# A Knowledge Based Approach to Facilitate Engineering Design

by

Satwiksai Seshasai

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 10, 2002



Author_____

Satwiksai Seshasai
Department of Electrical Engineering and Computer Science
May 10, 2002

Certified by_____

Dr. Amar Gupta
Co-Director, Productivity From Information Technology Initiative
Sloan School of Management
Thesis Supervisor

Accepted by_____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

*A Knowledge Based Approach to Facilitate Engineering Design*

by

*Satwiksai Seshasai*

Submitted to the
Department of Electrical Engineering and Computer Science

May 10, 2002

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

# Abstract

This thesis presents a knowledge-based approach to facilitate the engineering design process, especially in a distributed environment where collaboration is necessary. A major impediment to such collaboration is the issue of communications - clearly and efficiently expressing the desires of every stakeholder in the process, as well as the major decisions and the rationale behind these decisions. The approach described in this paper, embodied in the MIST system, provides a framework for making these decisions in the engineering design process, by eliciting and capturing the goals and desires of every stakeholder in the design process through utility and expense functions. An interactive system was designed using a four faceted knowledge-based framework of knowledge acquisition, knowledge discovery, knowledge management and knowledge dissemination to provide designers and stakeholders with valuable information about the design process. The MIST approach will be combined with the SSPARCy approach, also developed in this group, to address crucial applications that are today contingent on geographical proximity to occur with equal or superior effectiveness in a virtual world. While this paper analyzes a situation involving engineering design, the proposed knowledge-based approach is equally applicable to collaboration in business, healthcare, government, and other environments.

***Thesis Supervisor: Amar Gupta***
***Title: Co-Director, "PROFIT" Initiative, Sloan School of Management***

## Acknowledgements

I would like to thank all the people who have contributed to the development of the approach described in this paper and all those individuals who have supported me in the process. This work would not have been possible without the support of the entire team at the Space Systems, Policy and Architecture Research Consortium at MIT. Professors Daniel Hastings, Hugh McManus and Joyce Warmkessel have provided valuable guidance and feedback at every stage of the process. Adam Ross and Nathan Diller were the principle developers of the MATE interview process, and provided the motivation for developing a software system to implement this approach.

The implementation of this system would not have been possible without the contributions of the MIT SSPARC Information Technology team. Winston Chang, Anna Konfisakhar, Carolyn Ng, Frank Reyes, Tara Sainath, Connie Tao, and Jason Yeung were each responsible for a part of the software system, and helped determine the method of implementation for the project.

Finally, and most importantly, Dr. Amar Gupta has served as an invaluable advisor for this project, providing much needed advice, support and motivation. I would like to thank him for identifying this topic as appropriate for me, for his constant input into its progress and for his broad perspective on knowledge-based systems, which helped guide this project.

# TABLE OF CONTENTS

# 1 Introduction

One of the common characteristics of many design processes is the frequent repetition of fundamental decisions and designs from project to project. Often, as the design cycle becomes more pressured by time, the importance placed on capturing the decisions being made, as well as the rationale behind these decisions is reduced [1]. With the coming and going of team members, and the lack of any formalized means of representing knowledge about the design process, much knowledge is lost, and future design projects may go through the same processes for very similar projects. For example, in spacecraft design, it is common for the similar parts to be redesigned every five years, and go through the same design process [1]. This has tremendous effects on design cycle time and cost. Companies may spend years and millions of dollars making decisions that were made in previous projects in very similar settings.

Making decisions in the modern world, with multiple teams and multiple stakeholders in various locations presents many additional challenges. A major impediment to such collaboration is the issue of communications - clearly and efficiently expressing the desires of every stakeholder in the process, as well as the major decisions and the rationale behind these decisions. In a traditional design environment, this knowledge is often transferred through informal contact, and so transferring relevant and useful knowledge from one environment to another has the potential to provide significant benefits to decision-makers dealing with similar problems [2]. Emerging technologies have provided the ability to make knowledge much more available, but the vast amounts of knowledge collected makes it even harder for people to decide what knowledge is useful.

The ability to consult design rationale and reuse decisions has the power to greatly impact the time and cost of the design process. For example, NASA's Jet Propulsion Laboratory recently implemented a conceptual design model from the Aerospace Corporation which provided a framework for design representation and analysis. Many of the proposals developed by the Advanced Projects Design Team at JPL used knowledge from different subsystems and design models. By using these methods, JPL was able to reduce the time it took to write proposals from 6 months to 2 weeks, and reduce the cost from $250,000 to $80,000 [1]. The potential impact for the actual design process at major companies is analogous, and could save years and millions of dollars.

This thesis discusses an effort to pursue this goal by providing a common interface for members of the design process to collaborate, share goals and desires, and overall rationale. This approach not only builds a knowledge repository for the specific design process being conducted, but also facilitates knowledge discovery by immediately presenting and processing data from all stages of the design process. The immediate accessibility of significant knowledge about the design process has the potential to play a powerful role in an increasingly distributed and virtual design world. Although the specific case discussed here relates to the design of space systems, the proposed approach can be used to provide value to many other domains where increased knowledge is required in the decision making process.

Recognizing that interaction with the customer (or decision-maker) and meeting their needs is crucial to the design process, this endeavor focused on developing methods for gathering customer input and analyzing the utility of a design as a revolutionary

means of capturing design rationale. This method truly seeks to understand the "why" in spacecraft design, a "why" that only comes from capturing the true cost and utility to the customer. This process will collect specific data relating to the customer's preferences for easy updates over time, and will be integrated with data from the design process collected by the previously designed SSPARCy tool, discussed later in the thesis, to provide a true capture of the design drivers for the duration of the project.

# 2 Background

## 2.1 SSPARC Consortium

This research is being conducted as part of the Space Systems Policy and Architecture Consortium (SSPARC) at MIT. The purpose of this group is to examine space system design from a variety of perspectives, and specifically, produce optimal methods for choosing between various choices in space system architectures [3]. Current design methods do not provide efficient means for rationally choosing between a vast set of possible architectures. Simply making an a priori choice of architecture and designing the system around it allows optimization in a local (e.g., spacecraft) sense, but not necessarily in a global (e.g., architecture) sense[i]. To move towards making better higher-level decisions, it becomes especially vital to capture and process as much knowledge about the systems as possible.

## 2.2 PROFIT Initiative

The approach described in this thesis achieves the goal of facilitating the design process by using the four faceted knowledge-based approaches of knowledge acquisition, knowledge discovery, knowledge management and knowledge dissemination developed in the PROFIT initiative at the Sloan School of Management at MIT. The powerful impact of this method comes from the understanding that the missing piece of the engineering design process is a sincere effort to capture, understand and reuse knowledge about the process. Examples of this knowledge are the goals and requirements of various stakeholders in the process, the rationale behind major design decisions, the actual decisions made along the design process, and the history of the actual design. Such

knowledge exists within the system, but requires a formalized method of eliciting and processing to truly be utilized effectively. Once harnessed, this knowledge has the power to reduce design cycle time, allow for a more informed and intelligent exchange of ideas in the design process, and induce collaboration by sharing knowledge previously stored informally and dependant on geographic proximity. This approach centers the project on the fundamental notion that using knowledge within the design process can be used to more effectively achieve engineering goals by leveraging the following facets on a concurrent basis [4]:

• **Knowledge Acquisition** is the process of capturing information from various media, including people's minds and handwritten documents, into computer accessible media.

• **Knowledge Management** deals with mitigating issues relating to heterogeneities in underlying contexts of information coming from disparate sources such as multiple stakeholders, multiple projects and multiple stages of the process.

• **Knowledge Discovery** involves using emerging techniques to analyze huge amounts of information and to get better insights into such information than is possible using the best human domain experts.

• **Knowledge Dissemination** is the automated extraction of the most relevant pieces of information from a huge computer based information infrastructure with such extraction being tailored to the needs of different constituencies of users in an organization.

This four-faceted technique allows efficient exchange of knowledge vital to collaboration. In a later section, we demonstrate how each of these four building blocks has driven the design of these tools.

## 2.3 SSPARCy project

To support the process of capturing design rationale in an efficient fashion, the SSPARCy application was designed and implemented [5]. SSPARCy performs automatic information extraction of vital information from MATLAB source code and output from CalTech's ICEMaker, a tool used to extract data from a design process based in Microsoft Excel. It then allows users to view this information graphically, records the history of the states of the code, performs integration integrity checks and facilitates the capturing of design rationale associated with important code elements by providing an interface for designers to input text-based design rationale comments.

The SSPARCy application offers a suite of features that provide users with a centralized source of information regarding a simulation. It presents the current state of a simulation in order to enable a user to quickly access important information automatically. Additionally, it records the history of the various states of the entities of a simulation, allowing a designer to examine and analyze the evolution of a simulation over time. Furthermore, it performs automatic analysis of system integration integrity to alert an integrator of potential problems. This set of capabilities supplies a powerful set of tools to support the process of design rationale capture.

## 2.4 MATE

Multi-Attribute Tradespace Exploration is a method being developed by members of the SSPARC team. It provides a formalized means of exploring a tradespace by incorporating preferences into decision criteria with methods based in economic and operations research theory [6]. This is done by using both utility and cost-benefit analysis methods to obtain information from all of the stakeholders in the design process and

facilitate communication between them. The principle means of eliciting information is by conducting a series of interviews with the stakeholders to capture their preferences regarding the various attributes of the design architecture. This data is then used to drive the design process, by providing information about the utility and cost of each architecture being explored in the tradespace. The interview process is central to the goals of the MATE method, and will be one of the main focuses of this project.

# 3 Related Approaches

Before we present details of our knowledge-based approach, we present information of related efforts by other researchers.

## *3.1 Vehicles Knowledge-Based Design Approach*

The Vehicles Knowledge-based design environment, developed at the Aerospace Corporation [7], is a framework for managing knowledge related to the design environment of space systems. Analysis and modeling tools are combined with a historical database of previous projects to assist in building new architectures that benefit from the experiences of old architectures. The issues addressed in Vehicles are similar to the issues discussed in this paper, relating to what the best methods are for formalizing the knowledge related to the design environment. Vehicles proposes a flexible environment in which to describe systems and subsystems, and to analyze the effects of various changes to the design. The software environment provides a platform for designers to build their own tools, tailored for the specific types of analysis they wish to conduct. The design rationale, history and utility capture capabilities described in this paper can provide a complimentary value proposition to the Vehicles system.

## *3.2 Boeing's Knowledge System for Design*

The Boeing Company has developed a knowledge-based system to expedite design analysis by using an expert system to process computer programs relevant to understanding the design of the various subsystems [8]. The system understands the inputs and outputs of each subsystem's computer program, and can provide knowledge and insight into specific parts of the design as queried. For example, if the design analyst

wishes to start with a given output, the system can compute the required values for the parameters of the system to reach that particular output. The core contribution of this system is the separation of knowledge about how to execute the computer programs from the specific facts related to the programs. In other words, this system provides an interface to the design without actually having to understand the mechanics of the design. The MIST system described in this paper takes a complimentary approach to facilitating design analysis, by providing a means of assessing various architectures in terms of stakeholder utility. This takes a system such as the Boeing example one step further by potentially allowing design analysts to start with a goal of a certain utility, and determine which strategy for the design can best suit these goals.

## 3.3 Rule-based Algorithms from Chung-Hua University

One system, under development at the Chung-Hua University in Taiwan [9], uses a rule-based algorithm for transferring an individual customer's needs directly into specifications, by developing a matrix of weights between attributes and design factors. The weights are then used to determine rules, or relations, indicating how certain design parameters should change their respective values based on other values of parameters. The proposed paradigm can build on this approach by incorporating information on the goals and needs of multiple stakeholders, as well as by using the evolving knowledge repository to facilitate decisions on how specifications can be derived from customer preferences.

## 3.4 Artificial Intelligence Design of Aircraft

The Artificial Intelligence Design of Aircraft (AIDA) effort at the Delft University of Technology applies a case-based reasoning method to delineate initial

values for an aircraft lay-out using knowledge from previous cases [10]. Once again, however, the AIDA approach is dealing with the goals and desires of only one stakeholder. Additionally, the AIDA approach seeks to find one optimal architecture, rather than to provide a knowledge-based framework to be used by the designer to explore the various options.

## 3.5 SPOOL: Reverse Engineering for Design Rationale

The SPOOL project at the Université de Montréal takes a slightly different approach; it uses reverse engineering techniques to analyze existing projects and to determine which patterns in the design could be used to infer the rationale behind recurring patterns [11]. This minimizes the need for the designer to provide the rationale. By combining utility and expense data with the design history, our proposed approach will be able to make similar inferences regarding the design rationale at every stage of the design process, thereby offering an improvement over both the SPOOL approach and the other approaches mentioned above.

## 3.6 C-DeSS: Geometric Design Rationale

The C-DeSS system developed by Klein proposes a geometric design rationale tool [12]. This system provides a language through which a designer can describe the design geometry and express reasons for decisions made in forming this geometry. Elements of such a structured language can be integrated into the framework described in this paper, to form a more concrete representation for the entire design, not just for the geometry.

## 3.7 Distributed and Integrated Collaborative Engineering

The DICE (Distributed and Integrated Collaborative Engineering Environment) methodology initiated by Sriram offers a platform for collaborative engineering by decomposing each engineering project into a set of modules and allowing work to be conducted in parallel on each section of the project [13]. When the system encounters conflicting decisions about a particular design decision from engineers in different modules, it uses the design rationale to help negotiate the outcome. The approach described in this paper would complement the DICE approach by providing a concrete relationship between the history of the design parameters and the associated utility and expense functions which led to these decisions. This would allow engineers, working in a set of collaborative enterprises, who have conflicting solutions to a given problem to gauge the exact implications of each option on the different stakeholders of the system.

## 3.8 WAVE: Algorithm for Information Extraction

WAVE is an algorithm to learn information extraction rules [14]. Since it is intended to be an algorithm, it does not offer the broad functionality that is available in SSPARCy. However, the incorporation of the WAVE algorithm into SSPARCy would augment the latter's flexibility and ability to adapt to changes in the syntax of the designer code. Further, this could potentially enable the application to analyze other programming language by allowing SSPARCy to learn the information extraction rules for a new language over time.

## 3.9 Utility Evaluation from the University of Massachusetts

The Trade-off Based Robust Modeling and Design group at the University of Massachusetts has developed an online utility evaluation tool which conducts interviews

to determine stakeholder preferences in a similar fashion to the MIST approach [15]. The system incorporates mechanisms to deal with preference consistency and uncertainty and risk. These issues are addressed in the context of a given stakeholder for a given project. The MIST approach builds on this methodology by capturing information which will be useful in determining the relations between preferences of multiple stakeholders, at multiple stages of the design process, for multiple projects.

## 3.10 ICAD: Knowledge Based Engineering

The ICAD system is a Knowledge Based Engineering software solution used by world class manufacturers in aerospace, automotive and industrial equipment manufacturing, such as Boeing, British Aerospace, Pratt and Whitney, GM, Ford, Jaguar, Lotus and others, to automate system-level design, product design, tooling and product configuration [16]. This system uses generative technology to capture and apply generic product design knowledge, which includes product structures, development processes, and manufacturability rules. Such a system represents a potential application for the knowledge captured and managed by the approach described in this paper.

## 3.11 NASA's Virtual System Design Environment

The Virtual System Design Environment at NASA addresses the importance of facilitating collaborative environments in spacecraft design [17]. One of the focuses of the system is to enable the modification of a spacecraft structural component by various members of a team in different geographic locations. This issue can be extended by the MIST approach to incorporate the direct involvement of the stakeholders in the system. If designers are able to interact at the component design level over a network, the same

mechanisms can be used to demonstrate concepts, explain the status of the project, and gather input from the stakeholders of the system.

## 3.12 Design Repositories at NIST

The National Institute of Standards and Technology Design Repository Project addresses the importance of developing formal representations of knowledge in the design environment [18]. A specific language is being developed to represent design models, and interfaces for creating, browsing and searching for information on these models are being developed as well. The integration of design history and rationale with utility information from multiple stakeholders represents a complimentary means of representing a design model which addresses many of the same issues. For systems like these to work together, a set of standards must be developed, as described by the Engineering Design Technologies group at NIST [19].

## 3.13 Other Design Rationale Tools

The existing field of design rationale capture tools spans the spectrum from fully unstructured rationale to completely modeled rationale. Meeting minutes represent an unstructured, time delineated capture. QuestMap [20] and DRAMA [21] are examples of the next step – they provide basic structural elements and enable the user to devise a useful structure. At the other end of the spectrum from meeting minutes is DRIM [22], which is a completely specified model for the rationale underlying the design process. As a design rationale capture tool, SSPARCy lies somewhere on the spectrum between QuestMap and DRIM. SSPARCy creates a simple structure for design rationale by associating rationale with each simulation entity. Additionally, SSPARCy captures this rationale over time. Since this rationale can evolve at any level from project to variable

over time, a minimal logical structure is provided for the user to specify rationale in the manner and the level he or she perceives as being most beneficial. SSPARCy does not impose the rigid conceptual structure that DRIM proposes. Therefore, SSPARCy is a compromise in terms of design rationale capture between inflexible structure and amorphous disorder. Furthermore, the basic structure it provides is most appropriate for the domain-specific design process it endeavors to capture.

Design reuse is an obvious application for design rationale. Work at the Tsinghua University in Beijing demonstrates a prototype for using design rationale to support design reuse [23]. The Tsinghua system incorporates many intelligent mechanisms to process design rationale information and relate it to the design of the system. The knowledge captured by the system is used in future projects as a means of decision support.

# 4 Knowledge Based Framework

## *4.1 Overview*

MATE and its follow-on Concurrent Design method (MATE-CON) require a software platform to facilitate the higher level of designer and customer interaction. The system supplements the cumbersome face-to-face Multi-Attribute Utility Analysis (MAUA) interview process previously developed by the SSPARC team by using a Graphical User Interface (GUI) and potentially web-based computer interface with graphics that speed up the utility interview process and provide a facility to help document customer preference on a continuous basis. The designer is able to describe attributes, ranges of values, units and scenarios developed with input from the customer. The system then prepares an interview based on the attributes of the tradespace, and allows the designer to conduct the interview and enter the responses, as well as the customer to take the interview independent of the utility facilitator. The MATE interview process consists of multiple stages, dealing first with single attribute utility parameters and then with multiple attribute utility [24]. Each stage also contains a set of validation questions to ensure that the variables being considered are independent of each other. The structure for each of these interviews is consistent, and the system facilitates each of these stages.

The automated and customized interview sessions will present the customer with a scenario using the lottery equivalent probability (LEP) approach as developed in the field of decision theory. Each option is a situation with probabilities of two states for attribute analysis. The system continues in this form until the customer answers that they are indifferent to the two options. The software then tallies the indifference points for

later calculation and moves on to analyze the next attribute. This process can be conducted over time, and the results of each interview can serve as the basis for developing functions to assess the utility and the cost of design based on the set of the attributes specified by the customer or the set of customers. The integration of these data with the design parameter data collected by the SSPARCy system provides a comprehensive assessment of the rationale at various levels of design detail throughout the process.

### 4.1.1 Knowledge Acquisition

By conducting the interview with a software tool, the process can be made much quicker, and tailored to the individual responses. Calculations can be performed while the interview session is proceeding. Designers are provided with the options for adding and modifying attributes, and also for changing the scenario used to describe an attribute to the customer. As an improved communication interface is the core goal of this endeavor, the system allows not only the designer, but also the customer to provide feedback about the structure of the interview process. The choices made in structuring the interview are interesting independent of the results of the interview. This tool can also capture the choices made during the interview process, and use such information to provide knowledge about the design of the system as well as the design of the interview itself. The latter allows for future interviews to be conducted in a more efficient manner. The major issue raised in this area is how much control and flexibility is appropriate in customization of the interview session in real-time. Allowing excessive variability in the interview process may reduce the reliability and consistency of the results.

Although the prototype system is implemented in Microsoft Visual Basic and Excel for a variety of reasons, this system is designed to be generic and not be limited by the specific implementation described here. One of the major hurdles faced by the SSPARC team in implementing this process was educating the stakeholders and helping them to think in terms of the attribute system [24]. Visual Basic provides an interface familiar to most customers. For ease of the customer and designer, Visual Basic provides pre-existing GUI functionality in addition to familiar data analysis tools, especially for presenting data in graphical and other visual formats [25]. This allowed the team to focus primarily on developing additional functionality for the system. While the robustness of the Excel environment is less than the desired level, we opted for the use of a commonly used software interface since it reduces the time and effort needed to transfer data across platforms [26].

### 4.1.2 Knowledge Management

In this concept demonstration prototype endeavor, we have emphasized the visualization aspect. The paradigm of customized visualization alone adds value to customers by helping them understand the way in which their utility estimates evolve over time. The multi-attribute cost and utility functions can be created in graphical format. One can graphically witness how answers during the interview process impact the design process. This impacts the integrity of the interview process in both positive and negative ways - a very important design issue that needs additional exploration. The positive value-added is that with increased real-time knowledge, the customer may make more rational decisions: when faced with a graph which plots each of their answers for a given attribute, a customer may be alerted to reconsider the answer. On the negative side,

the customer may tend to give conventional and risk averse responses, as most customers probably prefer to give answers which appear to be consistent and rational. Further, a customer's true preferences may be changing over time, and it might be harmful to allow the customer to "game" the system. This is another issue which can be explored with the introduction of an interactive tool to facilitate the interview. The optimal level of customization will depend on the characteristics of the particular problem domain.

### 4.1.3 Knowledge Discovery

One of the major intellectual questions that MATE attempts to address is the role of utility and attribute based design to design rationale. The SSPARCy system, developed in 2001, solicits design rationale in a somewhat ambiguous manner, by asking the designer to manually describe major design decisions in a textual format. Collecting data from the customer, and monitoring the designer's choices in creating the utility functions provides the SSPARCy system with a numerical form for design rationale. The two systems, SSPARCy and MATE, have a common interest in collecting design parameters over time, and reacting to the major trends and changes. By using one integrated system to collect both forms of data, one possibility is to relate each moment of the design process to a specific interview, thereby associating an interview with each value of a design parameter. Major changes in the design parameters can then be easily mapped against the changing preferences elicited during the interviews. Such integration provides better information to both efforts, by relating the utility functions directly to any state of the design, as well as providing the motivation for major design decisions in a numerical format.

### 4.1.4 Knowledge Dissemination

The MIST system also contains a set of analysis tools to process the data collected from the interviews and the SSPARCy design parameter capture process. First, a set of reports can be produced to document both the customer's preferences and the various stages of the design process, as well as perhaps the design rationale specified by the designer. The use of an integrated tool allows for a consistent template to be used and for easy manipulation of the sets of data being reported. A comprehensive knowledge repository will evolve over time. When a new person joins the team, such a knowledge repository provides an invaluable mechanism for transferring the knowledge about the design process to the new member.

In addition to the reports, algorithms for utility and expense function generation are being integrated with the system to produce meaningful and useful tools with the interview results. The results of the interviews will be used to design a specific function for each stakeholder at the end of each interview. This will allow all iterations of an architecture to be assessed, and the changing utility of an architecture to be related to the changing preferences of the customer. The algorithms for developing these functions have already been delineated by the SSPARC team [24].

Finally, other data mining tools will be implemented to use the knowledge repository as a vehicle for knowledge discovery within the design process. Once adequate data have been collected about a specific design process, or multiple design processes, interpreting the patterns within these data can lead to better decisions in future endeavors.

# 5 Project Description

## 5.1 Attribute Interface

The MIST system utilizes the notion of *attributes* to represent the user-defined characteristics of the project that describe the important factors for the stakeholder. Before the interview process can commence, the engineers and the stakeholder must agree on a set of attributes to build the utility functions. This process involves a series of decisions, and a set of discussions between the different members of the engineering team, as well as the stakeholder. The attribute interface has been designed to structure the important pieces of knowledge related to each attribute, so that the discussion can be focused. Since the questions for the interviews are generated while the interview session is actually "on", it is technically possible to change the properties of an attribute, or add an attribute, at any time during the process. However, since an interview is given in the context of the current state of attributes, any chance in the properties of the attributes could invalidate the previously collected interview responses. For this reason, the attributes in the system must be defined before the interview session begins.

An attribute is represented in the system as a hybrid data type with various properties as shown in the figure below. The figure shows the properties related to an attribute, the data type of each property, the description and comments. Some of these properties are directly related to the description of the system, and others were implemented in order to facilitate the software system. When an attribute is created by the design team, an instance of the attribute data type is instantiated and modified as appropriate. The interview user forms and the analysis tools base their actions on the status of the attribute properties.

| Attribute Property | Data Type | Description | Comments |
|---|---|---|---|
| `Name` | `String` | Name of attribute | Understandable to all parties involved |
| `Min` | `Double` | Lowest value in range | |
| `Max` | `Double` | Highest value in range | |
| `Units` | `String` | Units for attribute values | Must be very specific |
| `Increment` | `Integer` | Number of indifference points to be collected | Enough to avoid large jumps in utility |
| `Direction` | `Boolean` | Direction of increasing utility | Towards max or min value? |
| `Scenario` | `String` | Scenario for single attribute interview | Describes context in which only the single attribute is considered |
| `Resolution` | `Double` | Probability resolution to which utility can be distinguished | Usually around 5% |
| `Format` | `String` | Format in which value of attribute is presented to user | Examples: “5x5”, “6” |
| `UnitsInc` | `Boolean` | Include units in format? | Usually `true`. |
| `Independent` | `Double` | Attribute value for independence interviews | Usually near middle of attribute range |
| `Definition` | `String` | Definition of attribute | Must be very specific |
| `LinearScale` | `Boolean` | Increase indifference point questions on linear or logarithmic scale? | |
| `Threshold` | `Double` | Difference in utility between adjacent indifference points which requires more questions | |

**Figure 1:** Table of properties for the attribute data type in MIST

### 5.1.1 Attribute Operations

A user can add, modify, or delete an attribute as described in the following paragraphs.

The user can add an attribute to the system from the main page. When the "Add Attribute" button is clicked, the Attribute Properties form is presented to the user, and the user can enter the properties for the attribute. Since part of the purpose of this form is to enable the engineers to conduct the discussion during the attribute definition period, it is not necessary for the user to put in a value for every property of the attribute. This process must be completed by the time the interviews are ready to be conducted – otherwise, the interview generation module will search for a value of the attribute which will not be there. The only required element for the attribute is the Name, since it is stored based on the name. After the properties are entered, the user can click on the Save button, and the attribute will be added to the system. At this point, the Attribute Rationale form is displayed and the user is asked to enter a rationale for adding the attribute to the project. The name of the attribute is added to the list of attributes on the main page, both in the completed interviews matrix and the random values section.

To modify an attribute, the user can click on the Modify Attribute button from the main page. This brings up the Attribute Navigator form, which shows a list of attributes currently in the system. The user selects the attribute they wish to modify, and clicks on the Modify button. The Attribute Properties form is displayed with whichever values of the attribute are in the system. Any of these values can be modified, however since the attribute is defined by its name, changing the name would result in the system creating a new attribute with the new name. When the Save button is clicked, the user is asked if the change to the attribute is a major change worth cataloging. If it is, the user is prompted for a rationale for the change in the Attribute Rationale form.

To delete an attribute, the user must click on the Modify Attribute button from the main page, select the attribute to be deleted, and then click the Delete button. Deletion is only available under the Modify Attribute form to ensure that the user is certain that he or she wishes to delete the attribute, instead of simply modifying it to make it appropriate. When an attribute is deleted, the user is prompted for the rationale behind deleting the attribute. This rationale is stored along with the final values of the attribute, in the attribute's history page. While the attribute name is deleted from the main page and all interviews, the history of values is still stored within the project, to preserve knowledge of the decision making process which led to this deletion.

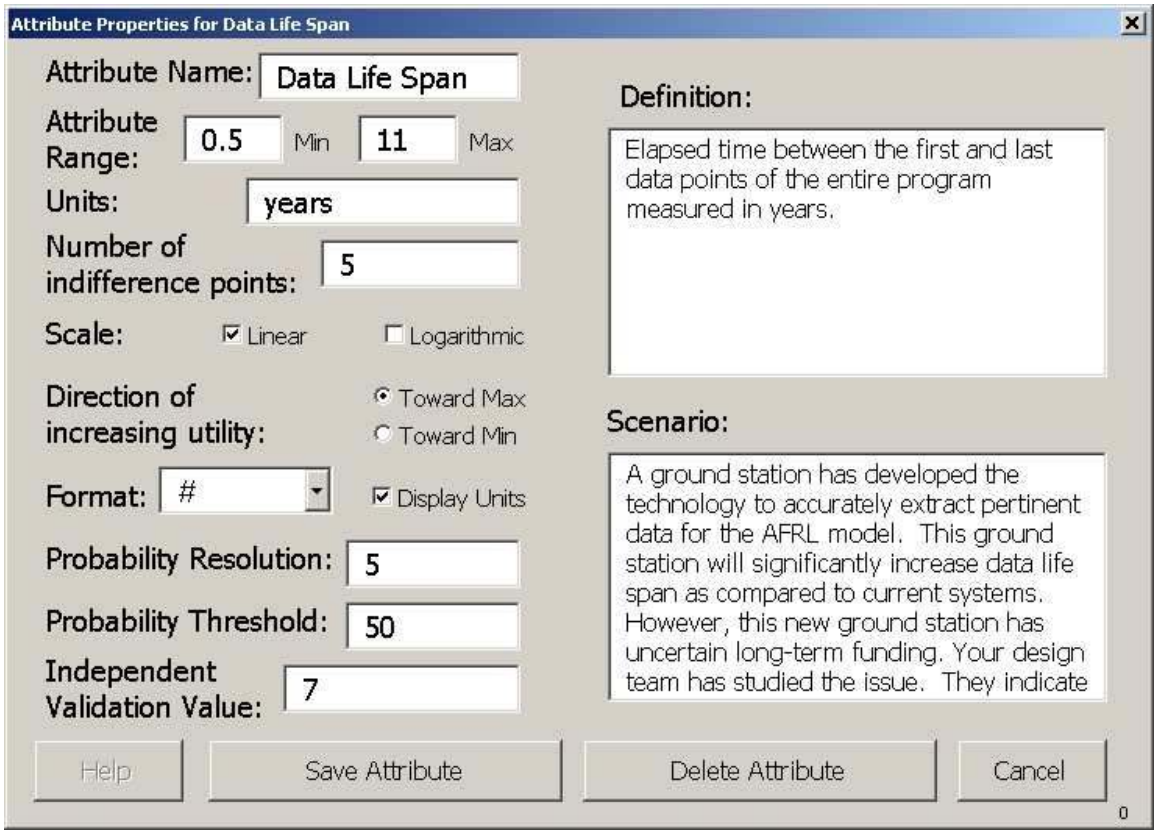


**Figure 2:** Attribute Modification interface

### 5.1.2 Attribute Storage

The data related to the attribute are stored in Excel worksheets with respect to each particular attribute. Within the same structure used to store properties for a given attribute, the results of each interview are also stored. Thus, the interview results are treated as properties of the attribute, because they directly describe the shape of the utility function for that attribute. The responses from the validation interviews are also stored with the attribute properties, because they represent the relative independence and integrity of each attribute. By storing the data with respect to the attributes, analysis of each attribute is made much easier.

Additionally, each attribute has an additional worksheet for storage of the history of each attribute. An entry is made in this structure each time the attribute is created or deleted, or the user agrees that the change made is major enough to warrant recording. The rationale for each state is stored as a property of that state. This allows the data to be used later in analysis, as well as be accessible for quick reference by users.

By using data storage structure familiar and accessible to the users, data about the attributes and interview results is much more easily accessible and editable. This storage mechanism removes the need for additional interfaces to impede the accessibility of data from the user.

## *5.2 Characteristics of Interview Process*

Each interview is designed to acquire a piece of knowledge intended to help build or validate a utility function for the stakeholder being interviewed. The interviews are meant to take place after the attribute definition phase has occurred, and the attributes have been approved by the stakeholder. The interviews can be conducted at multiple

points throughout the design process, to provide a continuing notion of the important goals and requirements of the stakeholder being interviewed.

### 5.2.1 Visualization

The issue of visualization is one that was examined in great detail by the designers of the MIST system. Using the MIST concept demonstration software, it is possible to display the curves representing the utility functions for the attributes being discussed as the interview is taking place. The figure below demonstrates this functionality for the single attribute interview, where the user's responses each help build the utility function for a single attribute. Similarly, the utility curves generated for all of the attributes can be displayed in any of the multi-attribute interviews.

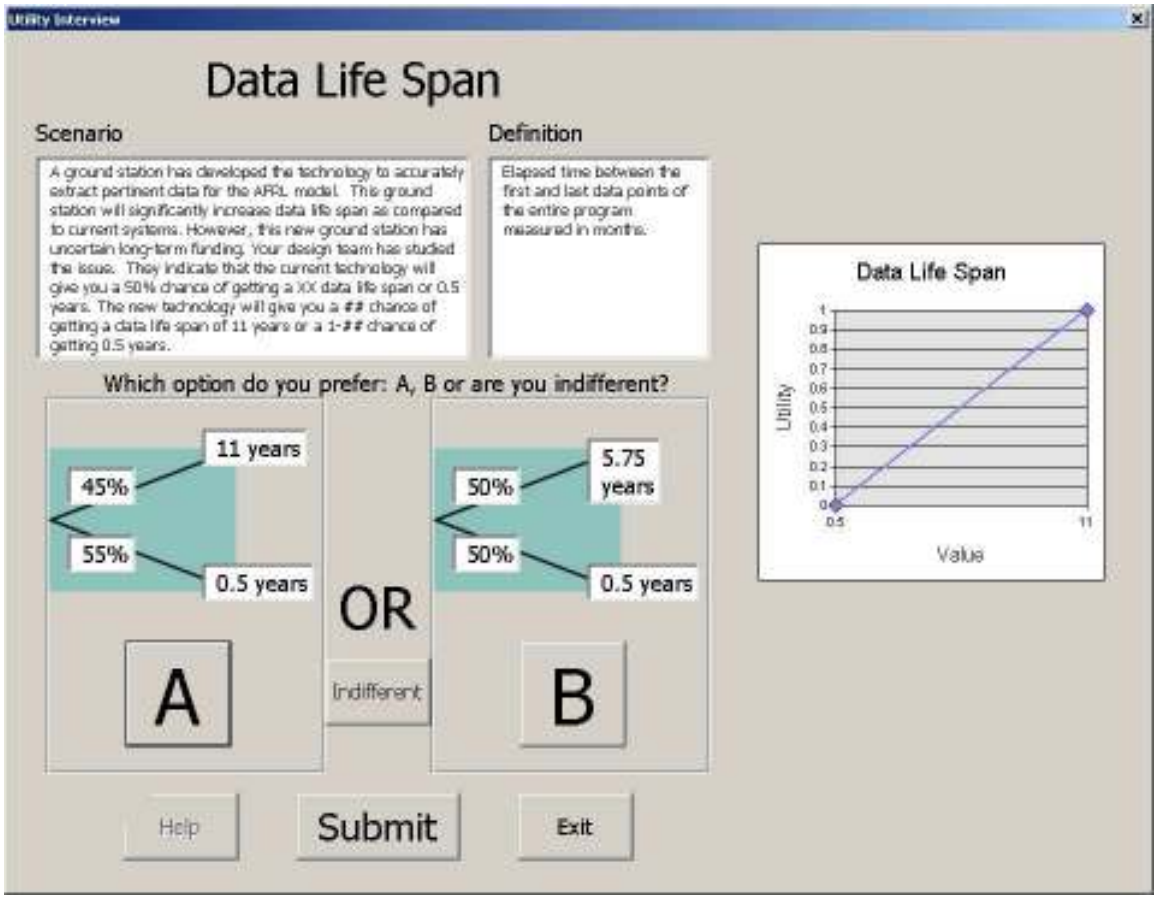



**Figure 3:** Single attribute interview with utility curve displayed.

If a response during the single attribute interview leads to a drastic change to the utility curve, the interviewee can see this immediately and think again about whether the response accurately represents his or her preferences. During the multi-attribute interviews, when the user is answering questions dealing with the relationships between attributes, having utility curves displayed might allow them to help evaluate in their minds how they actually feel about the different attributes.

When this idea was presented to the fellow designers, the decision was made to withhold the visualization features from the stakeholder, because it might provide too much information leading to bias in the responses. For example, during the latter part of a session, a stakeholder might be inclined to provide responses which fit the initially defined utility curve, to avoid appearing irrational. Another possible downfall of the visualization feature is the ability of the user to pre-determine a utility curve in their mind, and to provide answers which fit this curve. Based on further tests, one will decide whether to offer the visualization capability as an option in future versions of MIST.

### 5.2.2 Bracketing

Every interview generated by this system is designed to find the value at which the interviewee is indifferent between the two situations presented. The decision is basically between one option where the probabilities of certain outcomes are fixed, and a second situation where the probability of one outcome as opposed to another is varied. The interviews are designed so that this varied probability is between 0% and 50% -- a probability greater than 50% would result in one outcome always being better for the user.

The method for obtaining successive values is based on the algorithm used in DeLeqeue's thesis [27]. At any given time, there is a known range of probabilities in which the indifference point exists. For example, when an interview question begins, it is known that the indifference point is within the range of 0% to 50%. The software module in the MIST system which generates potential indifference points is described in the flow chart below:

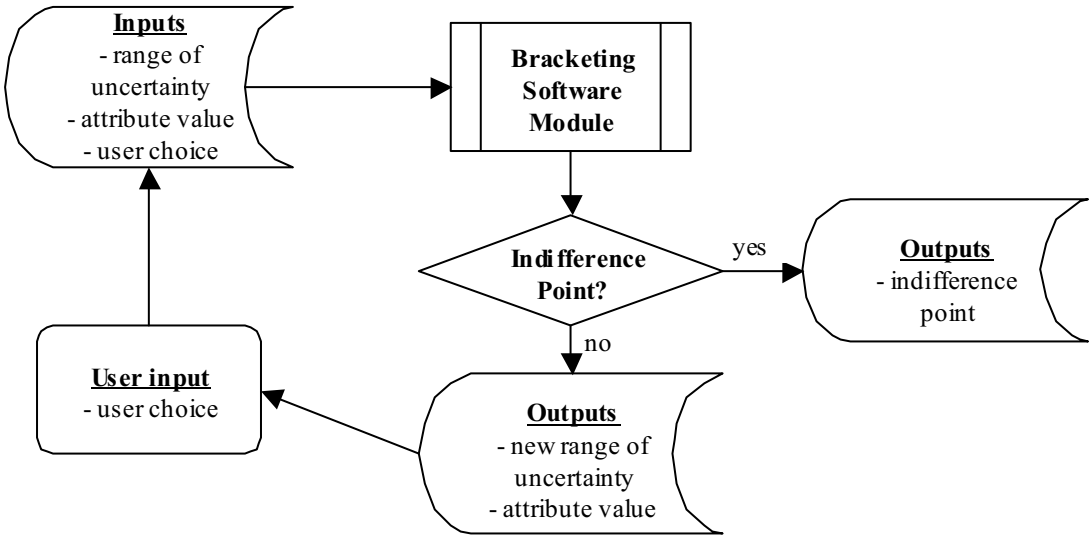



**Figure 4:** Flow chart describing bracketing software module.

Each successive question asks the user for his or her preference at the midpoint of this range. Based on the user's response, it will be clear whether the indifference point is either equal to, above or below the probability in the interview question. Either the interview ends with an indifference point selected, or the range is modified to reflect the latest response, and a new question is asked. Each probability chosen is a multiple of the attribute's probability resolution, which represents the degree to which the customer can distinguish between two situations. This process continues until the stakeholder declares a certain probability as the indifference point, or the size of the range is less than the

resolution. At this point, the stakeholder is informed that the indifference point will be set to the value exactly between the two endpoints of the range. A modification to this bracketing procedure was made for the implementation described in this paper, as described in the Test Implementation section. The figure below describes the example from the X-TOS project. The probability resolution for this example was 5%, and instead of using the midpoint of the range, values were chosen closer to the endpoints. Future versions will incorporate a more general version of this strategy.

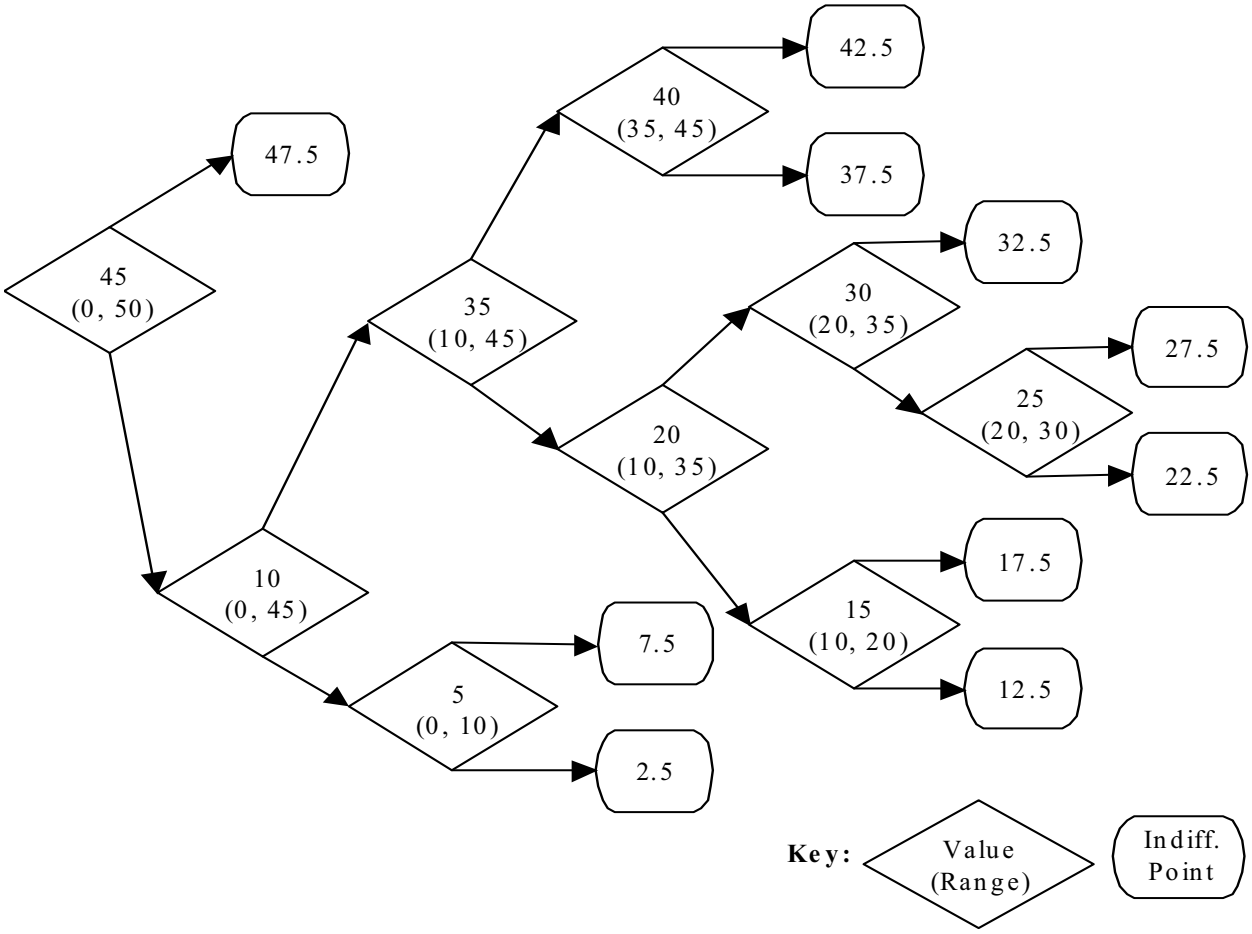



**Figure 5:** The X-TOS bracketing decision tree.

### 5.2.3 Single Attribute Interview

The goal of this interview is to build a utility function for each attribute, similar to the graph shown in the Output section of this paper. This function provides the mechanism for assessing the utility provided by differing values of the attribute for any proposed architecture. It provides decision makers with knowledge about whether one can obtain a significant gain in utility by increasing values at the lower end of the attribute range or at the high end of the range. The utility functions provide both a visual notional idea of the nature of the attribute, as well as a concrete input into a concurrent engineering simulation model. A concurrent engineering simulation can use the utility function, combined with a module which calculates the values of each of the attributes based on the architecture, to assess the utility provided for each attribute in any proposed architecture. This offers the potential to provide real-time feedback to engineers about any considered design decision without having to consult the various stakeholders each and every time.

After the attributes are defined and the properties are agreed upon, the system is deployed to the stakeholders with whom the attributes were designed. Depending on the nature of the roles involved, a single set of attributes may apply to a single stakeholder or a set of stakeholders with similar roles. The stakeholder begins an interview session with the single attribute interview.

The stakeholder navigates through the single attribute interview, as shown in the figure below, for each attribute. The user is provided with two pieces of information: the scenario and the definition of the attribute. The scenario is written to place the question in a context meant to emphasize that the specific attribute in the interview is the only aspect of the system to be considered at the time. Thus, a situation is described, such as

the discovery of a new technology which has the potential to affect the value of the attribute, but carries with it some risk of failure. Identifying the level of risk that the user is willing to take is one of the purposes of the single attribute interview. The definition of the attribute is also displayed because often in engineering situations, the specific interpretation of a concept varies among different parties.

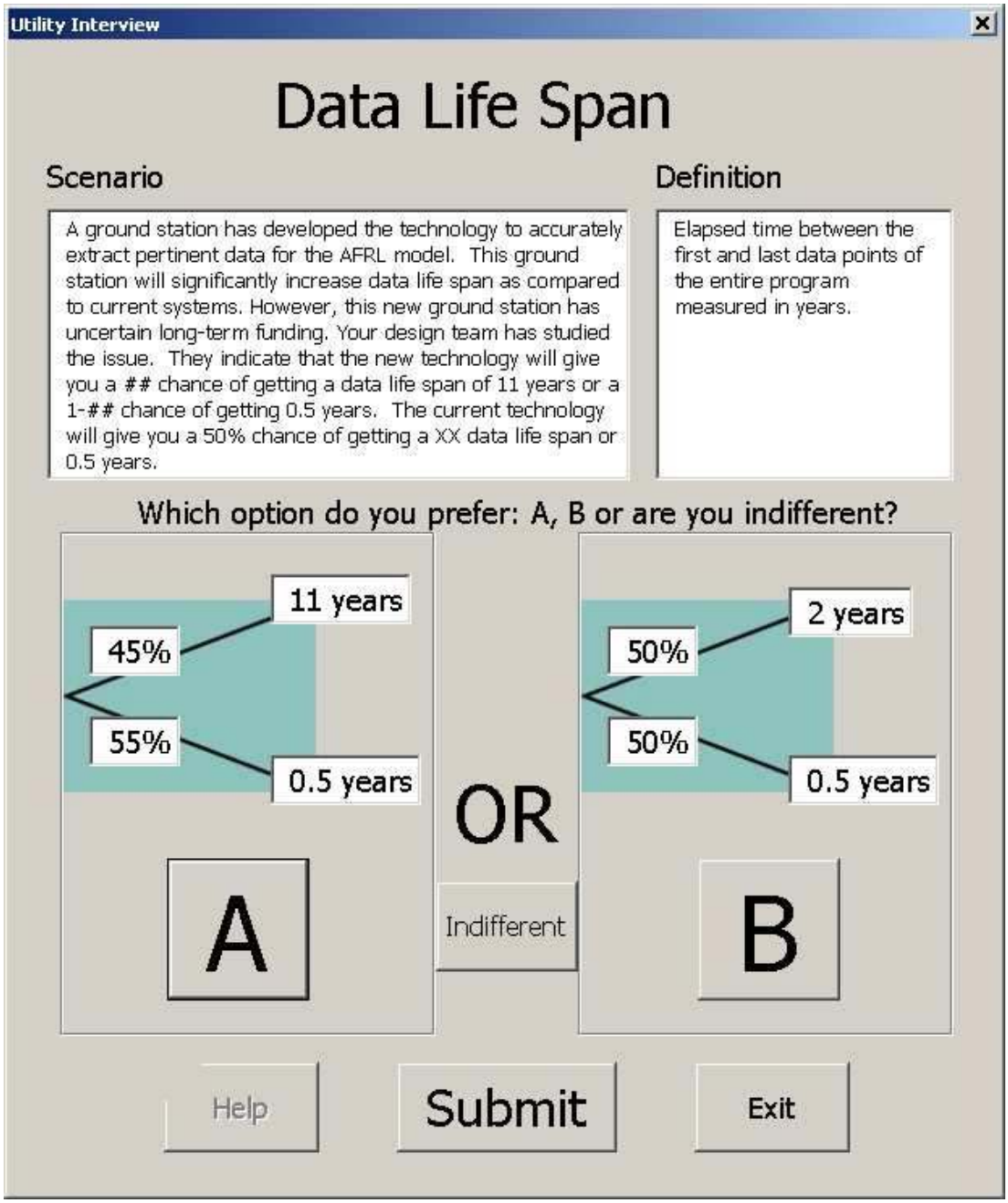



**Figure 6:** Single attribute interview form

The interview begins with a random value extracted from the list of values generated with the Attribute Property form. As described in an earlier section, these values represent increments along the attribute range at which indifference points are desired. Using this value, two situations are presented to the user. Each contains a certain chance of a favorable outcome, and a certain risk of an unfavorable outcome. Taken from multi-attribute utility theory, these questions are designed in the lottery equivalent manner [24]. One situation presents a 50% chance of obtaining the attribute value for which the indifference point is desired, and a 50% chance of obtaining the worst possible value for the attribute, as defined in the Attribute Property form. The other situation represents a certain probability *p%* that the best possible value will be obtained, and the probability *(100-p)%* that the worst possible values will be obtained. The value of *p* is varied based on the bracketing principle described above. Each time the user selects an option, the choice is recorded and the next probability in the bracketing sequence is displayed. This continues until an indifference point is reached – either by the user selecting the Indifferent button or by the system declaring an indifferent value. At this time, the indifference point is recorded, and the system displays the next random attribute value for which it desires an indifference point.

Once all of the indifference points are collected, the system uses the Utility Threshold collected by the Attribute Properties form to determine whether new indifference points need to be collected. If any two consecutive indifference points have a difference that is greater than the Utility Threshold, the system goes through the interview process for the midpoint value between the two indifference points. This

process is continued until no two consecutive indifference points have a difference in utility which is greater than the Utility Threshold, thus ensuring that the utility curve is captured to a desirable level of fidelity.

When this process is complete, the single attribute utility interview ends and a check mark is placed in the appropriate cell on the main page. The user is returned to the Attribute Navigator form, to conduct further interviews for remaining attributes, as appropriate. The following figure describes this process flow:

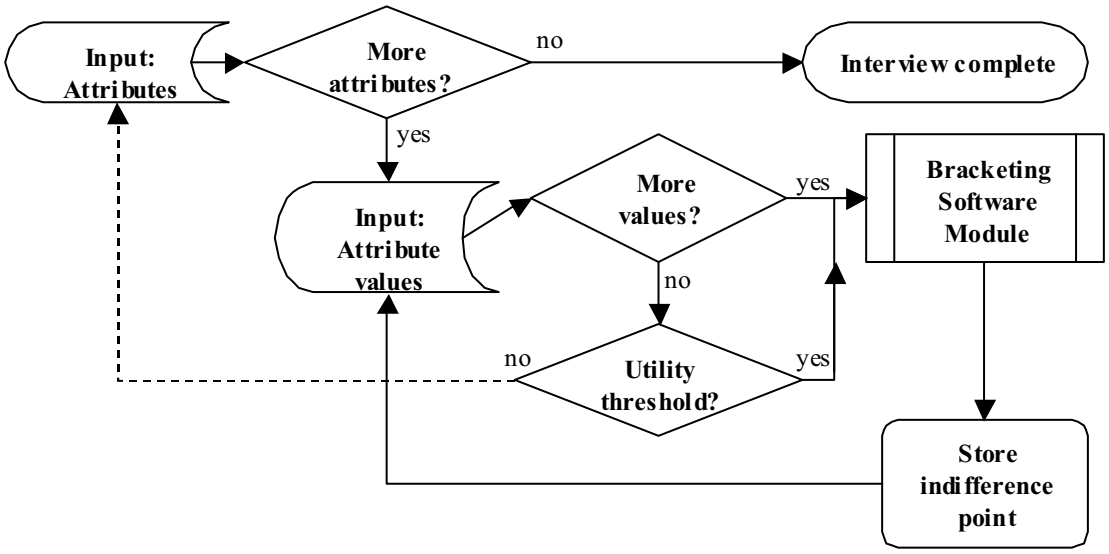



**Figure 7:** Process for single attribute interview

### 5.2.4 Corner Point Interview

The goal of the corner point interview is to determine the relative importance of each attribute with respect to the other attributes in the system by finding the "corner points" described in the MATE process [24]. This is done for each attribute by presenting the stakeholder with a situation similar to the one shown in the figure below. The user interface used for the corner point interview is similar to the interface for the

single attribute interview, except that individual values are replaced with lists that contain values for each attribute.

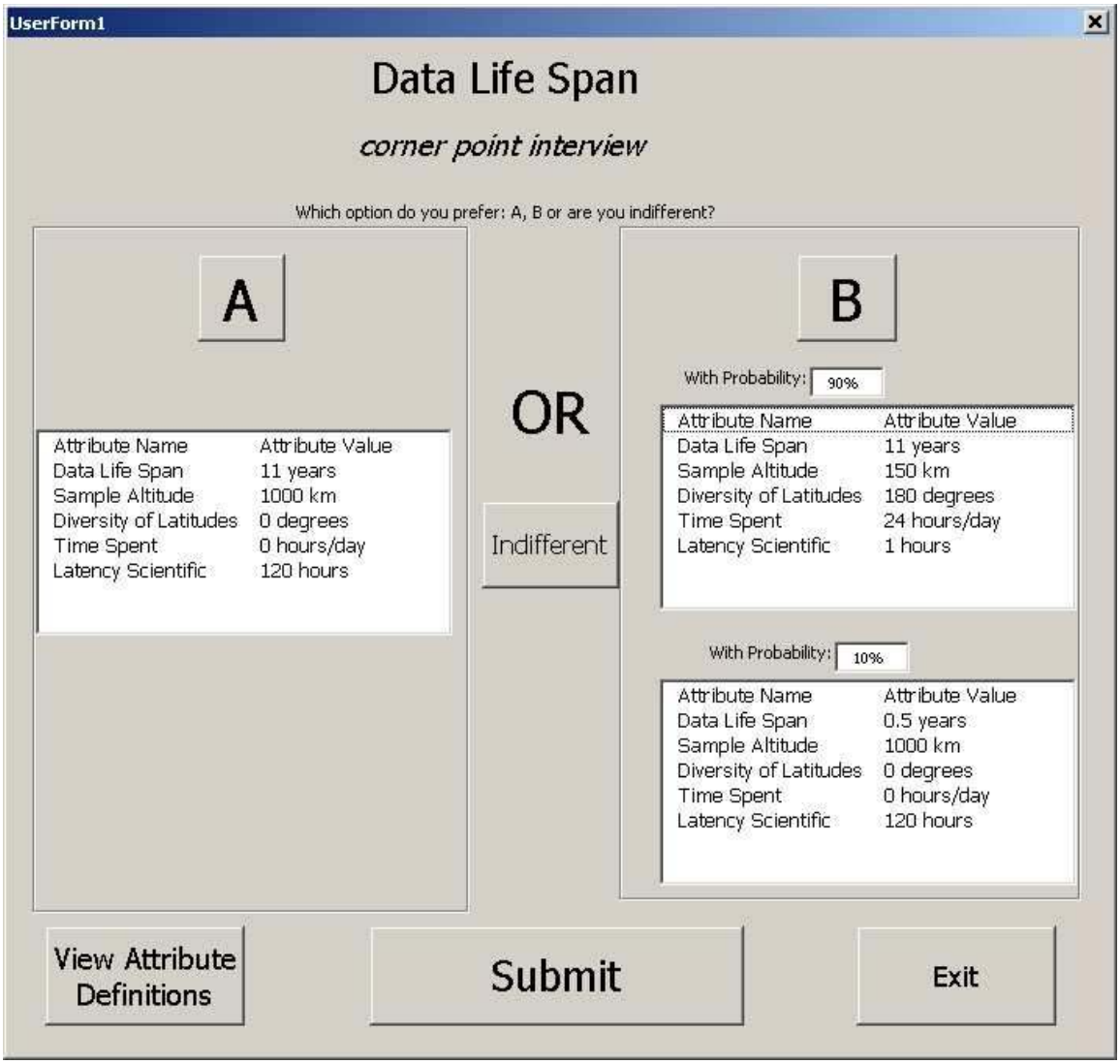



**Figure 8:** Multi-attribute interview form, used for corner point and random mix interviews.

Further, the certainty equivalent method is used, in place of the lottery equivalent method [24]. In the certainty equivalent method, the user has a choice between being certain of a particular outcome, or having a chance of another outcome. In the corner point interview, the two choices are as shown below:

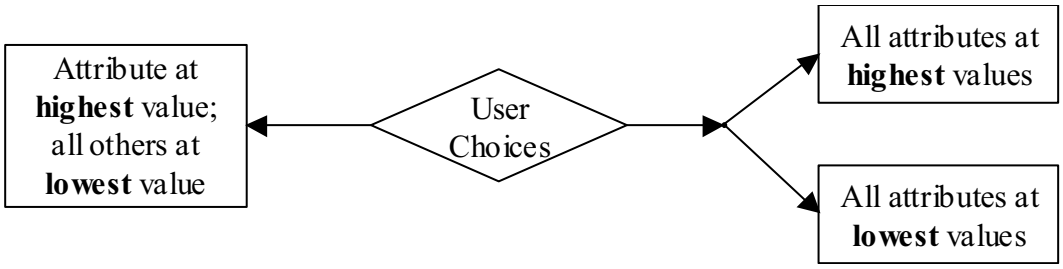



**Figure 9:** User choices in corner point interview

For each attribute, the above situations are created by populating the interview form's ListBoxes with the appropriate data from the attribute. Then, the bracketing module is used as described in the previous section, to find the percentage at which the stakeholder is indifferent between the two situations. At this point, the probability value, or "k-value", is stored with the attribute's parameter data. The "k-value" represents the relative weight, from 0 to 1, of the specific attribute being interviewed. A higher indifference point in a corner point interview means that the stakeholder was willing to give up a chance at a perfect system at a higher probability in exchange for the certainty of having a system where only the given attribute is perfect. The higher this probability, the more important the attribute, and thus the higher the k-value.

### 5.2.5 Attribute Independence Interview

After the data are collected, it is important to validate the selected attributes by ensuring that they are independent of each other. This is done by assessing and comparing the utility of two architectures, as described in the table below.

| | Architecture 1 | Architecture 2 |
|---|---|---|
| **Attribute** | Intermediate value from att. def. | Intermediate value from att. def. |
| **All Other Attributes** | Set to **highest** possible values | Set to **lowest** possible values |

**Figure 10:** Two architectures used to compare utility and determine independence

In each architecture, the value for the attribute being tested is left constant, at a value specified by the users in the attribute definition stage early in the process. The difference between the two architectures is that the remaining attributes are all set to the highest values in one, and the lowest values in the other. The stakeholder is asked to consider each system's utility with respect only to the attribute in question. If the attribute is truly independent, the utility of both architectures should be the same.

### 5.2.6 Random Mix Interview

The Random Mix module was developed to provide data to validate the utility functions developed by the MIST system. Users of the system can specify the number of random sets to generate. For each set, the system employs a random number generator which selects a value for each attribute along its range, as specified in the attribute definition stage. The resulting random sets, each representing a random architecture in the tradespace defined by the stakeholder and designers, are presented to the stakeholder as the culminating questions of the interview. Using the same interface as the corner point interview, the system uses the bracketing module to determine the utility of each of the random sets. These utilities are compared to the utilities generated by calculating the utility of each architecture with the utility functions. It is often the case that these utilities do not correlate, leading to the conclusion that humans cannot properly comprehend the

multi-dimensional problem of determining their own utility. Accordingly, a system such as MIST is invaluable in more accurately distilling the stakeholder's true utility values [24].

## *5.3 Output of MIST*

The MIST system produces a set of outputs, which can be used to facilitate the knowledge discovery and knowledge dissemination facets described earlier. The outputs described here are for one session of the MIST interview process. These results are provided in a generic format to the users, so that they can be imported into any third-party application for further processing. Work has begun on extending the MIST system to develop knowledge management tools as described in a later section.

### 5.3.1 Attributes and Interview Reports

Data in the MIST system are stored with respect to each attribute. As the interview is conducted, the responses are stored in the same spreadsheet which contains the attribute's properties as conceived in the attribute definition stage. Every choice is recorded; however, only the indifference points are used to calculate the utility functions and k-values. At any time, the user can choose to have all indifference points collected and stored in a separate table for analysis, as shown in Figure 11.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Attribute** | **Value** | **Utility** | | **Attribute** | **Corner Point** |
| 2 | Data Life Span | 0.5 | 0 | | Data Life Span | 0.1 |
| 3 | Data Life Span | 2 | 0.35 | | Sample Altitude | 0.425 |
| 4 | Data Life Span | 4 | 0.35 | | Diversity of Latitudes | 0.125 |
| 5 | Data Life Span | 6 | 0.65 | | Time Spent | 0.175 |
| 6 | Data Life Span | 8 | 0.75 | | Latency Scientific | 0.15 |
| 7 | Data Life Span | 10 | 0.95 | | | |
| 8 | Data Life Span | 11 | 1 | | | |
| 9 | Sample Altitude | 150 | 1 | | | |
| 10 | Sample Altitude | 300 | 0.55 | | | |
| 11 | Sample Altitude | 450 | 0.45 | | | |
| 12 | Sample Altitude | 575 | 0.35 | | | |
| 13 | Sample Altitude | 700 | 0.15 | | | |
| 14 | Sample Altitude | 850 | 0.05 | | | |
| 15 | Sample Altitude | 1000 | 0 | | | |
| 16 | Diversity of Latitudes | 0 | 0 | | | |
| 17 | Diversity of Latitudes | 30 | 0.45 | | | |
| 18 | Diversity of Latitudes | 60 | 0.65 | | | |
| 19 | Diversity of Latitudes | 90 | 0.65 | | | |
| 20 | Diversity of Latitudes | 120 | 0.85 | | | |
| 21 | Diversity of Latitudes | 150 | 0.85 | | | |
| 22 | Diversity of Latitudes | 180 | 1 | | | |
| 23 | Time Spent | 0 | 0 | | | |
| 24 | Time Spent | 2 | 0.05 | | | |
| 25 | Time Spent | 4 | 0.15 | | | |
| 26 | Time Spent | 8 | 0.35 | | | |
| 27 | Time Spent | 12 | 0.5 | | | |
| 28 | Time Spent | 16 | 0.6 | | | |
| 29 | Time Spent | 20 | 0.85 | | | |
| 30 | Time Spent | 24 | 1 | | | |
| 31 | Latency Scientific | 1 | 1 | | | |
| 32 | Latency Scientific | 20 | 0.85 | | | |
| 33 | Latency Scientific | 40 | 0.65 | | | |
| 34 | Latency Scientific | 60 | 0.55 | | | |
| 35 | Latency Scientific | 80 | 0.55 | | | |
| 36 | Latency Scientific | 100 | 0.15 | | | |
| 37 | Latency Scientific | 120 | 0 | | | |

**Figure 11:** Sample table of outputs for one interview session

### 5.3.2 Single Attribute Utility Functions

The core output for the MIST interview system is the utility functions for each attribute. As described in the MATE process, these utility functions describe the changing utility of an architecture as the value of the attribute changes. When the MIST user selects the Generate Reports button, they can generate a utility function for any attribute. The utility functions are represented as curves with each indifference point plotted and linear interpolation used to determine the function between indifference points. An example of such a curve is shown in figure 12.

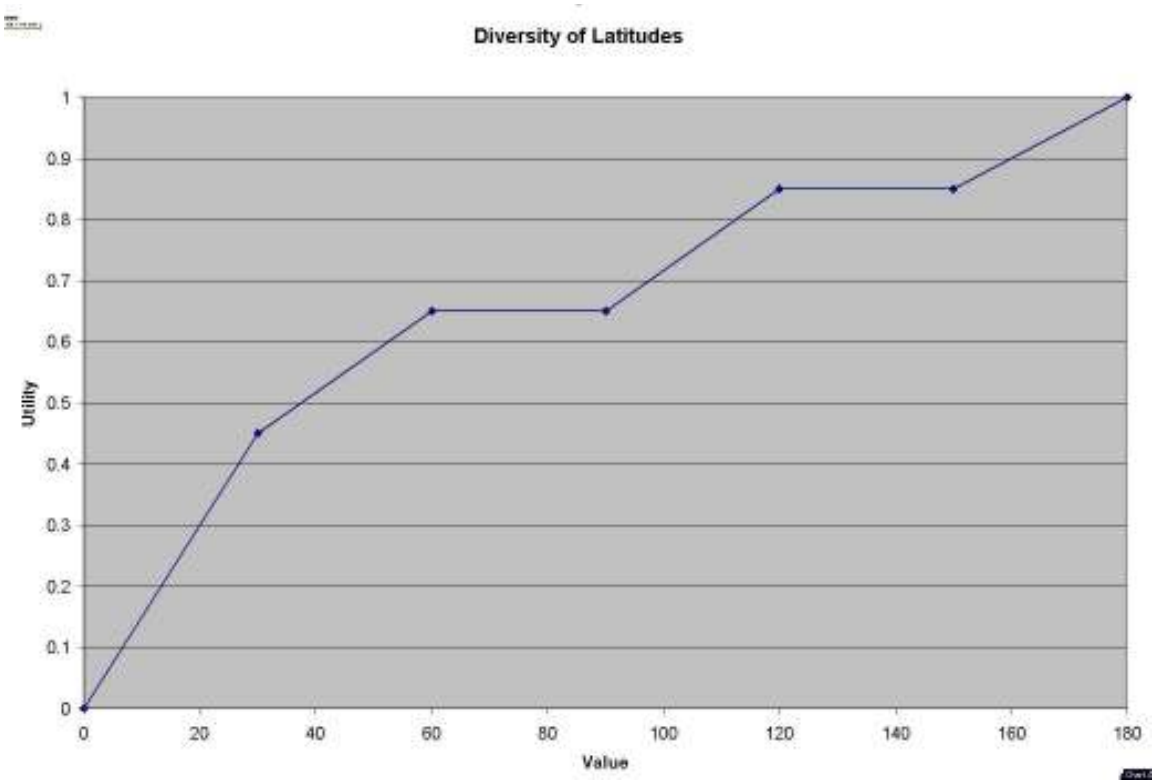



**Figure 12:** Sample utility curve for single attribute

### 5.3.3 Multi-Attribute Utility Function

The final output for the MIST system is the multi-attribute utility function, which calculates the utility of an architecture, given values for each of the attributes. The derivation of the actual function is described in the MATE process and shown in the equation below. In this equation, $k_i$ and $U_i(X_i)$ represent the k-value and utility function of a single attribute, and K is the aggregate scalar calculated as described in the MATE process [24].

$$KU(\underline{X})+1=\prod_{i=1}^{n=6}\left[Kk_iU_i(X_i)+1\right]$$

**Figure 13:** Multi-attribute utility function

In MIST, code modules carry out each step of the derivation on the fly, as shown in the flow chart below. Basically, once the value of each attribute is known, the utility of the system for each attribute can be determined by using linear interpolation between the two closest indifference points. Each attribute's utility is then multiplied by the k-value, and then a normalized product is produced; this represents the utility for the system with the given set of attribute values. This function can be used in conjunction with the knowledge management tools being developed for future versions, as described in a later section of this paper.

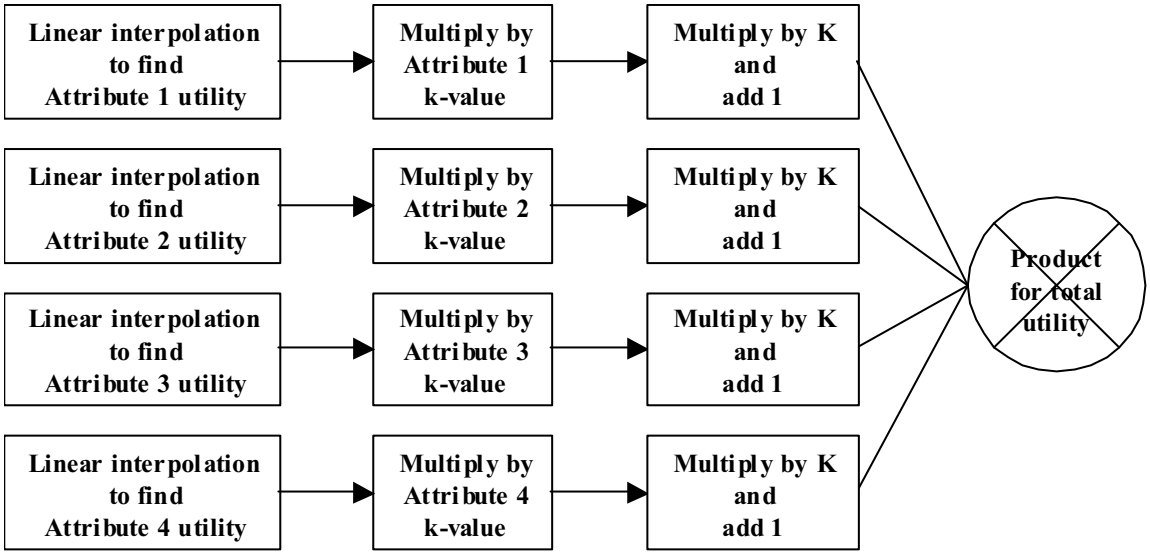



**Figure 14:** Multi-attribute utility calculation by software

### 5.3.4 Design rationale and history

One of the goals of the MIST system, especially taken in conjunction with the SSPARCy system, is to provide an efficient interface for capturing the history of decisions which are made along the design process. Many of the decisions made in defining the attributes and developing the interview process are crucial design decisions. The rationale captured for major design decisions by the Attribute Rationale module, as described above, is stored in a spreadsheet form very similar in format and structure to

the SSPARCy method of storing design history. This allows the SSPARCy suite of tools to be used to process the discussion of attribute definition, thus preserving and improving this important phase of knowledge exchange.

# 6 Extending MIST

The Multi-attribute Interview Software Tool described in this paper reflects the initial phase of an effort to build a complete and valuable knowledge-based tool to facilitate collaboration and informed design processes. This initial phase has built the core functionality to capture the utility of one stakeholder, at one point in time, and for one project. Each of these dimensions must be extended to enable this knowledge acquisition tool to transform into a source of knowledge management and discovery. As the element of time is incorporated into the system, it will be integrated with the SSPARCy system to provide a framework for relating utility and expense information to the design data and rationale of each project. Finally, for any of these software modules to be effective, the system must include a more user-friendly interface and help functionality for all users of the system. Work has begun in each of these areas and will be integrated into the next version of the system.

## *6.1 Network Mode*

As one of the main goals of the MIST system is to enable collaboration over geographic borders, functionality is being developed to allow the interview to take place over a network. This will reintroduce the possibility of real-time interaction into the interview, solving some of the issues described in the test implementation section, without requiring the users to travel.

The network mode has two main roles: observer and user. The user is the person directly interacting with the system, usually the stakeholder taking the interview. Observers are interested parties wishing to monitor and provide feedback on the

discussion surrounding the attribute definition and utility, such as other members of the stakeholder's team, designers, and engineers.

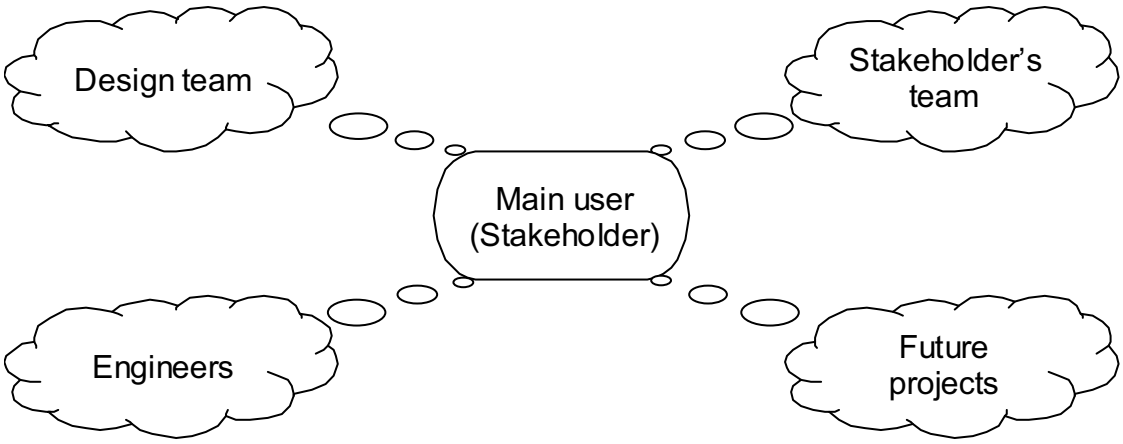



**Figure 15:** Potential roles for observers.

This mode is accomplished by building a library of possible actions which could take place during the various stages of the interview, and then using this as the basis for capturing the flow of the interview. The library includes a means of representation for commands such as opening a form, making a choice, clicking a particular button, etc. An entry in the library has the action, as well as the objects which were acted upon. When the user enables the observation mode, a text file is created and every action performed by the user is transcribed using a command from the library. This effectively creates a log file which can be used either in real-time or later in the process to observe the progress of the interview. The log file is made available in a secure manner to the other members of the team who wish to observe.

The observers run in a mode which does not allow user interaction, beyond being able to step forward and backwards through the interview. An observer selects the location for the log file, and then watches as the actions taken by the user are "played" on their local system. This allows observers to comment on the choices being made, and

provide advice on both the interview process and the utility decisions. Future versions will pursue the ability for multiple users to actively interact with the system, as well as integrated means of communication such as instant messaging, embedded within the MIST system.

## 6.2 Multiple Stakeholders

In order for the MIST system to become an integrated part of the concurrent engineering process, functionality must be developed to enable designers to examine the relationships and dependencies of each stakeholder's utility and expense values to the attributes of a system's design. This would also allow for the discovery of knowledge concerning the optimal attribute set that would maximize utility and expense for the entire stakeholder set.

The figure below demonstrates the main interface to the relationship analysis tool. This facilitates knowledge discovery in two ways. First, the tool can analyze and calculate utility and expense values for each stakeholder of a system design based on a given set of design attributes. This is done by using the utility equation as described in an earlier section. Secondly, by taking into account the utility and expense value of a stakeholder to a system design, the tool can generate and display all correlating utility and expense values for all stakeholders based on the attribute set that is under consideration.

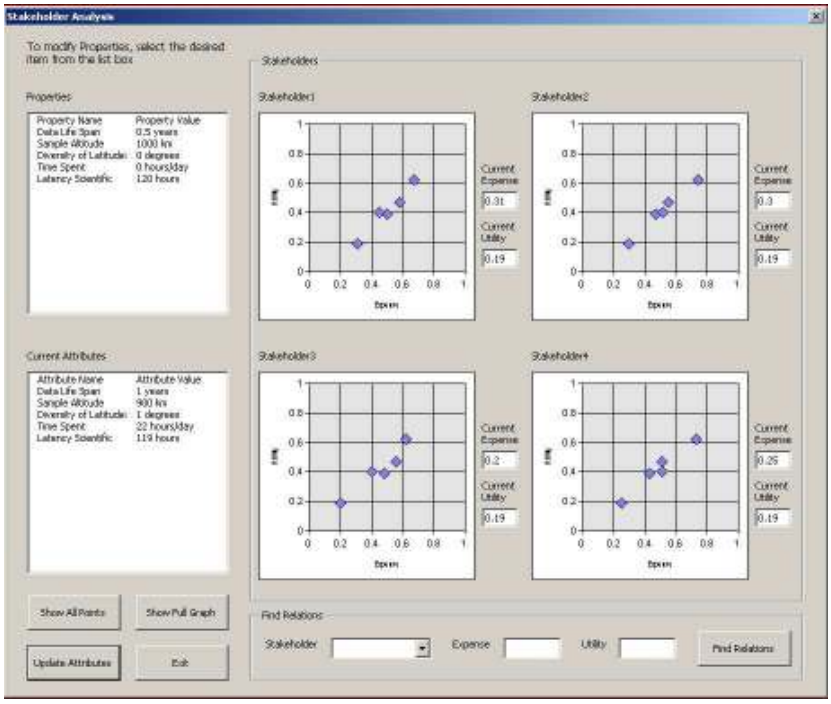



**Figure 16:** Interface for analyzing multiple stakeholders

With the relationship analysis tool, the effects of each design decision to system attributes can be recorded based on the changes to the utility and expense functions for all of the stakeholders in the system. Relationships between various roles in the design process are better explained through their utility and expense data, and future projects benefit from knowing how previous members in the same role operated under similar conditions.

Such a system simulates real-time feedback for designers in a concurrent process without having to consult every stakeholder with every design change. With a strong knowledge regarding stakeholder dependencies and relations, systems can be designed to optimize utility and expense values for all stakeholders involved.

## 6.3 Multiple Projects

Data from multiple spacecraft projects can be analyzed to develop models for future spacecraft project direction. Using data from the entire suite of tools, a complete set of design history, rationale and utility can be compiled for each project. An interface is being developed to process data from multiple projects and produce results which help determine similarities and differences. An initial version of the interface is shown below. Note that a user can select to compare either specific attributes, or entire projects. The utility curves for the attributes are compared using the least square method, which determines how closely the utility of one attribute resembles another. As a user discovers similarities to past projects, he or she can use the design history and lessons learned from similar design patterns to make their current design process more efficient.

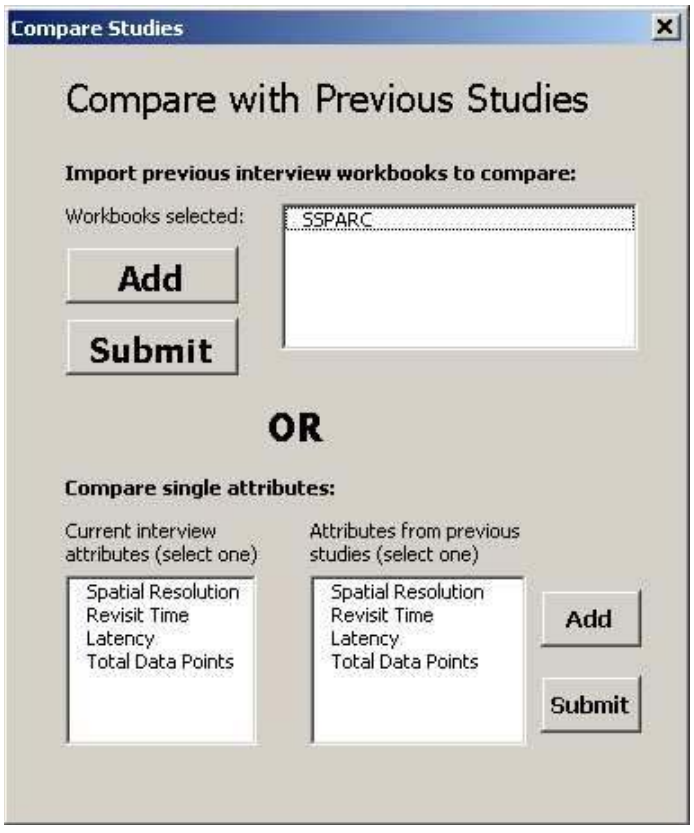



**Figure 17:** Interface for analyzing multiple projects

## 6.4 Integration with SSPARCy

Many of the extensions described to the system in this section will only truly be valuable with the integration of the SSPARCy and MIST systems. Physical integration would involve choosing one platform and designing both systems on that platform. The outputs of the integrated system will be a combination of both systems' current outputs, along with new, powerful outputs made possible. Design rationale can be linked directly to utility information from the stakeholders. The interviews could act as another means of separating the design process into phases for analysis. Emerging data mining techniques could be used to map attribute utilities to effects on system parameters.

At any given time, the following should be included as part of the state of the project: the k-value for each attribute, a utility function for each attribute, and an aggregate utility function. At any given time, the following should be calculated: exact utility for each attribute and the aggregate utility. This information will be made part of the SSPARCy data structure for design history. Design rationale input should allow the user to refer specifically to utility functions of specific attributes. For example, "we made this change to increase the utility of the X attribute." A link within the system would show you the utility function at that time, and how it has evolved. These initial steps provide the foundation for further research on how such innovative techniques can best facilitate the four faceted knowledge based structure described at the outset of this paper.

# 7 Technical Implementation

This section discusses the technical implementation of the design of the MIST system. The issues discussed here are not immediately relevant to the users of the system, although they directly impact the system's means of supporting the functionality described above. It is important to note that the interface described here is meant to provide an intelligent interface for the system to interact with various members of the design process. The operation of the system is different for designers conducting the interviews, and stakeholders who may be taking the interviews. The formalized structure for the knowledge was designed after careful study of the multi-attribute utility analysis process and the role of this analysis in the design process. The system is structured to provide a means of eliciting knowledge that the user may not even realize they have, and a means of forcing the user to think about each part of the design project in new terms. The back-end architecture of the system is designed to be flexible, so that interview questions can be formulated on the fly, and customized based on the user's previous answers. This architecture also allows the possibility of allowing multiple users to interact with the system concurrently, and provide feedback on each other's responses. Work has already begun on extending this customized interface to include intelligent feedback and guidance, so that users are better prepared to make the necessary decisions.

## *7.1 Platform*

This tool is implemented in Visual Basic and Microsoft Excel for a variety of reasons. Excel provides an interface familiar to most customers. For ease of the customer and designer, Visual Basic provides pre-existing GUI functionality in addition to familiar

data analysis tools, especially for presenting data in graphs and other visual formats. This allowed the IT team to focus primarily on developing more complete functionality of the system, instead of spending excessive time developing more complicated GUIs. A single software package will prevent the designer from wasting time transferring data from platform to platform.

The robustness of the Excel environment is an issue to be considered, but the more important issue was to quickly begin exploring the integration of utility analysis with pre-existing SSPARCy ideas as an initial framework for a complete design rationale capture tool. The functionality of SSPARCy consists primarily of multiple spreadsheets of design parameter values, and can be ported to Excel to make the integration seamless.

Data for this endeavor are stored in Microsoft Excel worksheets, which act as static arrays which are always viewable to the user. All interaction with the users of the system, and modification of data, is done through User Forms developed in Visual Basic. The worksheets and forms are described below.

## *7.2 Worksheets*

The four main types of worksheets used in the system are described here. For the initial implementation of the system, it was decided to leave these sheets viewable and editable to the users of the system, to provide an easy interface for making quick changes to the structure of the attributes or interviews without going through the user forms. This design decision was based on the need to deploy significant amounts of functionality in a short design timeframe, as well as the significant control we had over the initial users. In future versions, more effort will be spent on the interface functionality, to move the

worksheets to their own data file, and to implement security measures to ensure that data is only edited in the appropriate time and manner.

### 7.2.1 Main Interface

When the user opens the system, the main interface is displayed in a worksheet. The worksheet contains buttons organized in the same structure of the interview process, making it easily apparent what the steps to be taken are. Additional buttons are included for the features of the system being developed to extend the capabilities of MIST, as described above.

This worksheet also contains a table for every attribute and interview type. Microsoft Excel allows designers to designate a global variable to represent a range of cells anywhere in the system. The variable "Attribute_List" was used to designate the cells containing the names of the attributes in the main worksheet. When an attribute is added to the system, the size of this range is increased by one, and the name is included in this list. When an attribute is deleted, the opposite process is taken. This range of cells is also used in the Attribute Navigator form described below, to generate the list of current attributes.

The table in the top left of the main worksheet contains a cell which represents each interview for each attribute. When an interview is conducted, a check mark is placed in the appropriate cell, so that the user is always aware of the progress of the system. Responses can be deleted by using the button near this table. When a given set of interview responses is designated to be deleted, the values from the attribute's worksheet are deleted as well as the check mark from the main sheet.

Finally, the main worksheet also contains the values generated by the Random Mix interview. These are available here both because they do not relate to one specific attribute in particular, and so that users of the system have an easy means of editing the random values to be used in the interview. The Random Mix interview form reads the values directly from this worksheet, and stores the indifference point for a set of values in the cell above each set.

### 7.2.2 Attribute State

When an attribute is still active in the system, the current states of the properties of the attribute are stored in a worksheet. At all times, the user can know what the current values are. In the leftmost columns, each of the fields in the Attribute Properties form is stored in the appropriately labeled cell. The "Attribute Options" column represents the list of values for which the single attribute interview will seek indifference points for this specific attribute. This list is generated by the system every time the properties of the attribute are saved, either by adding or modifying the attribute. It is done by splitting the attribute range, specified by the minimum and maximum values, into the number of equal segments specified by the number of indifference points, either linearly or logarithmically. This list is then randomized and stored in the worksheet.

Along with the properties of the attribute, the worksheets are also used to store all responses given in interview sessions with respect to this specific attribute. These worksheets help to establish a complete history of the decisions made during the interview process. The history is used by the system to determine which interviews have been conducted, and also by the designers.

Although it is not designed to be modified by the users of the system, the values in the attribute worksheet can be modified as a backup method for resolving issues that may come up during the interview process. For example, if the attribute options generated are not round numbers, the designer might prefer to edit the values to provide the stakeholder with more reasonable questions during the interview process. Also, during the interview, if the stakeholder wishes to go backwards and revise his or her answer, the worksheets provide the means for editing or deleting responses. As the system evolves further, it will become less necessary to access the attribute worksheets.

### 7.2.3 Attribute History

Each attribute has a corresponding Attribute History worksheet that is created and modified along with the attribute. Since this worksheet can never be deleted, it provides a complete history of the entire attribute definition period. Each column of this worksheet contains a list of the properties of the attribute. It also contains a cell for the user-defined rationale for the specific iteration of attribute property values. These values are only stored when the attribute is created, deleted, or the user specifies that a change made to the attribute was a major change deserving cataloging.

### 7.2.4 Reports

Each attribute will also have a worksheet that shows a visual depiction of the utility function for that attribute. This is generated from the single attribute interview responses stored in the attribute's worksheet. The graph uses the Microsoft Excel "xlXYScatterLines" Chart Type [26] which plots each indifference point, and then performs a linear interpolation between each pair of adjacent points.

## 7.3 Forms

All user interaction with the system is handled through the use of Visual Basic User Forms [25]. In the Visual Basic environment, only one user form can be active at any given time [25]. A user form is made up of a set of controls, such as text boxes, command buttons, and list boxes. Thus, the state of the system is represented by the active user form, and the values of the controls on that user form. Please refer to the Appendix for figures depicting the actual forms. Figure 18 demonstrates the main types of forms which exist in the MIST system.

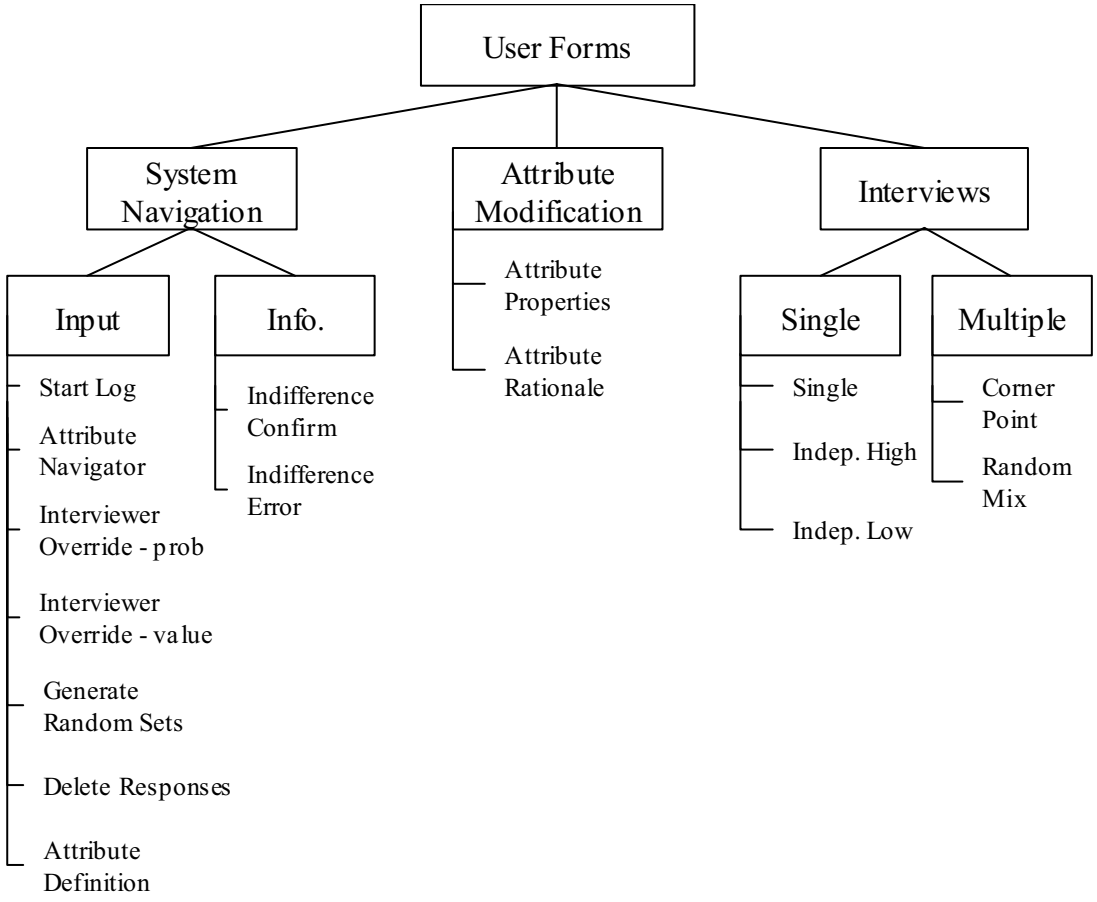



**Figure 18:** Different types of user forms

Note that the core of the system consists of an interview type being selected, then an attribute, and then the appropriate user form being called and populated with the relevant values, taken from the attribute current state worksheets.

### 7.3.1 Start Log

*Purpose*: open file for logging all user actions

*Called by*: button from main worksheet

*Inputs*: none

*State when called*: before process starts, to specify a text file to store all actions

*Relevant user input controls*: main text box used to specify filename

*Conditions*: filename is not accepted if it already exists

*Result*: filename stored as global variable; every module writes actions to specified file in formalized structure as shown in the Appendix

### 7.3.2 Attribute Navigator

*Purpose*: select attribute for modification, interview, response deletion or report generation

*Called by*: relevant buttons on main worksheet

*Inputs*: type of operation from button; attribute list from main worksheet

*State when called*: user wishes to perform operation listed in Purpose

*Relevant user input controls*: list box to select attribute; check box to select Interviewer Override Mode; button to confirm and continue

*Conditions*: system will not continue if specified interview for specified attribute was already conducted as specified in main worksheet

*Result*: if report generation is operation, chart worksheet created with graph of single attribute indifference points; otherwise, relevant user form made active, with attribute as input

### 7.3.3 Attribute Properties

*Purpose*: add, modify or delete an attribute

*Called by*: Attribute Navigator

*Inputs*: attribute name from Attribute Navigator; attribute properties from attribute's current state worksheet

*State when called*: attribute conception discussion occurring between designers and stakeholder

*Relevant user input controls*: various text boxes, radio buttons and drop down boxes to set properties

*Conditions*: properties must adhere to type specified in data structure as described in earlier section

*Result*: if new attribute, new worksheets created for current state and history; attribute properties stored in current state worksheet; if deleted attribute, current state worksheet deleted; if new, deleted or major change, Attribute Rationale form made active – when system returns, rationale added to attribute properties, and stored as new column in attribute history worksheet; Attribute Navigator made active

### 7.3.4 Attribute Rationale

*Purpose*: elicit rationale for major changes made to attributes

*Called by*: Attribute Properties

*Inputs*: attribute name

*State when called*: attribute properties entered, about to save state

*Relevant user input controls*: text box to enter rationale

Conditions: none

*Result*: rationale left in Attribute Rationale text box; Attribute Properties made active, immediately acquires rationale from Attribute Rationale text box

### 7.3.5 Interviewer Override – Value

*Purpose*: allows interviewer to override the attribute value for which an indifference point is being sought

*Called by*: Attribute Navigator or Single Attribute Interview

*Inputs*: attribute name, array of all values already interviewed, current value to be interviewed

*State when called*: about to begin single attribute interview for a particular value

*Relevant user input controls*: text box allows user to edit value, submit command button to confirm

*Conditions*: new value must fall within attribute range, must not have already been interviewed; Interviewer Override box must be checked in Attribute Navigator

*Result*: Single Attribute Interview form uses value from this form

### 7.3.6 Interviewer Override – Value

*Purpose*: allows interviewer to override the probability for which an interview question is being asked

*Called by*: Attribute Navigator or Single Attribute Interview or Interviewer Override - Value

*Inputs*: attribute name, current low, high bounds and probability value for range of uncertainty from bracketing module

*State when called*: about to ask single attribute interview question for a particular probability

*Relevant user input controls*: text box allows user to edit probability, submit command button to confirm

*Conditions*: new probability must fall within range of uncertainty, must not have already been interviewed; Interviewer Override box must be checked in Attribute Navigator

*Result*: Single Attribute Interview form uses probability from this form

### 7.3.7 Single Attribute Interview

*Purpose*: conduct single attribute or independence interviews

*Called by*: Attribute Navigator if beginning interview or Single Attribute Interview if continuing interview

*Inputs*: type of interview and attribute from Attribute Navigator; attribute scenario, definition, maximum and minimum points from attribute current state worksheet; current probability from bracketing module; current value from attribute current state worksheet or interview override module

*State when called*: in midst of either single attribute or independence interview, attempting to learn indifference point for certain value

*Relevant user input controls*: user selects A, B, or Indifferent command button based on preference; Submit button confirms choice

*Conditions*: if data exists from interview already, interview begins where it left off.

*Result*: user choice stored in attribute worksheet and returned to bracketing module; if interview completed, Attribute Navigator displayed; if interview override mode, override form displayed; otherwise, Single Attribute Interview displayed again with next probability or value from bracketing module

### 7.3.8 Multiple Attribute Interview

*Purpose*: conduct corner point or random mix interview

*Called by*: Attribute Navigator if beginning interview, or Multiple Attribute Interview if continuing interview

*Inputs*: type of interview from button on main worksheet; if corner point interview, attribute name from Attribute Navigator, attribute values from all attribute current state worksheets; if random mix interview, attribute values from main worksheet

*State when called*: in midst of either corner point or random mix interview, attempting to learn indifference point for certain value

*Relevant user input controls*: user selects A, B, or Indifferent command button based on preference; Submit button confirms choice

*Conditions*: if data exists from interview, interview begins where it stopped

*Result*: for corner point interview, user choice stored in attribute worksheet and if interview completed, Attribute Navigator displayed, otherwise return to bracketing module; for random mix, user choice stored in main worksheet if indifference point and next set of values displayed

### 7.3.9 Attribute Definition

*Purpose*: display definition of a particular attribute during multiple attribute interviews

*Called by*: Multiple Attribute Interview

*Inputs*: none

*State when called*: in midst of either corner point or random mix interview

*Relevant user input controls*: user selects attribute from list box

Conditions: none

*Result*: definition for attribute displayed; system returns to Multiple Attribute Interview form

### 7.3.10 Indifferent Confirm

*Purpose*: confirm indifference point

*Called by*: Single or Multiple Attribute Interview forms

*Inputs*: indifference point, either from user selecting "Indifferent" command button, or bracketing module generating indifference point

*State when called*: for given value or set of values, indifference point between two scenarios is apparently reached

*Relevant user input controls*: user may confirm or reject their indifference point

*Conditions*: none

*Result*: if confirmed, indifference point stored in relevant worksheet, system returns to interview form; if rejected, data deleted for given value or set of values, interview conducted again

### 7.3.11 Indifferent Error

*Purpose*: alert user to inconsistent responses, as determined by bracketing module

*Called by*: Single or Multiple Attribute Interview forms

*Inputs*: none

*State when called*: bracketing module determines that range of possible indifference points is null

*Relevant user input controls*: none

*Conditions*: none

*Result*: response data deleted for current value, interview conducted again.

### 7.3.12 Generate Random Sets

*Purpose*: generate sets of attribute values for random mix questions

*Called by*: button on main worksheet.

*Inputs*: none

*State when called*: before Random Mix Interview conducted

*Relevant user input controls*: user types number in text box; Submit button confirms choice

*Conditions*: integer must be entered

*Result*: random sets generated using attribute values taken from the attribute options in each respective worksheet; each set displayed on main worksheet

### 7.3.13 Delete Responses

*Purpose*: delete data from interview responses for particular interview

*Called by*: Attribute Navigator

*Inputs*: attribute selected by user from Attribute Navigator

*State when called*: user wishes to delete responses for a particular interview

*Relevant user input controls*: list box to select which interview to delete responses for given attribute

Conditions: none

*Result*: all data deleted from attribute current state worksheet which was generated by the selected interview; Delete Responses form displayed again.

## *7.4 Selected code modules*

While some of the code operation is intuitively obvious from the functionality description given in previous sections, particular modules are highlighted here which are not as immediately apparent.

### 7.4.1 Formatting Values

In the Attribute Properties form, a field specifies the "Format" for an attribute, and whether to "Include Units" in the description. For each attribute, at any time a value for that attribute is displayed as part of an interview, the numerical value is converted to the format prescribed by the Attribute Properties. This is done by casting the numerical value as text, and then conducting a text search and replace for the "#" symbol with the attribute value. Finally, if the "Include Units" box is checked, the text describing the units is concatenated to the end of the string. This string then is placed within the appropriate text box in the interview form. For example, if the Format for a given attribute were "#x#", the Units were "degrees" and the Include Units box was checked, then the formatted value for the integer 60 would be the text string "60x60 degrees". The following code represents this process.

### 7.4.2 Generating Attribute Values

When the Attribute Properties form is submitted to the system, values for the attribute are generated on the fly and stored in the current state worksheet for the attribute. The decision was made to allow the user to select either a linear or logarithmic scale for attribute values. The code for this operation takes in the minimum and maximum values of the attribute range, as well as the number of indifference points desired and the type of scale. If the scale is linear, the range is simply divided into equal segments. If the scale is logarithmic, the range is divided into segments with increasing orders of magnitude. For example, a range of 1 to 10,000 with three indifference points would have 10, 100 and 1000 as the indifference points. Each indifference point is stored in its own row in the attribute options column of the attribute current state worksheet. The following code represents this process.

### 7.4.3 Bracketing

The bracketing software module operates as described in the preceding sections. Since various user forms need the values relevant to the bracketing process, the decision was made to store the bracketing values in global variables. Thus, variables were created to represent the current indifference range and previous choice. This allows a set of Case statements and If-Then statements to navigate the decision tree depicted in an earlier section and constantly update the range and choice, as well as calculate new values as appropriate. If an indifference point is generated by the bracketing module (when the indifference range size drops below the probability resolution) the point is treated as a response and stored in the worksheet accordingly. Note that given the nature of this

design, only one interview may be conducted at any given time. This design limitation may be expanded in a future version of the system.

### 7.4.4 Saving State

The data storage model for the system requires that every response given to an interview is stored in the relevant attribute's current state worksheet. Rather than retain responses in memory for a particular interview's set of questions, the system writes each response to the worksheet immediately after it is given. This method of storing responses also acts as the means of representation for the state of the interview system. If an interview is not completed, and the user chooses to exit the interview form for any reason, the state is stored in the responses. When the interview form is re-opened, it will not ask questions related to a value for which an indifference point already exists in the attribute's current state worksheet. In effect, this allows the interview to "pick up where it left off" without having to store the state in any formalized or explicit manner.

### 7.4.5 Independence Interview Scenarios

The essence of the independence interviews is to ensure that the utility of each attribute is independent of the values of the other attributes. This is done by creating two scenarios for each attribute – one where every other attribute is set to its best value, and one where each attribute is at its worst. Since the single attribute interview form is used for these interviews, these scenarios are represented by replacing the text in the definition box with these values. The code builds a text string by sequentially adding the appropriate formatted values to the end for each attribute, and then displaying this string in the definition text box of the interview form.

### 7.4.6 Utility threshold

As alluded to in previous sections, at the end of every single attribute interview, the system checks to ensure that there is not a significant jump in utility between two indifference points, as determined by a threshold set in the Attribute Properties. The first step in this process is extracting the indifference points from the other interview responses. Then, the indifference points and their associated utilities are sorted by the attribute values at each point. Then, for each pair of adjacent points, if the difference between utility is above the threshold, a new single attribute interview is done for the attribute at the value exactly in between the two adjacent points. For example, if the utility of a 3 year mission lifetime is .2, and the utility of a 5 year mission lifetime is .8, and the utility threshold for mission lifetime is .5, then the system would attempt to find an indifference point for a 4 year mission lifetime.

# 8 Test Deployment

## *8.1 Project Description*

The approach described in this thesis was tested in the Spring of 2002 as part of the SSPARC consortium's X-TOS project. In an attempt to better understand and innovate space system design, the SSPARC team has been conducting a series of design exercises intended to simulate the development of a terrestrial observer swarm (TOS). The X-TOS project was conducted in conjunction with the MIT graduate course 16.89, "Space Systems Engineering", taught by Professors Hastings, Warmkessel and McManus [28]. In previous years, members of the design team assigned to assess the utility of the proposed system conducted a manual face-to-face interview with the customer, in the format described by the MATE approach.

By using the software system, the gathering of utility data was done in a much faster and more efficient manner. The role of the scientific customer in the X-TOS project was played by Kevin Ray, a member of the Air Force Research Laboratory at Hanscom Air Force Base. The knowledge acquisition process was much smoother, as the interview took under an hour, as opposed to the six hour manual process of previous projects. By having the data and questions displayed in a visual manner, it was much easier for the Air Force user to translate the notions of utility contained within his mind to the concrete data structure of the MATE approach [29].

The software system also allowed this interview to be conducted without the need to have all the interacting parties be present in the same geographical location. In the B-TOS project, Kevin Ray had to spend the day traveling to MIT to meet with the utility

team [24]. This required not only a commitment from him, but the time of every member of the team, who had to be present for the interview.

The interview was conducted in stages, as it fit in with the user's personal schedule, and communication between the user and engineers during the interview was made through the telephone and electronic mail. The Air Force user was also able to take time to consult with other members of the customer team, since he was working on his own schedule and his own location. While this may seem to be a rather small effect, this is an important functionality of the system because it allowed the acquisition of utility information to be collected in a much more fluid manner. In previous projects, the burden of conducting the interview led to the utility information only being collected from one stakeholder, and in a piecemeal manner. With the introduction of the software approach described in this paper, the utility data are being collected with a higher fidelity, with more stakeholders, and at frequent intervals throughout the design process.

This implementation has demonstrated the power of this approach to facilitate the other areas of the knowledge-based framework. Increasing the frequency of utility data will lead to a higher amount of knowledge discovery as the team can get concrete feedback as the design progresses. This approach has allowed the SSPARC team to conduct better knowledge management, because the utility information became a continuous driver for the design process rather than a one-time input into the system.

## *8.2 Issues Encountered*

As the initial version of this system was tested, a few issues were encountered, which provided valuable feedback in making the system useful and easy to use. Many of these issues dealt with the intuitive nature of the software and interview process. As this

system employs a manner of communication which is unusual and new to the users, making the process as intuitive as possible is a significant issue. While Kevin Ray suggested after using the system that it was very intuitive and made the interview process much more understandable, the experience still provided our team with both immediate and long-term issues to address. Some of these changes were incorporated immediately, as they were necessary for the implementation to continue, and some are discussed and will be pursued in future versions of this system.

### 8.2.1 Attribute Definition Stage

The largest issue this implementation revealed was that the attribute definition process is a much more important factor in the success of this system than we had assumed while designing the software. The knowledge exchange that takes place as the attributes are being defined would benefit from being incorporated into the knowledge-based framework of the entire system. Each iteration in the attribute definition process came as a result of a meeting with either members of the utility team, the entire design team, or the stakeholder being interviewed.

The software was designed to deal with this phase by using the Attribute Properties form described above. This form contains input boxes for each of the relevant properties of an attribute, and each time a modification is made to the attribute, it prompts the user to express the rationale behind the modification. While this allows a certain amount of the history to be retained, the implementation demonstrated that future versions of this system should have more fields to capture knowledge that will be more relevant to the discussion process that leads to the attribute definition.

First, the system must be better equipped to deal with the names of the attributes. By using the actual name of the attribute in the name of the worksheet containing all attribute properties, and thus having all references in the code depend on the attribute name, the system was ill-equipped to handle a change in the attribute name. Any change made to the name was recognized by the system as a new attribute being created, rather than an existing attribute being modified. Furthermore, since Microsoft Excel limits the number of characters in the name of a worksheet, the users of the system were constrained in how many characters they could use in the name of the attributes [26]. The next version of the system will treat the attribute name as another property of the system, thus removing this impediment.

Another key architecture change that was motivated by this implementation is the integration of multiple roles into the attribute definition process. The X-TOS design team was split into logical teams which each dealt with a specific module of the system [28]. One of these teams was the utility team, which was responsible for defining the attributes and developing the utility functions that would be used to explore the tradespace of options. This utility team took in input from each of the other teams, as well as from Kevin Ray and John Ballentin, the scientific users being interviewed for the utility functions. Each of these roles had very distinct interests within the system, and including a notion of which role had suggested or influenced a certain modification to the attributes would make the knowledge repository being collected much more valuable.

This issue will become increasingly important in future implementations when the design team is geographically dispersed. One of the major benefits of this system is that it provides a structure to knowledge that is otherwise shared in an informal and

unstructured manner. By using the Attribute Properties form of the MIST system, designers in different locations who do not have the informal contact can express their thoughts on the current state of the attributes by using this form. By including the role as part of this form, it becomes possible for a utility team to propose a set of attributes, along with a rationale for each, send this to designers of each module, gather input in the same form, and have the system act as an interface with which to negotiate changes. Obviously, this would not completely replace the need for meetings and discussions to occur, but it would make such meetings shorter and more efficient.

Finally, the attribute definition stage of the X-TOS project demonstrated the potential value of having information from previous projects available for information during the current project. At many times during the discussions between the design team and the scientific user, attributes from the B-TOS project were mentioned and decisions were made based on the successes and failures of the B-TOS experience [29]. However, with only a few of the members of the B-TOS project still working on the X-TOS system, input from the previous project was limited and focused mainly on the final decisions made in the B-TOS report. This observation made two things clear. First, the value of the knowledge repository being created along the attribute definition process is significant. But second, it demonstrated that the architecture of the system must be extended to allow attributes to link to other attributes from past projects. Just as it is valuable for the design rationale of the project to link directly to the utility function which drove a particular decision, it would make the attribute history much more powerful to have a direct link to past attributes. This is especially relevant because much of the discussion in this stage was conducted one attribute at a time. Thus, it would be

useful for designers to be able to notice that a past attribute was similar, then trace back the development of that attribute to attributes in past projects, and utilize the knowledge from as many previous projects as possible. This type of information is not traditionally catalogued, and would be very tedious and challenging without the framework provided by the MIST system.

Another useful property for the attribute form would be a feedback section for both the stakeholder and engineers to report on the actual value of the attribute, after the project is completed. This would involve both a freeform input as well as data captured by the system such as the k-value and perhaps the number of times the attribute was referred to in the design rationale of the system. Such information, produced automatically by the system, would allow future users to have a sense of how successful a certain choice of attributes was to a certain project.

### 8.2.2 Bracketing Intuitively

The method of bracketing used when acquiring the stakeholder's indifference point was changed during the implementation after the feedback of the X-TOS team. As described above, the goal of every interview question is to obtain the point at which the stakeholder is indifferent between the two options presented. The questions are framed in a manner such that the probability being varied is in the range from 0% to 50%, with the assumption that a probability of greater than 50% would render an obvious choice by the stakeholder (see above sections for the complete description of the interview logic). Developing an algorithm to find the indifference point seemed like a relatively easy task when this system was first designed. When it was deployed to the X-TOS project, the system simply asked about the midpoint of the range of probabilities with which it was

uncertain of the indifference point. For example, when the first question was asked, the system asked about the probability of 25%, since it was the midpoint between 0% and 50%. Depending on the response, the system would then ask about the midpoint between 0% and 25% or between 25% and 50%, and so forth. As described above, the system would round every probability to the resolution indicated in the attribute properties.

When the system was presented to the X-TOS team, the initial feedback suggested that this method of bracketing would not be most effective to the stakeholder. A key assumption behind the above-described approach was that the stakeholder knew where their indifference point would be. However, if this were the case, the system should just ask the user for the indifference point, and not bother with the interview. Adam Ross, one of the principle developers of the MATE process, suggested that part of the purpose of the interview was to help the stakeholder discover this indifference point, which existed within the stakeholder's mind but had not been formalized.

An immediate solution was designed to allow the X-TOS project to continue using the system. It was assumed that the probability resolution would be 5% for all attributes – in other words, no matter what the attribute, it was assumed that the stakeholder could not distinguish between two probabilities which were less than 5% apart. The sequence of probabilities for each question was also set to have the bracketing would take place in a slower manner, allowing the user to have time to understand the issues related to the specific attribute being asked about. The decision tree which was used by the X-TOS implementation to decide which probabilities would be chosen is shown in Figure 5. Every interview started near the top of the range, with 45%, and then moved to the low point in the range, with 10%. The bracketing continued in this manner,

alternating between high and low points, except when the stakeholder choose the option closer to the boundary of the range, at which point the system reverted to the normal method of selecting the midpoint of the range. The effect of this method was that it allowed the stakeholder to answer "easier" questions first – for example, the choice between a 45% chance of the dream value and a 50% chance of a mid-range value is relatively simple. This way, the stakeholder could spend the time while answering the easier questions developing a sense for what their own concerns were with the given attribute.

While the above solution enabled the X-TOS project to use the system, future versions must incorporate a more intelligent and automated method for bracketing. If the probability resolution changes to a value other than 5%, the above method would not work. Perhaps the most robust method would be for the system to create the probability sequence instantly, within the Attribute Properties form, and allow the interview designer to edit it to his or her own preference.

### 8.2.3 Interviewer Override Mode

Another immediate change necessitated by the X-TOS implementation was the incorporation of an Interviewer Override Mode. This was an essential element in being able to deploy the system to the Air Force user, rather than having the interview conducted by members of the utility team. The initial version of the system displayed both the probability and the attribute value for confirmation to the user. An interviewer using this system could then alter the values, to control the questions that were being asked. Members of the X-TOS team decided that even though it was possible for the user to simply click the interview button at each of these steps, it still provided too much

information to the user about the state of the system and the motivations behind the interview. To resolve this issue, an option was added to the Attribute Navigator interface which allows the user to declare whether they wish to run in the Interview Override Mode and have the ability to alter or add attribute values or probabilities to the interview process. In a future version which has the observation mode available, it will be beneficial to the system to have the ability to run one client in this override mode while the interviewee runs in the normal mode. This way, an interviewer in another location can decide on the fly whether they wish to acquire more values, or different values, based on the responses of the user.

### 8.2.4 Multi-mode Attributes

The X-TOS utility team decided to model the utility of the latency attribute in the context of two different types of missions: a scientific one, and a technology demonstration one [28]. This required the team to develop two different utility functions for latency and two k-values for each attribute in the system, one for each type of mission. Since the system was unable to handle this, the Air Force user was presented with two versions of the MIST software, one for the scientific mission and one for the technology demonstration mission. The attributes in each system were the same, with the only difference being the properties of the latency attribute. The user was asked to go through the entire interview system once for each mission.

This experienced demonstrated a characteristic of the knowledge our system is trying to capture which future versions must be able to handle. In characterizing an engineering project by attributes, it is important to recognize that the context in which the utility of the attributes is being described may depend on the environment in which the

project is being applied. For our system to most effectively represent this knowledge, a method must be designed to include the mode of application in the knowledge structure. This way, users can specify if an attribute's properties will change based on the mode, be able to enter multiple sets of properties based on the mode, and conduct the interviews in the context of each mode.

### 8.2.5 Observation

As described in the preceding sections, the system is prepared to handle very basic needs for observation. The X-TOS implementation demonstrated that this functionality is vital to being able to have this system enable collaboration as effectively as possible. At a few times during the day, the Air Force user contacted the utility team and the MIST designers via telephone with questions regarding the state of the interview. At each of these points, it would have been useful for the team members providing assistance to have been able to "play" the responses of the user on their own computer, and understand very quickly what point in the interview the user was struggling with. Early in the X-TOS interview, the Air Force user was not aware that he was supposed to go through the interview for more than one indifference point for each attribute. This simple error took a significant amount of time to realize, because the communication between the user and the MIST team was not clear. In the end, the mistake was realized only after the Air Force user sent the data to the MIST team, as if he had finished the interview, and the team was able to realize that he had not in fact conducted the entire interview.

### 8.2.6 Corner Point Concept

In the first set of results from the X-TOS interview, the k-values for every attribute except for Altitude were 0.05 [30]. If this were an accurate set of results, it would have indicated that none of the attributes were providing any real utility to the system except for the Altitude. In discussing this with the Air Force user, it was determined that the minimum value of the Altitude attribute was so low that under no circumstance would he chose to take a risk of having that be the value of the attribute. After explaining to the user that the motivation behind the corner point interviews was to consider the attributes at conceptually high and low values, it became clear that his answers were very tied to the actual values displayed, and not the concepts. This type of misunderstanding demonstrates the need for some sort of visual representation to be included in the multi-attribute interview forms, since most of the values in these forms are either the highest or lowest values of the attributes.

Also, the software definitely needs some sort of "Help" functionality to explain the motivation behind each interview, as will be discussed in a future section. Developing such functionality is a challenging task, because the goal of the system is to obtain preferences from the user that they are unaware of beforehand. If an interview such as the corner point interview is explained in great detail, the risk exists that the user would simply interpret the corner point interview as a request to weight the attribute against the other attributes. With such an interpretation, the interview process becomes meaningless as the customer would attempt to make a distinction which they would probably not be prepared to make. Therefore, any attempt at explaining the motivations behind an interview should carefully consider the implications of providing the user with too much independence in deciding what knowledge their responses provide.

### 8.2.7 Interview Scheduling

Without the interview scheduling functionality fully implemented, the X-TOS team had to send the Air Force user a complete set of detailed instructions to help him navigate the system. This method actually proved to not cause any significant problems, as the Air Force user was able to conduct the interviews in the order prescribed. However, if this system were to be used in a widespread manner, with many stakeholders taking the interview at many different times, it would be much more efficient to avoid the burden of sending detailed instructions to each user. One thing to note in observing the X-TOS implementation was that the user took advantage of the ability to stop and start the interview as he pleased. This functionality must be incorporated into any scheduling mechanism that is implemented in the system.

### 8.2.8 Indifference Point Attribute Values

A couple of issues were raised in the X-TOS implementation concerning the number of indifference points collected and the values for these points. First, although the system allows the designer to decide on the number of indifference points which will be collected, prior to the X-TOS implementation it simply split the attribute range into the number of equal segments specified by the designer. For example, if the minimum value of an attribute was 0 and the maximum was 100, and the user decided to collect 4 indifference points, the system would conduct the single-attribute interview for the values of 20, 40, 60 and 80. However, the possibility existed of a designer defining an attribute which has a logarithmic scale instead of a linear scale. The Attribute Properties form thus now allows for the possibility of both types of scales, so an attribute with a range from 0 to 10000 can collect indifference points at 10, 100 and 1000.

The other modification to the Attribute Properties form which this issue necessitates is the creation of the indifference values as the designer is defining the form, so that they can edit the values before they are saved to the interview script. Before the system was sent to the Air Force user, the utility team went through the worksheet for each attribute and changed the attribute values to round numbers, to make the interview more understandable to the user. This type of intelligent rounding can be done on the fly, with the system deciding whether to round the values to a multiple of 5, 10, 50, etc. This way, every time the number of indifference points is changed, the designers don't have to edit the individual worksheets containing the data.

Finally, the major change related to this issue developed in the X-TOS implementation was the addition of a probability threshold, which allowed the system to decide by itself if it needed to collect more indifference points. The utility team wanted to have the ability to collect more indifference points if an interview yielded a utility curve which had a major jump in utility between adjacent indifference points [31]. Since they would be deploying the system, the threshold was added to provide a bar above which the system would continue asking for indifference points.

### 8.2.9 Independence Interview Interface

Although the implementation and results of the independence interview went as planned, the actual results suggest that some reconsideration is needed with the independence interview interface. The indifference points selected for each of the attributes were identical for the high and low valued interviews. This would suggest that the attributes were selected perfectly, and are indeed independent. However, given the past experiences in the B-TOS project, it is unlikely that these responses truly represented

the user's preferences. Instead, it is likely that the ordering of the interviews, having the user take both independence interviews immediately after each other, may have played a role in the user giving identical responses. Furthermore, choosing to use the single attribute interface instead of the multi-attribute interface, might have placed too little emphasis on the changing values of the other attributes, and caused the interviewer to implicitly consider the attribute independently of the others. These issues must be taken into consideration when redesigning the independence interview.

## *8.3 Lessons Learned*

The issues raised above all suggest one very important theme: the effectiveness of knowledge acquisition is heavily dependent on the nature of the software interface. In designing such a system, it is important to consider the type of knowledge being sought, the form in which it exists prior to using the system, and the form in which it is to be disseminated. Much of the knowledge being acquired in this system exists on multiple levels. If users of the system are asked the questions our system is trying to answer, such as what the relative importance of an attribute, they might give a certain set of answers. However, when the interview questions are framed in the certainty equivalent method employed by this system, the user may give answers which are closer to their true preferences [24]. Effectively designing software that can negotiate the boundary between these two levels is the goal of this system. The X-TOS implementation helped demonstrate that this goal is not as simple as it may seem. Future versions of this software must ensure that it creates an environment that not only enables communication across locations but elicits knowledge which the users may not even be aware of.

# 9 Conclusion

In comparing the MIST system to related tools in the area of knowledge-based design management, the core contribution of MIST is a formalized means of integrating the input of the stakeholders of a project into the design process. Approaches described in section 3 represent a broad survey of the areas in which knowledge-based engineering design tools are being built. Figure 19 describes some examples of the specific means in which the MIST system improves upon the related approaches described earlier.

| **Approach** | **Relevant Contribution** | **MIST improvement** |
|---|---|---|
| 3.1 Vehicles | Environment for describing and analyzing systems | Utility used as means of assessing systems before, during and after |
| 3.2 Boeing's KSD | Expert system for interface with subsystem software modules | Attribute structure categorizes module inputs and outputs |
| 3.3 Rule-based algorithms | Transfer customer needs to specifications with rules | Avoids point design, allows tradespace exploration |
| 3.4 AIDA | Case-based reasoning to determine initial architecture from past cases | Current utility can be mapped to previous cases to assess in present |
| 3.5 SPOOL | Reverse engineering to infer rationale | Knowing each stakeholders utility aids in inferring design rationale |
| 3.6 C-DeSS | Geometric design description | Attributes provide broad structure for geometry |
| 3.7 DICE | Component-based design with negotiation mechanisms | Utility information removes need for constant input in negotiations |
| 3.8 Utility Evaluator | Utility interview tool with preference consistency, risk | Data from all stakeholders analyzed concurrently |
| 3.9 ICAD | Product design knowledge used to automate design and configuration | Automated design more tailored to all stakeholders' needs |
| 3.10 NASA's VSDE | Collaborative modification of system components | Incorporates geographically dispersed stakeholders in process |
| 3.11 NIST | Formal representations of design knowledge, specific language | Formal representation for utility and project attributes |
| 3.12 QuestMap and DRAMA | Devise structure for design rationale | Utility functions act as numerical means of expressing rationale |
| 3.12 DRIM | Specific model for design rationale | More flexible structure for attribute rationale input |
| 3.12 Tsinghua Design Reuse | Intelligent mechanisms to use design rationale for design reuse | Ability to map current utility to previous rationale - better solution |

**Figure 19**: Improvements offered by MIST system to related works

With the base functionality developed, the next step is to incorporate "intelligent" functionality to process the knowledge captured by each approach. The ultimate goal is to combine approaches into a single, powerful framework for knowledge-based engineering design. As the extensions described in section 6 are developed to provide the functional basis for pursuing these tasks, research must be done on effective methods for incorporating emerging methods into this process.

A major motivation for pursuing this goal is the staged implementation of data mining technology to analyze data from multiple perspectives. One of the major knowledge discovery efforts will be to analyze data from multiple design projects, to gain better insights into how the ultimate design is related to the set of spatially and temporally distributed inputs from the stakeholders. This would help to formulate the strategy for having the system provide “automated” baseline designs that might best satisfy a stakeholder's preferences based on past projects and past successes and failures. One aspect of this capability is to manage data over multiple stages throughout the design process. As requirements and situations change throughout the design process, it is important to capture the changing goals and desires of the stakeholders, and examine these data to discover trends in how these changes happen and how the project is affected by these changes.

Another issue to be addressed by data mining techniques is to compare the preferences of similar stakeholders. Relationships between the various roles in the design process will be better explained through the preference data, and future projects will benefit from knowing how previous members in the same role operated under similar conditions.

When employing these approaches in a virtual design world, another factor is the need for multimedia forms of input to ease communications over geographic boundaries. A text-to-speech engine in the interview system would make the interview process easier for those not familiar with the computer interface, as well as provide a human voice to a discussion which is usually carried out with humans. The other component of this is voice recognition - once again, this would allow the users to communicate with the system in a much more human manner. The functionality has already been developed to allow users from multiple locations to view the status of an interview in real-time. By integrating real-time communication aspects such as chat sessions, audio and video conferencing, the interview session could involve a large number of locations. It would then be feasible to involve more stakeholders in the process, because the need for geographical proximity is reduced.

Design environments such as the spacecraft environment are dependant on the successful interaction of multiple teams and multiple stakeholders based in various locations. Capturing the needs of every stakeholder in the process, as well as the details of the major decisions and rationale, provides a means of enabling knowledge transfer between teams without requiring intense interaction. Such a knowledge-based framework can have a tremendous impact on the design process, by providing designers with information that saves time, money and effort. The model presented in this thesis provides a means for innovating the spacecraft design process by truly exploiting the underlying knowledge within a traditional design environment.

# 10 References

*MIST software demonstration is available on-line upon request to author.*


[1] Smith, Patrick L., et. al. "Concurrent Design at Aerospace". *Crosslink,* Vol. 2: No. 1. Aerospace Corporation, 2001.

[2] Gillam, April and Stephen Presley. "A Paradigm Shift in Conceptual Design." *The International Journal for Manufacturing Science and Production*, Vol. 3: Nos. 2-4, 2000.

[3] McManus, H. and Warmkessel, J. "Creating Advanced Architectures for Space Systems: Product and Process". Presentation to SSPARC group, August 2001.

[4] Gupta, Amar. "A Four-Faceted Knowledge-Based Approach for Surmounting Borders". *Journal of Knowledge Management*, Vol. 5: No. 4, December 2001.

[5] Scott, Quincy R. "SSPARCy: A Software Integration Support and Design Rationale Capture System." Master's thesis, MIT, July 11, 2001.

[6] Ross, Adam. "MATE: Defining Roles, Stakes, and Decision Makers". Presentation to SSPARC group, November 20, 2001

[7] Gillam, April. "Vehicles Knowledge-Based Design Environment". *Journal of Spacecraft and Rockets*, Vol. 30: No. 3, pp 342-347. May-June 1993.

[8] Chalfan, M. K. "A Knowledge System that Integrates Heterogeneous Software for a Design Application". *The AI Magazine*, pp 80-84, Summer 1986.

[9] Chiu-Chi Wei, Ping-Hung Liu and Chie-Bein Chen, "An automated system for product specification and design". *Assembly Automation*: Vol. 20: No. 3, pp 225-232. MCB University Press, 2000.

[10] Rentema D.W.E., Jansen F.W., Torenbeek, E., "The application of AI and geometric modeling techniques in conceptual aircraft design". *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pg. 928-935. St.Louis, September 1998.

[11] Keller, Rudolf K. et. al. "Pattern-Based Reverse-Engineering of Design Components". *Proceedings of ICSE'99*, Los Angeles. IEEE, April 1999.

[12] Klein, Mark. "Capturing Design Rationale in Concurrent Engineering Teams". *IEEE Computer*, pp. 39-47. IEEE Computer Society, January 1993.

[13] Sriram, Ram D. *Distributed and Integrated Collaborative Engineering Environment*. Book to be published, 2002.

[14] Aseltine, J. "WAVE: An Incremental Algorithm for Information Extraction". *Proceedings of the AAAI 1999 Workshop on Machine Learning for Information Extraction*. 1999.

[15] Wan Jie and Krishnamurty, S., "Comparison based Decision Making in Engineering Design," *1999 ASME Design Theory and Methodology*. Las Vegas, 1999.

[16] Katragadda, Prasanna. "Knowledge-Based Systems Interoperability Workshop 9". Concentra Inc.

[17] Mapar, Jalal, et. al. "NASA Goddard Space Flight Center Virtual System Design Environment". *2001 IEEE Aerospace Conference Proceedings*. Vol. 7, Piscataway, NJ. IEEE, 2001.

[18] Szykman, Simon, et. al. "Design Repositories: Engineering Design's New Knowledge Base". *IEEE Intelligent Systems*, 2001.

[19] Sriram, Ram D. “Standards for the Collaborative Design Enterprise Response to Object Management Group”. http://www.omg.org/homepages/mfg/mfgppe.htm

[20] QuestMap v3.12. *The Soft Bicycle Company*, 2000.

[21] Brice, A. and Johns, B. “Improving Process Design by Improving the Design Process.” *A DRAMA white paper*. QuantiSci, Oct. 1998.

[22] Pena-Mora, F., Sriram, R., and Logcher, R. “Conflict Mitigation System for Collaborative Engineering.” *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*. pp. 101-124, 1995.

[23] Wang-Xin and Xiong-Guangleng. “Supporting Design Reuse Based on Integrated Design Rationale.” *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace*. Vol. 5. IEEE, Piscataway, NJ, 2001.

[24] Diller, Nathan, et. al. "B-TOS: Terrestrial Observer Swarm." Final Report, 16.89 Space Systems Engineering, MIT. Spring 2001.

[25] Microsoft Visual Basic 6.0. Help Manual. Microsoft Corporation, 1999

[26] Microsoft Excel 2000. Help Manual. Microsoft Corporation, 1999.

[27] Deleque, Phillippe. Master’s Thesis, Department of Civil Engineering and Technology and Policy, MIT, 1986.

[28] “X-TOS Final Report”. Final Report, 16.89 Space Systems Engineering, MIT. Spring 2002..

[29] Ray, K. Interviews. Spring 2002.

[30] Diller, N. Interviews. Fall 2001 and Spring 2002.

[31] Ross, A. Interviews. Fall 2001 and Spring 2002.

[32] McManus, H. Interviews. Fall 2001.

[33] Warmkessel, J. Interviews. Fall 2001.

[34] Jackson, Daniel. "Lecture 12: Object Models." 6.170 Laboratory in Software Engineering, MIT. March 1, 1999.

[35] Chang, W. Advanced Undergraduate Project, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[36] Reyes, F. Advanced Undergraduate Project, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[37] Ng, C. Advanced Undergraduate Project, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[38] Chang, W. Contributed code to MIST. Undergraduate Researcher, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[39] Ng, C. Contributed code to MIST. Undergraduate Researcher, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[40] Reyes, F. Contributed code to MIST. Undergraduate Researcher, Department of Electrical Engineering and Computer Science, MIT, Spring 2002.

[41] Ross, A. Contributed code to MIST. Graduate Student, Technology and Policy Program, MIT, Summer 2001.

[42] Sainath, T. Contributed code to MIST. Undergraduate Researcher, Department of Electrical Engineering and Computer Science, MIT, Winter/Spring 2002.

[43] Konfisakhar, A. Examined related research. Undergraduate Researcher, Sloan School of Management, MIT, Winter/Spring 2002.

# 11 Appendices

The following appendices contain the Visual Basic code, along with screen shots, of the system described in this paper. An on-line version of this software is available by making a request to the author. A couple of the modules in the system were done as Advanced Undergraduate Projects for the department of electrical engineering and computer science at MIT. These represent the only self-contained single-author bodies of work, and are indicated as such in the appropriate section. The rest of the system was designed, implemented and tested by different members of the team at different stages of the implementation process.

## *11.1 Forms*

### 11.1.1 Attribute Navigator


```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttNavOld
```

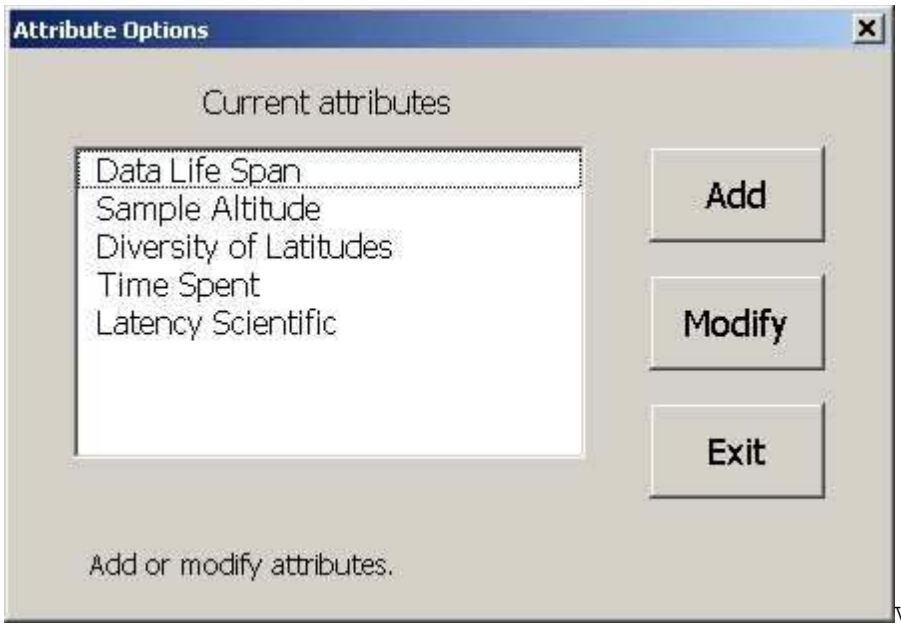
```
   Caption         =   "Attributes"
   ClientHeight    =   5340
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   4815
   OleObjectBlob   =   "AttNavOld.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttNavOld"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub CommandButtonInterview_Click() 'interview
    Dim att_choice As Integer

    att_choice = ListBoxAttributes.ListIndex
    If att_choice < 0 Then ListBoxAttributes.ListIndex = 0
    AttNavOld.Hide

    Select Case LabelInterview.Caption
    Case "single attribute interview"
        UtilInt.LabelAttribute.Caption = ListBoxAttributes.Value
        Single_Interview_Start (ListBoxAttributes.Value)
    Case "corner point interview"
        UtilIntII.LabelAttribute.Caption = ListBoxAttributes.Value
        Util_InterviewIIA_Start (ListBoxAttributes.Value)
    Case "independance high interview"
        UtilInt.LabelAttribute.Caption = ListBoxAttributes.Value
        Independance_Interview_High_Start (ListBoxAttributes.Value)
    Case "independance low interview"
        UtilInt.LabelAttribute.Caption = ListBoxAttributes.Value
        Independance_Interview_Low_Start (ListBoxAttributes.Value)
    Case "curve generation"
        CreateSingleAttributeCurve (ListBoxAttributes.Value)
    Case "response deletion"
        AttValuesDelete.LabelAttribute.Caption = ListBoxAttributes.Value
        AttValuesDelete.show
    Case "definition"
        AttDefinition.Caption = "Definition for " + ListBoxAttributes.Value
        AttDefinition.Label1.Caption = ListBoxAttributes.Value
        Dim Q_attribute_prop As AttributeProp
        Q_attribute_prop = Retrieve_selection(ListBoxAttributes.Value)
        AttDefinition.TextBoxDefinition.Text = Q_attribute_prop.Definition
        AttDefinition.show
    End Select

End Sub

Private Sub CommandButtonOptions_Click() 'options --> old code for remnant
button.
    AttNavOld.Hide
    AttSec.TextBoxPassword.Text = ""
    AttSec.show
    AttSec.TextBoxPassword.SetFocus
```

```
End Sub

Private Sub CommandButtonExit_Click() 'exit
    AttNavOld.Hide
End Sub

Private Sub ListBox1_Click()

End Sub

Private Sub Label1_Click()

End Sub
```

### 11.1.2 Attribute Definition


```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttDefinition
   Caption         =   "Definition for "
   ClientHeight    =   4260
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4710
   OleObjectBlob   =   "AttDefinition.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttDefinition"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub CommandButton1_Click()
    AttDefinition.Hide
End Sub

Private Sub Label1_Click()
End Sub
```

### 11.1.3 Attribute Options



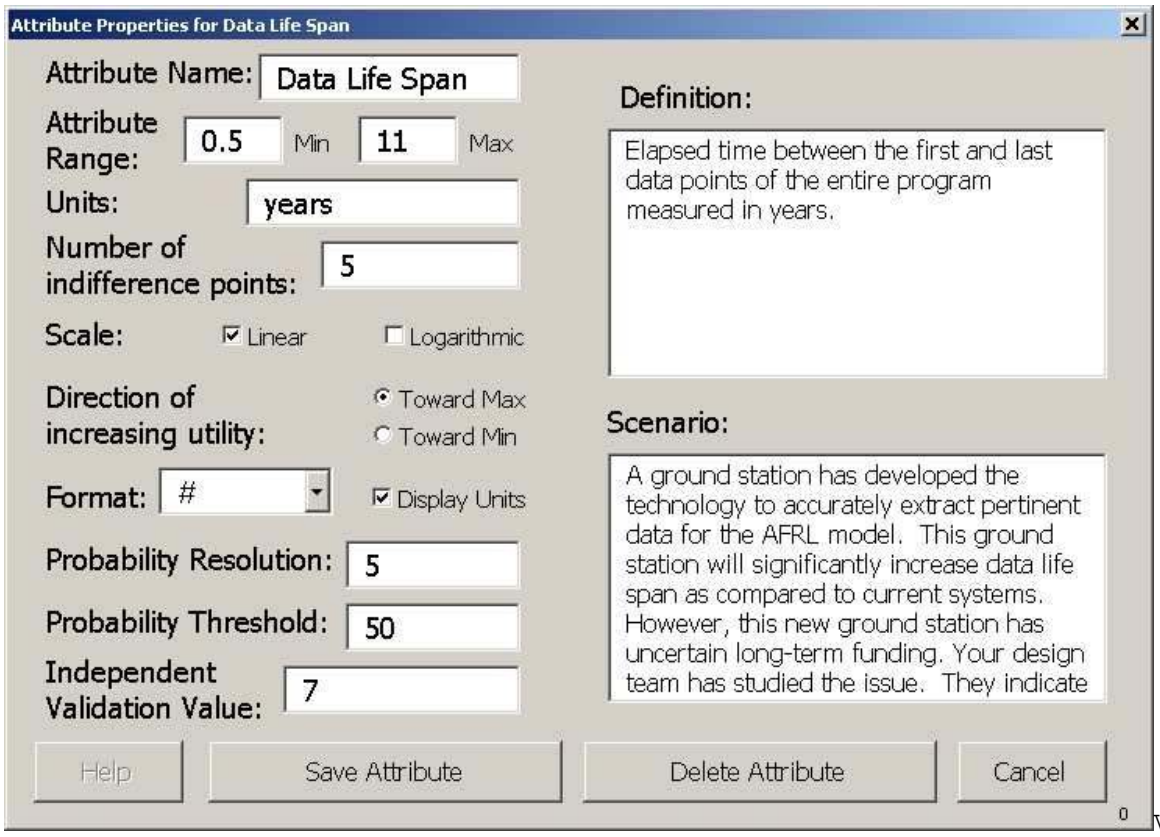

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttOpt
   Caption         =   "Attribute Options"
   ClientHeight    =   4245
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   6600
   OleObjectBlob   =   "AttOpt.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttOpt"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub CommandButtonAdd_Click() 'add
    AttOpt.Hide
    AttProp.TextBoxName.Text = ""
    AttProp.TextBoxMin.Text = ""
    AttProp.TextBoxMax.Text = ""
    AttProp.TextBoxUnits.Text = ""
    AttProp.TextBoxSteps.Text = ""
    AttProp.OptionButtonMin.Value = False
    AttProp.OptionButtonMax.Value = False
    'AttProp.ComboBoxFormat.Text = ""
    AttProp_Load
    AttProp.CheckBoxUnitsInc.Value = False
    AttProp.CommandButtonDelete.Visible = False
    AttProp.Label8.Caption = 1
    AttProp.TextBoxName.SetFocus
    AttProp.show
End Sub
```

```
Private Sub CommandButtonExit_Click()
    AttOpt.Hide
End Sub

Private Sub CommandButtonModify_Click() 'modify
    Dim selection As String
    Dim att_list As Range
    Dim attribute_param As AttributeProp

    AttOpt.Hide
    AttProp.CommandButtonDelete.Visible = True
    AttProp.Label8.Caption = 0
    selection = AttOpt.ListBoxAttributes.Value

    attribute_param = Retrieve_selection(selection)

    AttProp.TextBoxName.Text = attribute_param.Name
    AttProp.TextBoxMin.Text = attribute_param.min
    AttProp.TextBoxMax.Text = attribute_param.max
    AttProp.TextBoxUnits.Text = attribute_param.Units
    AttProp.TextBoxSteps.Text = attribute_param.Increment
    AttProp.TextBoxScenario.Text = attribute_param.Scenario
    AttProp.TextBoxResolution.Text = attribute_param.Resolution
    AttProp.ComboBoxFormat.Value = attribute_param.Format    'WDC, added for
format and units
    AttProp.CheckBoxUnitsInc.Value = attribute_param.UnitsInc 'WDC, added for
format and units
    If attribute_param.LinearScale Then
        AttProp.CheckBoxLinear.Value = True
        AttProp.CheckBoxLogarithmic.Value = False
    Else
        AttProp.CheckBoxLinear.Value = False
        AttProp.CheckBoxLogarithmic.Value = True
    End If

    If attribute_param.Direction Then
        AttProp.OptionButtonMax.Value = True
    Else
        AttProp.OptionButtonMin.Value = True
    End If
    AttProp.TextBoxIndependent.Text = attribute_param.Independent
    AttProp.TextBoxDefinition.Text = attribute_param.Definition
    AttProp.TextBoxThreshold.Text = attribute_param.Threshold

    AttProp_Load
    AttProp.show
End Sub
Sub AttProp_Load()
    AttProp.ComboBoxFormat.AddItem "#"
    AttProp.ComboBoxFormat.AddItem "#x#"
    AttProp.ComboBoxFormat.AddItem "Other"
End Sub
Private Sub CommandButton3_Click() 'exit
    Refresh_all
    AttOpt.Hide
    AttNav.ListBox1.ListIndex = 0
```

```
    AttNav.show
End Sub

VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttOptOld
   Caption         =   "Attribute Options"
   ClientHeight    =   4245
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   6600
   OleObjectBlob   =   "AttOptOld.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttOptOld"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub CommandButtonAdd_Click() 'add
    AttOpt.Hide
    AttProp.TextBoxName.Text = ""
    AttProp.TextBoxMin.Text = ""
    AttProp.TextBoxMax.Text = ""
    AttProp.TextBoxUnits.Text = ""
    AttProp.TextBoxSteps.Text = ""
    AttProp.OptionButtonMin.Value = False
    AttProp.OptionButtonMax.Value = False
    AttProp.CommandButtonDelete.Visible = False
    AttProp.Label8.Caption = 1
    AttProp.TextBoxName.SetFocus
    AttProp.show
End Sub

Private Sub CommandButtonExit_Click()
    AttOpt.Hide
End Sub

Private Sub CommandButtonModify_Click() 'modify
    Dim selection As String
    Dim att_list As Range
    Dim attribute_param As AttributeProp

    AttOpt.Hide
    AttProp.CommandButtonDelete.Visible = True
    AttProp.Label8.Caption = 0
    selection = AttOpt.ListBoxAttributes.Value

    attribute_param = Retrieve_selection(selection)

    AttProp.TextBoxName.Text = attribute_param.Name
    AttProp.TextBoxMin.Text = attribute_param.min
    AttProp.TextBoxMax.Text = attribute_param.max
    AttProp.TextBoxUnits.Text = attribute_param.Units
    AttProp.TextBoxSteps.Text = attribute_param.Increment
    AttProp.TextBoxScenario.Text = attribute_param.Scenario
    If attribute_param.Direction = 0 Then
```

```
        AttProp.OptionButtonMax.Value = True
    Else
        AttProp.OptionButtonMin.Value = True
    End If
    AttProp.show
End Sub

Private Sub CommandButton3_Click() 'exit
    Refresh_all
    AttOpt.Hide
    AttNav.ListBox1.ListIndex = 0
    AttNav.show
End Sub
```

### 11.1.4 Attribute Properties




```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttProp
   Caption         =   "Attribute Properties"
   ClientHeight    =   7545
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   10965
   OleObjectBlob   =   "AttProp.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttProp"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub CheckBoxLinear_Click()
    If CheckBoxLinear.Value = True Then
        CheckBoxLogarithmic.Value = False
    Else
        CheckBoxLogarithmic.Value = True
    End If
End Sub

Private Sub CheckBoxLogarithmic_Click()
```

```
    If CheckBoxLogarithmic.Value = True Then
        CheckBoxLinear.Value = False
    Else
        CheckBoxLinear.Value = True
    End If
End Sub

Private Sub CheckBoxUnitsInc_Click()

End Sub

Private Sub ComboBoxFormat_Change()

    Dim Format As String

    If ComboBoxFormat.Text = "Other" Then
        ComboBoxFormat.Text = ""
        ComboBoxFormat.Enabled = False
        Format = InputBox("Please enter in desired data format.", "Other data
format")
        ComboBoxFormat.Text = Format
    Else
    End If

End Sub

Private Sub CommandButtonHelp_Click() 'help

End Sub

Sub CommandButtonSave_Click() 'save
    Dim numrows As Integer
    Dim Att_Range As Range
    Dim New_Att_Name As String
    Dim New_sheet As Worksheet
    Dim AttHistory_Sheet As Worksheet
    Dim Attribute_Modified As AttributeProp
    Dim att_step As Double
    Dim i As Integer
    Dim NewAttribute As Integer
    Dim Entries As Integer
    Dim Save_History As Boolean
    Dim Change As Variant
    Dim intResponse As Integer

    If Allow_Observe = True Then    'WDC, allow observe
        'Need to pass in values for the save
        Put #IntFile, Output_Index, AttProp.TextBoxName.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.TextBoxMin.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.TextBoxMax.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.TextBoxUnits.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.TextBoxSteps.Text
        Output_Index = Output_Index + 1
```

```
        Put #IntFile, Output_Index, AttProp.OptionButtonMin.Value
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.ComboBoxFormat.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.CheckBoxUnitsInc.Value
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, AttProp.TextBoxResolution.Text
        Output_Index = Output_Index + 1
        Put #IntFile, Output_Index, "AttProp.CommandButtonSave_Click()"
        Output_Index = Output_Index + 1
    End If

    Attribute_Modified = Read_fields(AttProp)
    NewAttribute = AttProp.Label8.Caption

    If NewAttribute Then

        'increment attribute list
        Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
        numrows = Att_Range.Rows.Count
        ActiveWorkbook.Names("Attribute_list").RefersTo =
Att_Range.Resize(numrows + 1)

        'col 8 has attributes
        'row 9 starts the attributes
        New_Att_Name = TextBoxName.Text
        Sheets("Home").Cells(7 + numrows + 1, 8).Value = New_Att_Name 'adds
attribute to main list on main page
        Sheets("Home").Cells(20 + numrows + 1, 8).Value = New_Att_Name 'adds
attribute to random list on main page

        'create a new sheet for the attribute
        Set New_sheet = Worksheets.Add
        New_sheet.Move after:=Worksheets(Worksheets.Count)
        New_sheet.Name = New_Att_Name

        'create a new sheet for the attribute history and rationale capture
        Set AttHistory_Sheet = Worksheets.Add
        AttHistory_Sheet.Move after:=Worksheets(Worksheets.Count)
        AttHistory_Sheet.Name = New_Att_Name + " history"

        With AttHistory_Sheet
            .Range("A1").Value = 1
            .Range("A2").Value = "Attribute Name:"
            .Range("A3").Value = "Min:"
            .Range("A4").Value = "Max:"
            .Range("A5").Value = "Units:"
            .Range("A6").Value = "Increments:"
            .Range("A7").Value = "Inc. Utility:"
            .Range("A8").Value = "Scenario:"
            .Range("A9").Value = "Rationale:"
            .Range("A10").Value = "Format:"         'WDC, addition for format
data
            .Range("A11").Value = "Units Include:"  'WDC, addition for format
data
            .Range("A12").Value = "Resolution:"     'WDC, addition for
resolution
```

```
            .Range("A13").Value = "Independant Value"
            .Range("A14").Value = "Definition:"
            .Range("A15").Value = "Linear Scale:"
            .Range("A16").Value = "Threshold:"
        End With

    Else
        Set New_sheet = Worksheets(Attribute_Modified.Name)
        Set AttHistory_Sheet = Worksheets(Attribute_Modified.Name + "
history")
    End If

    If NewAttribute Then
        Save_History = True
    Else
        Change = MsgBox("Was this a major change to the attribute
properties?", vbYesNo, "Major change?")
        If Change = vbYes Then
            Save_History = True
        Else
            Save_History = False
        End If

        If Allow_Observe = True Then    'WDC, allow observe
            Put #IntFile, Output_Index, "Change = " & Change
            Output_Index = Output_Index + 1
        End If
    End If

    If Save_History Then
        AttProp.Hide
        AttRationale.LabelAttribute.Caption = Attribute_Modified.Name
        AttRationale.show

        With AttHistory_Sheet
            Entries = .Range("A1").Value + 1
            .Range("A1").Value = Entries
            .Cells(1, Entries).Value = Entries - 1
            .Cells(2, Entries).Value = Attribute_Modified.Name
            .Cells(3, Entries).Value = Attribute_Modified.min
            .Cells(4, Entries).Value = Attribute_Modified.max
            .Cells(5, Entries).Value = Attribute_Modified.Units
            .Cells(6, Entries).Value = Attribute_Modified.Increment
            If Attribute_Modified.Direction Then
                .Cells(7, Entries).Value = "Min"
            Else
                .Cells(7, Entries).Value = "Max"
            End If
            .Cells(8, Entries).Value = Attribute_Modified.Scenario
            .Cells(9, Entries).Value = AttRationale.TextBoxRationale.Text
            .Cells(10, Entries).Value = Attribute_Modified.Format
            If Attribute_Modified.UnitsInc Then
                .Cells(11, Entries).Value = "Yes"
            Else
                .Cells(11, Entries).Value = "No"
            End If
            .Cells(12, Entries).Value = Attribute_Modified.Resolution
```

```
        AttRationale.TextBoxRationale.Text = ""
        .Cells(13, Entries).Value = Attribute_Modified.Independent
        .Cells(14, Entries).Value = Attribute_Modified.Definition
        .Cells(15, Entries).Value = Attribute_Modified.LinearScale
        .Cells(16, Entries).Value = Attribute_Modified.Threshold
    End With

End If

With New_sheet
    .Range("A1").Value = "Attribute Name:"
    .Range("B1").Value = Attribute_Modified.Name
    .Range("A2").Value = "Min:"
    .Range("B2").Value = Attribute_Modified.min
    .Range("A3").Value = "Max:"
    .Range("B3").Value = Attribute_Modified.max
    .Range("A4").Value = "Units:"
    .Range("B4").Value = Attribute_Modified.Units
    .Range("A5").Value = "Increment Size:"
    .Range("B5").Value = Attribute_Modified.Increment
    .Range("A6").Value = "Inc. Utility:"
    If Attribute_Modified.Direction Then
        .Range("B6").Value = "Min"
    Else
        .Range("B6").Value = "Max"
    End If
    .Range("A7").Value = "Scenario:"
    .Range("B7").Value = Attribute_Modified.Scenario
    .Range("A8").Value = "Format:"
    .Range("B8").Value = Attribute_Modified.Format
    .Range("A9").Value = "Units Include:"
    If Attribute_Modified.UnitsInc Then
        .Range("B9").Value = "Yes"
    Else
        .Range("B9").Value = "No"
    End If
    .Range("A10").Value = "Resolution:"
    .Range("B10").Value = Attribute_Modified.Resolution

    .Range("A11").Value = "Independent:"
    .Range("B11").Value = Attribute_Modified.Independent

    .Range("A12").Value = "Definition:"
    .Range("B12").Value = Attribute_Modified.Definition

    .Range("A13").Value = "Linear Scale:"
    .Range("B13").Value = Attribute_Modified.LinearScale

    .Range("A14").Value = "Threshold:"
    .Range("B14").Value = Attribute_Modified.Threshold

    .Range("L1").Value = "Single Value"
    .Range("M1").Value = "Single Probability"
    .Range("N1").Value = "Single Preference"

    .Range("P1").Value = "Corner Probability"
    .Range("Q1").Value = "Corner Preference"
```

```
        .Range("S1").Value = "Independence Type"
        .Range("T1").Value = "Probability"
        .Range("U1").Value = "Preference"

        .Range("J:J").ClearContents
        .Range("J1").Value = "att_options"
        If Attribute_Modified.LinearScale Then
            att_step = (Attribute_Modified.max - Attribute_Modified.min) /
(Attribute_Modified.Increment + 1)
        Else
            att_step = (Attribute_Modified.max - Attribute_Modified.min) ^ (1
/ (Attribute_Modified.Increment + 1))
        End If

        'REWRITE THIS TO DEAL WITH TEXT VALUES

        Dim AttributeValues() As Integer

        AttributeValues = RandomizeList(Attribute_Modified.Increment)

        For i = 1 To Attribute_Modified.Increment
            If Attribute_Modified.LinearScale Then
                .Cells(1 + i, 10).Value = Attribute_Modified.min + (att_step
* AttributeValues(i))
            Else
                .Cells(1 + i, 10).Value = Attribute_Modified.min + (att_step
^ AttributeValues(i))
            End If
        Next i
    End With

    Sheets("Home").Select
    Refresh_all
    AttProp.Hide
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub
Private Sub CommandButtonDelete_Click()
    Dim numrows As Integer
    Dim Att_Range As Range
    Dim curr_sheet As Worksheet
    Dim att_list As Range
    Dim i As Integer
    Dim j As Integer
    Dim Response As Variant
    Dim attrow As Integer

    Set att_list = Worksheets("Home").Range("Attribute_list")
    'MsgBox prompt:="Are you sure you want to delete this attribute?",
Buttons:=vbYesNo, Title:="Confirm deletion"

    Response = MsgBox("Are you sure you want to delete this attribute?",
vbYesNo, "Delete Attribute?")

    If Response = vbNo Then
        Exit Sub
```

```
    Else
        AttRationale.LabelAttribute.Caption = AttProp.TextBoxName.Text
        AttRationale.show
    End If

    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count

    Set curr_sheet = Sheets(AttProp.TextBoxName.Text)

    For i = 1 To numrows
        attrow = i
        If att_list.Value(i, 1) = AttProp.TextBoxName.Text Then Exit For
    Next i

    If attrow <= numrows Then
        For i = attrow To numrows - 1
            Worksheets("Home").Cells(7 + i, 8).Value = att_list.Value(i + 1,
1)
            Worksheets("Home").Cells(20 + i, 8).Value = att_list.Value(i + 1,
1)
        Next i
    End If

    'decrement attribute list
    ActiveWorkbook.Names("Attribute_list").RefersTo =
Att_Range.Resize(numrows - 1)
    Worksheets("Home").Cells(7 + numrows, 8).ClearContents
    Worksheets("Home").Cells(20 + numrows, 8).ClearContents

    Application.DisplayAlerts = False
    curr_sheet.Delete   'Deletes worksheet with attribute parameters
    Application.DisplayAlerts = True
    Set curr_sheet = Sheets("Home")

    Refresh_all
    AttProp.Hide
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub

Private Sub CommandButtonCancel_Click() 'cancel
    AttProp.Hide
    Unload AttProp
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub

Private Sub CommandButtonScenario_Click() 'modify scenario
    modify_scenario_start
End Sub

Private Sub TextBox1_Change()

End Sub

Private Sub OptionButtonMax_Click()
```

```
End Sub

Private Sub TextBoxName_Change()
    AttProp.Caption = "Attribute Properties for " + TextBoxName.Text
End Sub

Private Function Read_fields(curr_userform As UserForm) As AttributeProp
Dim temp_attribute As AttributeProp

    temp_attribute.Name = curr_userform.TextBoxName.Text
    temp_attribute.min = curr_userform.TextBoxMin.Text
    temp_attribute.max = curr_userform.TextBoxMax.Text
    temp_attribute.Units = curr_userform.TextBoxUnits.Text
    temp_attribute.Increment = curr_userform.TextBoxSteps.Text
    temp_attribute.Direction = curr_userform.OptionButtonMin
    temp_attribute.Scenario = curr_userform.TextBoxScenario.Text
    temp_attribute.Format = curr_userform.ComboBoxFormat.Text       'WDC,
added for format and units
    temp_attribute.UnitsInc = curr_userform.CheckBoxUnitsInc    'WDC, added
for format and units
    temp_attribute.Resolution = curr_userform.TextBoxResolution.Text
    temp_attribute.Independent = curr_userform.TextBoxIndependent.Text
    temp_attribute.LinearScale = curr_userform.CheckBoxLinear.Value
    temp_attribute.Threshold = curr_userform.TextBoxThreshold.Text
    temp_attribute.Definition = curr_userform.TextBoxDefinition.Text

Read_fields = temp_attribute

End Function

Private Sub UserForm_Click()

End Sub

VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttPropOld
   Caption         =   "Attribute Properties"
   ClientHeight    =   8310
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   5310
   OleObjectBlob   =   "AttPropOld.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttPropOld"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub CommandButtonHelp_Click() 'help

End Sub

Private Sub CommandButtonSave_Click() 'save
```

```
    Dim numrows As Integer
    Dim Att_Range As Range
    Dim New_Att_Name As String
    Dim New_sheet As Worksheet
    Dim AttHistory_Sheet As Worksheet
    Dim Attribute_Modified As AttributeProp
    Dim att_step As Double
    Dim i As Integer
    Dim NewAttribute As Integer
    Dim Entries As Integer
    Dim Save_History As Boolean
    Dim Change As Variant

    Attribute_Modified = Read_fields(AttProp)
    NewAttribute = AttProp.Label8.Caption

    If NewAttribute Then

        'increment attribute list
        Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
        numrows = Att_Range.Rows.Count
        ActiveWorkbook.Names("Attribute_list").RefersTo =
Att_Range.Resize(numrows + 1)

        'col 9 has attributes
        'row 14 starts the attributes
        New_Att_Name = TextBoxName.Text
        Sheets("Home").Cells(14 + numrows + 1, 9).Value = New_Att_Name 'adds
attribute to list on main page

        'create a new sheet for the attribute
        Set New_sheet = Worksheets.Add
        New_sheet.Move after:=Worksheets(Worksheets.Count)
        New_sheet.Name = New_Att_Name

        'create a new sheet for the attribute history and rationale capture
        Set AttHistory_Sheet = Worksheets.Add
        AttHistory_Sheet.Move after:=Worksheets(Worksheets.Count)
        AttHistory_Sheet.Name = New_Att_Name + " history"

        With AttHistory_Sheet
            .Range("A1").Value = 1
            .Range("A2").Value = "Attribute Name:"
            .Range("A3").Value = "Min:"
            .Range("A4").Value = "Max:"
            .Range("A5").Value = "Units:"
            .Range("A6").Value = "Number of Steps:"
            .Range("A7").Value = "Inc. Utility:"
            .Range("A8").Value = "Scenario:"
            .Range("A9").Value = "Rationale:"
        End With

    Else
        Set New_sheet = Worksheets(Attribute_Modified.Name)
        Set AttHistory_Sheet = Worksheets(Attribute_Modified.Name + "
history")
    End If
```

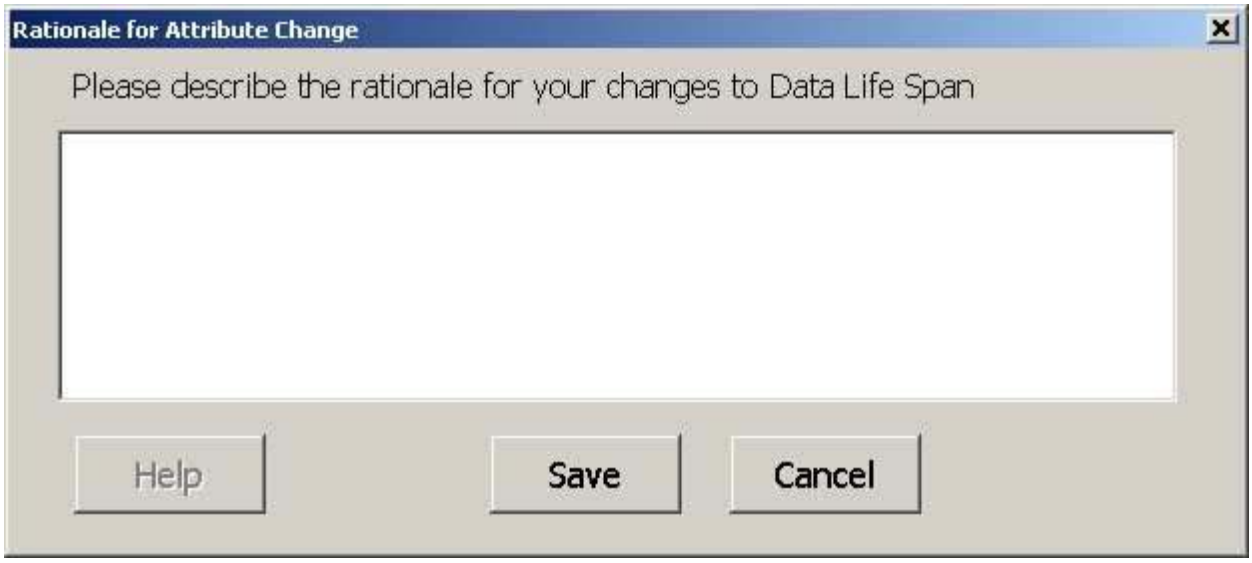
```
    If NewAttribute Then
        Save_History = True
    Else
        Change = MsgBox("Was this a major change to the attribute
properties?", vbYesNo, "Major change?")
        If Change = vbYes Then
            Save_History = True
        Else
            Save_History = False
        End If
    End If

    If Save_History Then
        AttProp.Hide
        AttRationale.LabelAttribute.Caption = Attribute_Modified.Name
        AttRationale.show

        With AttHistory_Sheet
            Entries = .Range("A1").Value + 1
            .Range("A1").Value = Entries
            .Cells(1, Entries).Value = Entries - 1
            .Cells(2, Entries).Value = Attribute_Modified.Name
            .Cells(3, Entries).Value = Attribute_Modified.min
            .Cells(4, Entries).Value = Attribute_Modified.max
            .Cells(5, Entries).Value = Attribute_Modified.Units
            .Cells(6, Entries).Value = Attribute_Modified.Increment
            If Attribute_Modified.Direction Then
                .Cells(7, Entries).Value = "Max"
            Else
                .Cells(7, Entries).Value = "Min"
            End If
            .Cells(8, Entries).Value = Attribute_Modified.Scenario
            .Cells(9, Entries).Value = AttRationale.TextBoxRationale.Text
            AttRationale.TextBoxRationale.Text = ""
        End With

    End If

    With New_sheet
        .Range("A1").Value = "Attribute Name:"
        .Range("B1").Value = Attribute_Modified.Name
        .Range("A2").Value = "Min:"
        .Range("B2").Value = Attribute_Modified.min
        .Range("A3").Value = "Max:"
        .Range("B3").Value = Attribute_Modified.max
        .Range("A4").Value = "Units:"
        .Range("B4").Value = Attribute_Modified.Units
        .Range("A5").Value = "Increment Size:"
        .Range("B5").Value = Attribute_Modified.Increment
        .Range("A6").Value = "Inc. Utility:"
        If Attribute_Modified.Direction Then
            .Range("B6").Value = "Max"
        Else
            .Range("B6").Value = "Min"
        End If
        .Range("A7").Value = "Scenario:"
```

```
        .Range("B7").Value = Attribute_Modified.Scenario
        .Range("I1").Value = "Sequence"
        .Range("J:J").ClearContents
        .Range("J1").Value = "att_options"
        att_step = (Attribute_Modified.max - Attribute_Modified.min) /
(Attribute_Modified.Increment - 1)

        'REWRITE THIS TO DEAL WITH TEXT VALUES

        For i = 0 To Attribute_Modified.Increment - 1
            .Cells(2 + i, 10).Value = Attribute_Modified.min + att_step * i
        Next i
'        If NewAttribute Then
'            .Shapes.AddTextbox(msoTextOrientationHorizontal, 50, 80, 300,
200).Name = Left(New_sheet.Name, 8) & "Box"
'        End If
'        .Shapes(Left(New_sheet.Name, 8) & "Box").TextFrame.Characters.Text =
Attribute_Modified.Scenario
    End With

    Sheets("Home").Select
    Refresh_all
    AttProp.Hide
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub
Private Sub CommandButtonDelete_Click()
    Dim numrows As Integer
    Dim Att_Range As Range
    Dim curr_sheet As Worksheet
    Dim att_list As Range
    Dim i As Integer
    Dim j As Integer
    Dim Response As Variant
    Dim attrow As Integer

    Set att_list = Worksheets("Home").Range("Attribute_list")
    'MsgBox prompt:="Are you sure you want to delete this attribute?",
Buttons:=vbYesNo, Title:="Confirm deletion"

    Response = MsgBox("Are you sure you want to delete this attribute?",
vbYesNo, "Delete Attribute?")

    If Response = vbNo Then
        Exit Sub
    Else
        AttRationale.LabelAttribute.Caption = AttProp.TextBoxName.Text
        AttRationale.show
    End If

    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count

    Set curr_sheet = Sheets(AttProp.TextBoxName.Text)

    For i = 1 To numrows
        attrow = i
```

```
        If att_list.Value(i, 1) = AttProp.TextBoxName.Text Then Exit For
    Next i

    If attrow < numrows Then
        For i = attrow To numrows - 1
            Worksheets("Home").Cells(16 + i, 9).Value = att_list.Value(i + 1,
1)
        Next i
    End If

    'decrement attribute list
    ActiveWorkbook.Names("Attribute_list").RefersTo =
Att_Range.Resize(numrows - 1)
    Worksheets("Home").Cells(16 + numrows, 9).ClearContents

    Application.DisplayAlerts = False
    curr_sheet.Delete   'Deletes worksheet with attribute parameters
    Application.DisplayAlerts = True
    Set curr_sheet = Sheets("Home")

    Refresh_all
    AttProp.Hide
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub

Private Sub CommandButtonCancel_Click() 'cancel
    AttProp.Hide
    AttOpt.ListBoxAttributes.ListIndex = 0
    AttOpt.show
End Sub

Private Sub CommandButtonScenario_Click() 'modify scenario
    modify_scenario_start
End Sub

Private Sub TextBox1_Change()

End Sub

Private Sub TextBoxName_Change()
    AttProp.Caption = "Attribute Properties for " + TextBoxName.Text
End Sub

Private Function Read_fields(curr_userform As UserForm) As AttributeProp
Dim temp_attribute As AttributeProp

    temp_attribute.Name = curr_userform.TextBoxName.Text
    temp_attribute.min = curr_userform.TextBoxMin.Text
    temp_attribute.max = curr_userform.TextBoxMax.Text
    temp_attribute.Units = curr_userform.TextBoxUnits.Text
    temp_attribute.Increment = curr_userform.TextBoxSteps.Text
    temp_attribute.Direction = curr_userform.OptionButtonMin
    temp_attribute.Scenario = curr_userform.TextBoxScenario.Text
    temp_attribute.Resolution = curr_userform.TextBoxResolution.Text
    temp_attribute.Independent = curr_userform.TextBoxIndependent.Text
```

```
Read_fields = temp_attribute

End Function

Private Sub UserForm_Click()

End Sub
```

### 11.1.5 Attribute Rationale


```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttRationale
   Caption         =   "Rationale for Attribute Change"
   ClientHeight    =   3765
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   9255
   OleObjectBlob   =   "AttRationale.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttRationale"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub CommandButtonHelp_Click() 'help

End Sub

Private Sub CommandButtonSave_Click() 'save
    AttRationale.Hide
'    AttProp.Show
End Sub
```

```
Private Sub CommandButtonCancel_Click() 'cancel
    AttRationale.Hide
'    AttProp.Show
End Sub

Private Sub TextBox1_Change()

End Sub

Private Sub CommandButton1_Click()

End Sub
```

### 11.1.6 Attribute Security

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttSec
   Caption         =   "Utility Interview Password Required"
   ClientHeight    =   3615
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   6240
   OleObjectBlob   =   "AttSec.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttSec"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Private Sub CommandButtonHelp_Click() 'help

End Sub

Private Sub CommandButtonOK_Click() 'ok
    Dim Entered_password As String
    Dim decrypt As String
    Const password = "adam"
    Const keyword = "□□□"

    Entered_password = TextBoxPassword.Text
    decrypt = dhXORText(Entered_password, keyword)
    If (decrypt = password) Then
        AttSec.Hide
        AttScen.TextBoxScenario.Text = "None"
        AttOpt.ListBoxAttributes.ListIndex = 0
        AttOpt.show
    Else
        TextBoxScenario.Text = ""
        Label3.Caption = "Invalid Password."
        TextBoxScenario.SetFocus
    End If
End Sub

Private Sub CommandButtonCancel_Click() 'cancel
    AttSec.Hide
    AttNav.ListBoxAttributes.ListIndex = 0
    AttNav.show
End Sub

Private Sub TextBox1_Change()

End Sub
```

### 11.1.7 Response Deletion


```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} AttValuesDelete
   Caption         =   "Clear Attribute Values"
   ClientHeight    =   6135
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   3810
   OleObjectBlob   =   "AttValuesDelete.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "AttValuesDelete"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub CommandButtonCancel_Click()
    AttValuesDelete.Hide
End Sub

Private Sub CommandButtonDelete_Click()
    Dim i As Integer

    For i = 0 To ListBoxInterviews.ListCount - 1
        If ListBoxInterviews.Selected(i) Then
            With Sheets(AttValuesDelete.LabelAttribute.Caption)
                Select Case ListBoxInterviews.List(i)
                    Case "Single Attribute"
                        .Range("L:L").ClearContents
                        .Range("M:M").ClearContents
                        .Range("N:N").ClearContents
                        .Range("O:O").ClearContents
                        .Cells(1, 12).Value = "Single Value"
                        .Cells(1, 13).Value = "Single Probability"
                        .Cells(1, 14).Value = "Single Preference"
                        Global_Interview_Type = "Single"
                        Mark_Deleted (AttValuesDelete.LabelAttribute.Caption)
                    Case "Corner Point"
                        .Range("P:P").ClearContents
                        .Range("Q:Q").ClearContents
                        .Range("R:R").ClearContents
                        .Cells(1, 16).Value = "Corner Probability"
                        .Cells(1, 17).Value = "Corner Preference"
                        Global_Interview_Type = "Corner Point"
                        Mark_Deleted (AttValuesDelete.LabelAttribute.Caption)
                    Case "Independence"
                        .Range("S:S").ClearContents
```

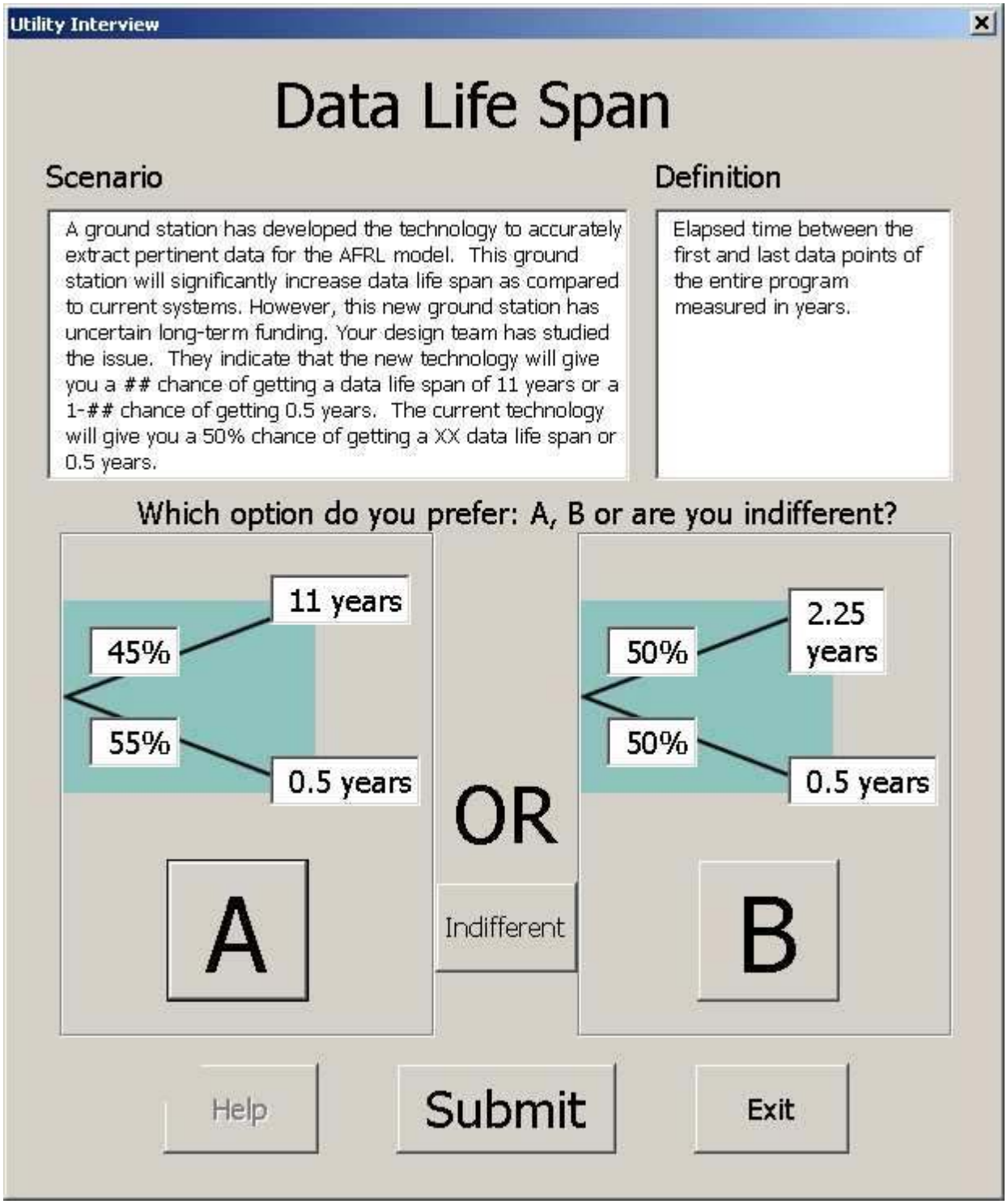
```
                        .Range("T:T").ClearContents
                        .Range("U:U").ClearContents
                        .Cells(1, 19).Value = "Independance Type"
                        .Cells(1, 20).Value = "Probability"
                        .Cells(1, 21).Value = "Preference"
                        Global_Interview_Type = "high"
                        Mark_Deleted (AttValuesDelete.LabelAttribute.Caption)
                        Global_Interview_Type = "low"
                        Mark_Deleted (AttValuesDelete.LabelAttribute.Caption)
                    Case "Curve Data"
                        .Range("W:W").ClearContents
                        .Range("X:X").ClearContents
                        .Cells(1, 23).Value = "Value"
                        .Cells(1, 24).Value = "Utility"
                    Case "Random Mix"
                        Dim Att_Range As Range
                        Dim AttCount As Integer, j As Integer
                        Set Att_Range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
                        AttCount = Att_Range.Rows.Count

                        For j = 1 To AttCount + 4
                            Sheets("Random Value
Answers").Rows(j).ClearContents
                        Next j
                        Sheets("Random Value Answers").Cells(1, 1).Value =
"Probability"
                        Sheets("Random Value Answers").Cells(2, 1).Value =
"Preference"
                        Sheets("Home").Rows(20).ClearContents
                        Sheets("Home").Cells(20, 8).Value = "Attribute"
                End Select
            End With
        End If
    Next i

    AttValuesDelete.Hide

End Sub
```

### 11.1.8 Generate Reports

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} GenReport
   Caption         =   "Generate Reports"
   ClientHeight    =   5565
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   9255
   OleObjectBlob   =   "GenReport.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "GenReport"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub InterviewsConducted_Click()
'Dim selectedInterview As String
'selectedInterview = GenReport.InterviewsConducted.Value



End Sub

Private Sub AllAttsButton_Click()
    For Attribute_index = 0 To NumOfAtts - 1
        Set InterviewsConducted.Selected(Interview_index) = True
    Next Interview_index

End Sub
Private Sub getReport_Click()

Dim selectedInterview As String
Dim selectedAtts() As String
Dim NumOfInterviews As Integer
Dim NumOfAtts As Integer
Dim Interview_index As Integer
Dim Attribute_index As Integer
Dim selectedAttIndex As Integer

NumOfInterviews = GenReport.InterviewsConducted.ListCount
NumOfAtts = GenReport.Attbox.ListCount

ReDim selectedAtts(NumOfAtts) 'SATWIK

'get the selected interview

selectedInterview = GenReport.InterviewsConducted.Value
Worksheets("TestSheet").Cells(3, 3) = selectedInterview

selectedAttIndex = 0
'for every attribute that got selected add it to the selectedAtts array of
Strings
```

```
For Attribute_index = 0 To NumOfAtts - 1
    If GenReport.Attbox.Selected(Attribute_index) = True Then
        selectedAtts(selectedAttIndex) =
GenReport.Attbox.List(Attribute_index) 'SATWIK
        Worksheets("TestSheet").Cells(4, 3) = selectedInterview
        selectedAttIndex = selectedAttIndex + 1
    End If
Next Attribute_index

Call GenerateReports(selectedInterview, selectedAtts())

GenReport.Hide

End Sub


Private Sub exitButton_Click()
    GenReport.Hide
End Sub

Private Sub Label3_Click()

End Sub
```

### 11.1.9 Single Attribute Interview




```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} UtilInt
   Caption         =   "Utility Interview"
   ClientHeight    =   10710
```

```
   ClientLeft      =   30
   ClientTop       =   390
   ClientWidth     =   9195
   OleObjectBlob   =   "UtilInt.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "UtilInt"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Option Explicit


Private Sub ChartSpaceUtility_DataSetChange()

End Sub

Private Sub Label4_Click()

End Sub

Private Sub ChartSpace1_DataSetChange()

End Sub

Private Sub end_interview_Click()

End Sub

Private Sub LabelAttribute_Click()

End Sub
Public Sub Store_Choice()
    Dim Interview_Storage As Worksheet
    Dim Preference As String
    Dim Interview_index As Integer

    'Figure out choice
    'If TextBoxA.BackColor = &HFF00& Then
    '    Preference = "A"
    'ElseIf TextBoxB.BackColor = &HFF00& Then
    '    Preference = "B"
    'ElseIf TextBoxI.BackColor = &HFF00& Then
    '    Preference = "I"
    'End If

    Preference = UtilInt.LabelPreference.Caption

    Global_Choice = Preference

    'If no choice made, do not allow submit
    'If (TextBoxA.BackColor = &H80000005) And (TextBoxI.BackColor =
&H80000005) And (TextBoxB.BackColor = &H80000005) Then Exit Sub
    If Preference = "" Then Exit Sub
```

```
    'save changes somewhere!!
    Set Interview_Storage = Worksheets(UtilInt.LabelAttribute.Caption)
    With Interview_Storage
        Select Case UtilInt.LabelInterviewType
            Case "single"
                'Determine index of interview
                Interview_index = .Cells(1, 15).Value + 2
                .Cells(1, 15).Value = Interview_index - 1

                .Cells(Interview_index, 12).Value =
CDbl(UtilInt.LabelCurrentValue.Caption)
                .Cells(Interview_index, 13).Value = TextBoxPe.Text
                .Cells(Interview_index, 14).Value = Preference
            Case "independence high"
                Interview_index = .Cells(1, 22).Value + 2
                .Cells(1, 22).Value = Interview_index - 1

                .Cells(Interview_index, 19).Value = "high"
                .Cells(Interview_index, 20).Value = TextBoxPe.Text
                .Cells(Interview_index, 21).Value = Preference
            Case "independence low"
                Interview_index = .Cells(1, 22).Value + 2
                .Cells(1, 22).Value = Interview_index - 1

                .Cells(Interview_index, 19).Value = "low"
                .Cells(Interview_index, 20).Value = TextBoxPe.Text
                .Cells(Interview_index, 21).Value = Preference
        End Select

    End With

    If Preference = "I" Then
        Call UpdateChart
    End If

    'clear all green buttons
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)

'    UtilInt.end_interview.Caption = 2
End Sub
Private Sub Submit_Click()
    UtilInt_Store_Choice
    'clear all green buttons
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)


    UtilInt.Hide
End Sub

Private Sub CommandButtonExit_Click()
    ChartSpace1.Clear
    UtilInt.Hide
    UtilInt.end_interview.Caption = "end"
```

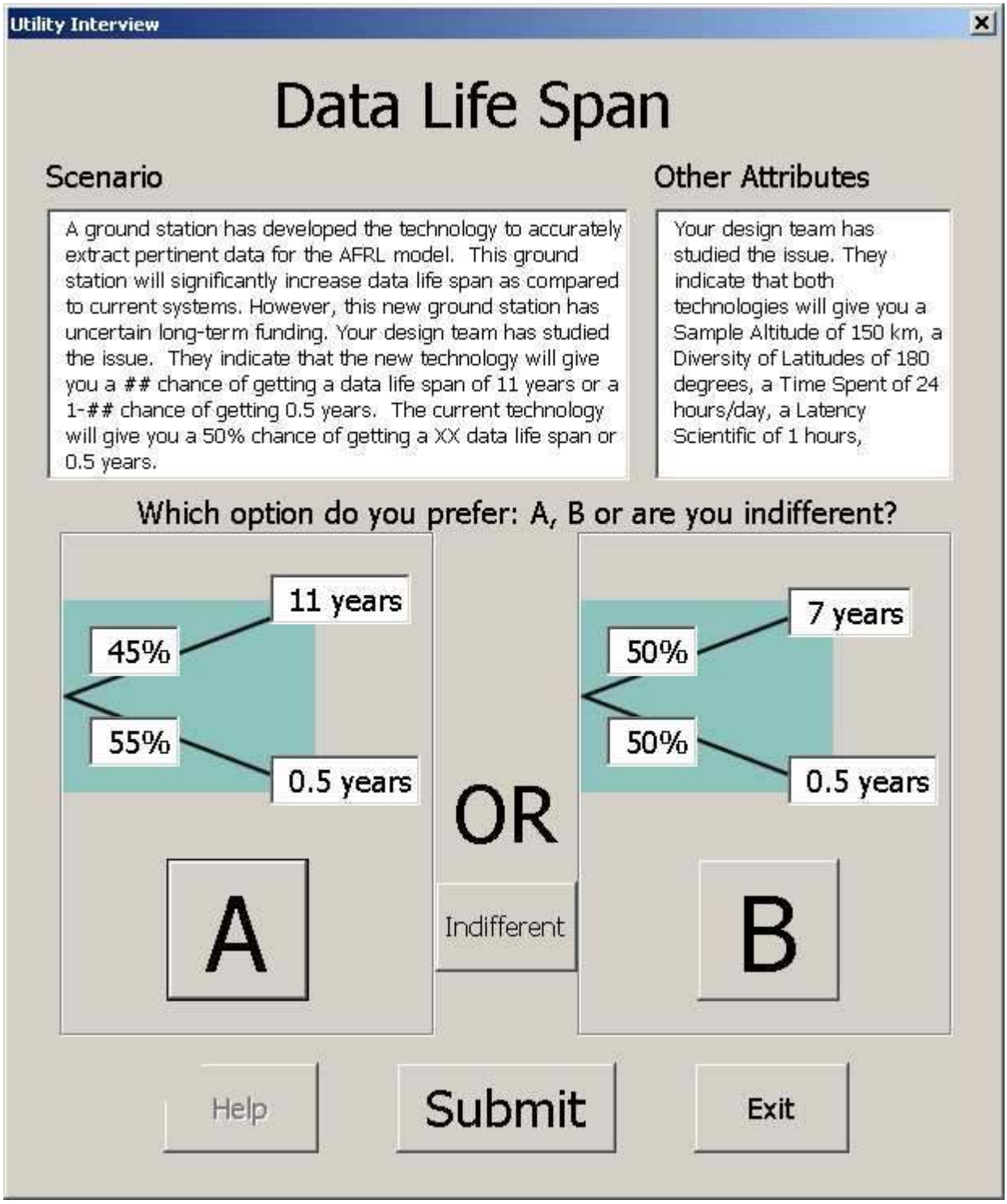
```
End Sub

Private Sub TxtBoxA_Change(Isclick As Boolean)
    If Isclick Then
'        TextBoxA.BackColor = &HFF00&
        UtilInt.LabelPreference.Caption = "A"
        Choice_A.BackColor = &HFF00&
    Else
'        TextBoxA.BackColor = &H80000005
        Choice_A.BackColor = &H8000000F
    End If
End Sub

Private Sub TxtBoxI_Change(Isclick As Boolean)
    If Isclick Then
'        TextBoxA.BackColor = &HFF00&
        UtilInt.LabelPreference.Caption = "I"
        Choice_Indiff.BackColor = &HFF00&
    Else
'        TextBoxA.BackColor = &H80000005
        Choice_Indiff.BackColor = &H8000000F
    End If
End Sub

Private Sub TxtBoxB_Change(Isclick As Boolean)
    If Isclick Then
'        TextBoxA.BackColor = &HFF00&
        UtilInt.LabelPreference.Caption = "B"
        Choice_B.BackColor = &HFF00&
    Else
'        TextBoxA.BackColor = &H80000005
        Choice_B.BackColor = &H8000000F
    End If

End Sub

Private Sub TextToSpeech1_ClickIn(ByVal x As Long, ByVal y As Long)

End Sub

Private Sub Choice_A_Click()
    TxtBoxA_Change (True)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)

End Sub

Private Sub Choice_B_Click()
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (True)

End Sub

Private Sub Choice_Indiff_Click()
    TxtBoxA_Change (False)
    TxtBoxI_Change (True)
```

```
    TxtBoxB_Change (False)

End Sub

Private Sub TextBoxPe_Change()

End Sub

Private Sub UserForm_Click()

End Sub

Private Sub UserForm_Activate()
    Call CreateChart
End Sub


Private Sub UpdateChart()
    Dim Chart2 As OWC.WCChart
    Dim Series2 As OWC.WCSeries
    Dim s As Integer
    Dim XData(1 To 3)
    Dim YData(1 To 3)
    Dim t As Integer
    Dim Att As String

    ChartSpace1.Clear
    ChartSpace1.Refresh

    t = Worksheets("Single Attribute
Interview").Range("A65536").End(xlUp).Row

    Set Chart2 = ChartSpace1.Charts.Add

    With ChartSpace1
        .HasChartSpaceTitle = True
        .ChartSpaceTitle.Caption = Worksheets("Single Attribute
Interview").Cells(t, 1)
        .ChartSpaceTitle.Font.Size = 14
    End With

    Att = LabelAttribute.Caption
    For s = 2 To 3
            XData(s - 1) = Worksheets(Att).Cells(s, 2)
    Next s
    YData(1) = 0
    YData(2) = 1

    XData(3) = Worksheets("Single Attribute Interview").Cells(t, 2)
    YData(3) = Worksheets("Single Attribute Interview").Cells(t, 3)
    YData(3) = YData(3) * 2

    Chart2.Type = OWC.chChartTypeSmoothLineMarkers

    Set Series2 = Chart2.SeriesCollection.Add

    With Series2
```

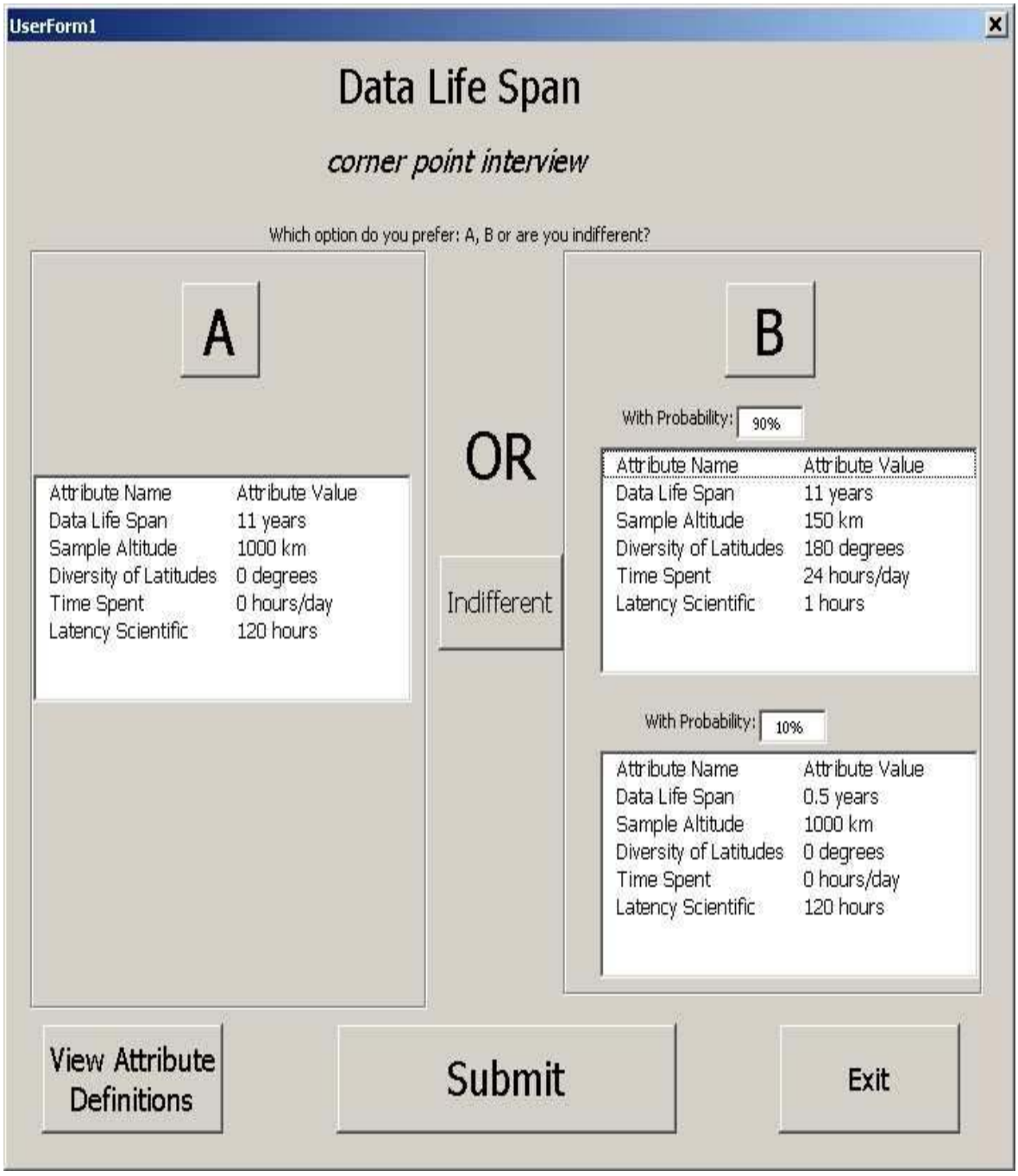
```
        .SetData OWC.chDimCategories, OWC.chDataLiteral, XData
        .SetData OWC.chDimValues, OWC.chDataLiteral, YData
    End With

    With Chart2.Axes(OWC.chAxisPositionLeft)
        .Scaling.Maximum = 1
        .Scaling.Minimum = 0
        .HasTitle = True
        .Title.Caption = "Utility"
        .Title.Font.Size = 12
        .MajorUnit = 0.1
    End With

    With Chart2.Axes(OWC.chAxisPositionBottom)
        .HasTitle = True
        .Title.Caption = "Value"
        .Title.Font.Size = 12
    End With

End Sub


Private Sub CreateChart()
    Dim r As Integer
    Dim XValues(1 To 2)
    Dim YValues As Variant
    Dim Attr As String
    Dim Chart1 As OWC.WCChart
    Dim Series1 As OWC.WCSeries
    Dim t As Integer

    ChartSpace1.Clear
    ChartSpace1.Refresh

'   Add a chart to the ChartSpace
    Set Chart1 = ChartSpace1.Charts.Add

'   Get the chart categories, values, and title
    Attr = LabelAttribute.Caption
    For r = 2 To 3
            XValues(r - 1) = Worksheets(Attr).Cells(r, 2)
    Next r
    YValues = Array(0, 1)
    With ChartSpace1
        .HasChartSpaceTitle = True
        .ChartSpaceTitle.Caption = Worksheets(Attr).Range("B1")
        .ChartSpaceTitle.Font.Size = 14
    End With

    Chart1.Type = OWC.chChartTypeSmoothLineMarkers

    Set Series1 = Chart1.SeriesCollection.Add

    With Series1
        .SetData OWC.chDimCategories, OWC.chDataLiteral, XValues
        .SetData OWC.chDimValues, OWC.chDataLiteral, YValues
    End With
```

```
    With Chart1.Axes(OWC.chAxisPositionLeft)
        .Scaling.Maximum = 1
        .Scaling.Minimum = 0
        .HasTitle = True
        .Title.Caption = "Utility"
        .Title.Font.Size = 12
        .MajorUnit = 0.1
    End With

    With Chart1.Axes(OWC.chAxisPositionBottom)
        .HasTitle = True
        .Title.Caption = "Value"
        .Title.Font.Size = 12
    End With

End Sub
```

### 11.1.10 Independence Interview



### 11.1.11 Multiple Attribute Interview




```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} UtilIntII
   Caption         =   "UserForm1"
   ClientHeight    =   9765
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   10620
   OleObjectBlob   =   "UtilIntII.frx":0000
```

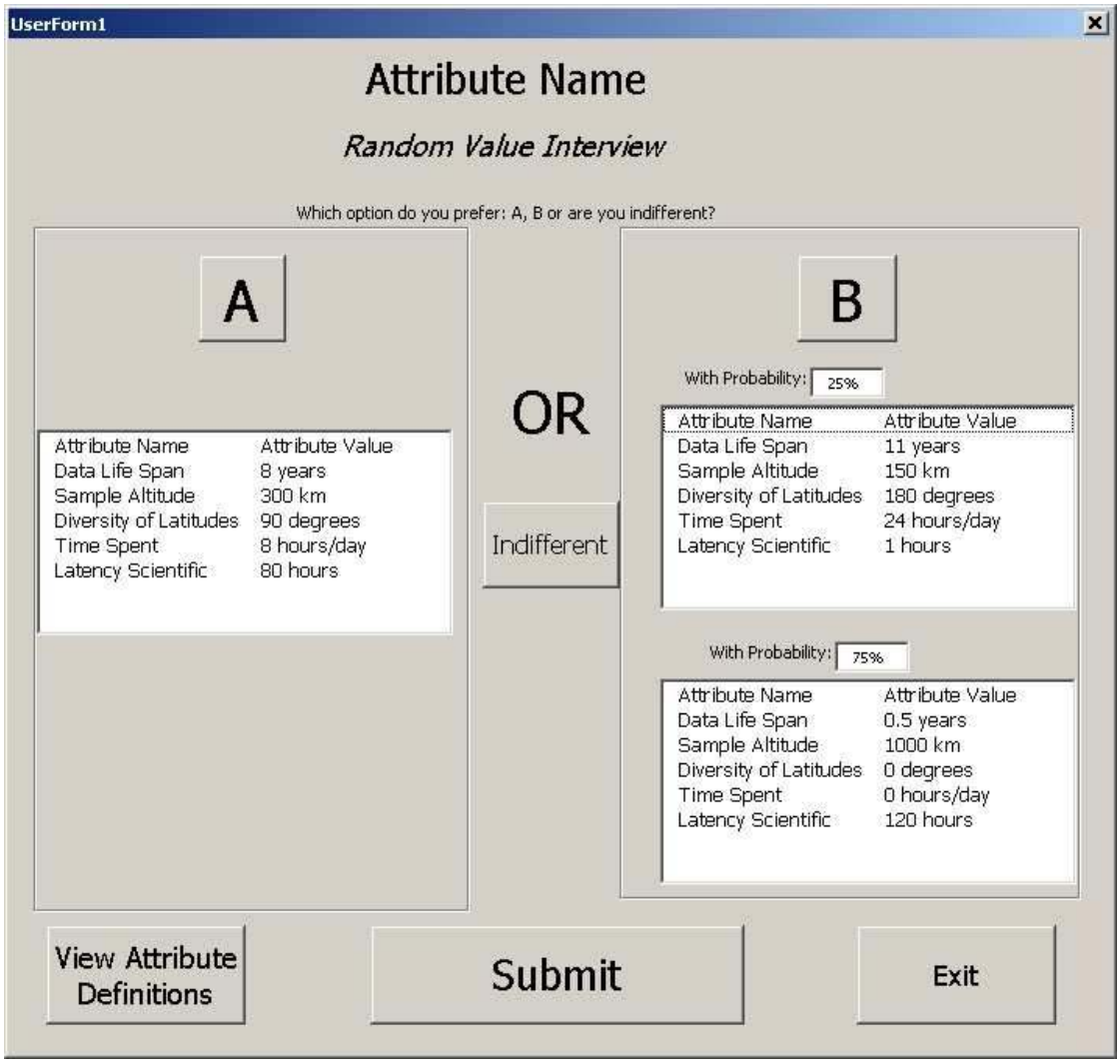
```
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "UtilIntII"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Option Explicit
Private Sub CommandButtonDefinition_Click()
    AttNavOld.CommandButtonInterview.Caption = "View Definition"
    Dim OldInterview As String
    OldInterview = AttNavOld.LabelInterview.Caption
    AttNavOld.LabelInterview.Caption = "definition"
    AttNavOld.show
    AttNavOld.LabelInterview.Caption = OldInterview
    AttNavOld.CommandButtonInterview.Caption = "Interview"
End Sub

Private Sub ChartSpaceUtility_DataSetChange()

End Sub

Private Sub Frame1_Click()

End Sub

Private Sub Frame2_Click()

End Sub

Private Sub Label5_Click()

End Sub

Private Sub ListBox2_Click()
'for Array_BI
End Sub

Private Sub ListBox3_Click()
'for Array_BII
End Sub

Private Sub ListBox1_Click()
 'for Array_A
End Sub
Public Sub Store_Choice()
    Dim Interview_index As Integer
    Dim Interview_Storage As Worksheet
    Dim Random_Values() As String

    'save changes somewhere!!
    Set Interview_Storage = Worksheets(UtilIntII.LabelAttribute.Caption)
    With Interview_Storage
        'Determine type of interview
        Select Case UtilIntII.LabelInterviewType.Caption
            Case "corner point interview"
```

```
                'Determine index of interview
                Interview_index = .Cells(1, 18).Value + 2
                .Cells(1, 18).Value = Interview_index - 1

                .Cells(Interview_index, 16).Value = UtilIntII.TextBoxPe.Text
                .Cells(Interview_index, 17).Value = Global_Choice
            Case "Random Value Interview"
                Interview_index = Sheets("Random Value Answers").Cells(2,
1).Value + 1
                Sheets("Random Value Answers").Cells(2, 1).Value =
Interview_index

                Sheets("Random Value Answers").Cells(1, Interview_index +
1).Value = UtilIntII.TextBoxPe.Text
                Sheets("Random Value Answers").Cells(2, Interview_index +
1).Value = Global_Choice

                Random_Values = UtilIntII.ListBox1.List

                Sheets("Random Value Answers").Cells(4, Interview_index +
1).Value = Random_Values
        End Select

    End With

End Sub

Private Sub Submit_Click()
    Dim Interview_Storage As Worksheet
    Dim Preference As String
    Dim Interview_index As Integer

    'If no choice made, do not allow submit
    If (Choice_A.BackColor = &H80000005) And (Choice_Indiff.BackColor =
&H80000005) And (Choice_B.BackColor = &H80000005) Then Exit Sub

    'Figure out choice
    If Choice_A.BackColor = &HFF00& Then
        Preference = "A"
    ElseIf Choice_B.BackColor = &HFF00& Then
        Preference = "B"
    ElseIf Choice_Indiff.BackColor = &HFF00& Then
        Preference = "I"
    End If

    'WDC, now selection stored globally
    Global_Choice = Preference

    UtilIntII_Store_Choice

    'clear all green boxes
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)
    UtilIntII.end_interview.Caption = 2
    UtilIntII.Hide
```

```
End Sub
Private Sub CommandButtonExit_Click()
    UtilIntII.Hide
    UtilIntII.end_interview.Caption = 1
End Sub

Private Sub TxtBoxA_Change(Isclick As Boolean)
    If Isclick Then
        Choice_A.BackColor = &HFF00&
        'TextBoxA.BackColor = &HFF00&
    Else
        Choice_A.BackColor = &H8000000F
        'TextBoxA.BackColor = &H80000005
    End If
End Sub

Private Sub TxtBoxI_Change(Isclick As Boolean)
    If Isclick Then
        Choice_Indiff.BackColor = &HFF00&
        'TextBoxI.BackColor = &HFF00&
    Else
        Choice_Indiff.BackColor = &H8000000F
        'TextBoxI.BackColor = &H80000005
    End If
End Sub

Private Sub TxtBoxB_Change(Isclick As Boolean)
    If Isclick Then
        Choice_B.BackColor = &HFF00&
        'TextBoxB.BackColor = &HFF00&
    Else
        Choice_B.BackColor = &H8000000F
        'TextBoxB.BackColor = &H80000005
    End If

End Sub

Private Sub TextToSpeech1_ClickIn(ByVal x As Long, ByVal y As Long)

End Sub

Private Sub Choice_A_Click()
    TxtBoxA_Change (True)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)

End Sub

Private Sub Choice_B_Click()
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (True)

End Sub

Private Sub Choice_Indiff_Click()
    TxtBoxA_Change (False)
```

```
    TxtBoxI_Change (True)
    TxtBoxB_Change (False)

End Sub

Private Sub TextBox12_Change()

End Sub

Private Sub TextBox13_Change()

End Sub

Private Sub UserForm_Click()

End Sub
```

### 11.1.12 Random Mix Interview



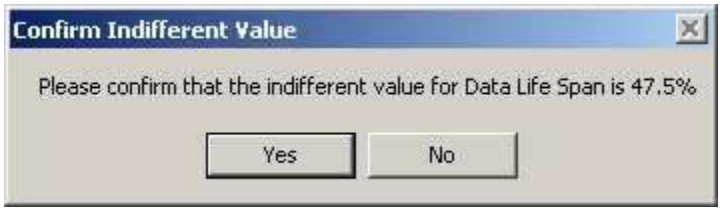

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} RandInt
   Caption         =   "RandInt"
   ClientHeight    =   8625
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   9315
   OleObjectBlob   =   "RandInt.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "RandInt"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

```
Option Explicit


Private Sub ChartSpaceUtility_DataSetChange()

End Sub

Private Sub Frame1_Click()

End Sub

Private Sub Frame2_Click()

End Sub

Private Sub ListBox2_Click()
'for Array_BI
End Sub

Private Sub ListBox3_Click()
'for Array_BII
End Sub

Private Sub ListBox1_Click()
 'for Array_A
End Sub

Private Sub Submit_Click()
    Dim Interview_Storage As Worksheet
    Dim Preference As String
    Dim Interview_index As Integer

    'If no choice made, do not allow submit
    If (TextBoxA.BackColor = &H80000005) And (TextBoxI.BackColor =
&H80000005) And (TextBoxB.BackColor = &H80000005) Then Exit Sub

    'Figure out choice
    If TextBoxA.BackColor = &HFF00& Then
        Preference = "A"
    ElseIf TextBoxB.BackColor = &HFF00& Then
        Preference = "B"
    ElseIf TextBoxI.BackColor = &HFF00& Then
        Preference = "I"
    End If

    'Determine index of interview
    Interview_index =
ActiveWorkbook.Names("Interview_Index").RefersToRange.Value

    'save changes somewhere!!
    Set Interview_Storage = Worksheets("Single Attribute Interview")
    With Interview_Storage
        .Cells(Interview_index, 1).Value = LabelAttribute.Caption
        '.Cells(Interview_Index, 2).Value = TextBoxCurr.Text
        .Cells(Interview_index, 3).Value = TextBoxPe.Text
        .Cells(Interview_index, 4).Value = Preference
    End With
```

```
    ActiveWorkbook.Names("Interview_Index").RefersToRange.Value =
Interview_index + 1

    'clear all green boxes
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)
    RandInt.end_interview.Caption = 2
    RandInt.Hide
End Sub

Private Sub CommandButtonExit_Click()
    RandInt.Hide
    RandInt.end_interview.Caption = 1
End Sub

Private Sub TxtBoxA_Change(Isclick As Boolean)
    If Isclick Then
        TextBoxA.BackColor = &HFF00&
    Else
        TextBoxA.BackColor = &H80000005
    End If
End Sub

Private Sub TxtBoxI_Change(Isclick As Boolean)
    If Isclick Then
        TextBoxI.BackColor = &HFF00&
    Else
        TextBoxI.BackColor = &H80000005
    End If
End Sub

Private Sub TxtBoxB_Change(Isclick As Boolean)
    If Isclick Then
        TextBoxB.BackColor = &HFF00&
    Else
        TextBoxB.BackColor = &H80000005
    End If

End Sub

Private Sub TextToSpeech1_ClickIn(ByVal x As Long, ByVal y As Long)

End Sub

Private Sub Choice_A_Click()
    TxtBoxA_Change (True)
    TxtBoxI_Change (False)
    TxtBoxB_Change (False)

End Sub

Private Sub Choice_B_Click()
    TxtBoxA_Change (False)
    TxtBoxI_Change (False)
    TxtBoxB_Change (True)
```

```
End Sub

Private Sub Choice_Indiff_Click()
    TxtBoxA_Change (False)
    TxtBoxI_Change (True)
    TxtBoxB_Change (False)

End Sub

Private Sub TextBox12_Change()

End Sub

Private Sub TextBox13_Change()

End Sub

Private Sub UserForm_Click()

End Sub
```

### 11.1.13 Indifference Point Confirmation




```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} IndifferentConfirm
   Caption         =   "Confirm"
   ClientHeight    =   2565
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   3630
   OleObjectBlob   =   "IndifferentConfirm.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "IndifferentConfirm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub CommandButtonConfirm_Click()
    IndifferentConfirm.Hide
    AttNav.show
End Sub

Private Sub CommandButtonRedo_Click()
    IndifferentConfirm.Hide
    Util_InterviewIIA_Start (AttNav.ListBoxAttributes.Value)
End Sub
```

### 11.1.14 Indifference Point Error

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} IndifferentError
   Caption         =   "Error"
   ClientHeight    =   2490
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   3660
   OleObjectBlob   =   "IndifferentError.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "IndifferentError"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub CommandButtonExit_Click()
    IndifferentError.Hide
    AttNav.show
End Sub

Private Sub CommandButtonRedo_Click()
    IndifferentError.Hide
    Util_InterviewIIA_Start (AttNav.ListBoxAttributes.Value)
End Sub
```

### 11.1.15 Start Log


```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} LogFile
   Caption         =   "Log File Name"
   ClientHeight    =   3210
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4440
   OleObjectBlob   =   "LogFile.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "LogFile"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub CommandButton1_Click()
    LogFile.Hide
End Sub
```

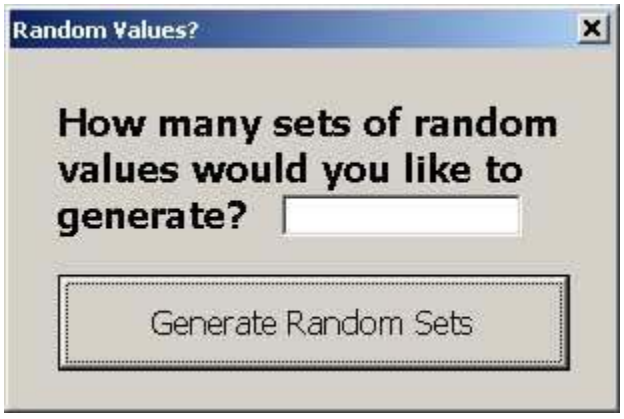
### 11.1.16 Interview Override - Value

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} NextValue
   Caption         =   "Next value?"
   ClientHeight    =   4935
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4710
   OleObjectBlob   =   "NextValue.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "NextValue"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub CommandButton1_Click()

End Sub

Private Sub TextBoxValue_Change()
    If NextValue.TextBoxValue.Text = "end" Then
        NextValue.CommandButtonInterview.Caption = "End Interview"
    Else
        NextValue.CommandButtonInterview.Caption = "Interview for " +
Format_Value(NextValue.LabelAttribute, NextValue.TextBoxValue.Text)
    End If
End Sub

Private Sub CommandButtonEnd_Click()
    NextValue.TextBoxValue.Text = "end"
    NextValue.Hide
End Sub

Private Sub CommandButtonInterview_Click()
    If NextValue.TextBoxValue.Text > "" Then
        NextValue.Hide
    End If
End Sub
```

### 11.1.17 Interview Override - Probability

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} NextProbability
   Caption         =   "Next probability?"
   ClientHeight    =   4935
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4710
   OleObjectBlob   =   "NextProbability.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "NextProbability"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False



Private Sub CommandButton1_Click()

End Sub

Private Sub Label1_Click()

End Sub

Private Sub TextBoxProbability_Change()
    If NextProbability.TextBoxProbability.Text = "end" Then
        NextProbability.CommandButtonInterview.Caption = "End Interview"
    Else
        NextProbability.CommandButtonInterview.Caption = "Interview for " +
NextProbability.TextBoxProbability.Text + "%"
        End If
End Sub

Private Sub CommandButtonEnd_Click()
    NextProbability.TextBoxProbability.Text = "end"
    NextProbability.Hide
End Sub

Private Sub CommandButtonInterview_Click()
    If NextProbability.TextBoxProbability.Text > "" Then
        NextProbability.Hide
    End If
End Sub

Private Sub Label2_Click()

End Sub
```

### 11.1.18 Generate Random Sets




```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} Random
   Caption         =   "Random Values?"
   ClientHeight    =   2670
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   4530
   OleObjectBlob   =   "Random.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "Random"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub CommandButtonGenerate_Click()
    Random.Hide
    Generate_Random_Values (CInt(TextBoxRandomSets.Text))
End Sub

Private Sub TextBoxRandomSets_Change()
    CommandButtonGenerate.Caption = "Generate " + TextBoxRandomSets.Text + "
Random Sets"
End Sub
```

### 11.1.19 Schedule Interview

```
VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} ScheduleInterview
   Caption         =   "Schedule Interview"
   ClientHeight    =   11790
   ClientLeft      =   45
   ClientTop       =   345
   ClientWidth     =   11130
   OleObjectBlob   =   "ScheduleInterview.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "ScheduleInterview"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Label1_Click()

End Sub

Private Sub Label3_Click()

End Sub

Private Sub Label4_Click()

End Sub

VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} Interview
   Caption         =   "UserForm1"
   ClientHeight    =   4335
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   7020
   OleObjectBlob   =   "Interview.frx":0000
   StartUpPosition =   1  'CenterOwner
End
Attribute VB_Name = "Interview"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False


Option Explicit

Private Sub AddAll_Click()
    Dim a As Integer
    ReDim Preserve Data(1 To lbIntType.ListCount, 1 To 2)
    For a = 0 To lbIntType.ListCount - 1
        lbIntType.Selected(a) = True
    Next a
    For a = 1 To 5
        Data(a, 1) = "Spatial Resolution"
```

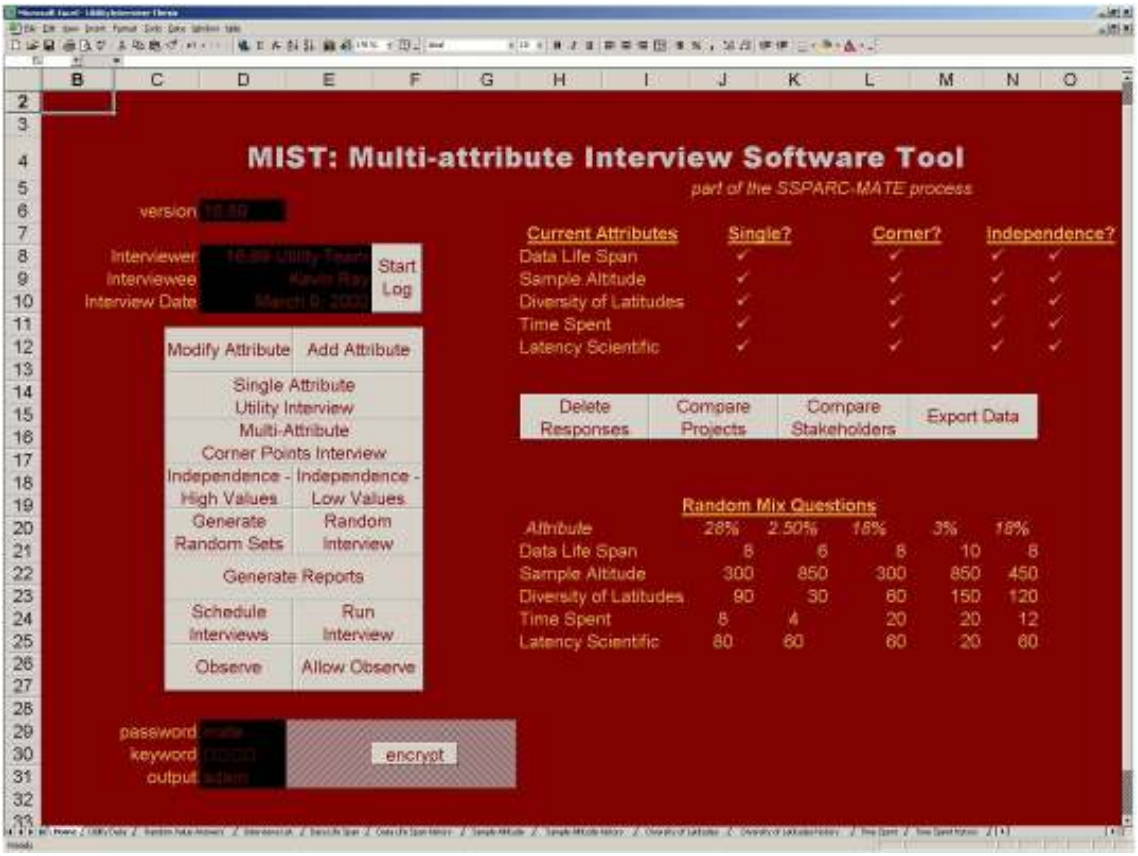
```
        Data(a, 2) = lbIntType.Selected(a - 1)
    Next a
End Sub

Private Sub CancelButton_Click()
    Dim Msg As String
    Dim Ans As Integer

    Msg = "Cancel the wizard?"
    Ans = MsgBox(Msg, vbQuestion + vbYesNo, APPNAME)
    If Ans = vbYes Then Unload Me
End Sub

Private Sub BackButton_Click()
    MultiPage1.Value = MultiPage1.Value - 1
    UpdateControls
End Sub

Private Sub AddInterview_Click()
    MultiPage1.Value = MultiPage1.Value - 1
    UpdateControls
End Sub

Private Sub NextButton_Click()
    MultiPage1.Value = MultiPage1.Value + 1
    UpdateControls
End Sub


Private Sub FinishButton_Click()
    Dim r As Long
    Dim s As Integer
    Dim t As Integer
    Dim b As Integer

    r = Application.WorksheetFunction. _
      CountA(Range("A:A")) + 1


    t = r + z
    For s = r To t
        Cells(r, 1) = tbName.Text
        b = z
        z = 1
        Cells(r, 2) = Data(z, 1)
        Cells(r, 3) = Data(z, 2)
        z = z + 1
        r = r + 1
    Next s

    Unload Me

End Sub

Private Sub MultiPage1_Change()
    Dim i As Integer
```

```
    If MultiPage1.Value = 2 Then
        ReDim Preserve Data(1 To z, 1 To 2)
        If obSpatial Then Data(z, 1) = "Spatial Resolution"
        If obRevisit Then Data(z, 1) = "Revisit Time"
        If obLatency Then Data(z, 1) = "Latency"
        Data(z, 2) = lbIntType.Value
        lbIntAdd.ColumnCount = 2
        lbIntAdd.List = Data
        z = z + 1
    End If

    If MultiPage1.Value = 3 Then
        lbIntSel.ColumnCount = 2
        lbIntSel.List = Data
        lbIntSched.ColumnCount = 2
    End If
End Sub

Private Sub AddButton_Click()
    Dim j As Integer
    If lbIntSel.ListIndex = -1 Then Exit Sub
    lbIntSel.BoundColumn = 1
    If Not cbDuplicates Then
        For j = 0 To lbIntSched.ListCount - 1
            If lbIntSel.Value = lbIntSched.List(j) Then
                Beep
                Exit Sub
            End If
        Next j
    End If
    lbIntSched.AddItem lbIntSel.Value
End Sub

Private Sub DeleteButton_Click()
    If lbIntSched.ListIndex = -1 Then Exit Sub
    lbIntSched.RemoveItem lbIntSched.ListIndex
End Sub


Private Sub tbName_Change()
    If tbName.Text = "" Then FinishButton.Enabled = False Else
FinishButton.Enabled = True
End Sub

Public Sub UserForm_Initialize()
    MultiPage1.Value = 0
    UpdateControls
    With lbIntType
        .RowSource = ""
        .AddItem "Single"
        .AddItem "Corner"
        .AddItem "Indep1"
        .AddItem "Indep2"
        .AddItem "Random"
    End With
End Sub
```

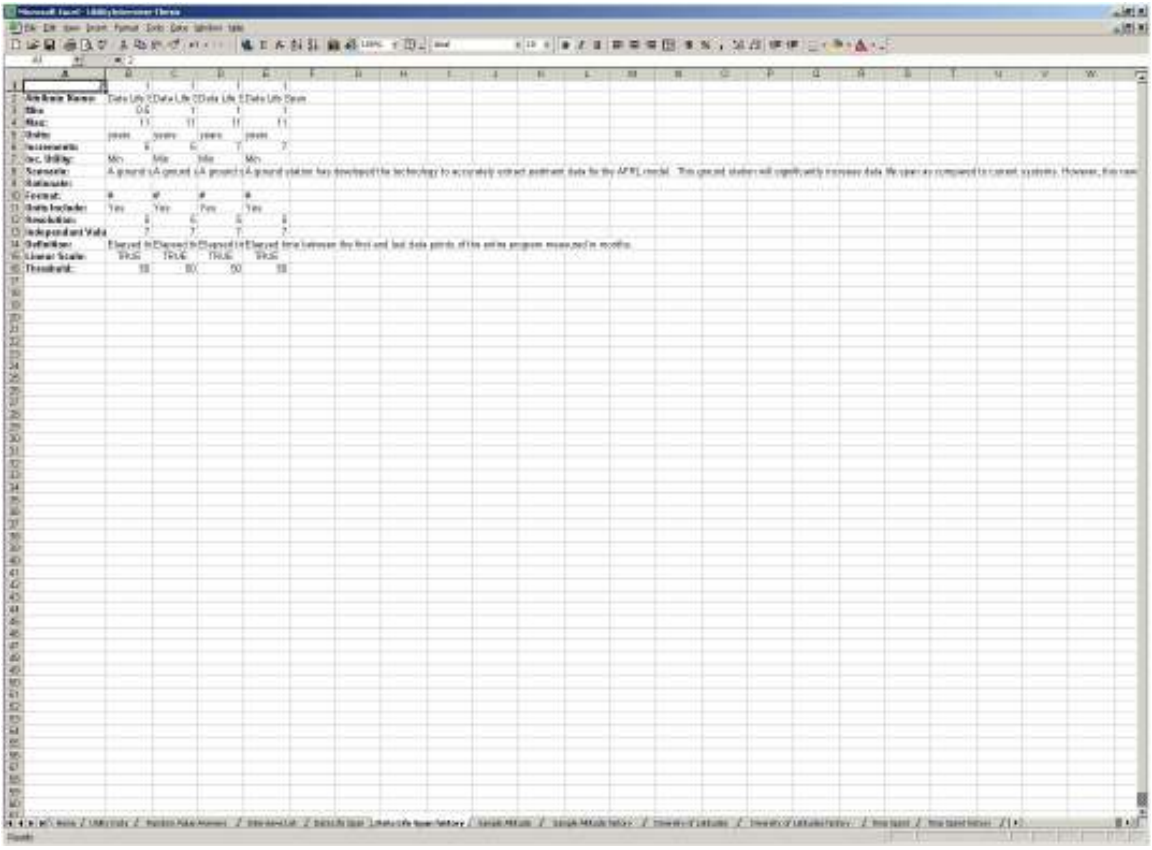
```
Sub UpdateControls()
    Select Case MultiPage1.Value
        Case 0
            BackButton.Enabled = False
            NextButton.Enabled = True
        Case MultiPage1.Pages.Count - 1
            BackButton.Enabled = True
            NextButton.Enabled = False
        Case Else
            BackButton.Enabled = True
            NextButton.Enabled = True
    End Select

'   Update the caption
    Me.Caption = APPNAME & " Step " _
      & MultiPage1.Value + 1 & " of " _
      & MultiPage1.Pages.Count

'   the Name field is required
    If tbName.Text = "" Then
        FinishButton.Enabled = False
    Else
        FinishButton.Enabled = True
    End If
End Sub
```

## *11.2 Data Storage Worksheets*

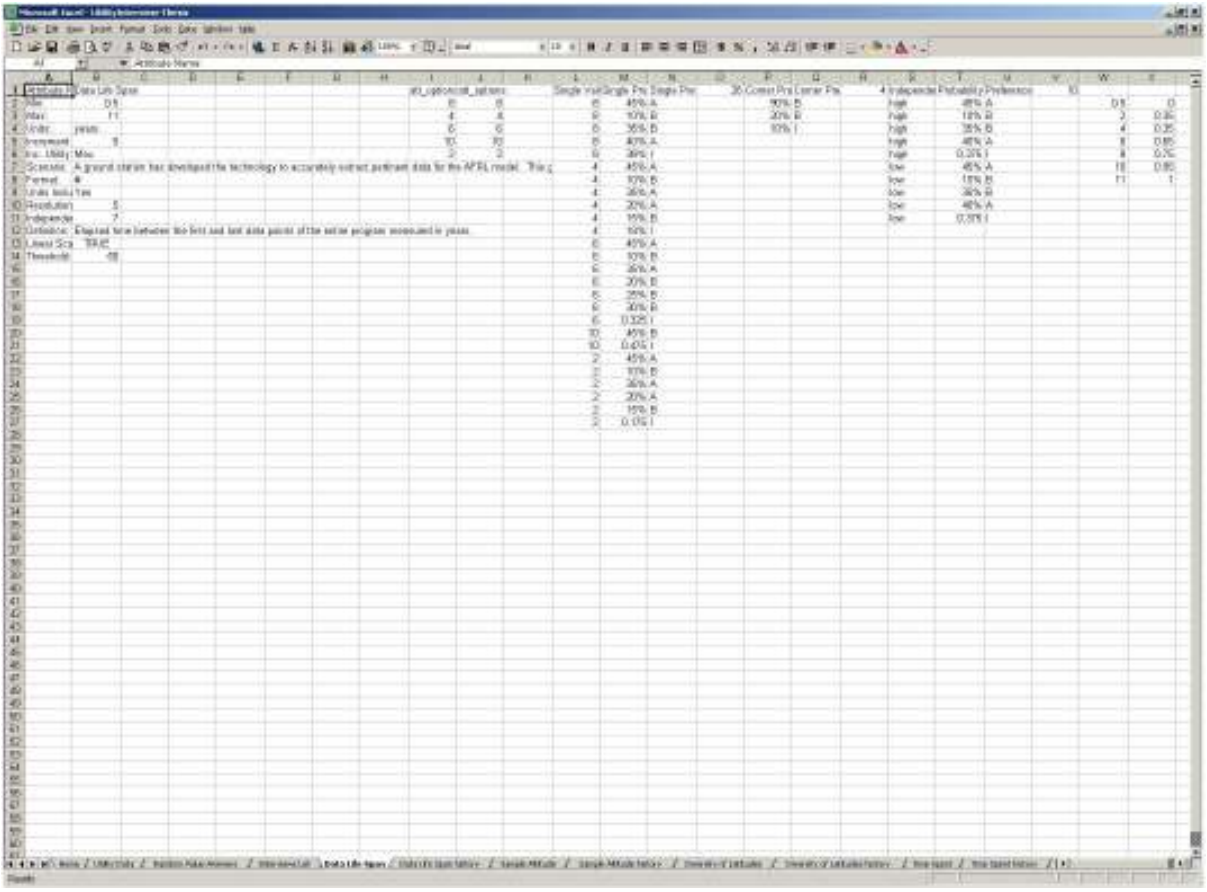
### 11.2.1 Main Interface



### 11.2.2 Attribute History and Rationale



### 11.2.3 Attribute Current State



## *11.3 Modules*

### 11.3.1 System Navigation

```
Sub Log_Start()
    Log_File = FreeFile
    LogFile.show
    Open LogFile.TextBoxFileName.Text For Random As #Log_File
End Sub
Sub SingleInterview_click()
    Interview_click ("single attribute interview")
End Sub

Sub CornerInterview_click()
    Interview_click ("corner point interview")
End Sub

Sub IndependanceInterview_click()
    Interview_click ("independance interview")
End Sub
Sub RandomGenerate_click()
    Load Random
    Random.show
    Unload Random
End Sub
Sub RandomInterview_click()
    Interview_click ("random interview")
End Sub
Sub IndependanceHigh_click()
    Interview_click ("independance high interview")
End Sub
Sub IndependanceLow_click()
    Interview_click ("independance low interview")
End Sub
Sub Delete_Start()

    AttNavOld.LabelInterview.Caption = "response deletion"
    AttNavOld.CommandButtonInterview.Caption = "Choose Interview"
    AttNavOld.show

End Sub

Sub Generate_Random_Values(Random_Sets As Integer)
    Dim Random_Set_Index As Integer, Current_Set As Integer, Current_Att As
Integer
    Dim Attribute_Value As Double
    Dim Steps As Integer
    Dim att_list As Range
    Dim Deleted_Set As Integer
    Dim Q_attribute As String

    Randomize

    Random_Set_Index = Sheets("Home").Cells(19, 16).Value
```

```
    Set att_list = Worksheets("Home").Range("Attribute_list")

    For Current_Set = 1 To Random_Sets
        Sheets("Home").Cells(20, 9 + Current_Set).Value = "Value"
        For Current_Att = 1 To att_list.Rows.Count
            Q_attribute = att_list.Cells(Current_Att, 1)
            Steps = Sheets(Q_attribute).Cells(5, 2).Value
            Attribute_Value = Sheets(Q_attribute).Cells(2 + Int(Rnd *
(Steps)), 10).Value
            Sheets("Home").Cells(20 + Current_Att, 9 + Current_Set).Value =
Attribute_Value
        Next Current_Att
    Next Current_Set

    If Random_Set_Index > Random_Sets Then
        For Deleted_Set = Random_Sets + 1 To Random_Set_Index
            For Current_Att = 0 To att_list.Rows.Count
                Sheets("Home").Cells(20 + Current_Att, 9 +
Deleted_Set).ClearContents
            Next Current_Att
        Next Deleted_Set
    End If

    Sheets("Home").Cells(19, 16).Value = Random_Sets

End Sub


Sub Display_AttNav()
Load AttNav
AttNav.show
Unload AttNav
End Sub
Sub Display_AttOpt()
Load AttOpt
AttOpt.show
Unload AttOpt
End Sub
Sub Display_AttProp()
Load AttProp
AttProp.show
Unload AttProp
End Sub
Sub Display_AttScen()
Load AttScen
AttScen.show
Unload AttScen
End Sub
Sub Display_AttSec()
Load AttSec
AttSec.show
Unload AttSec
End Sub
Sub Display_UtilInt()
Load UtilInt
UtilInt.show
Unload UtilInt
```

```
End Sub
'tara's addition
Sub Display_UtilIntII()
Load UtilIntII
UtilIntII.show
Unload UtilIntII
End Sub

Public Sub UtilIntII_Store_Choice()
    Dim Interview_index As Integer
    Dim Interview_Storage As Worksheet
    Dim Random_Values() As String

    Select Case Global_Choice
        Case "A"
            Global_Low = UtilIntII.LabelProbability.Caption
        Case "B"
            Global_High = UtilIntII.LabelProbability.Caption
        Case "I"
            Global_Low = "0"
            Global_High = "50"
    End Select

    'save changes somewhere!!
    If UtilIntII.LabelInterviewType.Caption = "Random Value Interview" Then
        Set Interview_Storage = Worksheets("Random Value Answers")
    Else
        Set Interview_Storage = Worksheets(UtilIntII.LabelAttribute.Caption)
    End If

    With Interview_Storage
        'Determine type of interview
        Select Case UtilIntII.LabelInterviewType.Caption
            Case "corner point interview"

                'Determine index of interview
                Interview_index = .Cells(1, 18).Value + 2
                .Cells(1, 18).Value = Interview_index - 1

                .Cells(Interview_index, 16).Value = UtilIntII.TextBoxPe.Text
                .Cells(Interview_index, 17).Value = Global_Choice
                Put #Log_File, , "Corner, " &
UtilIntII.LabelAttribute.Caption & ",X , " & UtilIntII.TextBoxPe.Text & ", "
& Global_Choice & ";"
            Case "Random Value Interview"
                Interview_index = .Cells(3, 1).Value + 1
                .Cells(3, 1).Value = Interview_index

                .Cells(1, Interview_index + 1).Value =
UtilIntII.TextBoxPe.Text
                .Cells(2, Interview_index + 1).Value = Global_Choice

                Put #Log_File, , "Random, " & CStr(UtilIntII.ListBox1.List(1,
1)) & ", X, " & UtilIntII.TextBoxPe.Text & ", " & Global_Choice & ";"
                'Put #Log_File, , UtilIntII.ListBox1.List

'                Random_Values = UtilIntII.ListBox1.List
```

```
'                .Cells(4, Interview_index + 1).Value = Random_Values

                Dim i As Integer
                Dim Number_of_Attributes As Integer
                Dim Attribute_List As Range

                Set Attribute_List =
ActiveWorkbook.Names("Attribute_list").RefersToRange
                Number_of_Attributes = Attribute_List.Rows.Count

                For i = 1 To Number_of_Attributes
                    .Cells(4 + i, Interview_index + 1).Value =
UtilIntII.ListBox1.List(i, 1)
                Next i

        End Select

    End With

End Sub

Public Sub UtilInt_Store_Choice()
    Dim Interview_Storage As Worksheet
    Dim Preference As String
    Dim Interview_index As Integer

    'Figure out choice
    'If TextBoxA.BackColor = &HFF00& Then
    '    Preference = "A"
    'ElseIf TextBoxB.BackColor = &HFF00& Then
    '    Preference = "B"
    'ElseIf TextBoxI.BackColor = &HFF00& Then
    '    Preference = "I"
    'End If

    Preference = UtilInt.LabelPreference.Caption

    Global_Choice = Preference

    'If no choice made, do not allow submit
    'If (TextBoxA.BackColor = &H80000005) And (TextBoxI.BackColor =
&H80000005) And (TextBoxB.BackColor = &H80000005) Then Exit Sub
    If Preference = "" Then Exit Sub

    Select Case Preference
        Case "A"
            Global_High = UtilInt.LabelProbability.Caption
        Case "B"
            Global_Low = UtilInt.LabelProbability.Caption
        Case "I"
            Global_Low = "0"
            Global_High = "50"
    End Select

    'save changes somewhere!!
    Set Interview_Storage = Worksheets(UtilInt.LabelAttribute.Caption)
    With Interview_Storage
```

```
        Select Case UtilInt.LabelInterviewType
            Case "single"
                'Determine index of interview
                Interview_index = .Cells(1, 15).Value + 2
                .Cells(1, 15).Value = Interview_index - 1

                .Cells(Interview_index, 12).Value =
CDbl(UtilInt.LabelCurrentValue.Caption)
                .Cells(Interview_index, 13).Value = UtilInt.TextBoxPe.Text
                .Cells(Interview_index, 14).Value = Preference
                Put #Log_File, , "Single, " & UtilInt.LabelAttribute.Caption
& ", " & UtilInt.LabelCurrentValue.Caption & ", " & UtilInt.TextBoxPe & ", "
& Preference & ";"
            Case "independence high"
                Interview_index = .Cells(1, 22).Value + 2
                .Cells(1, 22).Value = Interview_index - 1

                .Cells(Interview_index, 19).Value = "high"
                .Cells(Interview_index, 20).Value = UtilInt.TextBoxPe.Text
                .Cells(Interview_index, 21).Value = Preference
                Put #Log_File, , "Independence Low, " &
UtilInt.LabelAttribute.Caption & ", X, " & UtilInt.TextBoxPe & ", " &
Preference & ";"
            Case "independence low"
                Interview_index = .Cells(1, 22).Value + 2
                .Cells(1, 22).Value = Interview_index - 1

                .Cells(Interview_index, 19).Value = "low"
                .Cells(Interview_index, 20).Value = UtilInt.TextBoxPe.Text
                .Cells(Interview_index, 21).Value = Preference
                Put #Log_File, , "Independance High, " &
UtilInt.LabelAttribute.Caption & ", X, " & UtilInt.TextBoxPe & ", " &
Preference & ";"
        End Select

    End With

    'If Preference = "I" Then
    '    Call UpdateChart
    'End If


'    UtilInt.end_interview.Caption = 2
End Sub

Public Function dhXORText(strText As String, strPWD As String) As String
    'Encrypt or decrypt a string using the XOR operator.
    Dim abytText() As Byte
    Dim abytPWD() As Byte
    Dim intPWDPos As Integer
    Dim intPWDLen As Integer
    Dim intChar As Integer

    abytText = strText
    abytPWD = strPWD
    intPWDLen = LenB(strPWD)
    For intChar = 0 To LenB(strText) - 1
```

```
        'Get the next number between 0 and intPWD-1
        intPWDPos = (intChar Mod intPWDLen)
        abytText(intChar) = abytText(intChar) Xor abytPWD(intPWDPos)
    Next intChar
    dhXORText = abytText
End Function

Sub encrypt_cel()
    Dim input_str As String
    Dim output_str As String
    Dim password As String

    With Sheets("Home")
        input_str = .Range("D24").Value
        password = .Range("D23").Value
        output_str = dhXORText(input_str, password)
        .Range("D25").Value = output_str
    End With

End Sub

Sub Initialize_gui()


End Sub

Sub Mark_Completed(Q_attribute As String) 'SS
    Dim Attribute_Row As Integer
    Dim Att_Range As Range
    Dim numrows As Integer
    Dim i As Integer, Interview_Column As Integer

    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count

    For i = 8 To numrows + 7
        If Sheets("Home").Cells(i, 8).Value = Q_attribute Then Attribute_Row
= i
    Next i

    Select Case Global_Interview_Type
        Case "Single"
            Interview_Column = 10
        Case "Corner Point"
            Interview_Column = 12
        Case "high"
            Interview_Column = 14
        Case "low"
            Interview_Column = 15
    End Select

    Sheets("Home").Cells(Attribute_Row, Interview_Column).Value =
Sheets("Home").Cells(34, 16).Value

End Sub
Sub Mark_Deleted(Q_attribute As String)
    Dim Attribute_Row As Integer
```

```
    Dim Att_Range As Range
    Dim numrows As Integer
    Dim i As Integer, Interview_Column As Integer

    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count

    For i = 8 To numrows + 7
        If Sheets("Home").Cells(i, 8).Value = Q_attribute Then Attribute_Row
= i
    Next i

    Select Case Global_Interview_Type
        Case "Single"
            Interview_Column = 10
        Case "Corner Point"
            Interview_Column = 12
        Case "high"
            Interview_Column = 14
        Case "low"
            Interview_Column = 15
    End Select

    Sheets("Home").Cells(Attribute_Row, Interview_Column).Value = ""

End Sub
```

### 11.3.2 Attribute Modification

```
Attribute VB_Name = "Main_module"
Option Explicit

Type AttributeProp  'user-defined data type to hold attribute properties
    Name As String
    min As String
    max As String
    Units As String
    Increment As Integer
    Direction As Boolean
    Scenario As String
    Resolution As Double
    Format As String
    UnitsInc As Boolean
    Independent As Double
    Definition As String
    LinearScale As Boolean
    Threshold As Double
End Type

Sub modify_scenario_start()
    Dim att_sheet As Worksheet
    Dim att_index As Integer
    Dim currbox As Object
    Dim scene As String
```

```
    Unload AttScen
    Load AttScen
    If (Len(AttProp.TextBox1.Text) > 1) Then
        AttScen.Label2.Caption = AttProp.TextBox1.Text
    Else
        AttScen.Label2.Caption = "Unnamed Attribute"
    End If
    If AttProp.Label8.Caption = 0 Then
        Set att_sheet = Sheets(AttProp.TextBox1.Text)
        att_index = att_sheet.Index

        Set currbox = att_sheet.Shapes(Left(att_sheet.Name, 8) & "Box")
        scene = currbox.TextFrame.Characters.Text
        AttScen.RichTextBox1.Text = scene

        AttScen.RichTextBox1.SetFocus
    End If

    AttScen.show
End Sub

Attribute VB_Name = "Retrieve_Selection_module"
Function Retrieve_selection(attribute_choice As String) As AttributeProp
    Dim choice_sheet As Worksheet
    Dim current_attribute As AttributeProp
    Set choice_sheet = Worksheets(attribute_choice)

    With choice_sheet
        current_attribute.Name = .Range("B1") '.Name
        current_attribute.min = .Range("B2") '.Min
        current_attribute.max = .Range("B3") '.Max
        current_attribute.Units = .Range("B4") '.Units
        current_attribute.Increment = .Range("B5") 'Steps
        If .Range("B6") = "Min" Then
            current_attribute.Direction = 0 '.Direction
        Else
            current_attribute.Direction = 1
        End If
        current_attribute.Scenario = .Range("B7") 'Scenario
        current_attribute.Format = .Range("B8") 'Format
        If .Range("B9") = "Yes" Then
            current_attribute.UnitsInc = 1  'Units Include
        Else
            current_attribute.UnitsInc = 0
        End If
        current_attribute.Resolution = .Range("B10") 'Format
        current_attribute.Independent = .Range("B11") 'Independant Interview
Value
        current_attribute.Definition = .Range("B12")
        current_attribute.LinearScale = .Range("B13")
        current_attribute.Threshold = .Range("B14")

    End With

Retrieve_selection = current_attribute
End Function
```

```
Sub Refresh_all()
    Unload AttNavOld
    Unload AttOpt
    Unload AttProp
    Load AttNavOld
    Load AttOpt
    Load AttProp
End Sub
Attribute VB_Name = "Old_module"
Function Retrieve_selection_old(attribute_choice As String) As AttributeProp
    Dim choice_sheet As Worksheet
    Dim current_attribute As AttributeProp
    Set choice_sheet = Worksheets(attribute_choice)

    With choice_sheet
        current_attribute.Name = .Range("B1") '.Name
        current_attribute.min = .Range("B2") '.Min
        current_attribute.max = .Range("B3") '.Max
        current_attribute.Units = .Range("B4") '.Units
        current_attribute.Increment = .Range("B5") 'Steps
        If .Range("B6") = "Min" Then
            current_attribute.Direction = 0 '.Direction
        Else
            current_attribute.Direction = 1
        End If
        current_attribute.Scenario = .Range("B7") 'Scenario
    End With

Retrieve_selection = current_attribute
End Function

Sub Refresh_all_old()
    Unload AttNav
    Unload AttOpt
    Unload AttProp
    Load AttNav
    Load AttOpt
    Load AttProp

Attribute VB_Name = "Misc_module"
Sub show_attribute()
    Dim num As Integer
    Dim att_list As Range

    With Sheets("Home")
        num = .Range("A18")
        Set att_list = Worksheets("Home").Range("Attribute_list")
        .Range("B18").Value = att_list.Value(num, 1)
    End With

End Sub
Sub arename_sheet()
Attribute arename_sheet.VB_Description = "Macro recorded 8/10/2001 by Adam
Ross"
Attribute arename_sheet.VB_ProcData.VB_Invoke_Func = " \n14"
'
' arename_sheet Macro
```

```
' Macro recorded 8/10/2001 by Adam Ross
'

'
    Sheets("Sheet3").Select
    Sheets("Sheet3").Name = "LastOne"
End Sub
Sub aedit_text()
Attribute aedit_text.VB_Description = "Macro recorded 8/10/2001 by Adam Ross"
Attribute aedit_text.VB_ProcData.VB_Invoke_Func = " \n14"
'
' aedit_text Macro
' Macro recorded 8/10/2001 by Adam Ross
'

'
    ActiveSheet.Shapes("ScenarioBox4").Select
    selection.Characters.Text = "This is the answer I'm looking for!"
    With selection.Characters(Start:=1, Length:=35).Font
        .Name = "Arial"
        .FontStyle = "Regular"
        .Size = 10
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
    Range("I11").Select
End Sub

Function RandomizeList(NumberOfElements As Integer) As Integer()

    Dim OutputList() As Integer
    ReDim OutputList(0 To NumberOfElements)

    Dim i As Integer
    Dim RandElement As Integer
    Randomize

    For i = 1 To NumberOfElements
        OutputList(i) = 0
    Next i

    For i = 1 To NumberOfElements
        RandElement = Int(Rnd * NumberOfElements) + 1
        Do Until OutputList(RandElement) = 0
            RandElement = Int(Rnd * NumberOfElements) + 1
        Loop
        OutputList(RandElement) = i
    Next i

    RandomizeList = OutputList

End Function
```

### 11.3.3 Bracketing

```
Attribute VB_Name = "WDC_module"
Option Explicit
Public Global_Array() As Double
Public Global_Choice As String
Public Global_Hold As String
Public Indifferent_Value As Double
Public Allow_Observe As Boolean
Public Observe As Boolean
Public Output_File As String
Public Input_File As String
Public Output_Index As Integer
Public IntFile As Integer
Public CurrentInterviewForm As UserForm
Public Input_Index As Integer
Public Global_Interview_Type As String
Public Global_High As String
Public Global_Low As String
Public Val As Double
Public Indifferent_Index As Integer
Public Value_Index As Integer
Public IndifferentPointsAlreadyCollected As String
Public Log_File As Integer
Public Global_Value_Done As Boolean


Function Find_High(Q_attribute As String) As String

Dim inc_util As String
inc_util = Worksheets(Q_attribute).Cells(6, 2).Value

If inc_util = "Max" Then
Find_High = Worksheets(Q_attribute).Cells(3, 2).Value
Else
Find_High = Worksheets(Q_attribute).Cells(2, 2).Value
End If

Find_High = Format_Value(Q_attribute, Find_High)

End Function
Function Find_Low(Q_attribute As String) As String

Dim inc_util As String
inc_util = Worksheets(Q_attribute).Cells(6, 2).Value

If inc_util = "Max" Then
Find_Low = Worksheets(Q_attribute).Cells(2, 2).Value
Else
Find_Low = Worksheets(Q_attribute).Cells(3, 2).Value
End If

Find_Low = Format_Value(Q_attribute, Find_Low)
```

```
End Function
Function Format_Value(Q_attribute As String, Value As String) As String
    Dim Q_attribute_prop As AttributeProp
    Dim Units As String
    Dim Format As String
    Dim UnitsInc As Boolean
    Dim Formatted As String

    Q_attribute_prop = Retrieve_selection(Q_attribute)
    Units = Q_attribute_prop.Units
    Format = Q_attribute_prop.Format
    UnitsInc = Q_attribute_prop.UnitsInc

    Formatted = Replace(Format, "#", Value)

    If UnitsInc Then
        Formatted = Formatted & " " & Units
    Else
    End If

    Format_Value = Formatted

End Function
Sub Generate_Prob_UtilIntII(choice As String)

    Dim Q_attribute_prop As AttributeProp
    Dim Low As Double
    Dim high As Double
    Dim middle As Double
    Dim show As Double
    Dim Value As Double
    Dim Resolution As Double
    Dim InterviewForm As UserForm

'    low = Global_Array(0, 0)
'    high = Global_Array(0, 1)

    Low = CDbl(Global_Low)
    high = CDbl(Global_High)

'    InterviewForm = UtilIntII
'    UserForms.UtilIntII

    If UtilIntII.LabelInterviewType.Caption = "Random Value Interview" Then
        Resolution = 5
    Else
        Q_attribute_prop =
Retrieve_selection(UtilIntII.LabelAttribute.Caption)
        Resolution = Q_attribute_prop.Resolution
    End If

    If (high - Low) = Resolution Then
        Select Case Global_Choice
            Case "A"
'                If Global_Hold = "B" Then
                   'If low = 0 Then
'                       UtilIntII.Hide
```

```
'                       UtilIntII.end_interview.Caption = 1
                    '    Indifferent_Value = (low + high) / 2
                        Indifferent_Value = high - (Resolution / 2)
                        Indifferent_Start
                    'Else
                    '    UtilIntII.Hide
                    '    Indifferent_Value = low + 2.5
                    '    Indifferent_Start
                    'End If
'                Else
'                    UtilIntII.TextBoxPe.Text = high & "%"
'                    UtilIntII.TextBoxPeC.Text = (100 - high) & "%"
'                    Global_Array(0, 0) = high
'                    Global_Hold = "A"
'                End If
            Case "B"
'                If Global_Hold = "A" Then
'                    If high = 50 Then
'                        UtilIntII.Hide
'                        UtilIntII.end_interview.Caption = 1
'                        Indifferent_Value = (low + high) / 2
'                        Indifferent_Value = UtilIntII.TextBoxPe.Text - 2.5
                        Indifferent_Value = Low + (Resolution / 2)
                        Indifferent_Start
                    'Else
                    '    UtilIntII.Hide
                    '    Indifferent_Value = low - 2.5
                    '    Indifferent_Start
                    'End If
'                Else
'                    UtilIntII.TextBoxPe.Text = low & "%"
'                    UtilIntII.TextBoxPeC.Text = (100 - low) & "%"
'                    Global_Array(0, 1) = low
'                    Global_Hold = "B"
'                End If
            Case "I"
                UtilIntII.Hide
                UtilIntII.end_interview.Caption = 1
'                If Global_Hold = "A" Then
'                    Indifferent_Value = low
'                Else
'                    Indifferent_Value = high
'                End If
'
'                Indifferent_Value = UtilIntII.TextBoxPe.Text
'
'                Indifferent_Start
            Case Else
        End Select
    ElseIf high = Low Then
        Select Case Global_Choice
            Case "A"
                If Global_Hold = "B" Then
'                    UtilIntII.Hide
'                    UtilIntII.end_interview.Caption = 1
                    Indifferent_Value = high
                    Indifferent_Start
```

```
                Else
                    If Low = 0 Then
                        UtilIntII.Hide
                        UtilIntII.end_interview.Caption = 1
                        Error_Start
                    Else
                        UtilIntII.Hide
                        UtilIntII.end_interview.Caption = 1
                        Indifferent_Value = high + (Resolution / 2)
                        Indifferent_Start
                    End If
                End If
            Case "B"
                If Global_Hold = "A" Then
                    UtilIntII.Hide
                    UtilIntII.end_interview.Caption = 1
                    Indifferent_Value = high - (Resolution / 2)
                    Indifferent_Start
                Else
                    If high = 0 Then
                        UtilIntII.Hide
                        UtilIntII.end_interview.Caption = 1
                        Error_Start
                    Else
                        UtilIntII.Hide
                        UtilIntII.end_interview.Caption = 1
                        Indifferent_Value = high - (Resolution / 2)
                        Indifferent_Start
                    End If
                End If
            Case "I"
'                UtilIntII.Hide
                UtilIntII.end_interview.Caption = 1
'                Indifferent_Value = high
'                Indifferent_Start
            Case Else
        End Select
    Else

        If (Low = 0) And (high = 100) Then
            middle = 90
        Else
            If (Low = 0) And (high = 90) Then
                middle = 20
            Else
                If (Low = 20) And (high = 90) Then
                    middle = 70
                Else
                    If (Low = 20) And (high = 70) Then
                        middle = 40
                    Else
                        middle = (Low + high) / 2
                    End If
                End If
            End If
        End If
```

```
        middle = middle \ Resolution
        middle = middle * Resolution

        NextProbability.TextBoxProbability.Text = CStr(middle)

        Select Case Global_Choice
            Case "A"
'               middle = middle + Resolution
'               low = middle
'
''                If (high - low) > Resolution Then
 '                   show = (low + high) / 2
 '                   show = show \ Resolution
 '                   show = show * Resolution
 '                   UtilIntII.TextBoxPe.Text = show & "%"
 '                   UtilIntII.TextBoxPeC.Text = (100 - show) & "%"
 '                   Global_Array(0, 0) = low
 '               ElseIf (high - low) = Resolution Then
 '                   UtilIntII.TextBoxPe.Text = low & "%"
 '                   UtilIntII.TextBoxPeC.Text = (100 - low) & "%"
 '                   Global_Array(0, 0) = low
 '                   Global_Hold = "A"
 '               ElseIf low = high Then
 '                   UtilIntII.TextBoxPe.Text = low & "%"
 '                   UtilIntII.TextBoxPeC.Text = (100 - low) & "%"
 '                   Global_Array(0, 0) = low
                  Global_Hold = "A"
                'Else
                '    If Global_Hold = "B" Then
                '        UtilIntII.Hide
                '        Indifferent_Start
                '    Else
                '        UtilIntII.Hide
                '        Error_Start
                '    End If
 '              End If
            Case "B"
 '              middle = middle - Resolution
 '              high = middle
 '
 '              If (high - low) > Resolution Then
 '                  show = (low + high) / 2
 '                  show = show \ Resolution
 '                  show = show * Resolution
 '                  UtilIntII.TextBoxPe.Text = show & "%"
 '                  UtilIntII.TextBoxPeC.Text = (100 - show) & "%"
 '                  Global_Array(0, 1) = high
 '              ElseIf (high - low) = Resolution Then
 '                  UtilIntII.TextBoxPe.Text = high & "%"
 '                  UtilIntII.TextBoxPeC.Text = (100 - high) & "%"
 '                  Global_Array(0, 1) = high
 '                  Global_Hold = "B"
 '              ElseIf low = high Then
 '                  UtilIntII.TextBoxPe.Text = high & "%"
 '                  UtilIntII.TextBoxPeC.Text = (100 - high) & "%"
 '                  Global_Array(0, 1) = high
                  Global_Hold = "B"
```

```
                'Else
                '    If Global_Hold = "A" Then
                '        UtilIntII.Hide
                '        Indifferent_Start
                '    Else
                '        UtilIntII.Hide
                '        Error_Start
                '    End If
'                End If
            Case "I"
                UtilIntII.Hide
                UtilIntII.end_interview.Caption = 1
'                Indifferent_Value = (low + high) / 2
'                Indifferent_Start
            Case Else
'                UtilIntII.TextBoxPe.Text = middle & "%"
'                UtilIntII.TextBoxPeC.Text = (100 - middle) & "%"
'                Global_Array(0, 0) = 0
'                Global_Array(0, 1) = 50
        End Select
    End If

End Sub
Sub Generate_Prob_UtilInt(choice As String)

    Dim Q_attribute_prop As AttributeProp
    Dim Low As Double
    Dim high As Double
    Dim middle As Double
    Dim show As Double
    Dim Value As Double
    Dim Resolution As Double
    Dim InterviewForm As UserForm

'    low = Global_Array(0, 0)
'    high = Global_Array(0, 1)

    Low = CDbl(Global_Low)
    high = CDbl(Global_High)

    Q_attribute_prop = Retrieve_selection(UtilInt.LabelAttribute.Caption)
    Resolution = Q_attribute_prop.Resolution

    If (high - Low) = Resolution Then
        Select Case Global_Choice
            Case "A"
'                If Global_Hold = "B" Then
'                    UtilInt.Hide
'                    UtilInt.end_interview.Caption = 1
                   Indifferent_Value = high - (Resolution / 2)
                   Indifferent_Start_Single
'                Else
'                    UtilInt.TextBoxPe.Text = high & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - high) & "%"
'
'                    NextProbability.TextBoxProbability.Text = high
'
```

```
'                   Global_Array(0, 0) = high
'                   Global_Hold = "A"
'               End If
            Case "B"
'               If Global_Hold = "A" Then
'                       UtilInt.Hide
'                       UtilInt.end_interview.Caption = 1
                       Indifferent_Value = Low + (Resolution / 2)
                       Indifferent_Start_Single
'               Else
'                   UtilInt.TextBoxPe.Text = low & "%"
'                   UtilInt.TextBoxPeC.Text = (100 - low) & "%"
'
'                   NextProbability.TextBoxProbability.Text = low
'
'                   Global_Array(0, 1) = low
'                   Global_Hold = "B"
'               End If
            Case "I"
                UtilInt.Hide
                UtilInt.end_interview.Caption = 1
            Case Else
        End Select
    ElseIf high = Low Then
        Select Case Global_Choice
            Case "A"
                If Global_Hold = "B" Then
'                    Indifferent_Value = high + (Resolution / 2)
                    Indifferent_Value = high

                    Indifferent_Start_Single
                Else
                    If high = 0 Then
                        Error_Start
                    Else
'                        Indifferent_Value = high + (Resolution / 2)
                        Indifferent_Value = high

                        Indifferent_Start_Single
                    End If
                End If
            Case "B"
                If Global_Hold = "A" Then
'                    Indifferent_Value = high - (Resolution / 2)
                    Indifferent_Value = high

                    Indifferent_Start_Single
                Else
                    If high = 50 Then
                        Error_Start
                    Else
'                        Indifferent_Value = high - (Resolution / 2)
                        Indifferent_Value = high

                        Indifferent_Start_Single
                    End If
                End If
```

```
            Case "I"
                UtilInt.Hide
                UtilInt.end_interview.Caption = 1
            Case Else
        End Select
    Else
        If (Low = 0) And (high = 50) Then
            middle = 45
        Else
            If (Low = 0) And (high = 45) Then
                middle = 10
            Else
                If (Low = 10) And (high = 45) Then
                    middle = 35
                Else
                    If (Low = 10) And (high = 35) Then
                        middle = 20
                    Else
                        middle = (Low + high) / 2
                    End If
                End If
            End If
        End If

        middle = middle \ Resolution
        middle = middle * Resolution

        NextProbability.TextBoxProbability.Text = CStr(middle)

        Select Case Global_Choice
            Case "A"
                Global_Hold = "A"

'                middle = middle + Resolution
'                low = middle
'
'                If (high - low) > Resolution Then
'                    show = (low + high) / 2
'                    show = show \ Resolution
'                    show = show * Resolution
'                    UtilInt.TextBoxPe.Text = show & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - show) & "%"
'                    Global_Array(0, 0) = low
'                ElseIf (high - low) = Resolution Then
'                    UtilInt.TextBoxPe.Text = low & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - low) & "%"
'                    Global_Array(0, 0) = low
'                    Global_Hold = "A"
'                ElseIf low = high Then
'                    UtilInt.TextBoxPe.Text = low & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - low) & "%"
'                    Global_Array(0, 0) = low
'                    Global_Hold = "A"
'                End If
            Case "B"
                Global_Hold = "B"
```

```
'                middle = middle - Resolution
'                high = middle
'
'                If (high - low) > Resolution Then
'                    show = (low + high) / 2
'                    show = show \ Resolution
'                    show = show * Resolution
'                    UtilInt.TextBoxPe.Text = show & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - show) & "%"
'                    Global_Array(0, 1) = high
'                ElseIf (high - low) = Resolution Then
'                    UtilInt.TextBoxPe.Text = high & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - high) & "%"
'                    Global_Array(0, 1) = high
'                    Global_Hold = "B"
'                ElseIf low = high Then
'                    UtilInt.TextBoxPe.Text = high & "%"
'                    UtilInt.TextBoxPeC.Text = (100 - high) & "%"
'                    Global_Array(0, 1) = high
'                    Global_Hold = "B"
'                End If
            Case "I"
                UtilInt.Hide
                UtilInt.end_interview.Caption = 1
            Case Else
'                UtilInt.TextBoxPe.Text = middle & "%"
'                UtilInt.TextBoxPeC.Text = (100 - middle) & "%"
'                Global_Array(0, 0) = 0
'                Global_Array(0, 1) = 50
        End Select
    End If

End Sub
Sub Error_Start()
    Dim Change As Integer

    Change = MsgBox("The values selected so far will result in an indifferent
value outside of the current min and max values for " &
UtilIntII.LabelAttribute.Caption & ".  Would you like to redo the Multi-
Attribute Interview?", vbYesNo + vbExclamation, "Error")

    If Allow_Observe = True Then    'WDC, allow observe
        Put #IntFile, Output_Index, "Change = " & Change
        Output_Index = Output_Index + 1
    End If

    If Change = vbYes Then
        IndifferentError.Hide
        Util_InterviewIIA_Start (AttNavOld.ListBoxAttributes.Value)
    Else
        IndifferentError.Hide
        AttNavOld.show
    End If
'    IndifferentError.Label1.Caption = "The values selected so far result in
an indifferent value outside of the current min and max values for " &
UtilIntII.LabelAttribute.Caption & ".  Please redo or exit."
'    IndifferentError.show
```

```
End Sub
Sub Error_Start_Single()
    Dim Change As Integer

    Change = MsgBox("The values selected so far will result in an indifferent
value outside of the current min and max values for " &
UtilInt.LabelAttribute.Caption & ".  Would you like to redo the Single
Attribute Interview?", vbYesNo + vbExclamation, "Error")

    If Allow_Observe = True Then    'WDC, allow observe
        Put #IntFile, Output_Index, "Change = " & Change
        Output_Index = Output_Index + 1
    End If

    If Change = vbYes Then
        IndifferentError.Hide
'        Util_InterviewIIA_Start (AttNavOld.ListBoxAttributes.Value)
    Else
        IndifferentError.Hide
'        AttNavOld.show
    End If
'    IndifferentError.Label1.Caption = "The values selected so far result in
an indifferent value outside of the current min and max values for " &
UtilIntII.LabelAttribute.Caption & ".  Please redo or exit."
'    IndifferentError.show
End Sub
Sub Indifferent_Start()
    Dim Change As Integer

    Change = MsgBox("Please confirm that the indifferent value for " &
UtilIntII.LabelAttribute.Caption & " is " & Indifferent_Value & "%", vbYesNo,
"Confirm Indifferent Value")

    If Allow_Observe = True Then    'WDC, allow observe
        Put #IntFile, Output_Index, "Change = " & Change
        Output_Index = Output_Index + 1
    End If

    If Change = vbYes Then
        Global_Choice = "I"
        UtilIntII.TextBoxPe.Text = CStr(Indifferent_Value) + "%"
        UtilIntII_Store_Choice
        UtilIntII.Hide
        IndifferentConfirm.Hide
'        AttNavOld.show
    Else
        IndifferentConfirm.Hide
        Util_InterviewIIA_Start (AttNavOld.ListBoxAttributes.Value)
    End If
End Sub
Sub Indifferent_Start_Single()
    Dim Change As Integer

    Change = MsgBox("Please confirm that the indifferent value for " &
UtilInt.LabelAttribute.Caption & " is " & Indifferent_Value & "%", vbYesNo,
"Confirm Indifferent Value")
```

```
    If Allow_Observe = True Then    'WDC, allow observe
        Put #IntFile, Output_Index, "Change = " & Change
        Output_Index = Output_Index + 1
    End If

    If Change = vbYes Then
        UtilInt.LabelPreference.Caption = "I"
        UtilInt.TextBoxPe = Indifferent_Value / 100
'        If UtilInt.LabelInterviewType = "single" Then
        UtilInt_Store_Choice
'        End If
        IndifferentConfirm.Hide
'        AttNavOld.show
    Else
        IndifferentConfirm.Hide
'        Util_InterviewIIA_Start (AttNavOld.ListBoxAttributes.Value)
    End If
End Sub

Sub Initialize_Bracketing()
    ReDim Global_Array(0 To 2, 0 To 2)
    Global_Array(0, 0) = 0
    Global_Choice = "nil"
    If Global_Interview_Type = "Corner Point" Then
        Global_Array(0, 1) = 100
        Global_High = "100"
    Else
        Global_Array(0, 1) = 50
        Global_High = "50"
    End If
    Global_Low = "0"
End Sub

Sub Initialize_Bracketing2(Att As String, Interview As String)
    Dim Attribute_Row As Integer
    Dim Att_Range As Range
    Dim numrows As Integer
    Dim i As Integer
    Dim Q_attribute As String
    Dim Interview_Type As String

    Q_attribute = "Spatial Resolution"
    Interview_Type = "Single"

    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count

    For i = 9 To numrows + 8
        If Sheets("Home").Cells(i, 8).Value = Q_attribute Then Attribute_Row
= i
    Next i

    Select Case Interview_Type
        Case "Single"
            Sheets("Home").Cells(Attribute_Row, 10).Value =
Sheets("Home").Cells(30, 16).Value
    End Select
```

```
End Sub


Function Request_Value(Q_attribute As String, Q_Value As Double, Value_Index
As Integer, Number_of_Values As Integer) As Integer
    NextValue.LabelQuestion.Caption = "What is the next value of " +
Q_attribute + " you would like to find an indifferent point for?"
    NextValue.LabelAttribute.Caption = Q_attribute
    If Value_Index > Number_of_Values Then
        NextValue.TextBoxValue.Text = "end"
    Else
        NextValue.TextBoxValue.Text = CStr(Q_Value)
    End If
    If AttNavOld.CheckBoxOverride Then NextValue.show
    Request_Value = 0
End Function
```

### 11.3.4 Interviews

```
Sub Util_Interview_Start(Q_attribute As String)
    Dim curr_prob As Double, curr_Xi As Double, curr_value As Double
    Dim X_best As Double, X_worst As Double
    Dim Q_attribute_prop As AttributeProp
    Dim prob_index As Integer, Value_Index As Integer, AttributeDataPoints As
Integer

'    UtilInt.TextBoxScenario.Text =
Sheets(Q_attribute).Shapes(Left(Q_attribute, 8) &
"Box").TextFrame.Characters.Text

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    UtilInt.TextBoxScenario.Text = Q_attribute_prop.Scenario

    If Not (Q_attribute_prop.Direction) Then
        X_best = Q_attribute_prop.min
        X_worst = Q_attribute_prop.max
    Else
        X_best = Q_attribute_prop.max
        X_worst = Q_attribute_prop.min
    End If

    UtilInt.TextBoxHigh.Text = Format_Value(Q_attribute, X_best)
    UtilInt.TextBoxLow1.Text = Format_Value(Q_attribute, X_worst)
    UtilInt.TextBoxLow2.Text = Format_Value(Q_attribute, X_worst)

    'Create graph
    'UtilInt.ChartSpaceUtility.HasChartSpaceTitle = True
    'UtilInt.ChartSpaceUtility.ChartSpaceTitle.Caption =
UtilInt.LabelAttribute.Caption + " Utility"
    '
    'AttributeDataPoints = Sheets(Q_attribute).Cells(1, 13).Value
    'UtilInt.ChartSpaceUtility.Charts(0).Type = chChartTypeScatterSmoothLine
```

```
    'UtilInt.ChartSpaceUtility.Charts(0).SetData chDimXValues, 0,
Sheets(Q_attribute).Range("L4..L7")

'    ActiveChart.ChartType = xlXYScatterSmooth
'    ActiveChart.SetSourceData Source:=Sheets(Q_attribute).Range("L3:M7")

'    Type = chChartTypeSmoothLineMarkers
'    UtilInt.ChartSpaceUtility.Charts(0).SetData
chDimXValues:=Sheets(Q_attribute).Range("L4..L7")
'    UtilInt.ChartSpaceUtility.Charts(0).SetData
chDimYValues:=Sheets(Q_attribute).Range("M4..M7")

'Pick initial probability to ask as a multiple of 5 between 0 and 50

    UtilInt.show
    Randomize
    curr_prob = Int((9 * Rnd) + 1) * 5


'    prob_index = 2
    Value_Index = 2

'    curr_prob = Sheets(Q_attribute).Cells(prob_index, 9).Value * 100
'    curr_value = Sheets(Q_attribute).Cells(value_index, 10).Value

'    UtilInt.TextBoxPe.Text = curr_prob & "%"
'    UtilInt.TextBoxPeC.Text = (100 - curr_prob) & "%"
'    UtilInt.TextBoxCurr.Text = curr_value
'
'    UtilInt.LabelPreference.Caption = ""
'    UtilInt.Show
'    UtilInt.Choice_A.SetFocus

    Do While Not (UtilInt.end_interview.Caption = 1)
        If UtilInt.end_interview.Caption = 2 Then
'            prob_index = prob_index + 1
            curr_value = Sheets(Q_attribute).Cells(Value_Index, 10).Value

            UtilInt.show


            If curr_value = 0 Then
                UtilInt.end_interview.Caption = 1
            Else
                If UtilInt.LabelPreference.Caption = "I" Then
                    Value_Index = Value_Index + 1
'                    prob_index = 2
                Else
                    curr_prob = Int((9 * Rnd) + 1) * 5
                    UtilInt.TextBoxPe.Text = curr_prob & "%"
                    UtilInt.TextBoxPeC.Text = (100 - curr_prob) & "%"
                    UtilInt.TextBoxCurr.Text = Format_Value(Q_attribute,
curr_value)
                    UtilInt.end_interview.Caption = 0
                    UtilInt.LabelPreference.Caption = ""
                    UtilInt.show
                    UtilInt.Choice_A.SetFocus
```

```
                End If
            End If
        End If
    Loop
    AttNav.show
End Sub

'Tara's Addition
Sub Util_InterviewII_Start()
    Dim curr_prob As Double, X_best As Double, X_worst As Double
    Dim Q_attribute_prop As AttributeProp
    Dim prob_index As Integer, AttributeDataPoints As Integer

    'UtilIntII.TextBoxScenario.Text =
Sheets(Q_attribute).Shapes(Left(Q_attribute, 8) &
"Box").TextFrame.Characters.Text

    'Q_attribute_prop = Retrieve_selection(Q_attribute)

    'If Not (Q_attribute_prop.Direction) Then
    '    X_best = Q_attribute_prop.Min
    '    X_worst = Q_attribute_prop.Max
    'Else
    '    X_best = Q_attribute_prop.Max
    '    X_worst = Q_attribute_prop.Min
    'End If

    'UtilIntII.TextBoxHigh.Text = X_best
    'UtilIntII.TextBoxLow1.Text = X_worst
    'UtilIntII.TextBoxLow2.Text = X_worst
    prob_index = 2
    'value_index = 2

    curr_prob = Sheets("Att_Name&Val").Cells(9, prob_index).Value * 100
    'curr_value = Sheets(Q_attribute).Cells(value_index, 10).Value

    UtilIntII.TextBoxPe.Text = curr_prob & "%"
    UtilIntII.TextBoxPeC.Text = (100 - curr_prob) & "%"

'tara's additions
    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_Range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count 'numrows contains the number of Attributes

    Dim Array_A() As String
    Dim Array_BI() As String
    Dim Array_BII() As String
    ReDim Array_A(0 To numrows + 1, 0 To 2)
    ReDim Array_BI(0 To numrows + 1, 0 To 2)
    ReDim Array_BII(0 To numrows + 1, 0 To 2)

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"
    Array_BI(0, 0) = "Attribute Name"
```

```
    Array_BI(0, 1) = "Attribute Value"
    Array_BII(0, 0) = "Attribute Name"
    Array_BII(0, 1) = "Attribute Value"

    Dim att_name As String
    Dim a As Integer
    'for ListBox1
    a = 0

    Dim column_index As Integer
    column_index = 2

    Do While a <= numrows
        Array_A(a + 1, 0) = Worksheets("Att_Name&Val").Cells(a + 2, 1).Value
        Array_A(a + 1, 1) = Worksheets("Att_Name&Val").Cells(a + 2,
column_index).Value
        a = a + 1
    Loop

    'for ListBox2,3
    Dim b, rowindex As Integer
    b = 0
    rowindex = 12

    Do While b <= numrows
        Array_BI(b + 1, 0) = Worksheets("Att_Name&Val").Cells(b + 2, 1).Value
        Array_BI(b + 1, 1) = Worksheets("Att_Name&Val").Cells(rowindex,
2).Value
        Array_BII(b + 1, 0) = Worksheets("Att_Name&Val").Cells(b + 2,
1).Value
        Array_BII(b + 1, 1) = Worksheets("Att_Name&Val").Cells(rowindex,
3).Value
        rowindex = rowindex + 1
        b = b + 1
        Loop

    UtilIntII.ListBox1.List = Array_A
    UtilIntII.ListBox2.List = Array_BI
    UtilIntII.ListBox3.List = Array_BII
    'end additions

    'Create graph
    'UtilIntII.ChartSpaceUtility.HasChartSpaceTitle = True
    'UtilIntII.ChartSpaceUtility.ChartSpaceTitle.Caption =
UtilIntII.LabelAttribute.Caption + " Utility"

    'AttributeDataPoints = Sheets(Q_attribute).Cells(1, 13).Value
    'UtilIntII.ChartSpaceUtility.Charts(0).Type =
chChartTypeScatterSmoothLine
    'UtilIntII.ChartSpaceUtility.Charts(0).SetData chDimXValues, 0,
Sheets(Q_attribute).Range("L4..L7")

'    ActiveChart.ChartType = xlXYScatterSmooth
'    ActiveChart.SetSourceData Source:=Sheets(Q_attribute).Range("L3:M7")

'    Type = chChartTypeSmoothLineMarkers
```

```
'    UtilIntII.ChartSpaceUtility.Charts(0).SetData
chDimXValues:=Sheets(Q_attribute).Range("L4..L7")
'    UtilIntII.ChartSpaceUtility.Charts(0).SetData
chDimYValues:=Sheets(Q_attribute).Range("M4..M7")

    UtilIntII.show
    UtilIntII.Choice_A.SetFocus

'    Do While 1 = 1
'        i = i + 1
'    Loop

    Do While Not (UtilIntII.end_interview.Caption = 1)
        If UtilIntII.end_interview.Caption = 2 Then ' have not reached end of
prob, increment prob
            prob_index = prob_index + 1
            curr_prob = Sheets("Att_Name&Val").Cells(9, prob_index).Value *
100
            'curr_value = Sheets(Q_attribute).Cells(value_index, 10).Value

            If Sheets("Att_Name&Val").Cells(1, column_index).Value = "" Then
'have reached end of value, end
                UtilIntII.end_interview.Caption = 1
            Else
                If curr_prob = 0 Then
                    'value_index = value_index + 1
                    column_index = column_index + 1
                    prob_index = 1
                Else
                'for ListBox1
                a = 0
                    Do While a <= numrows
                        Array_A(a + 1, 0) =
Worksheets("Att_Name&Val").Cells(a + 2, 1).Value
                        Array_A(a + 1, 1) =
Worksheets("Att_Name&Val").Cells(a + 2, column_index).Value
                        a = a + 1
                    Loop

                UtilIntII.ListBox1.List = Array_A
                UtilIntII.TextBoxPe.Text = curr_prob & "%"
                UtilIntII.TextBoxPeC.Text = (100 - curr_prob) & "%"
                'UtilIntII.TextBoxCurr.Text = curr_value
                UtilIntII.end_interview.Caption = 0
                UtilIntII.show
                UtilIntII.Choice_A.SetFocus
                End If
          End If
        End If
    Loop
End Sub


Sub RandomInterview_start()
    Dim Number_of_Interviews As Integer
    Dim Number_of_Attributes As Integer
    Dim Attribute_List As Range
```

```
    Dim Interview_index As Integer

    Number_of_Interviews = Sheets("Home").Cells(19, 16).Value
    Set Attribute_List = ActiveWorkbook.Names("Attribute_list").RefersToRange
    Number_of_Attributes = Attribute_List.Rows.Count

    Dim Array_A() As String
    Dim Array_BI() As String
    Dim Array_BII() As String
    ReDim Array_A(0 To Number_of_Attributes + 1, 0 To 2)
    ReDim Array_BI(0 To Number_of_Attributes + 1, 0 To 2)
    ReDim Array_BII(0 To Number_of_Attributes + 1, 0 To 2)

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"
    Array_BI(0, 0) = "Attribute Name"
    Array_BI(0, 1) = "Attribute Value"
    Array_BII(0, 0) = "Attribute Name"
    Array_BII(0, 1) = "Attribute Value"

    Dim a As Integer

    For a = 1 To Number_of_Attributes
        Array_BI(a, 0) = Attribute_List(a, 1)
        Array_BI(a, 1) = Find_High(Attribute_List(a, 1))
        Array_BII(a, 0) = Attribute_List(a, 1)
        Array_BII(a, 1) = Find_Low(Attribute_List(a, 1))
    Next a

    UtilIntII.ListBox2.List = Array_BI
    UtilIntII.ListBox3.List = Array_BII

    UtilIntII.LabelInterviewType.Caption = "Random Value Interview"

    For Interview_index = 1 To Number_of_Interviews
        For a = 1 To Number_of_Attributes
            Array_A(a, 0) = Attribute_List(a, 1)
            Array_A(a, 1) = Format_Value(Attribute_List(a, 1),
CStr(Sheets("Home").Cells(20 + a, 9 + Interview_index).Value))
        Next a
        UtilIntII.ListBox1.List = Array_A

        Initialize_Bracketing

        Do While Not (Global_Choice = "I" Or UtilIntII.end_interview.Caption
= 1)
            Generate_Prob_UtilIntII (Global_Choice)

            If Global_Choice <> "I" Then
                NextProbability.TextBoxMaxProbability.Text = Global_High +
"%"
                NextProbability.TextBoxMinProbability.Text = Global_Low + "%"
                If AttNavOld.CheckBoxOverride Then NextProbability.show
                If NextProbability.TextBoxProbability.Text <> "end" Then
                    UtilIntII.TextBoxPe.Text =
NextProbability.TextBoxProbability.Text + "%"
```

```
                    UtilIntII.TextBoxPeC.Text = CStr(100 -
CDbl(NextProbability.TextBoxProbability.Text)) + "%"
                    UtilIntII.LabelProbability.Caption =
NextProbability.TextBoxProbability.Text

                    UtilIntII.end_interview.Caption = 0
                    UtilIntII.show
                    UtilIntII.Choice_A.SetFocus
                End If
            End If
        Loop
        Sheets("Home").Cells(20, 9 + Interview_index).Value =
UtilIntII.TextBoxPe.Text
    Next Interview_index

End Sub

Function Value_Done(Q_attribute As String, Val As Double) As Boolean

    Dim VDone As Boolean
    VDone = False

    With Sheets(Q_attribute)

        Dim i As Integer
        i = 2

        Dim Pref As String
        Pref = .Cells(i, 14).Value

        Do While Pref > ""
            If Pref = "I" Then
                If .Cells(i, 12).Value = Val Then
                    VDone = True
                End If
            End If
            i = i + 1
            Pref = .Cells(i, 14).Value
        Loop

    End With

    Global_Value_Done = VDone
    Value_Done = VDone

End Function

Sub Interview_click(InterviewType As String)
Dim Increment As Integer
Dim curr_value As Integer
Dim curr_gui As Object

Increment = 1

Load UtilIntII 'tara's addition
Load UtilInt
Load AttNavOld
```

```
Load AttOpt
Load AttProp
Load AttRationale
Load AttSec

'Need initialization procedueres here
Initialize_gui

Select Case InterviewType
    Case "random interview"
        RandomInterview_start
    Case "curve generation"
        AttNavOld.ListBoxAttributes.ListIndex = 0
        AttNavOld.LabelInterview.Caption = InterviewType
        AttNavOld.CommandButtonInterview.Caption = "Generate curve"
        AttNavOld.show
        AttNavOld.CommandButtonInterview.Caption = "Interview"
    Case Else
        AttNavOld.ListBoxAttributes.ListIndex = 0
        AttNavOld.LabelInterview.Caption = InterviewType
        AttNavOld.show
End Select

Unload UtilIntII 'tara's addition
Unload UtilInt
Unload AttNavOld
Unload AttOpt
Unload AttProp
Unload AttRationale
Unload AttSec

End Sub
Sub Independance_Interview_High_Start(Q_attribute As String)
    Global_Interview_Type = "high"
    UtilInt.LabelInterviewType.Caption = "independence high"
    Independance_Interview_Start (Q_attribute)
End Sub
Sub Independance_Interview_Low_Start(Q_attribute As String)
    Global_Interview_Type = "low"
    UtilInt.LabelInterviewType.Caption = "independence low"
    Independance_Interview_Start (Q_attribute)
End Sub
Sub Independance_Interview_Start(Q_attribute As String)
    Dim curr_prob As Double, curr_Xi As Double, curr_value As String
    Dim X_best As String, X_worst As String
    Dim Q_attribute_prop As AttributeProp
    Dim prob_index As Integer, Value_Index As Integer, AttributeDataPoints As
Integer
    Dim Val As Double
    Dim att_list As Range

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    Dim ArrayFixedValues() As String
    Dim num_atts As Integer

    Set att_list = ActiveWorkbook.Names("Attribute_list").RefersToRange
```

```
    num_atts = att_list.Rows.Count

    ReDim ArrayFixedValues(0 To num_atts, 0 To 2)

    Dim i As Integer
    Dim Att As String
    For i = 1 To num_atts
        Att = att_list.Cells(i, 1)
        If Att <> Q_attribute Then
            ArrayFixedValues(i, 0) = Att
            Select Case Global_Interview_Type
                Case "high"
                    ArrayFixedValues(i, 1) = Find_High(Att)
                Case "low"
                    ArrayFixedValues(i, 1) = Find_Low(Att)
            End Select
        End If
    Next i

    Dim ScenarioAddition As String

    ScenarioAddition = "Your design team has studied the issue. They indicate
that both technologies will give you"

    For i = 1 To num_atts
        If ArrayFixedValues(i, 0) > "" Then
            ScenarioAddition = ScenarioAddition + " a " + ArrayFixedValues(i,
0) + " of " + ArrayFixedValues(i, 1) + ","
        End If
    Next i

    UtilInt.TextBoxScenario.Text = Q_attribute_prop.Scenario
    UtilInt.LabelDefinition.Caption = "Other Attributes"
    UtilInt.TextBoxDefinition.Text = ScenarioAddition

    X_best = Find_High(Q_attribute)
    X_worst = Find_Low(Q_attribute)

    UtilInt.TextBoxHigh.Text = X_best
    UtilInt.TextBoxLow1.Text = X_worst
    UtilInt.TextBoxLow2.Text = X_worst

    Initialize_Bracketing
    Generate_Prob_UtilInt (Global_Choice)

    Val = Q_attribute_prop.Independent

    UtilInt.TextBoxCurr.Text = Format_Value(Q_attribute, CStr(Val))
    UtilInt.LabelCurrentValue.Caption = Val

    Do While UtilInt.LabelPreference.Caption <> "I"
        UtilInt.LabelPreference.Caption = "none"

        NextProbability.TextBoxMaxProbability.Text = Global_High + "%"
        NextProbability.TextBoxMinProbability.Text = Global_Low + "%"
        If AttNavOld.CheckBoxOverride Then NextProbability.show
        If NextProbability.TextBoxProbability.Text <> "end" Then
```

```
            UtilInt.TextBoxPe.Text = NextProbability.TextBoxProbability.Text
+ "%"
            UtilInt.TextBoxPeC.Text = CStr(100 -
CDbl(NextProbability.TextBoxProbability.Text)) + "%"
            UtilInt.LabelProbability.Caption =
NextProbability.TextBoxProbability.Text

            UtilInt.show
            UtilInt.Choice_A.SetFocus
        End If
        Generate_Prob_UtilInt (Global_Choice)
    Loop

    UtilInt.LabelPreference.Caption = "none"
    Mark_Completed (Q_attribute)

    AttNavOld.show

End Sub

'WDC's Addition, will replace current Util_InterviewII_Start
Sub Util_InterviewIIA_Start(Q_attribute As String)
    Dim curr_prob As Double, X_best As Double, X_worst As Double
    Dim Q_attribute_prop As AttributeProp
    Dim prob_index As Integer, AttributeDataPoints As Integer

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    UtilIntII.LabelInterviewType.Caption = "corner point interview"

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_Range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_Range.Rows.Count 'numrows contains the number of Attributes

    Dim Array_att(6) As String
    Dim i As Integer
    i = 0

    Do While i < numrows
        Array_att(i) = Att_Range(i + 1).Value
        i = i + 1
    Loop

    Dim Array_A() As String
    Dim Array_BI() As String
    Dim Array_BII() As String
    ReDim Array_A(0 To numrows + 1, 0 To 2)
    ReDim Array_BI(0 To numrows + 1, 0 To 2)
    ReDim Array_BII(0 To numrows + 1, 0 To 2)

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"
    Array_BI(0, 0) = "Attribute Name"
    Array_BI(0, 1) = "Attribute Value"
```

```
    Array_BII(0, 0) = "Attribute Name"
    Array_BII(0, 1) = "Attribute Value"

    Dim a As Integer
    a = 0

    Do While a < numrows
        If Q_attribute = Array_att(a) Then
        Array_A(a + 1, 0) = Array_att(a)
        Array_A(a + 1, 1) = Find_High(Array_att(a))
        Array_BI(a + 1, 0) = Array_att(a)
        Array_BI(a + 1, 1) = Find_High(Array_att(a))
        Array_BII(a + 1, 0) = Array_att(a)
        Array_BII(a + 1, 1) = Find_Low(Array_att(a))
        a = a + 1
        Else
        Array_A(a + 1, 0) = Array_att(a)
        Array_A(a + 1, 1) = Find_Low(Array_att(a))
        Array_BI(a + 1, 0) = Array_att(a)
        Array_BI(a + 1, 1) = Find_High(Array_att(a))
        Array_BII(a + 1, 0) = Array_att(a)
        Array_BII(a + 1, 1) = Find_Low(Array_att(a))
        a = a + 1
        End If
    Loop

    UtilIntII.ListBox1.List = Array_A
    UtilIntII.ListBox2.List = Array_BI
    UtilIntII.ListBox3.List = Array_BII

    'Dim column_index As Integer
    'column_index = 2
    Global_Interview_Type = "Corner Point"
    Initialize_Bracketing

    Generate_Prob_UtilIntII (Global_Choice)

    NextProbability.TextBoxMaxProbability.Text = Global_High + "%"
    NextProbability.TextBoxMinProbability.Text = Global_Low + "%"
    If AttNavOld.CheckBoxOverride Then NextProbability.show
    If NextProbability.TextBoxProbability.Text <> "end" Then
        UtilIntII.TextBoxPe.Text = NextProbability.TextBoxProbability.Text +
"%"
        UtilIntII.TextBoxPeC.Text = CStr(100 -
CDbl(NextProbability.TextBoxProbability.Text)) + "%"
        UtilIntII.LabelProbability.Caption =
NextProbability.TextBoxProbability.Text

        UtilIntII.show
        UtilIntII.Choice_A.SetFocus
    End If

    Do While Not (Global_Choice = "I" Or UtilIntII.end_interview.Caption = 1)
        If UtilIntII.end_interview.Caption = 2 Then ' have not reached end of
prob, increment prob

            Generate_Prob_UtilIntII (Global_Choice)
```

```
            NextProbability.TextBoxMaxProbability.Text = Global_High + "%"
            NextProbability.TextBoxMinProbability.Text = Global_Low + "%"
            If AttNavOld.CheckBoxOverride Then NextProbability.show
            If NextProbability.TextBoxProbability.Text <> "end" Then
                UtilIntII.TextBoxPe.Text =
NextProbability.TextBoxProbability.Text + "%"
                UtilIntII.TextBoxPeC.Text = CStr(100 -
CDbl(NextProbability.TextBoxProbability.Text)) + "%"
                UtilIntII.LabelProbability.Caption =
NextProbability.TextBoxProbability.Text

                If Global_Choice <> "I" Then
                    UtilIntII.end_interview.Caption = 0
                    UtilIntII.show
                    UtilIntII.Choice_A.SetFocus
                End If
            End If
        End If
    Loop

    Global_Interview_Type = "Corner Point"
    Mark_Completed (Q_attribute)

    AttNavOld.show
End Sub

Sub Single_Interview_Start(Q_attribute As String)

'IndifferentPointsAlreadyCollected, Val, Value_Index, Indifferent_Index are
Global

    Dim curr_prob As Double, curr_Xi As Double, curr_value As String
    Dim X_best As String, X_worst As String
    Dim Q_attribute_prop As AttributeProp
    Dim prob_index As Integer, AttributeDataPoints As Integer

    UtilInt.LabelInterviewType.Caption = "single"

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    UtilInt.TextBoxScenario.Text = Q_attribute_prop.Scenario
    UtilInt.TextBoxDefinition.Text = Q_attribute_prop.Definition

    X_best = Find_High(Q_attribute)
    X_worst = Find_Low(Q_attribute)

    UtilInt.TextBoxHigh.Text = X_best
    UtilInt.TextBoxLow1.Text = X_worst
    UtilInt.TextBoxLow2.Text = X_worst

    'CurrentInterviewForm = UtilIntII

    Value_Index = 2
    Global_Interview_Type = "Single Attribute"
    Initialize_Bracketing
    Generate_Prob_UtilInt (Global_Choice)
```

```
'    curr_value = Format_Value(Q_attribute,
Sheets(Q_attribute).Cells(value_index, 10).Value)
'    UtilInt.TextBoxCurr.Text = curr_value

'    UtilInt.show
    Dim Dummy As Integer

    Val = Sheets(Q_attribute).Cells(Value_Index, 10).Value
    Indifferent_Index = 1
    NextValue.TextBoxAlreadyCollected.Text = ""
    Dummy = Request_Value(Q_attribute, Val, Indifferent_Index,
Q_attribute_prop.Increment)

    Do While Not (NextValue.TextBoxValue.Text = "end" Or
NextProbability.TextBoxProbability.Text = "end" Or
UtilInt.end_interview.Caption = "end")
            ConductInterview (Q_attribute)
    Loop

    ExtractIndifferentPoints (Q_attribute)
    NextValue.TextBoxValue.Text = "start"

    Dim i As Integer
    i = 2
    With Sheets(Q_attribute)
        Do Until (.Cells(i, 23).Value = Q_attribute_prop.max Or
NextValue.TextBoxValue.Text = "end")

            If (Abs(.Cells(i + 1, 24).Value - .Cells(i, 24).Value) >
Q_attribute_prop.Threshold) Then
                Value_Index = 2
                Indifferent_Index = 1
                Val = (.Cells(i, 23) + .Cells(i + 1, 23)) / 2
                Global_Low = .Cells(i, 24).Value * 100 / 2
                Global_High = .Cells(i + 1, 24).Value * 100 / 2

                ConductInterview (Q_attribute)

                ExtractIndifferentPoints (Q_attribute)
                i = 1

            End If
            i = i + 1

        Loop
    End With

'    If UtilInt.end_interview.Caption <> "end" Then
    Global_Interview_Type = "Single"
    Mark_Completed (Q_attribute)
'    End If

    AttNavOld.show
End Sub
Sub ConductInterview(Q_attribute As String)
```

```
    Dim Dummy As Integer
    Dim Q_attribute_prop As AttributeProp

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    'IndifferentPointsAlreadyCollected, Val, Value_Index, Indifferent_Index

    If UtilInt.LabelPreference.Caption = "I" Or Global_Value_Done Then
        If IndifferentPointsAlreadyCollected > "" Then
            IndifferentPointsAlreadyCollected =
IndifferentPointsAlreadyCollected + ", "
        End If
        IndifferentPointsAlreadyCollected = IndifferentPointsAlreadyCollected
+ CStr(UtilInt.LabelCurrentValue.Caption)
        NextValue.TextBoxAlreadyCollected.Text =
IndifferentPointsAlreadyCollected
        Dummy = Request_Value(Q_attribute, Val, Value_Index - 1,
Q_attribute_prop.Increment)
    End If

    If NextValue.TextBoxValue.Text <> "end" Then
        If Not Value_Done(Q_attribute, CDbl(NextValue.TextBoxValue.Text))
Then
            UtilInt.TextBoxCurr.Text = Format_Value(Q_attribute,
NextValue.TextBoxValue.Text)
            UtilInt.LabelCurrentValue.Caption =
CDbl(NextValue.TextBoxValue.Text)

            UtilInt.LabelPreference.Caption = "none"

            NextProbability.TextBoxMaxProbability.Text = Global_High + "%"
            NextProbability.TextBoxMinProbability.Text = Global_Low + "%"
            If AttNavOld.CheckBoxOverride Then NextProbability.show
            If NextProbability.TextBoxProbability.Text <> "end" Then
                UtilInt.TextBoxPe.Text =
NextProbability.TextBoxProbability.Text + "%"
                UtilInt.TextBoxPeC.Text = CStr(100 -
CDbl(NextProbability.TextBoxProbability.Text)) + "%"
                UtilInt.LabelProbability.Caption =
NextProbability.TextBoxProbability.Text

                UtilInt.show
                UtilInt.Choice_A.SetFocus

                If UtilInt.end_interview.Caption <> "end" Then
                    Generate_Prob_UtilInt (Global_Choice)

                    If UtilInt.LabelPreference.Caption = "I" Then
                        Value_Index = Value_Index + 1
                        Initialize_Bracketing
                        Generate_Prob_UtilInt (Global_Choice)
                    End If

                    Val = Sheets(Q_attribute).Cells(Value_Index, 10).Value
                    Indifferent_Index = Indifferent_Index + 1
                Else
'                   Indifferent_Index = Q_attribute_prop.Increment + 1
```

```
                    NextValue.TextBoxValue.Text = "end"
                End If
            End If
        Else
            Value_Index = Value_Index + 1
            Val = Sheets(Q_attribute).Cells(Value_Index, 10).Value
        End If
    End If

End Sub
```

### 11.3.5 Report Generation

```
Attribute VB_Name = "Module4"
Sub Macro4()
Attribute Macro4.VB_Description = "Macro recorded 2/22/2002 by Satwik
Seshasai"
Attribute Macro4.VB_ProcData.VB_Invoke_Func = " \n14"
'
' Macro4 Macro
' Macro recorded 2/22/2002 by Satwik Seshasai
'

'
    Range("J21:K23").Select
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmooth
    ActiveChart.SetSourceData Source:=Sheets("Home").Range("J21:K23"),
PlotBy:= _
        xlColumns
    ActiveChart.Location Where:=xlLocationAsNewSheet
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "blah"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "x stuf"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "y stuf"
    End With
    ActiveChart.HasLegend = False


Sub ExtractIndifferentPoints(Q_attribute As String)
    Dim i As Integer
    Dim Indifferent_Index As Integer
    Dim Pref As String
    Dim Q_attribute_prop As AttributeProp

    Q_attribute_prop = Retrieve_selection(Q_attribute)

    i = 2
    Indifferent_Index = 4

    Pref = "Start"
```

```
    With Sheets(Q_attribute)
        .Range("W:W").ClearContents
        .Range("X:X").ClearContents

        .Cells(2, 23).Value = Q_attribute_prop.min
        .Cells(3, 23).Value = Q_attribute_prop.max
        If Q_attribute_prop.Direction Then
            .Cells(2, 24).Value = 0
            .Cells(3, 24).Value = 1
        Else
            .Cells(2, 24).Value = 1
            .Cells(3, 24).Value = 0
        End If

        Do Until Pref = ""
            Pref = .Cells(i, 14).Value
            If Pref = "I" Then
                .Cells(Indifferent_Index, 23).Value = .Cells(i, 12).Value
                .Cells(Indifferent_Index, 24).Value = .Cells(i, 13).Value * 2
                Indifferent_Index = Indifferent_Index + 1
            End If
            i = i + 1
        Loop

        Worksheets(Q_attribute).Range("W2:X6000").Sort _
            Key1:=Worksheets(Q_attribute).Range("W2")
    End With

End Sub

Attribute VB_Name = "Carolyn_Module"
Sub GenerateReports(Interview_Name As String, selected_attributes() As
String)
If Interview_Name = "Single Attribute Interview" Then
'iterate through the selected attributes
'for each selected attribute create
Dim Attribute_index As Integer
Attribute_index = 0
Do While selected_attributes(Attribute_index) <> Null
    'create
    Worksheets("TestSheet").Cells(3, 3) = selected_attributes.Size

'create a sheet to graph these values

'get all the attributes for single attribute interviews
'get the max and min values of these attributes
'get the utility and values of the attributes involved in the interview
'construct tables and graphs on this sheet

End If
If Interview_Name = "Multi Attribute Interview" Then


End If


End Sub
```

```
'Carolyn's Generate Report stuff
Sub Display_GenReport()
Load GenReport
GenReport.show
Unload GenReport
      End Sub

Attribute VB_Name = "Module3"
Sub MacroCreateChart()
Attribute MacroCreateChart.VB_Description = "Macro recorded 2/20/2002 by
Satwik Seshasai"
Attribute MacroCreateChart.VB_ProcData.VB_Invoke_Func = " \n14"
'
' MacroCreateChart Macro
' Macro recorded 2/20/2002 by Satwik Seshasai
'

'
    'Dim i As Integer
    'Dim Val As Double
'
'    i = 1
'
'    Do Until Val = 0
'
'        Val = Sheets("Latency").Cells(i, 12).Value
'        i = i + 1
'
'    Loop

    Worksheets("Latency").Range(Worksheets("Latency").Cells(1, 12),
Worksheets("Latency").Cells(100, 14)).Sort Key1:=Cells(2, 3),
Order1:=xlAscending, Key2:=Cells(2, 1), Order2:=xlAscending, Header:=xlGuess,
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

    Sheets("Latency").Columns("L:N").Select
    selection.Sort Key1:=Range("N2"), Order1:=xlAscending, Key2:=Range("L2")
_
        , Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub

Sub MacroGraphREAL()
Attribute MacroGraphREAL.VB_Description = "Macro recorded 2/20/2002 by Satwik
Seshasai"
Attribute MacroGraphREAL.VB_ProcData.VB_Invoke_Func = " \n14"
'
' MacroGraphREAL Macro
' Macro recorded 2/20/2002 by Satwik Seshasai
'

'
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmooth
    ActiveChart.SetSourceData Source:=Sheets("Spatial
Resolution").Range("L9:M13" _
```

```
        ), PlotBy:=xlColumns
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Spatial
Resolution"
End Sub
Sub GenerateReports_Click()
    Interview_click ("curve generation")
End Sub

Sub CreateSingleAttributeCurve(Q_attribute As String)

    ExtractIndifferentPoints (Q_attribute)

    With Sheets(Q_attribute)
        Charts.Add
        With ActiveChart
            .ChartType = xlXYScatterSmooth
            .Name = Q_attribute + " Curve"
            .Move after:=Worksheets(Worksheets.Count)
'            .HasTitle = True
'            .ChartTitle.Characters.Text = Q_attribute + " Utility"
'            .Axes(xlCategory, xlPrimary).HasTitle = True
'            .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text =
Q_attribute + " Value"
'            .Axes(xlValue, xlPrimary).HasTitle = True
'            .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Utility"
            .SetSourceData Source:=Sheets(Q_attribute).Range("W2:X100"),
PlotBy:=xlColumns
        End With

    End With

End Sub
Sub Macro3()
Attribute Macro3.VB_Description = "Macro recorded 2/21/2002 by Satwik
Seshasai"
Attribute Macro3.VB_ProcData.VB_Invoke_Func = " \n14"
'
' Macro3 Macro
' Macro recorded 2/21/2002 by Satwik Seshasai
'

'
    Sheets("Latency").Select
    Range("L2:N51").Select
    selection.Sort Key1:=Range("L2"), Order1:=xlAscending, Header:=xlGuess, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
    selection.Sort Key1:=Range("N2"), Order1:=xlAscending, Key2:=Range("L2")
_
        , Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub


Attribute VB_Name = "Module3"
Sub MacroCreateChart()
```

```
Attribute MacroCreateChart.VB_Description = "Macro recorded 2/20/2002 by
Satwik Seshasai"
Attribute MacroCreateChart.VB_ProcData.VB_Invoke_Func = " \n14"
'
' MacroCreateChart Macro
' Macro recorded 2/20/2002 by Satwik Seshasai
'

'
    'Dim i As Integer
    'Dim Val As Double
'
'    i = 1
'
'    Do Until Val = 0
'
'        Val = Sheets("Latency").Cells(i, 12).Value
'        i = i + 1
'
'    Loop

    Worksheets("Latency").Range(Worksheets("Latency").Cells(1, 12),
Worksheets("Latency").Cells(100, 14)).Sort Key1:=Cells(2, 3),
Order1:=xlAscending, Key2:=Cells(2, 1), Order2:=xlAscending, Header:=xlGuess,
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

    Sheets("Latency").Columns("L:N").Select
    selection.Sort Key1:=Range("N2"), Order1:=xlAscending, Key2:=Range("L2")
_
        , Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub

Sub MacroGraphREAL()
Attribute MacroGraphREAL.VB_Description = "Macro recorded 2/20/2002 by Satwik
Seshasai"
Attribute MacroGraphREAL.VB_ProcData.VB_Invoke_Func = " \n14"
'
' MacroGraphREAL Macro
' Macro recorded 2/20/2002 by Satwik Seshasai
'

'
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmooth
    ActiveChart.SetSourceData Source:=Sheets("Spatial
Resolution").Range("L9:M13" _
        ), PlotBy:=xlColumns
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Spatial
Resolution"
End Sub
Sub GenerateReports_Click()
    Interview_click ("curve generation")
End Sub

Sub CreateSingleAttributeCurve(Q_attribute As String)
```

```
    ExtractIndifferentPoints (Q_attribute)

    With Sheets(Q_attribute)
        Charts.Add
        With ActiveChart
            .ChartType = xlXYScatterSmooth
            .Name = Q_attribute + " Curve"
            .Move after:=Worksheets(Worksheets.Count)
'            .HasTitle = True
'            .ChartTitle.Characters.Text = Q_attribute + " Utility"
'            .Axes(xlCategory, xlPrimary).HasTitle = True
'            .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text =
Q_attribute + " Value"
'            .Axes(xlValue, xlPrimary).HasTitle = True
'            .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Utility"
            .SetSourceData Source:=Sheets(Q_attribute).Range("W2:X100"),
PlotBy:=xlColumns
        End With

    End With

End Sub
Sub Macro3()
Attribute Macro3.VB_Description = "Macro recorded 2/21/2002 by Satwik
Seshasai"
Attribute Macro3.VB_ProcData.VB_Invoke_Func = " \n14"
'
' Macro3 Macro
' Macro recorded 2/21/2002 by Satwik Seshasai
'

'
    Sheets("Latency").Select
    Range("L2:N51").Select
    selection.Sort Key1:=Range("L2"), Order1:=xlAscending, Header:=xlGuess, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
    selection.Sort Key1:=Range("N2"), Order1:=xlAscending, Key2:=Range("L2")
_
        , Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub


Attribute VB_Name = "Module2"
Sub Macro1()
Attribute Macro1.VB_Description = "Macro recorded 12/27/2001 by Satwik
Seshasai"
Attribute Macro1.VB_ProcData.VB_Invoke_Func = " \n14"
'
' Macro1 Macro
' Macro recorded 12/27/2001 by Satwik Seshasai
'

'
    Range("L3:M7").Select
    Charts.Add
```

```
    ActiveChart.ChartType = xlXYScatterSmooth
    ActiveChart.SetSourceData Source:=Sheets("Spatial
Resolution").Range("L3:M7")
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Spatial
Resolution"
End Sub
Sub Macro2()
Attribute Macro2.VB_Description = "Macro recorded 12/27/2001 by Quincy Scott"
Attribute Macro2.VB_ProcData.VB_Invoke_Func = " \n14"
'
' Macro2 Macro
' Macro recorded 12/27/2001 by Quincy Scott
'

'
    ActiveCell.Range("A1:B5").Select
    selection.Sort Key1:=ActiveCell, Order1:=xlAscending, Header:=xlGuess, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmooth
    ActiveChart.SetSourceData Source:=Sheets("Spatial
Resolution").Range("L3:M7") _
        , PlotBy:=xlColumns
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Spatial
Resolution"
    With ActiveChart
        .HasTitle = False
        .Axes(xlCategory, xlPrimary).HasTitle = False
        .Axes(xlValue, xlPrimary).HasTitle = False
    End With
End Sub

Sub MacroAttRange()
    Dim Att_Range As Range


    Set Att_Range = ActiveWorkbook.Names("Attribute_list").RefersToRange
    ActiveWorkbook.Names("Attribute_list").RefersTo = Att_Range.Resize(6)
End Sub
```

### 11.3.6 Compare Projects

*All code in this module written by Frank Reyes* [36].

```
Attribute VB_Name = "Frank_Module"
Public Const DECLARATIONS_ROW As Integer = 1    'Row of AttributeCache
Sheet where declarations are made such as NumberStudies, etc
Public Const NUM_STUDIES_COL As Integer = 1     'Column of
AttributeCache Sheet containing the NumberStudies value
Public Const NUM_ATTRIBUTES_COL As Integer = 2  'Column of
AttributeCache Sheet containing the NumberAttributes value

Public Const DATA_START_ROW As Integer = 3      'Row of AttributeCache
Sheet where data entry rows begin
Public Const TYPE_COL As Integer = 1            'Column of
AttributeCache Sheet containing the type field
Public Const ID_COL As Integer = 2              'Column of
AttributeCache Sheet containing the id field
Public Const NAME_COL As Integer = 3            'Column of
AttributeCache Sheet containing the name field
Public Const DESC_COL As Integer = 4            'Column of
AttributeCache Sheet containing the description field
Public Const VALUES_START_COL As Integer = 5    'Column of
AttributeCache Sheet where the data entries values begin

Public Const ATTRIBUTE_TYPE As String = "1"     'Value used to denote
that a data entry in the AttributeCache Sheet is an Attribute
Public Const STUDY_TYPE As String = "2"         'Value used to denote
that a data entry in the AttributeCache Sheet is a Study

Type DoublePoint            'Emulates a point (X,Y)
    X As Double
    Y As Double
End Type

Type DoubleLine             'Emulates a line Y = MX + C
    M As Double
    C As Double
End Type

Function VerticalPointToLineDistanceSquared(InLine As DoubleLine,
InPoint As DoublePoint) As Double
'Returns the squared distance along the y axis between a point and a
line
    'Find the Y Value of the line at the same X value as the point
    Dim YVal As Double
    YVal = InLine.M * InPoint.X + InLine.C

    'Return the squared distance between the two Y Values
    VerticalPointToLineDistance = (YVal - InPoint.Y) * (YVal -
InPoint.Y)
End Function
```

```
Function DifferenceSquared(InValue1 As Double, InValue2 As Double) As
Double
'Returns the squared difference between two values
    DifferenceSquared = (InValue2 - InValue1) * (InValue2 - InValue1)
End Function

Function PointToLineDistance(InLine As DoubleLine, InPoint As
DoublePoint) As Double
'Returns the shortest distance between a point and a line
    'Find equation of a line through the point and perpendicular to the
line
    Dim PerpLine As DoubleLine
    PerpLine.M = -1 / InLine.M
    PerpLine.C = InPoint.X / InLine.M + InPoint.Y

    'Find the point of intersection of the two lines
    Dim PerpIntersectPoint As DoublePoint
    PerpIntersectPoint.X = (PerpLine.C - InLine.C) / (InLine.M -
PerpLine.M)
    PerpIntersectPoint.Y = (PerpLine.M * InLine.C - InLine.M *
PerpLine.C) / (PerpLine.M - InLine.M)

    'Shortest distance between point and line is the distance between
the point
    'and the point of intersection of the line through the point and
perpendicular to the line
    PointToLineDistance = PointToPointDistance(InPoint,
PerpIntersectPoint)
End Function

Function PointToPointDistance(InPoint1 As DoublePoint, InPoint2 As
DoublePoint) As Double
'Returns the distance between two points using the standard distance
formula
    PointToPointDistance = Sqr((InPoint2.X - InPoint1.X) * (InPoint2.X
- InPoint1.X) + (InPoint2.Y - InPoint1.Y) * (InPoint2.Y - InPoint1.Y))
End Function

Function GetSlope(InPoint1 As DoublePoint, InPoint2 As DoublePoint) As
Double
'Returns the slope of two points
    GetSlope = (InPoint2.Y - InPoint1.Y) / (InPoint2.X - InPoint1.X)
End Function

Function GetLine(InPoint1 As DoublePoint, InPoint2 As DoublePoint) As
DoubleLine
'Returns the line between two points
    Dim LineToReturn As DoubleLine
    LineToReturn.M = GetSlope(InPoint1, InPoint2)
    LineToReturn.C = -GetSlope(InPoint1, InPoint2) * InPoint1.X +
InPoint1.Y

    GetLine = LineToReturn
End Function

Function CurveCorrelationValue(CurveToCompare() As DoublePoint,
BaseCurve() As DoublePoint) As Double
```

```
'Returns a value similar to the sum of squares distance between the two
curves
    Dim CorrelationToReturn As Double
    CorrelationToReturn = 0

    'Loop and temporary variables
    Dim intLoop As Integer
    Dim CurrentLine As DoubleLine
    Dim CurrentLeftX As Double
    Dim CurrentRightX As Double

    'Loop through all of the X values of the curve to compare and
    'add their squared vertical Y distance to the running sum
    For intLoop = 1 To UBound(CurveToCompare)
        If intLoop = 1 Then     'Left endpoint, squared difference will
be either 0 or 1
            CorrelationToReturn = CorrelationToReturn +
DifferenceSquared(CurveToCompare(1).Y, BaseCurve(1).Y)

        Else
            If intLoop = UBound(CurveToCompare) Then      'Right
Endpoint, squared difference will be either 0 or 1
                CorrelationToReturn = CorrelationToReturn +
DifferenceSquared(CurveToCompare(UBound(CurveToCompare)).Y,
BaseCurve(UBound(BaseCurve)).Y)

            Else    'Determine in between which consecutive X values of
the base curve the current point's X value lies
                CurrentLeftX = BaseCurve(1).X
                CurrentRightX = BaseCurve(2).X
                CurrentRightIndex = 2

                'Loop until you find the correct bounding X values
                Do Until ((CurrentLeftX <= CurveToCompare(intLoop).X)
And (CurveToCompare(intLoop).X <= CurrentRightX))
                    CurrentLeftX = CurrentRightX
                    CurrentRightX = BaseCurve(CurrentRightIndex + 1).X
                    CurrentRightIndex = CurrentRightIndex + 1
                Loop

                'Handle case where bounding X values and current
point's X value are all the same value
                'Test for case
                If ((CurrentRightX = BaseCurve(CurrentRightIndex + 1))
And (CurrentRightX = CurveToCompare(intLoop).X)) Then
                    'Shift values
                    CurrentLeftX = CurrentRightX
                    CurrentRightX = BaseCurve(CurrentRightIndex + 1).X
                    CurrentRightIndex = CurrentRightIndex + 1

                    'Add squared Y distance from the current point to
the closest of the two other points
                    If (Abs(CurveToCompare(intLoop).Y -
BaseCurve(CurrentLeftX).Y) <= Abs(CurveToCompare(intLoop).Y -
BaseCurve(CurrentRightX).Y)) Then
                        CorrelationToReturn = CorrelationToReturn +
DifferenceSquared(CurveToCompare(intLoop).Y, BaseCurve(CurrentLeftX).Y)
```

```
                    Else
                        CorrelationToReturn = CorrelationToReturn +
DifferenceSquared(CurveToCompare(intLoop).Y,
BaseCurve(CurrentRightX).Y)
                    End If

                Else        'create a line between the two X values and
add squared vertical distance of current point to line
                    CurrentLine = GetLine(BaseCurve(CurrentLeftX),
BaseCurve(CurrentRightX))
                    CorrelationToReturn = CorrelationToReturn +
VerticalPointToLineDistanceSquared(CurrentLine,
CurveToCompare(intLoop))
                End If
            End If
        End If
    Next

    CurveCorrelationValue = CorrelationToReturn
End Function

Sub CompressCurve(InCurve() As DoublePoint)
'Compresses the X values of the curve such that the values are all in
the range of 0 to 1
    Dim intLoop As Integer
    Dim XRange As Double
    XRange = InCurve(UBound(InCurve)).X - InCurve(1).X
    Dim XMin As Double
    XMin = InCurve(1).X

    'Loop through all of the X values and compress them using the
minimum value and range of values
    For intLoop = 1 To UBound(InCurve)
        InCurve(intLoop).X = (InCurve(intLoop).X - XMin) / XRange
    Next
End Sub

Function EqualsStringValue(StringToParse As String) As Variant
'Returns the value of a string of the form Expression=value
    EqualsStringValue = Right(StringToParse, Len(StringToParse) -
InStr(StringToParse, "="))
End Function

Sub PopulateCurveArrayFromCache(AttributeID As Integer, CurveArray() As
DoublePoint)
'Populates a curve array with the values in the attribute cache, given
an attribute ID
    'Create a worksheet variable to access the attribute cache
worksheet
    Dim AttCacheWorksheet As Worksheet
    Set AttCacheWorksheet =
ThisWorkbook.Worksheets.Item("AttributeCache")

    'Variable to hold the row number of the attribute with the given id
in the worksheet
    Dim AttributeRow As Integer
    AttributeRow = DATA_START_ROW
```

```
    'Loop through rows until you find the one containing the correct
attribute
    Do Until ((AttCacheWorksheet.Cells(AttributeRow, TYPE_COL).Value =
ATTRIBUTE_TYPE) And (CInt(AttCacheWorksheet.Cells(AttributeRow,
ID_COL).Value) = AttributeID))
        AttributeRow = AttributeRow + 1
    Loop

    'Loop variable for the current values column you are at for the
given attribute in the worksheet
    Dim currentCol As Integer
    currentCol = VALUES_START_COL

    'String to hold cell data of the form "X=value,Y=value"
    Dim currentCellString As String
    currentCellString = CStr(AttCacheWorksheet.Cells(AttributeRow,
currentCol).Value)

    'Loop through all of the values available for the attribute and add
it to the curve array
    While Not IsEmpty(AttCacheWorksheet.Cells(AttributeRow,
currentCol))
        ReDim Preserve CurveArray(1 To (currentCol - VALUES_START_COL +
1))
        CurveArray(UBound(CurveArray)).X =
CDbl(EqualsStringValue(Left(currentCellString, InStr(currentCellString,
",") - 1)))
        CurveArray(UBound(CurveArray)).Y =
CDbl(EqualsStringValue(Right(currentCellString, Len(currentCellString)
- InStr(currentCellString, ","))))

        currentCol = currentCol + 1
        currentCellString = CStr(AttCacheWorksheet.Cells(AttributeRow,
currentCol).Value)
    Wend

End Sub

Sub PopulateAndDisplayComparisonForm()
'Populates the comparison form with the values from the attribute cache
worksheet and displays it
    'Load the form
    Load AttCompare

    'Create a worksheet variable to access the attribute cache
worksheet
    Dim AttCacheWorksheet As Worksheet
    Set AttCacheWorksheet =
ThisWorkbook.Worksheets.Item("AttributeCache")

    'Variables to hold the number of data entries
    Dim TotalStudies As Integer
    Dim TotalAttributes As Integer
    Dim TotalStudiesAndAttributes As Integer

    'Get total number of studies and attributes
```

```
    TotalStudies =
CInt(EqualsStringValue(AttCacheWorksheet.Cells(DECLARATIONS_ROW,
NUM_STUDIES_COL).Value))
    TotalAttributes =
CInt(EqualsStringValue(AttCacheWorksheet.Cells(DECLARATIONS_ROW,
NUM_ATTRIBUTES_COL).Value))
    TotalStudiesAndAttributes = TotalStudies + TotalAttributes

    'loop through the data rows, add row to appropriate list box in
form
    'based on whether it is a study or attribute
    Dim currentRow As Integer
    For currentRow = DATA_START_ROW To (DATA_START_ROW +
TotalStudiesAndAttributes - 1)
        If (AttCacheWorksheet.Cells(currentRow, TYPE_COL).Value =
STUDY_TYPE) Then
            AttCompare.Controls.Item("Studies").AddItem
AttCacheWorksheet.Cells(currentRow, NAME_COL).Value
        Else
            AttCompare.Controls.Item("CachedAttributes").AddItem
AttCacheWorksheet.Cells(currentRow, NAME_COL).Value
        End If
    Next

    'Variables used to get information about current attributes
    Dim numrows As Integer
    Dim r As Integer
    Dim Att_range As Range

    'Determine the number of current attributes
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count

    'Add all the current attributes to the appropriate list box on the
form
    For r = 8 To (8 + numrows - 1)
        AttCompare.Controls.Item("CurrentAttributes").AddItem
Sheets("Home").Cells(r, 8).Value
    Next

    'Display the form
    AttCompare.show
    Unload AttCompare
End Sub

Sub TestCode()
'Temporary procedure to test code
    Dim Attribute3() As DoublePoint
    Call PopulateCurveArrayFromCache(3, Attribute3)
    'MsgBox (UBound(Attribute3))

End Sub
```

### 11.3.7 Compare Stakeholders

*All code in this module written by Winston Chang* [35].

```
Stakeholder Analysis Form Load Code
Sub Load_StakeholderAnalysis()

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    Dim Array_att(20) As String
    Dim i As Integer
    i = 0

    Do While i < numrows
        Array_att(i) = Att_range(i + 1).Value
        i = i + 1
    Loop

    Dim Array_A() As String
    ReDim Array_A(0 To numrows + 1, 0 To 2)
    Dim Array_B() As String
    ReDim Array_B(0 To numrows + 1, 0 To 2)
    Dim Array_C(20) As String
    Dim Array_D(20) As String

    Array_A(0, 0) = "Property Name"
    Array_A(0, 1) = "Property Value"
    Array_B(0, 0) = "Attribute Name"
    Array_B(0, 1) = "Attribute Value"

    If Worksheets("Test").Cells(1, 14) <> 0 Then
        Dim b As Integer

        b = 0
        i = 2

        Do Until (Worksheets("Test").Cells(i, 11) =
Worksheets("Test").Cells(1, 14))
            i = i + 1
        Loop

        Do While b < numrows
            Array_A(b + 1, 0) = Worksheets("Test").Cells(i, 12).Value
            Array_C(b + 1) = Worksheets("Test").Cells(b + 1, 17).Value
            Array_D(b + 1) = Worksheets("Test").Cells(i, 14).Value
            Array_A(b + 1, 1) = Array_C(b + 1) & " " & Array_D(b + 1)
            Array_B(b + 1, 0) = Worksheets("Test").Cells(i, 12).Value
```

```
            Array_C(b + 1) = Worksheets("Test").Cells(i, 13).Value
            Array_B(b + 1, 1) = Array_C(b + 1) & " " & Array_D(b + 1)
            b = b + 1
            i = i + 1
        Loop

        StakeholderAnalysis.PropertiesListBox.List = Array_A
        StakeholderAnalysis.AttributesListBox.List = Array_B
    End If


    If Worksheets("Test").Cells(1, 14) = 0 Then

        Dim a As Integer
        a = 0

        Do While a < numrows
            Array_A(a + 1, 0) = Worksheets("Test").Cells(a + 1,
16).Value
            Array_C(a + 1) = Worksheets("Test").Cells(a + 1, 17).Value
            Array_D(a + 1) = Worksheets("Test").Cells(a + 1, 19).Value
            Array_A(a + 1, 1) = Array_C(a + 1) & " " & Array_D(a + 1)
            Array_B(a + 1, 0) = Worksheets("Test").Cells(a + 1,
16).Value
            Array_C(a + 1) = Worksheets("Test").Cells(a + 1, 18).Value
            Array_B(a + 1, 1) = Array_C(a + 1) & " " & Array_D(a + 1)
            a = a + 1
        Loop

        StakeholderAnalysis.PropertiesListBox.List = Array_A
        StakeholderAnalysis.AttributesListBox.List = Array_B

        'begin filling in of "Test" wkst
        Worksheets("Test").Cells(1, 14) = 1
        Worksheets("Test").Cells(2, 1) = Worksheets("Test").Cells(1,
14)

        a = 0
        i = 0

        Do While a < numrows
            Worksheets("Test").Cells(i + 2, 11) =
Worksheets("Test").Cells(1, 14)
            Worksheets("Test").Cells(i + 2, 12) = Array_B(a + 1, 0)
            Worksheets("Test").Cells(i + 2, 13) = Array_C(a + 1)
            Worksheets("Test").Cells(i + 2, 14) = Array_D(a + 1)
            a = a + 1
            i = i + 1
        Loop

        Worksheets("Test").Cells(2, 2) =
StakeholderAnalysis.CalcExpense("Stakeholder1")
        Worksheets("Test").Cells(2, 3) =
StakeholderAnalysis.CalcUtility("Stakeholder1")
        Worksheets("Test").Cells(2, 4) =
StakeholderAnalysis.CalcExpense("Stakeholder2")
```

```
        Worksheets("Test").Cells(2, 5) =
StakeholderAnalysis.CalcUtility("Stakeholder2")
        Worksheets("Test").Cells(2, 6) =
StakeholderAnalysis.CalcExpense("Stakeholder3")
        Worksheets("Test").Cells(2, 7) =
StakeholderAnalysis.CalcUtility("Stakeholder3")
        Worksheets("Test").Cells(2, 8) =
StakeholderAnalysis.CalcExpense("Stakeholder4")
        Worksheets("Test").Cells(2, 9) =
StakeholderAnalysis.CalcUtility("Stakeholder4")
    End If

    Dim tmp As Integer

    tmp = Worksheets("Test").Cells(1, 14) + 1

    Worksheets("Test").Cells(tmp, 1) = Worksheets("Test").Cells(1, 14)

    StakeholderAnalysis.UtilityTextBox1.Text =
Worksheets("Test").Cells(tmp, 3)
    StakeholderAnalysis.UtilityTextBox2.Text =
Worksheets("Test").Cells(tmp, 5)
    StakeholderAnalysis.UtilityTextBox3.Text =
Worksheets("Test").Cells(tmp, 7)
    StakeholderAnalysis.UtilityTextBox4.Text =
Worksheets("Test").Cells(tmp, 9)

    StakeholderAnalysis.ExpenseTextBox1.Text =
Worksheets("Test").Cells(tmp, 2)
    StakeholderAnalysis.ExpenseTextBox2.Text =
Worksheets("Test").Cells(tmp, 4)
    StakeholderAnalysis.ExpenseTextBox3.Text =
Worksheets("Test").Cells(tmp, 6)
    StakeholderAnalysis.ExpenseTextBox4.Text =
Worksheets("Test").Cells(tmp, 8)

'graphing
    Dim Chart1 As owc.WCChart
    Dim Chart2 As owc.WCChart
    Dim Chart3 As owc.WCChart
    Dim Chart4 As owc.WCChart
    Dim Series1 As owc.WCSeries
    Dim Series2 As owc.WCSeries
    Dim Series3 As owc.WCSeries
    Dim Series4 As owc.WCSeries
    Dim XData1(1 To 40)
    Dim YData1(1 To 40)
    Dim XData2(1 To 40)
    Dim YData2(1 To 40)
    Dim XData3(1 To 40)
    Dim YData3(1 To 40)
    Dim XData4(1 To 40)
    Dim YData4(1 To 40)

    StakeholderAnalysis.ChartSpace1.Clear
    StakeholderAnalysis.ChartSpace1.Refresh
    StakeholderAnalysis.ChartSpace2.Clear
```

```
StakeholderAnalysis.ChartSpace2.Refresh
StakeholderAnalysis.ChartSpace3.Clear
StakeholderAnalysis.ChartSpace3.Refresh
StakeholderAnalysis.ChartSpace4.Clear
StakeholderAnalysis.ChartSpace4.Refresh

Set Chart1 = StakeholderAnalysis.ChartSpace1.Charts.Add
Set Chart2 = StakeholderAnalysis.ChartSpace2.Charts.Add
Set Chart3 = StakeholderAnalysis.ChartSpace3.Charts.Add
Set Chart4 = StakeholderAnalysis.ChartSpace4.Charts.Add

For i = 1 To 40
    XData1(i) = Worksheets("Test").Cells(i + 1, 2)
    YData1(i) = Worksheets("Test").Cells(i + 1, 3)
    XData2(i) = Worksheets("Test").Cells(i + 1, 4)
    YData2(i) = Worksheets("Test").Cells(i + 1, 5)
    XData3(i) = Worksheets("Test").Cells(i + 1, 6)
    YData3(i) = Worksheets("Test").Cells(i + 1, 7)
    XData4(i) = Worksheets("Test").Cells(i + 1, 8)
    YData4(i) = Worksheets("Test").Cells(i + 1, 9)
Next

Chart1.Type = owc.chChartTypeScatterMarkers
Chart2.Type = owc.chChartTypeScatterMarkers
Chart3.Type = owc.chChartTypeScatterMarkers
Chart4.Type = owc.chChartTypeScatterMarkers
Set Series1 = Chart1.SeriesCollection.Add
Set Series2 = Chart2.SeriesCollection.Add
Set Series3 = Chart3.SeriesCollection.Add
Set Series4 = Chart4.SeriesCollection.Add

With Series1
    .SetData owc.chDimXValues, owc.chDataLiteral, XData1
    .SetData owc.chDimYValues, owc.chDataLiteral, YData1
End With

With Series2
    .SetData owc.chDimXValues, owc.chDataLiteral, XData2
    .SetData owc.chDimYValues, owc.chDataLiteral, YData2
End With

With Series3
    .SetData owc.chDimXValues, owc.chDataLiteral, XData3
    .SetData owc.chDimYValues, owc.chDataLiteral, YData3
End With

With Series4
    .SetData owc.chDimXValues, owc.chDataLiteral, XData4
    .SetData owc.chDimYValues, owc.chDataLiteral, YData4
End With

With Chart1.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
```

```
    .Title.Font.Size = 6
End With

With Chart1.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart2.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With

With Chart2.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart3.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With

With Chart3.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart4.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With
```

```
    With Chart4.Axes(owc.chAxisPositionBottom)
        .Scaling.Maximum = 1
        .Scaling.Minimum = 0
        .MajorUnit = 0.2
        .HasTitle = True
        .Title.Caption = "Expense"
        .Title.Font.Size = 6
    End With

    FindRelations_Load

End Sub

Sub FindRelations_Load()
    StakeholderAnalysis.FRStakeholderComboBox.Clear
    StakeholderAnalysis.FRStakeholderComboBox.AddItem "Stakeholder 1"
    StakeholderAnalysis.FRStakeholderComboBox.AddItem "Stakeholder 2"
    StakeholderAnalysis.FRStakeholderComboBox.AddItem "Stakeholder 3"
    StakeholderAnalysis.FRStakeholderComboBox.AddItem "Stakeholder 4"
End Sub
Stakeholder Analysis Main Form Code
Private Sub CommandButtonExit_Click()
    Unload StakeholderAnalysis
    StakeholderAnalysis.Hide
End Sub

Private Sub CommandButtonFindRelations_Click()
    Dim a As Integer

    Select Case FRStakeholderComboBox.Value
        Case "Stakeholder 1"
            a = 2
        Case "Stakeholder 2"
            a = 4
        Case "Stakeholder 3"
            a = 6
        Case "Stakeholder 4"
            a = 8
        Case Else
            MsgBox ("Error.  Please select a stakeholder.")
    End Select

    Dim i As Integer

    i = 2

    Do Until (Worksheets("Test").Cells(i, a + 1) =
FRUtilityTextBox.Text) And (Worksheets("Test").Cells(i, a) =
FRExpenseTextBox.Text)
        i = i + 1
    Loop
    'now i contains the row number which the attribute set is at

    'GRAPHING
    Dim Chart1 As owc.WCChart
    Dim Chart2 As owc.WCChart
    Dim Chart3 As owc.WCChart
```

```
    Dim Chart4 As owc.WCChart
    Dim Series1 As owc.WCSeries
    Dim Series2 As owc.WCSeries
    Dim Series3 As owc.WCSeries
    Dim Series4 As owc.WCSeries

    StakeholderAnalysis.ChartSpace1.Clear
    StakeholderAnalysis.ChartSpace1.Refresh
    StakeholderAnalysis.ChartSpace2.Clear
    StakeholderAnalysis.ChartSpace2.Refresh
    StakeholderAnalysis.ChartSpace3.Clear
    StakeholderAnalysis.ChartSpace3.Refresh
    StakeholderAnalysis.ChartSpace4.Clear
    StakeholderAnalysis.ChartSpace4.Refresh

    Set Chart1 = StakeholderAnalysis.ChartSpace1.Charts.Add
    Set Chart2 = StakeholderAnalysis.ChartSpace2.Charts.Add
    Set Chart3 = StakeholderAnalysis.ChartSpace3.Charts.Add
    Set Chart4 = StakeholderAnalysis.ChartSpace4.Charts.Add

    Chart1.Type = owc.chChartTypeScatterMarkers
    Chart2.Type = owc.chChartTypeScatterMarkers
    Chart3.Type = owc.chChartTypeScatterMarkers
    Chart4.Type = owc.chChartTypeScatterMarkers
    Set Series1 = Chart1.SeriesCollection.Add
    Set Series2 = Chart2.SeriesCollection.Add
    Set Series3 = Chart3.SeriesCollection.Add
    Set Series4 = Chart4.SeriesCollection.Add

    With Series1
        .SetData owc.chDimXValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 2)
        .SetData owc.chDimYValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 3)
    End With

    With Series2
        .SetData owc.chDimXValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 4)
        .SetData owc.chDimYValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 5)
    End With

    With Series3
        .SetData owc.chDimXValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 6)
        .SetData owc.chDimYValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 7)
    End With

    With Series4
        .SetData owc.chDimXValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 8)
        .SetData owc.chDimYValues, owc.chDataLiteral,
Worksheets("Test").Cells(i, 9)
    End With
```

```
With Chart1.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With

With Chart1.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart2.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With

With Chart2.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart3.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Utility"
    .Title.Font.Size = 6
End With

With Chart3.Axes(owc.chAxisPositionBottom)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
    .MajorUnit = 0.2
    .HasTitle = True
    .Title.Caption = "Expense"
    .Title.Font.Size = 6
End With

With Chart4.Axes(owc.chAxisPositionLeft)
    .Scaling.Maximum = 1
    .Scaling.Minimum = 0
```

```
        .MajorUnit = 0.2
        .HasTitle = True
        .Title.Caption = "Utility"
        .Title.Font.Size = 6
    End With

    With Chart4.Axes(owc.chAxisPositionBottom)
        .Scaling.Maximum = 1
        .Scaling.Minimum = 0
        .MajorUnit = 0.2
        .HasTitle = True
        .Title.Caption = "Expense"
        .Title.Font.Size = 6
    End With

    FRStakeholderComboBox.Value = ""
    FRUtilityTextBox.Text = ""
    FRExpenseTextBox.Text = ""

    UtilityTextBox1.Text = Worksheets("Test").Cells(i, 3)
    UtilityTextBox2.Text = Worksheets("Test").Cells(i, 5)
    UtilityTextBox3.Text = Worksheets("Test").Cells(i, 7)
    UtilityTextBox4.Text = Worksheets("Test").Cells(i, 9)

    ExpenseTextBox1.Text = Worksheets("Test").Cells(i, 2)
    ExpenseTextBox2.Text = Worksheets("Test").Cells(i, 4)
    ExpenseTextBox3.Text = Worksheets("Test").Cells(i, 6)
    ExpenseTextBox4.Text = Worksheets("Test").Cells(i, 8)

    Dim scenario_num As Integer
    scenario_num = Worksheets("Test").Cells(i, 1)

    i = 2

    Do Until (Worksheets("Test").Cells(i, 11) = scenario_num)
        i = i + 1
    Loop


    'fill in Attribute List
    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    Dim Array_A() As String
    ReDim Array_A(0 To numrows + 1, 0 To 2)
    Dim Array_C(20) As String
    Dim Array_D(20) As String

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"
```

```
    Dim b As Integer

    b = 0

    Do While b < numrows
        Array_A(b + 1, 0) = Worksheets("Test").Cells(i, 12).Value
        Array_C(b + 1) = Worksheets("Test").Cells(i, 13).Value
        Array_D(b + 1) = Worksheets("Test").Cells(i, 14).Value
        Array_A(b + 1, 1) = Array_C(b + 1) & " " & Array_D(b + 1)
        b = b + 1
        i = i + 1
    Loop

    StakeholderAnalysis.AttributesListBox.List = Array_A
End Sub

Private Sub CommandButtonShowAll_Click()
    Load_StakeholderAnalysis
End Sub

Private Sub CommandButtonShowGraph_Click()

    'Deletes charts
    Dim ch As Chart

    For Each ch In ActiveWorkbook.Charts
        With ch
            If .Name = "Stakeholder 1" Or .Name = "Stakeholder 2" Or
.Name = "Stakeholder 3" Or .Name = "Stakeholder 4" Then
                ch.Delete
            End If
        End With
    Next

    Dim a As Integer
    Dim cellRange As String

    Select Case FRStakeholderComboBox.Value
        Case "Stakeholder 1"
            a = 2
            cellRange = "B2:C42"
        Case "Stakeholder 2"
            a = 4
            cellRange = "D2:E42"
        Case "Stakeholder 3"
            a = 6
            cellRange = "F2:G42"
        Case "Stakeholder 4"
            a = 8
            cellRange = "H2:I42"
        Case Else
            MsgBox ("Error.  Please select a stakeholder.")
    End Select

    Charts.Add

    With ActiveChart
```

```
        .ChartType = xlXYScatter
        .Name = FRStakeholderComboBox.Value

        .SetSourceData Source:=Sheets("Test").Range(cellRange),
PlotBy:=x1Columns
        .HasTitle = True
        .ChartTitle.Characters.Text = FRStakeholderComboBox.Value & ":
Multiple Attribute Scenario Chart"
        .Axes(1, 1).MinimumScale = 0
        .Axes(1, 1).MaximumScale = 1
        .Axes(2, 1).MinimumScale = 0
        .Axes(2, 1).MaximumScale = 1
        .Axes(1, 1).HasTitle = True
        .Axes(1, 1).AxisTitle.Characters.Text = "Expense"
        .Axes(2, 1).HasTitle = True
        .Axes(2, 1).AxisTitle.Characters.Text = "Utility"
        .Legend.Delete
    End With

    Unload StakeholderAnalysis
    StakeholderAnalysis.Hide
End Sub

Private Sub CommandButtonUpdate_Click()

    ReDim ArrayAttribute(20)
    ReDim ArrayValue(20)
    ReDim ArrayUnit(20)
    Dim i As Integer
    Dim j As Integer

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    i = 2

    Do Until (Worksheets("Test").Cells(i, 11) =
Worksheets("Test").Cells(1, 14))
        i = i + 1
    Loop

    j = 0

    Do While j < numrows
        ArrayAttribute(j) = Worksheets("Test").Cells(j + i, 12).Value
        ArrayValue(j) = Worksheets("Test").Cells(j + i, 13).Value
        ArrayUnit(j) = Worksheets("Test").Cells(j + i, 14).Value
        j = j + 1
    Loop

    Dim Array_A() As String
```

```
    ReDim Array_A(0 To numrows + 1, 0 To 2)

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"

    i = 0

    Do While i < numrows
        Array_A(i + 1, 0) = ArrayAttribute(i)
        Array_A(i + 1, 1) = ArrayValue(i) & " " & ArrayUnit(i)
        i = i + 1
    Loop

    AttributeScenario.AttributesListBox.List = Array_A

    Load AttributeScenario
    AttributeScenario.show
End Sub

Public Function CalcUtility(Stakeholder As String) As Double

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    Dim Array_att(20) As String
    Dim i As Integer
    i = 0

    Do While i < numrows
        Array_att(i) = Att_range(i + 1).Value
        i = i + 1
    Loop

    'this allows to find index of newest Attribute Scenario
    i = 2

    Do Until (Worksheets("Test").Cells(i, 11) =
Worksheets("Test").Cells(1, 14))
        i = i + 1
    Loop

    'Finds all Ui(Xi) values
    Dim UtilVal(20) As Double
    Dim AttVal As Double

    Dim j As Integer
    Dim rise As Double
    Dim run As Double
    Dim yint As Double

    j = 0
```

```
    g = 0

    Do While j < numrows
        AttVal = Worksheets("Test").Cells(j + i, 13)
        Do While AttVal > Worksheets(Array_att(j)).Cells(g + 2, 23)
            g = g + 1
        Loop
        'now at cell k, the val is great than AttVal

        If g = 0 Then
            rise = Worksheets(Array_att(j)).Cells(g + 2, 24)
            run = Worksheets(Array_att(j)).Cells(g + 2, 23)
            If run = 0 Then
                rise = 0
                run = 1
            End If
        Else
            rise = Worksheets(Array_att(j)).Cells(g + 2, 24) -
Worksheets(Array_att(j)).Cells(g + 1, 24)
            run = Worksheets(Array_att(j)).Cells(g + 2, 23) -
Worksheets(Array_att(j)).Cells(g + 1, 23)
        End If

        yint = Worksheets(Array_att(j)).Cells(g + 2, 24) - (rise / run)
* Worksheets(Array_att(j)).Cells(g + 2, 23)
        UtilVal(j) = AttVal * rise / run + yint
        g = 0
        j = j + 1
    Loop


    'Finds all k corner points
    Dim K(20) As Double

    j = 0
    g = 0

    Do While j < numrows
        Do While Worksheets(Array_att(j)).Cells(g + 2, 17) <> "I"
            g = g + 1
        Loop

        K(j) = Worksheets(Array_att(j)).Cells(g + 2, 16)
        g = 0
        j = j + 1
    Loop


    'Find Kconstant
    Dim Kconstant As Double
    Dim temp As Double
    Dim zeroval As Double
    Dim smallest As Double
    Dim smaller As Double
    Dim Ksave1 As Double
    Dim Ksave2 As Double
```

```
j = 0
smallest = 1000
smaller = 1000
Ksave1 = 0
Ksave2 = 0

Kconstant = -1
temp = 1

Do While Kconstant < 1.1
    Do While j < numrows
        temp = (Kconstant * K(j) + 1) * temp
        j = j + 1
    Loop

    zeroval = Abs(temp - (Kconstant + 1))

    If zeroval < smallest Then
        smaller = smallest
        Ksave2 = Ksave1

        smallest = zeroval
        Ksave1 = Kconstant
    ElseIf zeroval < smaller Then
        smaller = zeroval
        Ksave2 = Kconstant
    End If

    Kconstant = Kconstant + 0.1
    temp = 1
    j = 0
Loop

Dim Kfinal As Double
j = 0
smallest = 1000
temp = 1
Ksave1 = Round(Ksave1, 2)
Ksave2 = Round(Ksave2, 2)

Do While Ksave1 < Ksave2 + 0.01
    If Ksave1 = 0 Then
        Ksave1 = Ksave1 + 0.01
    End If

    Do While j < numrows
        temp = (Ksave1 * K(j) + 1) * temp
        j = j + 1
    Loop

    zeroval = Abs(temp - (Ksave1 + 1))

    If zeroval < smallest Then
        smallest = zeroval
        Kfinal = Ksave1
    End If
```

```
        Ksave1 = Ksave1 + 0.01
        temp = 1
        j = 0
    Loop

    Dim finalUtil As Double

    j = 0
    temp = 1

    Do While j < numrows
        temp = (Kfinal * K(j) * UtilVal(j) + 1) * temp
        j = j + 1
    Loop

    finalUtil = Round(((temp - 1) / Kfinal), 2)
    CalcUtility = finalUtil

End Function

Public Function CalcExpense(Stakeholder As String) As Double
    Dim randomnumber As Double

    randomnumber = Round((0 + Rnd * (0.15)), 2)
    CalcExpense = CalcUtility(Stakeholder) + randomnumber

End Function
Attribute Scenario Update Form Code
Private Sub CommandButtonCancel_Click()
    AttributeScenario.Hide
    Unload AttributeScenario
End Sub

Private Sub CommandButtonScenarioComplete_Click()
    Dim i As Integer

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    i = 2

    Do Until (Worksheets("Test").Cells(i, 11) =
Worksheets("Test").Cells(1, 14))
        i = i + 1
    Loop

    'i is the row where the new Attribute Scenario can be put down
    i = i + numrows
    'add one more row for spacing of attribute sets
    i = i + 1
```

```
    'the empty space index is incremented
    Worksheets("Test").Cells(1, 14) = Worksheets("Test").Cells(1, 14) +
1

    Dim j As Integer

    j = 0

    Do While j < numrows
        Worksheets("Test").Cells(i, 11) = Worksheets("Test").Cells(1,
14)
        Worksheets("Test").Cells(i, 12) = ArrayAttribute(j)
        Worksheets("Test").Cells(i, 13) = ArrayValue(j)
        Worksheets("Test").Cells(i, 14) = ArrayUnit(j)
        j = j + 1
        i = i + 1
    Loop

    'number the Attribute Scenario for naming of Utility and Expenses

    Dim tmp As Integer

    tmp = Worksheets("Test").Cells(1, 14) + 1

    Worksheets("Test").Cells(tmp, 1) = Worksheets("Test").Cells(1, 14)

    Worksheets("Test").Cells(tmp, 2) =
StakeholderAnalysis.CalcExpense("Stakeholder1")
    Worksheets("Test").Cells(tmp, 3) =
StakeholderAnalysis.CalcUtility("Stakeholder1")
    Worksheets("Test").Cells(tmp, 4) =
StakeholderAnalysis.CalcExpense("Stakeholder2")
    Worksheets("Test").Cells(tmp, 5) =
StakeholderAnalysis.CalcUtility("Stakeholder2")
    Worksheets("Test").Cells(tmp, 6) =
StakeholderAnalysis.CalcExpense("Stakeholder3")
    Worksheets("Test").Cells(tmp, 7) =
StakeholderAnalysis.CalcUtility("Stakeholder3")
  Worksheets("Test").Cells(tmp, 8) = StakeholderAnalysis.CalcExpense("Stakeholder4")
    Worksheets("Test").Cells(tmp, 9) =
StakeholderAnalysis.CalcUtility("Stakeholder4")

    Load_StakeholderAnalysis

    AttributeScenario.Hide
    Unload AttributeScenario
End Sub

Private Sub CommandButtonUpdateAttribute_Click()

    Dim i As Integer

    i = 0

    Do Until (ArrayAttribute(i) = AttributesListBox.Value)
        i = i + 1
    Loop
```

```
    UpdateAttr.PromptLabel.Caption = "Current value for " &
ArrayAttribute(i) & " is " & ArrayValue(i) & " " & ArrayUnit(i)
    UpdateAttr.Caption = ArrayAttribute(i)
    UpdateAttr.MinMaxLabel.Caption = "(Min value, Max value) : (" &
Find_Low(ArrayAttribute(i)) & ", " & Find_High(ArrayAttribute(i)) & ")"

    Load UpdateAttr
    UpdateAttr.show
End Sub
Update Attribute Form Code
Private Sub CommandButtonCancel_Click()
    UpdateAttr.Hide
    Unload UpdateAttr
End Sub

Private Sub CommandButtonUpdate_Click()
    Dim i As Integer

    i = 0

    Do Until (ArrayAttribute(i) = UpdateAttr.Caption)
        i = i + 1
    Loop

    ArrayValue(i) = TextBoxAttributeValue.Text
    ArrayUnit(i) = TextBoxAttributeUnit.Text

    UpdateAttributeScenario

    UpdateAttr.Hide
    Unload UpdateAttr
End Sub

Sub UpdateAttributeScenario()

    Dim i As Integer

    Dim att_list As Range
    Dim numrows As Integer
    Dim Att_range As Range
    Set att_list = Worksheets("Home").Range("Attribute_list")
    Set Att_range =
ActiveWorkbook.Names("Attribute_list").RefersToRange
    numrows = Att_range.Rows.Count 'numrows contains the number of
Attributes

    Dim Array_A() As String
    ReDim Array_A(0 To numrows + 1, 0 To 2)

    Array_A(0, 0) = "Attribute Name"
    Array_A(0, 1) = "Attribute Value"

    i = 0

    Do While i < numrows
```

```
        Array_A(i + 1, 0) = ArrayAttribute(i)
        Array_A(i + 1, 1) = ArrayValue(i) & " " & ArrayUnit(i)
        i = i + 1
    Loop

    AttributeScenario.AttributesListBox.List = Array_A

    Load AttributeScenario
End Sub
Find Relations Code
Sub Find_Relations(Stakeholder As String, Utility As String, Expense As
String)

    Dim incr As Integer
    Dim attr As Integer
    Dim i As Integer

    Select Case Stakeholder
        Case "Stakeholder 1"
            incr = 2
        Case "Stakeholder 2"
            incr = 4
        Case "Stakeholder 3"
            incr = 6
        Case "Stakeholder 4"
            incr = 8
    End Select


    attr = 0

    For i = 1 To 40
        If (Utility = Worksheets("Test").Cells(i + 1, incr + 1)) And
(Expense = Worksheets("Test").Cells(i + 1, incr)) Then
            attr = Worksheets("Test").Cells(i + 1, 1)
        End If
    Next

    If attr = 0 Then
        MsgBox "The entered Utility and Expense Values do not correlate
to an Attribute set for the specified Stakeholder"
    ElseIf attr <> 0 Then
    'display relevant numbers
    End If
```

### 11.3.8 Network Mode

```
Sub Allow_Observe_On()
    'need to address deleting of old file, not just overwriting it

    Dim File_Name As String

    Allow_Observe = 1
    Observe = 0
    Output_Index = 1
    IntFile = FreeFile

    File_Name = InputBox("Please enter desired file for observation data.
(ex. c:\test.txt)", "Open File")

    If File_Name <> "" Then
        Open File_Name For Random As #IntFile
    Else
        File_Name = MsgBox("Allow Observe is off.", vbOKOnly, "Allow
Observe")
    End If

'    Open "c:\test.txt" For Random As #IntFile
End Sub

Sub Allow_Observe_Off()
    Allow_Observe = 0
    Observe = 0
    Output_Index = 1
    Close #IntFile
End Sub

Sub Observe_On()

    Dim Command_Run As String
    Observe = 1
    Allow_Observe = 0
    Input_Index = 1
    IntFile = FreeFile

    Open "c:\test.txt" For Random As #IntFile

    Command_Run = "test"
Do While Command_Run <> ""
    Get #IntFile, Input_Index, Command_Run
    Command_Run = Command_Run
    Command_Select (Command_Run)
'    Adams_module.Start_Interview_click
'    AttNav.ListBoxAttributes.Value = "Latency"
'    AttNav.CommandButtonMulti_Click
    Input_Index = Input_Index + 1
Loop

End Sub

Sub Observe_Off()
```

```
    Observe = 0
    Allow_Observe = 0
    Input_Index = 1
    Close #IntFile
End Sub

Sub Command_Select(strCommand As String)
        Select Case strCommand
            Case "Adams_module.Start_Interview_click"
                Adams_module.Start_Interview_click
            Case "AttNav.CommandButtonMulti_click"
                'Selecting which Attribute
                Input_Index = Input_Index + 1
                Get #IntFile, Input_Index, strCommand
'                AttNav.ListBoxAttributes.Value = strCommand
                AttNav.ListBoxAttributes.Value = "Latency"

                AttNav.CommandButtonMulti_Click
            Case Else
        End Select

End Sub
```