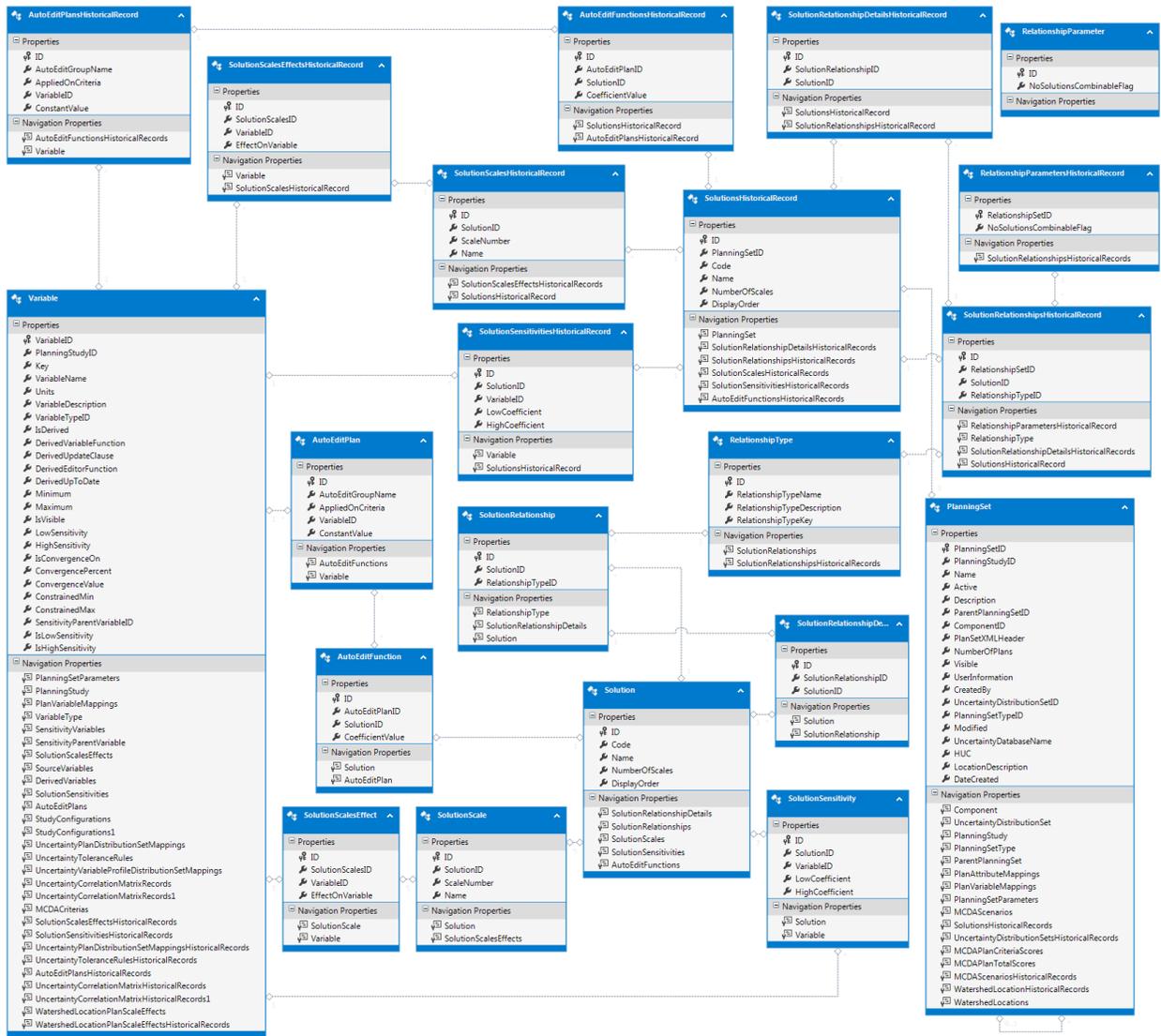


Figure 13
Generator Diagram

D.3.4 Uncertainty Diagram

The database diagram in Figure 13 contains all the data structures required by the Uncertainty Module.

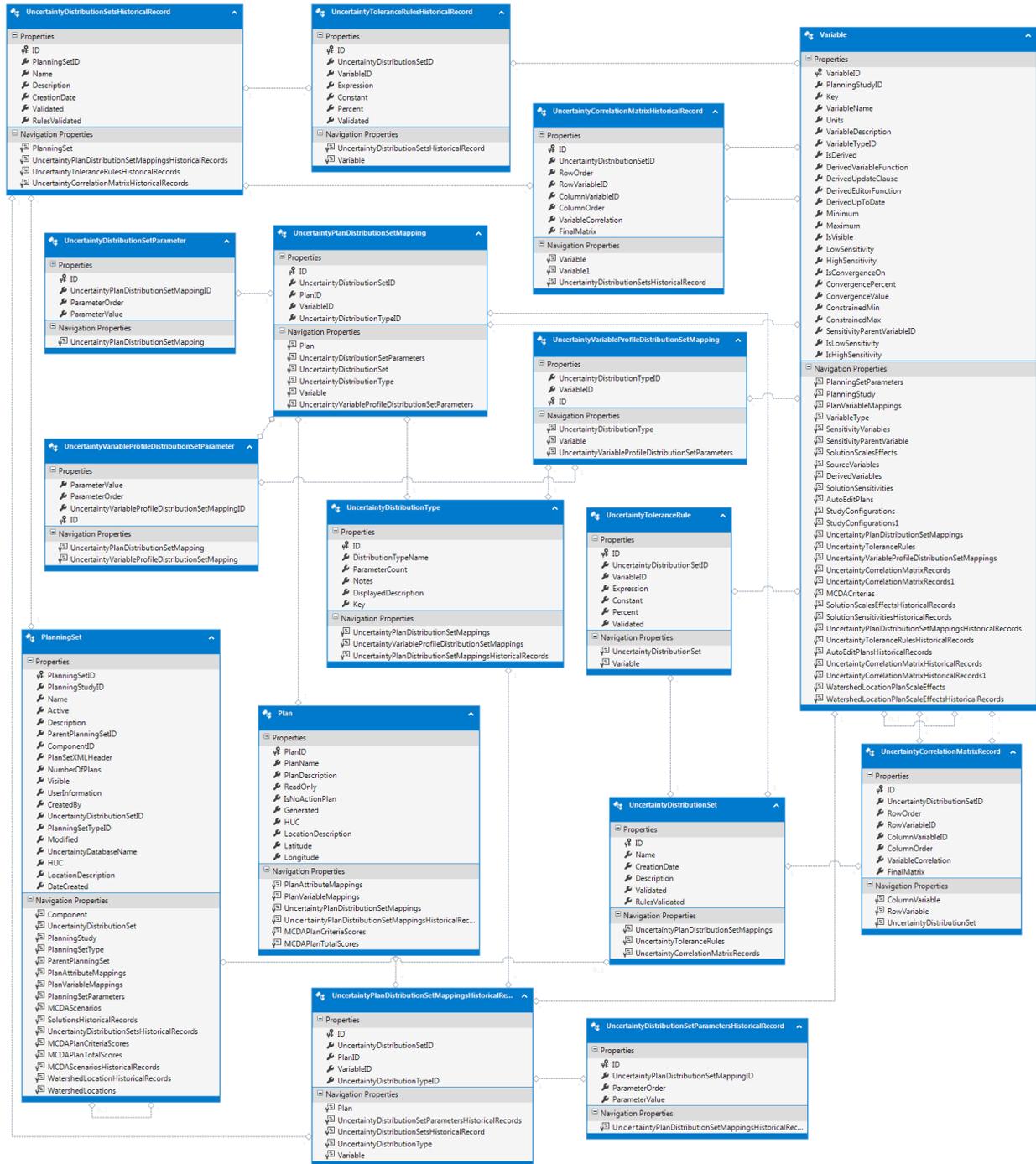


Figure 15
Uncertainty Diagram

D.3.5 Annualizer Diagram

The database diagram in Figure 14 contains all the data structures required by the Annualizer tool.

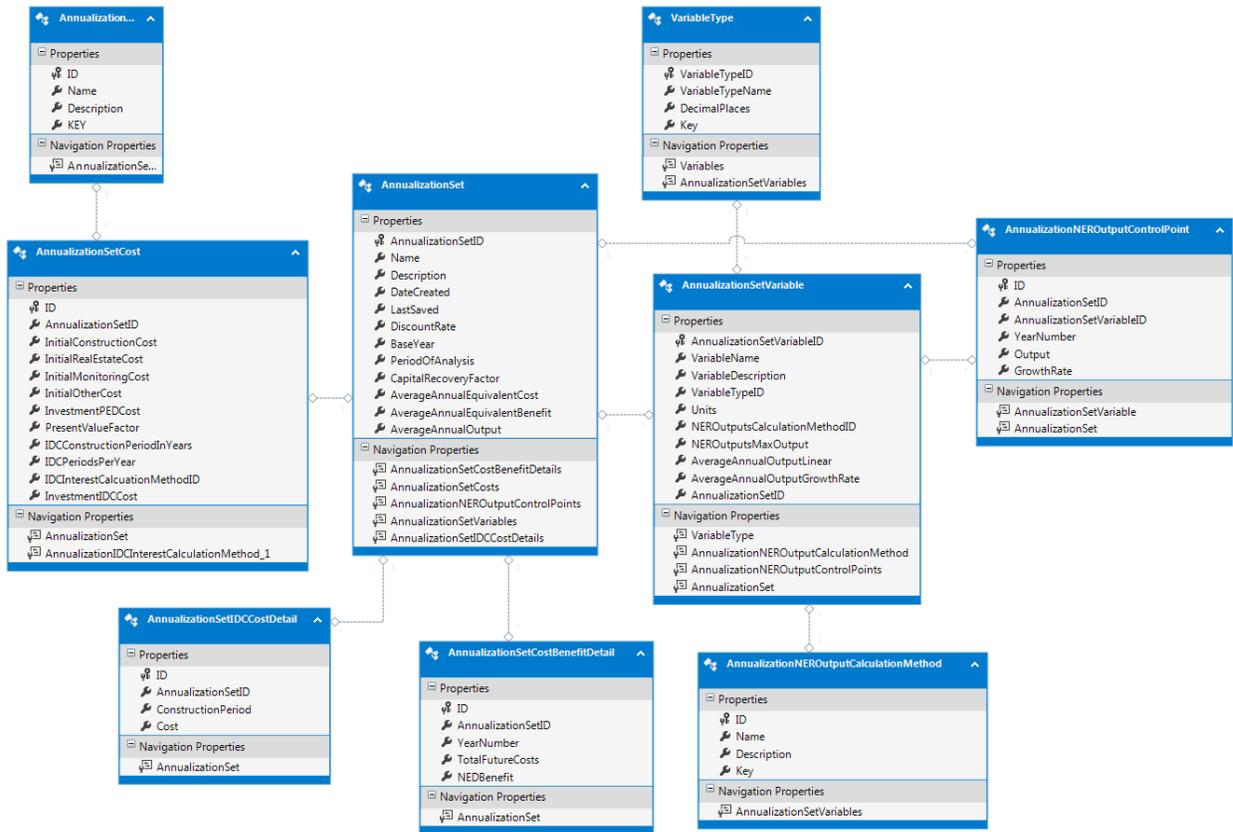


Figure 16
Annualizer Diagram

D.3.6 Watershed Diagram

The database diagram in Figure 15 contains all the data structures required by the Watershed Wizard.

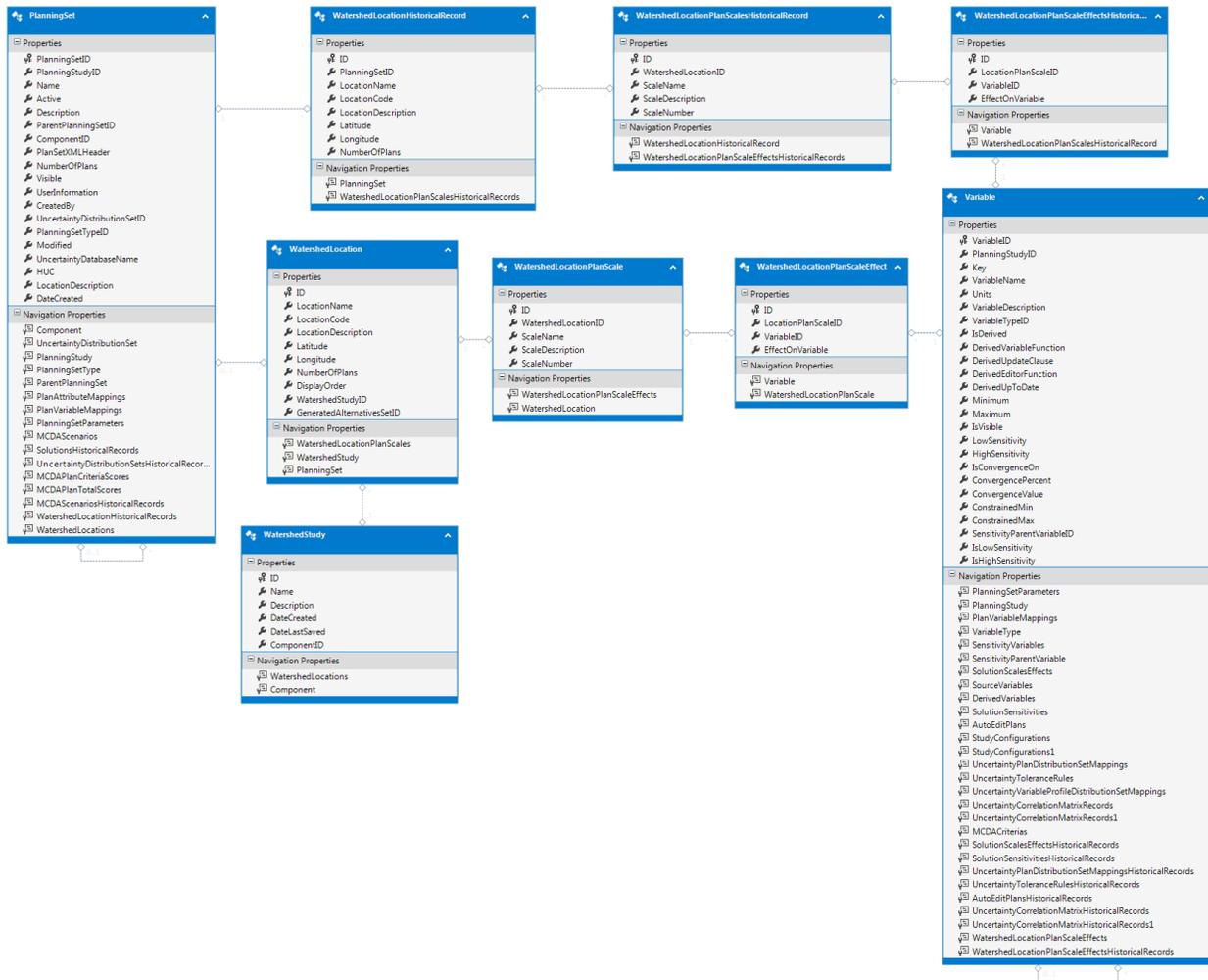


Figure 17
Watershed Diagram

D.3.7 Planning Study Database DDL

The DDL for the Planning Study Database is included below:

```
CREATE TABLE [AnnualizationIDCInterestCalculationMethods] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [KEY] TEXT NOT NULL);
```

```
CREATE UNIQUE INDEX [UIDX_AnnualizationIDCCalcMethod_Name] ON
[AnnualizationIDCInterestCalculationMethods] ([Name]);
```

```

CREATE UNIQUE INDEX [UIDX_AnnualizationICDCalcMethod_Key] ON
[AnnualizationIDCInterestCalculationMethods] ([KEY]);

CREATE TABLE [AnnualizationNEROutputCalculationMethods] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [Key] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_NEROutputCalculationMethods_Name] ON
[AnnualizationNEROutputCalculationMethods] ([Name]);

CREATE UNIQUE INDEX [UIDX_NEROutputCalculationMethods_Key] ON
[AnnualizationNEROutputCalculationMethods] ([Key]);

CREATE TABLE [AnnualizationSets] (
  [AnnualizationSetID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [DateCreated] DATETIME NOT NULL DEFAULT current_timestamp,
  [LastSaved] DATETIME NOT NULL DEFAULT current_timestamp,
  [DiscountRate] DOUBLE(18, 2) NOT NULL DEFAULT 0,
  [BaseYear] INTEGER NOT NULL,
  [PeriodOfAnalysis] INTEGER NOT NULL DEFAULT 50,
  [CapitalRecoveryFactor] DOUBLE(18, 6) NOT NULL DEFAULT 0,
  [AverageAnnualEquivalentCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
  [AverageAnnualEquivalentBenefit] DOUBLE(18, 2) NOT NULL DEFAULT 0,
  [AverageAnnualOutput] DOUBLE(18, 2) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX [UIDX_AnnualizationSets_Name] ON [AnnualizationSets]
([Name]);

CREATE TABLE [VariableTypes] (
  [VariableTypeID] INTEGER PRIMARY KEY AUTOINCREMENT,
  [VariableTypeName] [TEXT(255)] NOT NULL,
  [DecimalPlaces] INTEGER NOT NULL DEFAULT 0,
  [Key] [TEXT(255)] NOT NULL);

CREATE INDEX [IDX_VariableTypes_VariableTypeName] ON [VariableTypes]
([VariableTypeName]);

CREATE UNIQUE INDEX [IDX_VariableTypes_Key] ON [VariableTypes] ([Key]);

CREATE TABLE [AnnualizationSetVariables] (
  [AnnualizationSetVariableID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [AnnualizationSetID] INTEGER NOT NULL CONSTRAINT
  [FK_AnnualizationSetVariables_AnnualizationSets] REFERENCES
  [AnnualizationSets] ([AnnualizationSetID]),

```

```

[VariableName] TEXT NOT NULL DEFAULT 255,
[VariableDescription] TEXT(500),
[VariableTypeID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationVariables_VariableTypeID] REFERENCES
[VariableTypes]([VariableTypeID]),
[Units] TEXT(255),
[NEROutputsCalculationMethodID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationVariables_NEROutputsCalculationID] REFERENCES
[AnnualizationNEROutputCalculationMethods]([ID]) DEFAULT 1,
[NEROutputsMaxOutput] DOUBLE(18, 3) NOT NULL DEFAULT 100,
[AverageAnnualOutputLinear] DOUBLE(18, 3) NOT NULL DEFAULT 0,
[AverageAnnualOutputGrowthRate] DOUBLE(18, 3) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_AnnualizationSetVariables_AnnualizationSetID_VariableName] ON
[AnnualizationSetVariables] ([AnnualizationSetID], [VariableName]);

CREATE TABLE [AnnualizationNEROutputControlPoints] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [AnnualizationSetID] INTEGER NOT NULL CONSTRAINT
[FK_NEROutputControlPoints_AnnualizationSetID] REFERENCES
[AnnualizationSets]([AnnualizationSetID]),
  [AnnualizationSetVariableID] INTEGER NOT NULL CONSTRAINT
[FK_NEROutputControlPoints_VariableID] REFERENCES
[AnnualizationSetVariables]([AnnualizationSetVariableID]),
  [YearNumber] INTEGER NOT NULL,
  [Output] DOUBLE(18, 2) NOT NULL DEFAULT 0,
  [GrowthRate] DOUBLE(18, 2) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_NEROutputControlPoints_AnnualizationSetID_VariableID_YearNumber] ON
[AnnualizationNEROutputControlPoints] ([AnnualizationSetID],
[AnnualizationSetVariableID], [YearNumber]);

CREATE TABLE [AnnualizationSetCostBenefitDetails] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [AnnualizationSetID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationSetCostBenefitDetails_AnnualizationSetID] REFERENCES
[AnnualizationSets]([AnnualizationSetID]),
  [YearNumber] INTEGER NOT NULL,
  [TotalFutureCosts] DOUBLE(18, 2) NOT NULL DEFAULT 0,
  [NEDBenefit] DOUBLE(18, 2) NOT NULL DEFAULT 0);

CREATE INDEX [IDX_AnnualizationSetCostBenefitDetails_AnnualizationSetID] ON
[AnnualizationSetCostBenefitDetails] ([AnnualizationSetID]);

CREATE UNIQUE INDEX
[UIDX_AnnualizationSetCostBenefitDetails_AnnualizationSetID_YearNumber] ON
[AnnualizationSetCostBenefitDetails] ([AnnualizationSetID], [YearNumber]);

```

```

CREATE TABLE [AnnualizationSetCosts] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [AnnualizationSetID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationSetCost_AnnualizationSets] REFERENCES
[AnnualizationSets]([AnnualizationSetID]),
    [InitialConstructionCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [InitialRealEstateCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [InitialMonitoringCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [InitialOtherCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [InvestmentPEDCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [InvestmentIDCCost] DOUBLE(18, 2) NOT NULL DEFAULT 0,
    [PresentValueFactor] DOUBLE NOT NULL DEFAULT 1,
    [IDCConstructionPeriodInYears] INTEGER NOT NULL DEFAULT 10,
    [IDCPeriodsPerYear] INTEGER NOT NULL DEFAULT 2,
    [IDCInterestCalcuationMethodID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationSetCosts_IDCInterestCalcuationMethod] REFERENCES
[AnnualizationIDCInterestCalculationMethods]([ID]) DEFAULT 2);

CREATE INDEX [IDX_AnnualizationSetCosts_IDCInterestCalculationMethod] ON
[AnnualizationSetCosts] ([IDCInterestCalcuationMethodID]);

CREATE UNIQUE INDEX [UIDX_AnnualizationSetCost_AnnualizationSetID] ON
[AnnualizationSetCosts] ([AnnualizationSetID]);

CREATE TABLE [AnnualizationSetIDCCostDetails] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [AnnualizationSetID] INTEGER NOT NULL CONSTRAINT
[FK_AnnualizationSetIDCCostDetai_AnnualizationSet] REFERENCES
[AnnualizationSets]([AnnualizationSetID]),
    [ConstructionPeriod] INTEGER NOT NULL,
    [Cost] DOUBLE NOT NULL DEFAULT 0);

CREATE INDEX [IDX_AnnualizationSetIDCCostDetai_AnnualizationSet] ON
[AnnualizationSetIDCCostDetails] ([AnnualizationSetID]);

CREATE UNIQUE INDEX
[UIDX_AnnualizationSetICDCostDetails_AnnualziationSetID_ConstructionPeriod]
ON [AnnualizationSetIDCCostDetails] ([AnnualizationSetID],
[ConstructionPeriod]);

CREATE TABLE [DataTypes] (
    [DataTypeID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [DataTypeName] [TEXT(255)] NOT NULL,
    [Description] [TEXT(500)],
    [Key] [TEXT(255)] NOT NULL);

CREATE UNIQUE INDEX [UIDX_DataTypes_DataTypeName] ON [DataTypes]
([DataTypeName]);

```

```

CREATE UNIQUE INDEX [UIDX_DataTypes_Key] ON [DataTypes] ([Key]);

CREATE TABLE [PlanningStudy] (
    [PlanningStudyID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT(255),
    [Description] TEXT(500),
    [HUC] TEXT(12),
    [LocationDescription] TEXT(500),
    [XMLSchema] MEMO,
    [StudyManagerID] INTEGER,
    [ApplicationVersion] TEXT(255));

CREATE INDEX [IDX_PlanningStudy_StudyManagerID] ON [PlanningStudy]
([StudyManagerID]);

CREATE TABLE [VisibilityTypes] (
    [VisibilityTypeID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [VisibilityTypeName] [TEXT(255)] NOT NULL,
    [Order] [NUMERIC(18, 2)] NOT NULL,
    [VisibilityTypeKey] [TEXT(255)] NOT NULL);

CREATE INDEX [IDX_VisibilityTypes_Order] ON [VisibilityTypes] ([Order]);

CREATE UNIQUE INDEX [UIDX_VisibilityTypes_Name] ON [VisibilityTypes]
([VisibilityTypeName]);

CREATE TABLE [Attributes] (
    [AttributeID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [PlanningStudyID] INTEGER NOT NULL CONSTRAINT
[FK_Attributes_PlanningStudyID] REFERENCES
[PlanningStudy]([PlanningStudyID]),
    [Key] [TEXT(255)] NOT NULL,
    [Name] [TEXT(255)] NOT NULL,
    [Description] [TEXT(500)],
    [DataTypeID] INTEGER CONSTRAINT [FK_Attributes_DataTypeID] REFERENCES
[DataTypes]([DataTypeID]),
    [IsComplex] BOOLEAN NOT NULL DEFAULT 0,
    [XMLSchema] [TEXT(255)],
    [IsDefaultAttribute] BOOLEAN NOT NULL DEFAULT 0,
    [VisibilityTypeID] INTEGER NOT NULL CONSTRAINT
[FK_Attributes_VisibilityTypeID] REFERENCES
[VisibilityTypes]([VisibilityTypeID]));

CREATE INDEX [IDX_Attributes_DataTypeID] ON [Attributes] ([DataTypeID]);

CREATE INDEX [IDX_Attributes_IsDefaultAttribute] ON [Attributes]
([IsDefaultAttribute]);

```

```

CREATE INDEX [IDX_Attributes_PlanningStudyID] ON [Attributes]
([PlanningStudyID]);

CREATE INDEX [IDX_Attributes_VisibilityTypeID] ON [Attributes]
([VisibilityTypeID]);

CREATE UNIQUE INDEX [UIDX_Attributes_Key] ON [Attributes] ([Key]);

CREATE UNIQUE INDEX [UIDX_Attributes_Name] ON [Attributes] ([Name]);

CREATE TABLE [Variables] (
    [VariableID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [PlanningStudyID] INTEGER NOT NULL CONSTRAINT
[FK_Variables_PlanningStudyID] REFERENCES
[PlanningStudy]([PlanningStudyID]),
    [Key] TEXT(255) NOT NULL,
    [VariableName] TEXT(255) NOT NULL,
    [Units] TEXT(100),
    [VariableDescription] TEXT(255),
    [VariableTypeID] INTEGER NOT NULL CONSTRAINT [FK_Variables_VariableTypeID]
REFERENCES [VariableTypes]([VariableTypeID]),
    [IsDerived] BOOLEAN NOT NULL DEFAULT 0,
    [DerivedVariableFunction] TEXT(255),
    [DerivedUpdateClause] TEXT,
    [DerivedEditorFunction] TEXT,
    [DerivedUpToDate] BOOLEAN NOT NULL DEFAULT 0,
    [Minimum] INTEGER,
    [Maximum] INTEGER,
    [IsVisible] BOOLEAN NOT NULL DEFAULT 1,
    [LowSensitivity] NUMERIC(18, 4),
    [HighSensitivity] NUMERIC(18, 4),
    [IsConvergenceOn] BOOLEAN NOT NULL DEFAULT 0,
    [ConvergencePercent] REAL DEFAULT 0,
    [ConvergenceValue] NUMERIC(18, 2) DEFAULT 0,
    [ConstrainedMin] INTEGER,
    [ConstrainedMax] INTEGER,
    [SensitivityParentVariableID] INTEGER CONSTRAINT [FK_Variables_Variables]
REFERENCES [Variables]([VariableID]),
    [IsLowSensitivity] BOOLEAN NOT NULL DEFAULT 0,
    [IsHighSensitivity] BOOLEAN NOT NULL DEFAULT 0);

CREATE INDEX [IDX_Variables_IsDerived] ON [Variables] ([IsDerived]);

CREATE INDEX [IDX_Variables_PlanningStudyID] ON [Variables]
([PlanningStudyID]);

CREATE UNIQUE INDEX [IDX_Variables_VariableName] ON [Variables]
([VariableName]);

CREATE INDEX [IDX_Variables_VariableTypeID] ON [Variables]
([VariableTypeID]);

```

```

CREATE UNIQUE INDEX [UIDX_Variables_Key] ON [Variables] ([Key]);

CREATE INDEX [IDX_Variables_SensitivityParentID] ON [Variables]
([SensitivityParentVariableID]);

CREATE TABLE [AutoEditPlans] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [AutoEditGroupName] TEXT(255) NOT NULL,
    [AppliedOnCriteria] TEXT(255) NOT NULL,
    [VariableID] INTEGER NOT NULL CONSTRAINT [FK_AutoEditPlans_VariableID]
REFERENCES [Variables]([VariableID]),
    [ConstantValue] DOUBLE NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX [UIDX_AutoEditPlans_GroupName] ON [AutoEditPlans]
([AutoEditGroupName]);

CREATE INDEX [IDX_AutoEditPlans_VariableID] ON [AutoEditPlans]
([VariableID]);

CREATE TABLE [Solutions] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [Code] TEXT(10) NOT NULL,
    [Name] TEXT(255) NOT NULL,
    [NumberOfScales] INTEGER NOT NULL DEFAULT 0,
    [DisplayOrder] DOUBLE NOT NULL DEFAULT 0);

CREATE INDEX [IDX_Solutions_DisplayOrder] ON [Solutions] ([DisplayOrder]);

CREATE UNIQUE INDEX [UIDX_Solutions_Code] ON [Solutions] ([Code]);

CREATE TABLE [AutoEditFunctions] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [AutoEditPlanID] INTEGER NOT NULL CONSTRAINT
[FK_AutoEditFunctions_AutoEditPlanID] REFERENCES [AutoEditPlans]([ID]),
    [SolutionID] INTEGER NOT NULL CONSTRAINT [FK_AutoEditFunctions_SolutionID]
REFERENCES [Solutions]([ID]),
    [CoefficientValue] DOUBLE NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX [UIDX_AutoEditFunctions_AutoEditPlanID_SolutionID] ON
[AutoEditFunctions] ([AutoEditPlanID], [SolutionID]);

CREATE TABLE [AutoEditPlansHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [AutoEditGroupName] TEXT(255) NOT NULL,
    [AppliedOnCriteria] TEXT(255) NOT NULL,

```

```

    [VariableID] INTEGER NOT NULL CONSTRAINT
[FK_AutoEditPlansHistoricalRecords_VariableID] REFERENCES
[Variables] ([VariableID]),
    [ConstantValue] DOUBLE NOT NULL DEFAULT 0);

CREATE INDEX [IDX_AutoEditPlansHistoricalRecords_VariableID] ON
[AutoEditPlansHistoricalRecords] ([VariableID]);

CREATE TABLE [ComponentTypes] (
    [ComponentTypeID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] [TEXT(255)] NOT NULL,
    [Description] [TEXT(500)],
    [Key] [TEXT(255)] NOT NULL);

CREATE UNIQUE INDEX [UIDX_ComponentTypes_Key] ON [ComponentTypes] ([Key]);

CREATE TABLE [Components] (
    [ComponentID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] [TEXT(255)],
    [Description] [TEXT(500)],
    [Key] [TEXT(255)],
    [ComponentTypeID] INTEGER NOT NULL CONSTRAINT
[FK_Components_ComponentTypeID] REFERENCES
[ComponentTypes] ([ComponentTypeID]),
    [VisibilityTypeID] INTEGER NOT NULL CONSTRAINT
[FK_Components_VisibilityTypeID] REFERENCES
[VisibilityTypes] ([VisibilityTypeID]));

CREATE UNIQUE INDEX [UIDX_Components_Name] ON [Components] ([Name]);

CREATE UNIQUE INDEX [UIDX_Components_Key] ON [Components] ([Key]);

CREATE INDEX [IDX_Components_VisibilityType] ON [Components]
([VisibilityTypeID]);

CREATE TABLE [PlanningSetTypes] (
    [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] [TEXT(255)] NOT NULL,
    [Description] [TEXT(500)],
    [Key] [TEXT(255)] NOT NULL);

CREATE UNIQUE INDEX [UIDX_PlanningSetTypes_Name] ON [PlanningSetTypes]
([Name]);

CREATE UNIQUE INDEX [UIDX_PlanningSetTypes_Key] ON [PlanningSetTypes]
([Key]);

CREATE TABLE [UncertaintyDistributionSets] (

```

```

[ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
[Name] TEXT(255) NOT NULL,
[Description] TEXT(500),
[CreationDate] DATETIME,
[Validated] BOOL NOT NULL DEFAULT 1,
[RulesValidated] BOOL NOT NULL DEFAULT 1);

CREATE UNIQUE INDEX [UIDX_UncertaintyDistributionSets_Name] ON
[UncertaintyDistributionSets] ([Name]);

CREATE TABLE [PlanningSets] (
  [PlanningSetID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [PlanningStudyID] INTEGER NOT NULL CONSTRAINT
  [FK_PlanningSets_PlanningStudyID] REFERENCES
  [PlanningStudy] ([PlanningStudyID]),
  [Name] TEXT(255) NOT NULL,
  [Active] BOOLEAN NOT NULL DEFAULT 0,
  [Description] TEXT(500),
  [HUC] TEXT(12),
  [LocationDescription] TEXT(500),
  [ParentPlanningSetID] INTEGER CONSTRAINT
  [FK_PlanningSets_ParentPlanningSetID] REFERENCES
  [PlanningSets] ([PlanningSetID]),
  [ComponentID] INTEGER CONSTRAINT [FK_PlanningSets_ComponentID] REFERENCES
  [Components] ([ComponentID]),
  [PlanSetXMLHeader] TEXT,
  [NumberOfPlans] INTEGER,
  [Visible] BOOLEAN DEFAULT 0,
  [UserInformation] TEXT(255),
  [CreatedBy] TEXT(255),
  [PlanningSetTypeID] INTEGER CONSTRAINT [FK_PlanningSets_PlanningSetTypeID]
  REFERENCES [PlanningSetTypes] ([ID]),
  [Modified] BOOLEAN NOT NULL DEFAULT 0,
  [UncertaintyDistributionSetID] INTEGER CONSTRAINT
  [FK_PlanningSets_UncertaintyDistributionSetID] REFERENCES
  [UncertaintyDistributionSets] ([ID]),
  [UncertaintyDatabaseName] TEXT(255),
  [DateCreated] DATETIME);

CREATE INDEX [IDX_PlanningSets_Active] ON [PlanningSets] ([Active]);

CREATE INDEX [IDX_PlanningSets_ComponentID] ON [PlanningSets]
([ComponentID]);

CREATE INDEX [IDX_PlanningSets_ParentPlanningSetID] ON [PlanningSets]
([ParentPlanningSetID]);

CREATE INDEX [IDX_PlanningSets_PlanningSetTypeID] ON [PlanningSets]
([PlanningSetTypeID]);

```

```

CREATE INDEX [IDX_PlanningSets_UncertaintyDistributionSetID] ON
[PlanningSets] ([UncertaintyDistributionSetID]);

CREATE INDEX [IDX_PlanningSets_Visible] ON [PlanningSets] ([Visible]);

CREATE TABLE "SolutionsHistoricalRecords" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionsHistorical_PlanningSets] REFERENCES
[PlanningSets]([PlanningSetID]),
  [Code] TEXT(10) NOT NULL,
  [Name] TEXT(255) NOT NULL,
  [NumberOfScales] INTEGER NOT NULL DEFAULT 0,
  [DisplayOrder] DOUBLE NOT NULL DEFAULT 0);

CREATE INDEX [IDX_SolutionsHistorical_PlanningSetID_DisplayOrder] ON
[SolutionsHistoricalRecords] ([PlanningSetID], [DisplayOrder]);

CREATE UNIQUE INDEX [UIDX_SolutionsHistorical_Code] ON
[SolutionsHistoricalRecords] ([PlanningSetID], [Code]);

CREATE TABLE [AutoEditFunctionsHistoricalRecords] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [AutoEditPlanID] INTEGER NOT NULL CONSTRAINT
[FK_AutoEditFunctionsHistoricalRecords_AutoEditPlanID] REFERENCES
[AutoEditPlansHistoricalRecords]([ID]),
  [SolutionID] INTEGER NOT NULL CONSTRAINT
[FK_AutoEditFunctionsHistoricalRecords_SolutionID] REFERENCES
[SolutionsHistoricalRecords]([ID]),
  [CoefficientValue] DOUBLE NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_AutoEditFunctionsHistoricalRecords_AutoEditPlanID_SolutionID] ON
[AutoEditFunctionsHistoricalRecords] ([AutoEditPlanID], [SolutionID]);

CREATE TABLE CostEffectiveAnalysis (
  PlanID INTEGER PRIMARY KEY,
  CEID INTEGER,
  CECID INTEGER,
  BBID INTEGER,
  CostID INTEGER,
  OutputID INTEGER,
  CostValue NUMERIC(18,3),
  OutputValue NUMERIC(18,3),
  CostEffective INTEGER,
  IsNoActionPlan BOOLEAN
);

```

```

CREATE TABLE [DerivedVariableMappings] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [DerivedVariableID] INTEGER NOT NULL CONSTRAINT
    [FK_DerivedVariableMappings_Variables_1] REFERENCES
    [Variables]([VariableID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT
    [FK_DerivedVariableMappings_Variables_2] REFERENCES
    [Variables]([VariableID]));

CREATE UNIQUE INDEX
[UIDX_DerivedVariableMappings_DerivedVariableID_VariableID] ON
[DerivedVariableMappings] ([DerivedVariableID], [VariableID]);

CREATE INDEX [IDX_DerivedVariableMappings_VariableID] ON
[DerivedVariableMappings] ([VariableID]);

CREATE TABLE [MCDACompromiseProgrammingExponents] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [ExponentName] TEXT(255) NOT NULL,
    [ExponentDescription] TEXT(500),
    [ExponentKey] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_MCDACPExponents_ExponentName] ON
[MCDACompromiseProgrammingExponents] ([ExponentName]);

CREATE UNIQUE INDEX [UIDX_MCDACPExponents_ExponentKey] ON
[MCDACompromiseProgrammingExponents] ([ExponentKey]);

CREATE TABLE [MCDACriteria] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [VariableID] INTEGER CONSTRAINT [FK_MCDACriteria_VariableID] REFERENCES
    [Variables]([VariableID]),
    [AttributeID] INTEGER CONSTRAINT [FK_MCDACriteria_AttributeID] REFERENCES
    [Attributes]([AttributeID]));

CREATE TABLE [MCDACriteriaOptimizationMethods] (
    [ID] INTEGER NOT NULL PRIMARY KEY,
    [Name] TEXT(255) NOT NULL,
    [Description] TEXT(500),
    [Key] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_MCDACriteriaOptimizationMethods_Key] ON
[MCDACriteriaOptimizationMethods] ([Key]);

CREATE UNIQUE INDEX [UIDX_MCDACriteriaOptimizationMethods_Name] ON
[MCDACriteriaOptimizationMethods] ([Name]);

CREATE TABLE [MCDAEigenValues] (

```

```

[EigenValue] INTEGER NOT NULL PRIMARY KEY,
[EigenDescription] TEXT(500) NOT NULL,
[Key] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_MCDAEigenValues_Key] ON [MCDAEigenValues] ([Key]);

CREATE TABLE [MCDANormalizationMethods] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [Key] TEXT(25) NOT NULL);

CREATE UNIQUE INDEX [UIDX_MCDANormalizationMethods_Name] ON
[MCDANormalizationMethods] ([Name]);

CREATE UNIQUE INDEX [UIDX_MCDANormalizationMethods_Key] ON
[MCDANormalizationMethods] ([Key]);

CREATE TABLE [MCDARankingMethods] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [Key] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_MCDARankingMethods_Key] ON [MCDARankingMethods]
([Key]);

CREATE UNIQUE INDEX [UIDX_MCDARankingMethods_Name] ON [MCDARankingMethods]
([Name]);

CREATE TABLE [MCDAScenarios] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAScenarios_PlanningSetID] REFERENCES [PlanningSets]([PlanningSetID]),
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [RankingMethodID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAScenarios_RankingMethod] REFERENCES [MCDARankingMethods]([ID]),
  [NormalizationMethodID] INTEGER CONSTRAINT
[FK_MCDAScenarios_NormalizationMethod] REFERENCES
[MCDANormalizationMethods]([ID]),
  [CPExponentID] INTEGER CONSTRAINT [FK_MCDAScenarios_CPExponent] REFERENCES
[MCDACompromiseProgrammingExponents]([ID]),
  [DateCreated] DATETIME,
  [DateLastSaved] DATETIME);

CREATE INDEX [IDX_MCDAScenarios_CPExponentID] ON [MCDAScenarios]
([CPExponentID]);

```

```
CREATE INDEX [IDX_MCDAScenarios_NormalizationMethodID] ON [MCDAScenarios]
([NormalizationMethodID]);
```

```
CREATE UNIQUE INDEX [UIDX_MCDAScenarios_PlanningSetID] ON [MCDAScenarios]
([PlanningSetID], [Name]);
```

```
CREATE INDEX [IDX_MCDAScenarios_RankingMethodID] ON [MCDAScenarios]
([RankingMethodID]);
```

```
CREATE TABLE [MCDAEigenVectors] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [ScenarioID] INTEGER NOT NULL CONSTRAINT [FK_MCDAEigenVectors_ScenarioID]
REFERENCES [MCDAScenarios]([ID]),
  [CriteriaIDy] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectors_CriteriaIDy] REFERENCES [MCDACriteria]([ID]),
  [CriteriaIDx] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectors_CriteriaIDx] REFERENCES [MCDACriteria]([ID]),
  [EigenValue] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectors_EigenValueID] REFERENCES
[MCDAEigenValues]([EigenValue]));
```

```
CREATE INDEX [IDX_MCDAEigenVectors_ScenarioID_CriteriaIDx_CriteriaIDy] ON
[MCDAEigenVectors] ([ScenarioID], [CriteriaIDx], [CriteriaIDy]);
```

```
CREATE TABLE [MCDAScenariosHistoricalRecords] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAScenariosHistoricalRecords_PlanningSetID] REFERENCES
[PlanningSets]([PlanningSetID]),
  [Name] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [RankingMethodID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAScenariosHistoricalRecords_RankingMethod] REFERENCES
[MCDARankingMethods]([ID]),
  [NormalizationMethodID] INTEGER CONSTRAINT
[FK_MCDAScenariosHistoricalRecords_NormalizationMethod] REFERENCES
[MCDANormalizationMethods]([ID]),
  [CPExponentID] INTEGER CONSTRAINT
[FK_MCDAScenariosHistoricalRecords_CPExponent] REFERENCES
[MCDACompromiseProgrammingExponents]([ID]),
  [DateCreated] DATETIME,
  [DateLastSaved] DATETIME);
```

```
CREATE INDEX [IDX_MCDAScenariosHistoricalRecords_CPExponentID] ON
[MCDAScenariosHistoricalRecords] ([CPExponentID]);
```

```
CREATE INDEX [IDX_MCDAScenariosHistoricalRecords_NormalizationMethodID] ON
[MCDAScenariosHistoricalRecords] ([NormalizationMethodID]);
```

```

CREATE INDEX [IDX_MCDAScenariosHistoricalRecords_RankingMethodID] ON
[MCDAScenariosHistoricalRecords] ([RankingMethodID]);

CREATE UNIQUE INDEX [UIDX_MCDAScenariosHistoricalRecords_PlanningSetID] ON
[MCDAScenariosHistoricalRecords] ([PlanningSetID]);

CREATE TABLE [MCDAEigenVectorsHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [ScenarioID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectorsHistoricalRecords_ScenarioID] REFERENCES
[MCDAScenariosHistoricalRecords] ([ID]),
    [CriteriaIdx] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectorsHistoricalRecords_CriteriaIdx] REFERENCES
[MCDACriteria] ([ID]),
    [CriteriaIDy] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectorsHistoricalRecords_CriteriaIDy] REFERENCES
[MCDACriteria] ([ID]),
    [EigenValue] INTEGER NOT NULL CONSTRAINT
[FK_MCDAEigenVectorsHistoricalRecords_EigenValueID] REFERENCES
[MCDAEigenValues] ([EigenValue]));

CREATE UNIQUE INDEX [UIDX1_MCDAEigenVectorsHistoricalRecords] ON
[MCDAEigenVectorsHistoricalRecords] ([ScenarioID], [CriteriaIdx],
[CriteriaIDy]);

CREATE TABLE [Plans] (
    [PlanID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [PlanName] TEXT(255) NOT NULL,
    [PlanDescription] TEXT(500),
    [ReadOnly] BOOLEAN NOT NULL DEFAULT 0,
    [IsNoActionPlan] BOOLEAN NOT NULL DEFAULT 0,
    [Generated] BOOLEAN NOT NULL DEFAULT 0,
    [HUC] TEXT(12),
    [LocationDescription] TEXT(500),
    [Latitude] DOUBLE,
    [Longitude] DOUBLE);

CREATE INDEX [IDX_Plans_Generated] ON [Plans] ([Generated]);

CREATE INDEX [IDX_Plans_IsNoActionPlan] ON [Plans] ([IsNoActionPlan]);

CREATE INDEX [IDX_Plans_PlanName] ON [Plans] ([PlanName]);

CREATE INDEX [IDX_Plans_ReadOnly] ON [Plans] ([ReadOnly]);

CREATE TABLE "MCDAPlanCriteriaScores" (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

```

```

    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
    [FK_MCDACriteriaContributions_PlanningSetID] REFERENCES
    [PlanningSets] ([PlanningSetID]),
    [PlanID] INTEGER NOT NULL CONSTRAINT [FK_MCDACriteriaContributions_PlanID]
    REFERENCES [Plans] ([PlanID]),
    [CriteriaID] INTEGER NOT NULL CONSTRAINT
    [FK_MCDACriteriaContributions_CriteriaID] REFERENCES [MCDACriteria] ([ID]),
    [CriteriaScore] DOUBLE(18, 3) DEFAULT 0);

```

```

CREATE INDEX [IDX_MCDACriteriaContributions_CriteriaID] ON
"MCDAPlanCriteriaScores" ([CriteriaID]);

```

```

CREATE UNIQUE INDEX
[UIDX_MCDACriteriaContributions_PlanningSetID_PlanID_CriteriaID] ON
"MCDAPlanCriteriaScores" ([PlanningSetID], [PlanID], [CriteriaID]);

```

```

CREATE TABLE "MCDAPlanTotalScores" (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
    [FK_MCDAScenarioPlanTotalScores_PlanningSets] REFERENCES
    [PlanningSets] ([PlanningSetID]),
    [PlanID] INTEGER NOT NULL CONSTRAINT
    [FK_MCDAScenarioPlanTotalScores_PlanID] REFERENCES [Plans] ([PlanID]),
    [Rank] INTEGER DEFAULT 0,
    [TotalScore] DOUBLE(18, 3) DEFAULT 0);

```

```

CREATE UNIQUE INDEX [UIDX_MCDAScenarioPlanTotalScores_PlanningSetID_PlanID]
ON "MCDAPlanTotalScores" ([PlanningSetID], [PlanID]);

```

```

CREATE TABLE "MCDAPrometheeTypes" (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT(255) NOT NULL,
    [Description] TEXT(500),
    [Key] TEXT(255) NOT NULL);

```

```

CREATE UNIQUE INDEX [UIDX_PrometheeTypes_Name] ON "MCDAPrometheeTypes"
([Name]);

```

```

CREATE UNIQUE INDEX [UIDX_PrometheeTypes_Key] ON "MCDAPrometheeTypes"
([Key]);

```

```

CREATE TABLE [MCDARankingMethodCriterionTypes] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [MCDARankingMethodID] INTEGER NOT NULL CONSTRAINT
    [FK_MCDARankingMethodCriterionTypes_MCDARankingMethods] REFERENCES
    [MCDARankingMethods] ([ID]),
    [CriterionTypeName] TEXT(255) NOT NULL,
    [CriterionTypeDescription] TEXT(500),
    [Key] TEXT(255));

```

```
CREATE UNIQUE INDEX
[IDX_MCDARankingMethodCriterionTypes_MCDARankingMethodID_CriterionTypeName]
ON [MCDARankingMethodCriterionTypes] ([MCDARankingMethodID],
[CriterionTypeName]);
```

```
CREATE TABLE [MCDAScenarioCriteriaDetails] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [ScenarioID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetails_ScenarioID] REFERENCES
[MCDAScenarios]([ID]),
  [CriteriaID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetails_CriteriaID] REFERENCES
[MCDACriteria]([ID]),
  [Weight] DOUBLE(18, 3) NOT NULL DEFAULT 1,
  [OptimizationMethodID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetails_OptimizationMethods] REFERENCES
[MCDACriteriaOptimizationMethods]([ID]) DEFAULT 1,
  [PrometheeTypeID] INTEGER CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetails_PrometheeTypeID] REFERENCES
[MCDAPrometheeTypes]([ID]) DEFAULT 1,
  [PrometheeQ] DOUBLE(18, 3) DEFAULT 0,
  [PrometheeP] DOUBLE(18, 3) DEFAULT 0,
  [Minimum] DOUBLE(18, 3) DEFAULT (-10000000),
  [Maximum] DOUBLE(18, 3) DEFAULT 10000000);
```

```
CREATE INDEX [IDX_MCDAPLanningSetCriteriaDetails_ScenarioID_CriteriaID] ON
[MCDAScenarioCriteriaDetails] ([ScenarioID], [CriteriaID]);
```

```
CREATE INDEX [IDX_MCDAPlanningSetCriteriaDetails_OptimizationMethodID] ON
[MCDAScenarioCriteriaDetails] ([OptimizationMethodID]);
```

```
CREATE TABLE [MCDAScenarioCriteriaDetailsHistoricalRecords] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [ScenarioID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetailsHistoricalRecords_ScenarioID] REFERENCES
[MCDAScenariosHistoricalRecords]([ID]),
  [CriteriaID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetailsHistoricalRecords_CriteriaID] REFERENCES
[MCDACriteria]([ID]),
  [Weight] DOUBLE(18, 3) NOT NULL DEFAULT 1,
  [OptimizationMethodID] INTEGER NOT NULL CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetailsHistoricalRecords_OptimizationMethods]
REFERENCES [MCDACriteriaOptimizationMethods]([ID]) DEFAULT 1,
  [PrometheeTypeID] INTEGER CONSTRAINT
[FK_MCDAPlanningSetCriteriaDetailsHistoricalRecords_PrometheeTypeID]
REFERENCES [MCDAPrometheeTypes]([ID]) DEFAULT 1,
  [PrometheeQ] DOUBLE(18, 3) DEFAULT 0,
  [PrometheeP] DOUBLE(18, 3) DEFAULT 0,
  [Minimum] DOUBLE(18, 3) DEFAULT (-10000000),
```

```

[Maximum] DOUBLE(18, 3) DEFAULT 10000000);

CREATE INDEX
[IDX_MCDAPlanningSetCriteriaDetailsHistoricalRecords_OptimizationMethodID]
ON [MCDAScenarioCriteriaDetailsHistoricalRecords] ([OptimizationMethodID]);

CREATE UNIQUE INDEX
[UIDX2_MCDAPlanningSetCriteriaDetailsHistoricalRecords_ScenarioID] ON
[MCDAScenarioCriteriaDetailsHistoricalRecords] ([ScenarioID], [CriteriaID]);

CREATE TABLE "Parameters" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [Name] TEXT(255) NOT NULL,
  [Key] TEXT(255) NOT NULL,
  [Description] TEXT(500),
  [ComponentID] INTEGER NOT NULL CONSTRAINT [FK_ParameterTypes_ComponentID]
REFERENCES [Components] ([ComponentID]),
  [DataTypeID] INTEGER NOT NULL CONSTRAINT [FK_ParameterTypes_DataTypeID]
REFERENCES [DataTypes] ([DataTypeID]));

CREATE INDEX [IDX_ParameterTypes_ComponentID] ON "Parameters"
([ComponentID]);

CREATE UNIQUE INDEX [UIDX_ParameterTypes_Key] ON "Parameters" ([Key]);

CREATE TABLE "PlanAttributeMappings" (
  [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
  [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternativeAttributes_PlanningSetID] REFERENCES
[PlanningSets] ([PlanningSetID]),
  [PlanID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternativeAttributes_PlanID] REFERENCES [Plans] ([PlanID]),
  [AttributeID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternativeAttributes_AttributeID] REFERENCES
[Attributes] ([AttributeID]),
  [Value] [TEXT(50)]);

CREATE INDEX [IDX_PlanAttributeMatrix_AttributeID] ON
[PlanAttributeMappings] ([AttributeID]);

CREATE INDEX [IDX_PlanAttributeMatrix_PlanID_AttributeID] ON
[PlanAttributeMappings] ([PlanID], [AttributeID]);

CREATE UNIQUE INDEX
[UIDX_PlanAttributeMatrix_PlanningSetID_PlanID_AttributeID] ON
[PlanAttributeMappings] ([PlanningSetID], [PlanID], [AttributeID]);

CREATE TABLE [PlanningSetParameters] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

```

```

[ParameterID] INTEGER NOT NULL CONSTRAINT
[FK_PlanningSetParameters_ParameterID] REFERENCES [Parameters]([ID]),
[PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_PlanningSetParameters_PlanningSetID] REFERENCES
[PlanningSets]([PlanningSetID]),
[VariableID] INTEGER CONSTRAINT [FK_PlanningSetParameters_VariableID]
REFERENCES [Variables]([VariableID]),
[Criteria] TEXT(255),
[Value] TEXT(255));

CREATE INDEX [IDX_PlanningSetParameters_ParameterID] ON
[PlanningSetParameters] ([ParameterID]);

CREATE INDEX [IDX_PlanningSetParameters_PlanningSet_Variable_Parameter] ON
[PlanningSetParameters] ([PlanningSetID], [VariableID], [ParameterID]);

CREATE TABLE [PlanVariableMappings] (
  [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
  [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternatives_PlanningSetID] REFERENCES
[PlanningSets]([PlanningSetID]),
  [PlanID] INTEGER NOT NULL CONSTRAINT [FK_SharedPlanAlternatives_PlanID]
REFERENCES [Plans]([PlanID]),
  [Description] TEXT(255),
  [VariableID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternatives_VariableID] REFERENCES [Variables]([VariableID]),
  [Value] NUMERIC(18, 3) DEFAULT 0);

CREATE INDEX [IDX_PlanVariableMatrix_PlanID_VariableID] ON
[PlanVariableMappings] ([PlanID], [VariableID]);

CREATE INDEX [IDX_PlanVariableMatrix_VariableID] ON [PlanVariableMappings]
([VariableID]);

CREATE UNIQUE INDEX
[UIDX_PlanVariableMatrix_PlanningSetID_PlanID_VariableID] ON
[PlanVariableMappings] ([PlanningSetID], [PlanID], [VariableID]);

CREATE TABLE [RelationshipParameters] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [NoSolutionsCombinableFlag] BOOL NOT NULL DEFAULT 0);

CREATE TABLE [RelationshipParametersHistoricalRecords] (
  [RelationshipSetID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [NoSolutionsCombinableFlag] BOOL NOT NULL DEFAULT 0);

CREATE TABLE [RelationshipTypes] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

```

```

    [RelationshipTypeName] TEXT(255) NOT NULL,
    [RelationshipTypeDescription] TEXT(500),
    [RelationshipTypeKey] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_RelationshipTypes_Key] ON [RelationshipTypes]
([RelationshipTypeKey]);

CREATE TABLE [SolutionRelationships] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [SolutionID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationships_Solutions] REFERENCES [Solutions]([ID]),
    [RelationshipTypeID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionsRelationships_RelationshipTypes] REFERENCES
[RelationshipTypes]([ID]);

CREATE INDEX [IDX_SolutionRelationships_SolutionsID] ON
[SolutionRelationships] ([SolutionID]);

CREATE TABLE [SolutionRelationshipDetails] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [SolutionRelationshipID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationshipDetails_SolutionRelationshipID] REFERENCES
[SolutionRelationships]([ID]),
    [SolutionID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationshipDetails_SolutionID] REFERENCES [Solutions]([ID]));

CREATE INDEX [IDX_SolutionsRelationshipDetails_SolutionRelationshipID] ON
[SolutionRelationshipDetails] ([SolutionRelationshipID]);

CREATE INDEX [IDX_SolutionRelationshipDetails_SolutionID] ON
[SolutionRelationshipDetails] ([SolutionID]);

CREATE INDEX
[IDX_SolutionRelationshipDetails_SolutionRelationshipID_SolutionID] ON
[SolutionRelationshipDetails] ([SolutionRelationshipID], [SolutionID]);

CREATE TABLE [SolutionRelationshipsHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [RelationshipSetID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionsRelationshipsHistoricalRecords_RelationshipSetID] REFERENCES
[RelationshipParametersHistoricalRecords]([RelationshipSetID]),
    [SolutionID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationshipsHistoricalRecords_SolutionsHistoricalRecords]
REFERENCES [SolutionsHistoricalRecords]([ID]),
    [RelationshipTypeID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionsRelationshipsHistoricalRecords_RelationshipTypes] REFERENCES
[RelationshipTypes]([ID]);

```

```

CREATE INDEX
[IDX_SolutionRelationshipsHistoricalRecords_RelationshipSetID_SolutionsID]
ON [SolutionRelationshipsHistoricalRecords] ([RelationshipSetID],
[SolutionID]);

CREATE TABLE [SolutionRelationshipDetailsHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [SolutionRelationshipID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationshipDetailsHistoricalRecords_SolutionRelationshipID]
REFERENCES [SolutionRelationshipsHistoricalRecords]([ID]),
    [SolutionID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionRelationshipDetailsHistoricalRecords_SolutionID] REFERENCES
[SolutionsHistoricalRecords]([ID]));

CREATE INDEX
[IDX_SolutionsRelationshipDetailsHistoricalRecords_SolutionRelationshipID]
ON [SolutionRelationshipDetailsHistoricalRecords]
([SolutionRelationshipID]);

CREATE INDEX [IDX_SolutionRelationshipDetailsHistoricalRecords_SolutionID]
ON [SolutionRelationshipDetailsHistoricalRecords] ([SolutionID]);

CREATE INDEX
[IDX_SolutionRelationshipDetailsHistoricalRecords_SolutionRelationshipID_Sol
utionID] ON [SolutionRelationshipDetailsHistoricalRecords]
([SolutionRelationshipID], [SolutionID]);

CREATE TABLE [SolutionScales] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [SolutionID] INTEGER NOT NULL CONSTRAINT [FK_SolutionScales_Solutions]
REFERENCES [Solutions]([ID]),
    [ScaleNumber] integer NOT NULL,
    [Name] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [IDX_SolutionScales_SolutionID_ScaleNumber] ON
[SolutionScales] ([SolutionID], [ScaleNumber]);

CREATE TABLE [SolutionScalesEffects] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [SolutionScalesID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionScalesEffects_SolutionScales] REFERENCES [SolutionScales]([ID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT
[FK_SolutionScalesEffects_Variables] REFERENCES [Variables]([VariableID]),
    [EffectOnVariable] NUMERIC(18, 3) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX [IDX_SolutionScalesEffects_SolutionScalesID_VariableID]
ON [SolutionScalesEffects] ([SolutionScalesID], [VariableID]);

```

```

CREATE TABLE "SolutionScalesHistoricalRecords" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [SolutionID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionScalesHistorical_SolutionsHistorical] REFERENCES
  "SolutionsHistoricalRecords"([ID]),
  [ScaleNumber] integer NOT NULL,
  [Name] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [IDX_SolutionScalesHistorical_SolutionID_ScaleNumber] ON
"SolutionScalesHistoricalRecords" ([SolutionID], [ScaleNumber]);

CREATE TABLE "SolutionScalesEffectsHistoricalRecords" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [SolutionScalesID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionScalesEffectsHistorical_SolutionScalesHistorical] REFERENCES
  "SolutionScalesHistoricalRecords"([ID]),
  [VariableID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionScalesEffectsHistorical_Variables] REFERENCES
  [Variables]([VariableID]),
  [EffectOnVariable] NUMERIC(18, 3) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[IDX_SolutionScalesEffectsHistorical_SolutionScalesID_VariableID] ON
"SolutionScalesEffectsHistoricalRecords" ([SolutionScalesID], [VariableID]);

CREATE TABLE [SolutionSensitivities] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [SolutionID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionSensitivites_SolutionID] REFERENCES [Solutions]([ID]),
  [VariableID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionSensitivities_VariableID] REFERENCES [Variables]([VariableID]),
  [LowCoefficient] DOUBLE DEFAULT 0,
  [HighCoefficient] DOUBLE DEFAULT 0);

CREATE UNIQUE INDEX [UIDX_SolutionSensitivities_SolutionID_Varaible_ID] ON
[SolutionSensitivities] ([SolutionID], [VariableID]);

CREATE INDEX [IDX_SolutionSensitivities_VariableID] ON
[SolutionSensitivities] ([VariableID]);

CREATE TABLE [SolutionSensitivitiesHistoricalRecords] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [SolutionID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionSensitivitesHistoricalRecords_SolutionID] REFERENCES
  [SolutionsHistoricalRecords]([ID]),
  [VariableID] INTEGER NOT NULL CONSTRAINT
  [FK_SolutionSensitivitiesHistoricalRecords_VariableID] REFERENCES
  [Variables]([VariableID]),
  [LowCoefficient] DOUBLE DEFAULT 0,

```

```

[HighCoefficient] DOUBLE DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_SolutionSensitivitiesHistoricalRecords_SolutionID_Variable_ID] ON
[SolutionSensitivitiesHistoricalRecords] ([SolutionID], [VariableID]);

CREATE INDEX [IDX_SolutionSensitivitiesHistoricalRecords_VariableID] ON
[SolutionSensitivitiesHistoricalRecords] ([VariableID]);

CREATE TABLE [StudyConfiguration] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [CostVariableID] INTEGER CONSTRAINT [StudyConfiguration_CostVariableID_FK]
REFERENCES [Variables] ([VariableID]),
    [OutputVariableID] INTEGER CONSTRAINT
[StudyConfiguration_OutputVariableID_FK] REFERENCES
[Variables] ([VariableID]));

CREATE UNIQUE INDEX [StudyConfiguration_CostVariableID_IDX] ON
[StudyConfiguration] ([CostVariableID]);

CREATE UNIQUE INDEX [StudyConfiguration_OutputVariableID_IDX] ON
[StudyConfiguration] ([OutputVariableID]);

CREATE TABLE [TempUncertaintyCEICAValidatedPlans] (
    [PlanID] INTEGER NOT NULL PRIMARY KEY);

CREATE TABLE [TestPlanData] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT NOT NULL,
    [Cost] DOUBLE NOT NULL DEFAULT 0,
    [Output] DOUBLE NOT NULL DEFAULT 0,
    [Variable1] DOUBLE NOT NULL,
    [Variable2] DOUBLE NOT NULL,
    [Variable3] DOUBLE NOT NULL);

CREATE TABLE [UncertaintyDistributionSetsHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[IDX_UncertaintyDistributionSetsHistoricalRecords_PlanningSets] REFERENCES
[PlanningSets] ([PlanningSetID]),
    [Name] TEXT(255) NOT NULL,
    [Description] TEXT(500),
    [CreationDate] DATETIME,
    [Validated] BOOL NOT NULL DEFAULT 1,
    [RulesValidated] BOOL NOT NULL DEFAULT 1);

CREATE UNIQUE INDEX [UIDX_UncertaintyDistributionSetsHistoricalRecords_Name]
ON [UncertaintyDistributionSetsHistoricalRecords] ([Name]);

```

```

CREATE INDEX
[IDX_UncertaintyDistributionSetsHistoricalRecords_PlanningSetID] ON
[UncertaintyDistributionSetsHistoricalRecords] ([PlanningSetID]);

CREATE TABLE "UncertaintyCorrelationMatrixHistoricalRecords" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
[FK1_UncertaintyCorrelationMatrixHistoricalRecords] REFERENCES
[UncertaintyDistributionSetsHistoricalRecords] ([ID]),
  [RowOrder] INTEGER NOT NULL,
  [RowVariableID] INTEGER NOT NULL CONSTRAINT
[FK2_UncertaintyCorrelationMatrixHistoricalRecords] REFERENCES
[Variables] ([VariableID]),
  [ColumnVariableID] INTEGER NOT NULL CONSTRAINT
[FK3_UncertaintyCorrelationMatrixHistoricalRecords] REFERENCES
[Variables] ([VariableID]),
  [ColumnOrder] INTEGER NOT NULL,
  [VariableCorrelation] DOUBLE,
  [FinalMatrix] BOOLEAN DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyCorrelationMatrixHistoricalRecords] ON
"UncertaintyCorrelationMatrixHistoricalRecords"
([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyCorrelationMatrixHistoricalRecords] ON
"UncertaintyCorrelationMatrixHistoricalRecords" ([RowVariableID]);

CREATE INDEX [IDX3_UncertaintyCorrelationMatrixHistoricalRecords] ON
"UncertaintyCorrelationMatrixHistoricalRecords" ([ColumnVariableID]);

CREATE TABLE "UncertaintyCorrelationMatrixRecords" (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
[FK1_UncertaintyCorrelationMatrix] REFERENCES
[UncertaintyDistributionSets] ([ID]),
  [RowOrder] INTEGER NOT NULL,
  [RowVariableID] INTEGER NOT NULL CONSTRAINT
[FK2_UncertaintyCorrelationMatrix] REFERENCES [Variables] ([VariableID]),
  [ColumnVariableID] INTEGER NOT NULL CONSTRAINT
[FK3_UncertaintyCorrelationMatrix] REFERENCES [Variables] ([VariableID]),
  [ColumnOrder] INTEGER NOT NULL,
  [VariableCorrelation] DOUBLE,
  [FinalMatrix] BOOLEAN DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyCorrelationMatrix] ON
"UncertaintyCorrelationMatrixRecords" ([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyCorrelationMatrix] ON
"UncertaintyCorrelationMatrixRecords" ([RowVariableID]);

```

```

CREATE INDEX [IDX3_UncertaintyCorrelationMatrix] ON
"UncertaintyCorrelationMatrixRecords" ([ColumnVariableID]);

CREATE TABLE [UncertaintyDistributionTypes] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [DistributionTypeName] TEXT(255) NOT NULL,
    [ParameterCount] INTEGER NOT NULL,
    [Notes] TEXT(500),
    [DisplayedDescription] teXT(500),
    [Key] TEXT(100));

CREATE UNIQUE INDEX [UIDX_UncertaintyDistributionTypes_DistributionTypeName]
ON [UncertaintyDistributionTypes] ([DistributionTypeName]);

CREATE TABLE [UncertaintyPlanDistributionSetMappings] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
[FK1_UncertaintyPlanDistributionSetMappings] REFERENCES
[UncertaintyDistributionSets]([ID]),
    [PlanID] INTEGER NOT NULL CONSTRAINT
[FK2_UncertaintyPlanDistributionSetMappings] REFERENCES [Plans]([PlanID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT
[FK3_UncertaintyPlanDistributionSetMappings] REFERENCES
[Variables]([VariableID]),
    [UncertaintyDistributionTypeID] INTEGER NOT NULL CONSTRAINT
[FK4_UncertaintyPlanDistributionSetMappings] REFERENCES
[UncertaintyDistributionTypes]([ID]));

CREATE UNIQUE INDEX [UIDX1_UncertaintyPlanDistributionSetMappings] ON
[UncertaintyPlanDistributionSetMappings] ([UncertaintyDistributionSetID],
[PlanID], [VariableID]);

CREATE INDEX [IDX1_UncertaintyPlanDistributionSetMappings] ON
[UncertaintyPlanDistributionSetMappings] ([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyPlanDistributionSetMappings] ON
[UncertaintyPlanDistributionSetMappings] ([PlanID]);

CREATE INDEX [IDX3_UncertaintyPlanDistributionSetMappings] ON
[UncertaintyPlanDistributionSetMappings] ([VariableID]);

CREATE INDEX [IDX4_UncertaintyPlanDistributionSetMappings] ON
[UncertaintyPlanDistributionSetMappings] ([UncertaintyDistributionTypeID]);

CREATE TABLE [UncertaintyDistributionSetParameters] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

```

```

    [UncertaintyPlanDistributionSetMappingID] INTEGER NOT NULL CONSTRAINT
    [FK1_UncertaintyDistributionSetParameters] REFERENCES
    [UncertaintyPlanDistributionSetMappings] ([ID]),
    [ParameterOrder] INTEGER NOT NULL,
    [ParameterValue] DOUBLE DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyDistributionSetParameters] ON
[UncertaintyDistributionSetParameters]
([UncertaintyPlanDistributionSetMappingID], [ParameterOrder]);

CREATE TABLE [UncertaintyPlanDistributionSetMappingsHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
    [FK1_UncertaintyPlanDistributionSetMappingsHistoricalRecords] REFERENCES
    [UncertaintyDistributionSetsHistoricalRecords] ([ID]),
    [PlanID] INTEGER NOT NULL CONSTRAINT
    [FK2_UncertaintyPlanDistributionSetMappingsHistoricalRecords] REFERENCES
    [Plans] ([PlanID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT
    [FK3_UncertaintyPlanDistributionSetMappingsHistoricalRecords] REFERENCES
    [Variables] ([VariableID]),
    [UncertaintyDistributionTypeID] INTEGER NOT NULL CONSTRAINT
    [FK4_UncertaintyPlanDistributionSetMappingsHistoricalRecords] REFERENCES
    [UncertaintyDistributionTypes] ([ID]));

CREATE UNIQUE INDEX
[UIDX1_UncertaintyPlanDistributionSetMappingsHistoricalRecords] ON
[UncertaintyPlanDistributionSetMappingsHistoricalRecords]
([UncertaintyDistributionSetID], [PlanID], [VariableID]);

CREATE INDEX [IDX1_UncertaintyPlanDistributionSetMappingsHistoricalRecords]
ON [UncertaintyPlanDistributionSetMappingsHistoricalRecords]
([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyPlanDistributionSetMappingsHistoricalRecords]
ON [UncertaintyPlanDistributionSetMappingsHistoricalRecords] ([PlanID]);

CREATE INDEX [IDX3_UncertaintyPlanDistributionSetMappingsHistoricalRecords]
ON [UncertaintyPlanDistributionSetMappingsHistoricalRecords] ([VariableID]);

CREATE INDEX [IDX4_UncertaintyPlanDistributionSetMappingsHistoricalRecords]
ON [UncertaintyPlanDistributionSetMappingsHistoricalRecords]
([UncertaintyDistributionTypeID]);

CREATE TABLE [UncertaintyDistributionSetParametersHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyPlanDistributionSetMappingID] INTEGER NOT NULL CONSTRAINT
    [FK1_UncertaintyDistributionSetParametersHistoricalRecords] REFERENCES
    [UncertaintyPlanDistributionSetMappingsHistoricalRecords] ([ID]),
    [ParameterOrder] INTEGER NOT NULL,

```

```

[ParameterValue] DOUBLE DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyDistributionSetParametersHistoricalRecords] ON
[UncertaintyDistributionSetParametersHistoricalRecords]
([UncertaintyPlanDistributionSetMappingID], [ParameterOrder]);

CREATE TABLE [UncertaintyToleranceRules] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
[FK1_UncertaintyToleranceRules] REFERENCES
[UncertaintyDistributionSets]([ID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT [FK2_UncertaintyToleranceRules]
REFERENCES [Variables]([VariableID]),
    [Expression] TEXT(500),
    [Constant] DOUBLE,
    [Percent] DOUBLE,
    [Validated] BOOLEAN DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyToleranceRules] ON [UncertaintyToleranceRules]
([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyToleranceRules] ON [UncertaintyToleranceRules]
([VariableID]);

CREATE TABLE [UncertaintyToleranceRulesHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyDistributionSetID] INTEGER NOT NULL CONSTRAINT
[FK1_UncertaintyToleranceRulesHistoricalRecords] REFERENCES
[UncertaintyDistributionSetsHistoricalRecords]([ID]),
    [VariableID] INTEGER NOT NULL CONSTRAINT
[FK2_UncertaintyToleranceRulesHistoricalRecords] REFERENCES
[Variables]([VariableID]),
    [Expression] TEXT(500),
    [Constant] DOUBLE,
    [Percent] DOUBLE,
    [Validated] BOOLEAN DEFAULT 0);

CREATE INDEX [IDX1_UncertaintyToleranceRulesHistoricalRecords] ON
[UncertaintyToleranceRulesHistoricalRecords]
([UncertaintyDistributionSetID]);

CREATE INDEX [IDX2_UncertaintyToleranceRulesHistoricalRecords] ON
[UncertaintyToleranceRulesHistoricalRecords] ([VariableID]);

CREATE TABLE [UncertaintyVariableProfileDistributionSetMappings] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [VariableID] INTEGER NOT NULL CONSTRAINT
[FK3_UncertaintyVariableProfileDistributionSetMappings] REFERENCES
[Variables]([VariableID]),

```

```

    [UncertaintyDistributionTypeID] INTEGER NOT NULL CONSTRAINT
    [FK4_UncertaintyVariableProfileDistributionSetMappings] REFERENCES
    [UncertaintyDistributionTypes] ([ID]);

CREATE INDEX [IDX3_UncertaintyVariableProfileDistributionSetMappings] ON
    [UncertaintyVariableProfileDistributionSetMappings] ([VariableID]);

CREATE INDEX [IDX4_UncertaintyVariableProfileDistributionSetMappings] ON
    [UncertaintyVariableProfileDistributionSetMappings]
    ([UncertaintyDistributionTypeID]);

CREATE TABLE [UncertaintyVariableProfileDistributionSetParameters] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [UncertaintyVariableProfileDistributionSetMappingID] INTEGER NOT NULL
    CONSTRAINT [FK1_UncertaintyVariableProfileDistributionSetParameters]
    REFERENCES [UncertaintyVariableProfileDistributionSetMappings] ([ID]),
    [ParameterValue] DOUBLE DEFAULT 0,
    [ParameterOrder] INTEGER NOT NULL);

CREATE INDEX [IDX1_UncertaintyVariableProfileDistributionSetParameters] ON
    [UncertaintyVariableProfileDistributionSetParameters]
    ([UncertaintyVariableProfileDistributionSetMappingID], [ParameterOrder]);

CREATE TABLE [WatershedLocationHistoricalRecords] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
    [FK_WatershedLocationHistoricalRecords_PlanningSetID] REFERENCES
    [PlanningSets] ([PlanningSetID]),
    [LocationName] TEXT(255) NOT NULL,
    [LocationCode] TEXT(10) NOT NULL,
    [LocationDescription] TEXT(500),
    [Latitude] DOUBLE,
    [Longitude] DOUBLE,
    [NumberOfPlans] INTEGER NOT NULL DEFAULT 1);

CREATE INDEX [UIDX_WatershedLocations_PlanningSetID_LocationCode] ON
    [WatershedLocationHistoricalRecords] ([PlanningSetID], [LocationCode]);

CREATE UNIQUE INDEX [UIDX_WatershedLocations_PlanningSetID_LocationName] ON
    [WatershedLocationHistoricalRecords] ([PlanningSetID], [LocationName]);

CREATE TABLE [WatershedStudies] (
    [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT(255) NOT NULL,
    [Description] TEXT(500),
    [DateCreated] DATETIME,
    [DateLastSaved] DATETIME,
    [ComponentID] INTEGER NOT NULL CONSTRAINT [FK_WatershedStudies_Components]
    REFERENCES [Components] ([ComponentID]);

```

```
CREATE UNIQUE INDEX [UIDX_WatershedStudies_Name] ON [WatershedStudies]
([Name]);
```

```
CREATE INDEX [IDX_WatershedStudies_Components] ON [WatershedStudies]
([ComponentID]);
```

```
CREATE TABLE [WatershedLocations] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [WatershedStudyID] INTEGER NOT NULL CONSTRAINT
  [FK_WatershedStudies_WatershedLocations] REFERENCES
  [WatershedStudies]([ID]),
  [LocationName] TEXT(255) NOT NULL,
  [LocationCode] TEXT(10) NOT NULL,
  [LocationDescription] TEXT(500),
  [Latitude] DOUBLE,
  [Longitude] DOUBLE,
  [NumberOfPlans] INTEGER NOT NULL DEFAULT 1,
  [DisplayOrder] DOUBLE NOT NULL DEFAULT 0,
  [GeneratedAlternativesSetID] INTEGER CONSTRAINT
  [FK_WatershedLocations_PlanningSets] REFERENCES
  [PlanningSets]([PlanningSetID]));
```

```
CREATE UNIQUE INDEX [UIDX_WatershedLocations_LocationCode] ON
[WatershedLocations] ([WatershedStudyID], [LocationCode]);
```

```
CREATE UNIQUE INDEX [UIDX_WatershedLocations_LocationName] ON
[WatershedLocations] ([WatershedStudyID], [LocationName]);
```

```
CREATE INDEX [IDX_WatershedLocations_GeneratedAlternativesSetID] ON
[WatershedLocations] ([GeneratedAlternativesSetID]);
```

```
CREATE TABLE [WatershedLocationPlanScales] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [WatershedLocationID] INTEGER NOT NULL CONSTRAINT
  [FK_WatershedLocationPlanScales_WatershedLocationID] REFERENCES
  [WatershedLocations]([ID]),
  [ScaleName] TEXT(255) NOT NULL,
  [ScaleDescription] TEXT(500),
  [ScaleNumber] INTEGER NOT NULL DEFAULT 1);
```

```
CREATE INDEX [IDX_WatershedLocationPlanScales_WatershedLocationID] ON
[WatershedLocationPlanScales] ([WatershedLocationID]);
```

```
CREATE TABLE [WatershedLocationPlanScaleEffects] (
  [ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  [LocationPlanScaleID] INTEGER NOT NULL CONSTRAINT
  [FK_WatershedLocationPlanScaleEffects_LocationPlanScaleID] REFERENCES
  [WatershedLocationPlanScales]([ID]),
```

```

[VariableID] INTEGER NOT NULL CONSTRAINT
[FK_WatershedLocationPlanScaleEffects_VariableID] REFERENCES
[Variables]([VariableID]),
[EffectOnVariable] NUMERIC(18, 3) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_WatershedLocationPlanScaleEffects_LocationPlanScaleID_VariableID] ON
[WatershedLocationPlanScaleEffects] ([LocationPlanScaleID], [VariableID]);

CREATE TABLE [WatershedLocationPlanScalesHistoricalRecords] (
[ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
[WatershedLocationID] INTEGER NOT NULL CONSTRAINT
[FK_WatershedLocationPlanScalesHistoricalRecords_WatershedLocationID]
REFERENCES [WatershedLocationHistoricalRecords]([ID]),
[ScaleName] TEXT(255) NOT NULL,
[ScaleDescription] TEXT(500),
[ScaleNumber] INTEGER NOT NULL DEFAULT 1);

CREATE INDEX
[IDX_WatershedLocationPlanScalesHistoricalRecords_WatershedLocationID] ON
[WatershedLocationPlanScalesHistoricalRecords] ([WatershedLocationID]);

CREATE TABLE [WatershedLocationPlanScaleEffectsHistoricalRecords] (
[ID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
[LocationPlanScaleID] INTEGER NOT NULL CONSTRAINT
[FK_WatershedLocationPlanScaleEffectsHistoricalRecords_LocationPlanScaleID]
REFERENCES [WatershedLocationPlanScalesHistoricalRecords]([ID]),
[VariableID] INTEGER NOT NULL CONSTRAINT
[FK_WatershedLocationPlanScaleEffectsHistoricalRecords_VariableID]
REFERENCES [Variables]([VariableID]),
[EffectOnVariable] NUMERIC(18, 3) NOT NULL DEFAULT 0);

CREATE UNIQUE INDEX
[UIDX_WatershedLocationPlanScaleEffectsHistoricalRecords_LocationPlanScaleID
_VariableID] ON [WatershedLocationPlanScaleEffectsHistoricalRecords]
([LocationPlanScaleID], [VariableID]);

```

D.4 Uncertainty Database for Child Sets

When users generate a planning set through the Uncertainty module, the Monte Carlo simulation generates a planning set for each iteration. These child sets are used to construct a final parent set, as well as perform CE/ICA, but they are typically not directly accessed and viewed by users. In order to minimize performance degradation of the database by storing large quantities of infrequently accessed data, the development team decided to store the child planning sets in a separate database. Therefore each uncertainty planning set will have a corresponding database containing all child sets created during the Monte Carlo analysis. An entity relationship diagram (ERD) of this database is displayed in Figure 16.

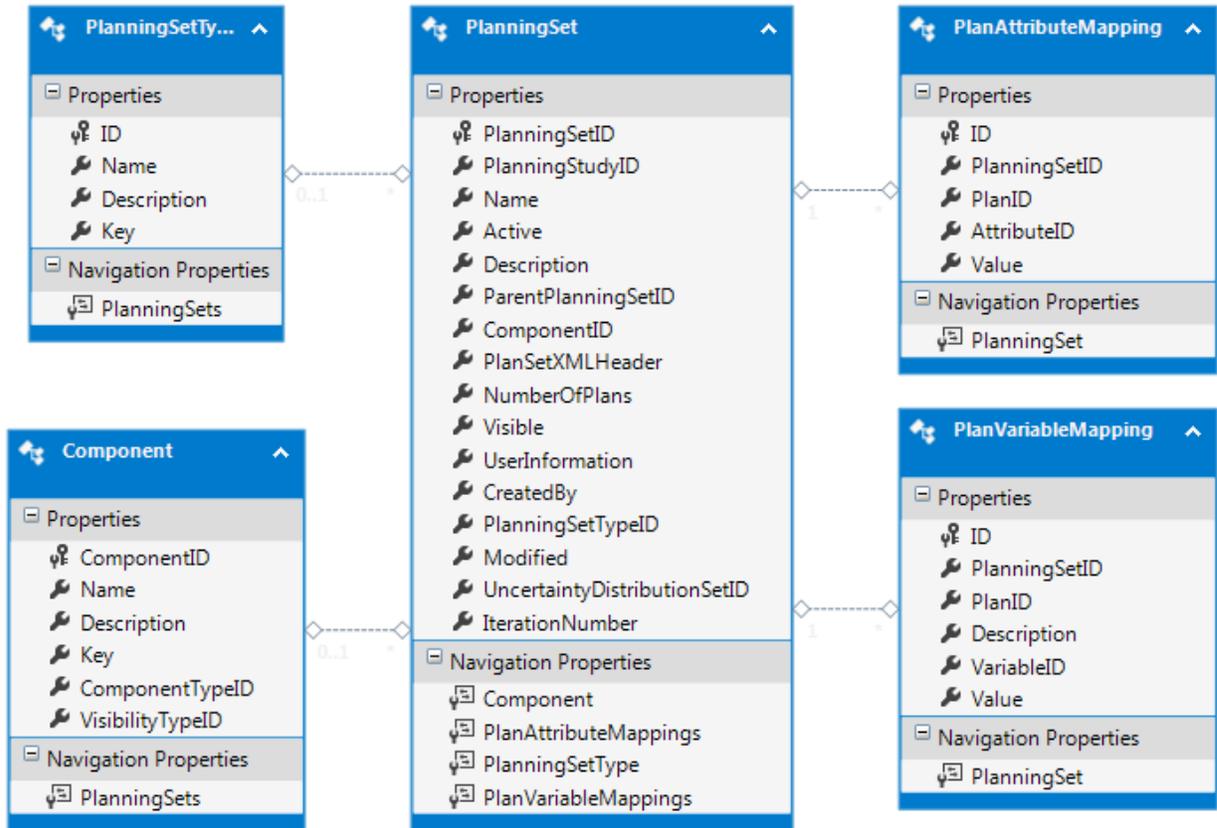


Figure 18
Uncertainty Child Database Diagram

D.5 Uncertainty Database DDL

The DDL for all Uncertainty child databases is shown below:

```
CREATE TABLE [Components] (
    [ComponentID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT(255),
    [Description] TEXT(500),
    [Key] TEXT(255),
    [ComponentTypeID] INTEGER NOT NULL CONSTRAINT
[FK_Components_ComponentTypeID] REFERENCES
[ComponentTypes] ([ComponentTypeID]),
    [VisibilityTypeID] INTEGER NOT NULL CONSTRAINT
[FK_Components_VisibilityTypeID] REFERENCES
[VisibilityTypes] ([VisibilityTypeID]);

CREATE INDEX [IDX_Components_VisibilityType] ON [Components]
([VisibilityTypeID]);

CREATE UNIQUE INDEX [UIDX_Components_Key] ON [Components] ([Key]);

CREATE UNIQUE INDEX [UIDX_Components_Name] ON [Components] ([Name]);
```

```

CREATE TABLE [PlanningSetTypes] (
    [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [Name] TEXT(255) NOT NULL,
    [Description] TEXT(500),
    [Key] TEXT(255) NOT NULL);

CREATE UNIQUE INDEX [UIDX_PlanningSetTypes_Key] ON [PlanningSetTypes]
([Key]);

CREATE UNIQUE INDEX [UIDX_PlanningSetTypes_Name] ON [PlanningSetTypes]
([Name]);

CREATE TABLE [PlanningSets] (
    [PlanningSetID] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    [PlanningStudyID] INTEGER NOT NULL,
    [Name] TEXT(255) NOT NULL,
    [Active] BOOLEAN NOT NULL DEFAULT 0,
    [Description] TEXT(500),
    [ParentPlanningSetID] INTEGER,
    [ComponentID] INTEGER CONSTRAINT [FK_PlanningSets_ComponentID] REFERENCES
[Components]([ComponentID]),
    [PlanSetXMLHeader] TEXT,
    [NumberOfPlans] INTEGER,
    [Visible] BOOLEAN DEFAULT 0,
    [UserInformation] TEXT(255),
    [CreatedBy] TEXT(255),
    [PlanningSetTypeID] INTEGER CONSTRAINT [FK_PlanningSets_PlanningSetTypeID]
REFERENCES [PlanningSetTypes]([ID]),
    [Modified] BOOLEAN NOT NULL DEFAULT 0,
    [UncertaintyDistributionSetID] INTEGER,
    [IterationNumber] INTEGER);

CREATE INDEX [IDX_PlanningSets_Active] ON [PlanningSets] ([Active]);

CREATE INDEX [IDX_PlanningSets_ComponentID] ON [PlanningSets]
([ComponentID]);

CREATE INDEX [IDX_PlanningSets_ParentPlanningSetID] ON [PlanningSets]
([ParentPlanningSetID]);

CREATE INDEX [IDX_PlanningSets_PlanningSetTypeID] ON [PlanningSets]
([PlanningSetTypeID]);

CREATE INDEX [IDX_PlanningSets_UncertaintyDistributionSetID] ON
[PlanningSets] ([UncertaintyDistributionSetID]);

CREATE INDEX [IDX_PlanningSets_Visible] ON [PlanningSets] ([Visible]);

```

```

CREATE TABLE [PlanAttributeMappings] (
    [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternativeAttributes_PlanningSetID] REFERENCES
[PlanningSets]([PlanningSetID]),
    [PlanID] INTEGER NOT NULL,
    [AttributeID] INTEGER NOT NULL,
    [Value] TEXT(50));

CREATE INDEX [IDX_PlanAttributeMatrix_AttributeID] ON
[PlanAttributeMappings] ([AttributeID]);

CREATE INDEX [IDX_PlanAttributeMatrix_PlanID_AttributeID] ON
[PlanAttributeMappings] ([PlanID], [AttributeID]);

CREATE UNIQUE INDEX
[UIDX_PlanAttributeMatrix_PlanningSetID_PlanID_AttributeID] ON
[PlanAttributeMappings] ([PlanningSetID], [PlanID], [AttributeID]);

CREATE TABLE [PlanVariableMappings] (
    [ID] INTEGER PRIMARY KEY AUTOINCREMENT,
    [PlanningSetID] INTEGER NOT NULL CONSTRAINT
[FK_SharedPlanAlternatives_PlanningSetID] REFERENCES
[PlanningSets]([PlanningSetID]),
    [PlanID] INTEGER NOT NULL,
    [Description] TEXT(255),
    [VariableID] INTEGER NOT NULL,
    [Value] NUMERIC(18, 3) DEFAULT 0);

CREATE INDEX [IDX_PlanVariableMatrix_PlanID_VariableID] ON
[PlanVariableMappings] ([PlanID], [VariableID]);

CREATE INDEX [IDX_PlanVariableMatrix_VariableID] ON [PlanVariableMappings]
([VariableID]);

CREATE UNIQUE INDEX
[UIDX_PlanVariableMatrix_PlanningSetID_PlanID_VariableID] ON
[PlanVariableMappings] ([PlanningSetID], [PlanID], [VariableID]);

```

