SOFTWARE IMPLEMENTATION OF HIERARCHICAL DECISION MODELING AND HIERARCHICAL DECISION MODELING SENSITIVITY ANALYSIS

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

By

ADNAN MAHMOOD

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

ADVISOR: DR. HONGYI CHEN

December 2011

ACKNOWLEDGMENTS

I am heartily thankful to my supervisor, Dr. Hongyi Chen, whose encouragement, guidance and support from the preliminary to the concluding level enabled me to develop an understanding of the subject and also for giving me the privilege of working under her supervision.

I would also like to thank Ms. Li, Jingrui, Mr. Khamesra, Utkarsh, Mr. Acharya, Nikhil, Mr. Natarajan, Ganapathy, Mr. Singh, Siddharth, Mr. Ahmad, Mushtaq, Mr. Taha, Muhammad and Ms. Khan, Roshna without whose support, this task wouldn't have been achieved.

Adnan Mahmood

ABSTRACT

Decision making can be regarded as the mental processes (cognitive process) resulting in the selection of a course of action among several alternative scenarios. Every decision making process produces a final choice. The output can be an action or an opinion of choice. To facilitate multi-criteria multi-level decision makings, different algorithms including the Hierarchical Decision Modeling (HDM) and its Sensitivity Analysis (HDM SA) have been developed. This research project implements the HDM and HDM SA algorithms developed by Dr. Dundar F. Kocaoglu and Dr. Hongyi Chen in software application to ease the cognitive burden of the users and assist decision makers in effective decision making. The project is carried out from the conception stage to the final manifestation of the software in a planned and structured process. Its application is demonstrated through two examples, one using a complete hierarchy to assess technologies and the other using an incomplete hierarchy to create a technology development envelope. A user guide of the software application is included at the end of this thesis.

Table of Contents

LIST OF FIGURES

1. INTRODUCTION

Decision making is a complicated process which results in selecting a course of action when subjected to multiple alternatives. The output is usually an action or an opinion of choice. From a cognitive perspective, the decision making process must be regarded as a continuous process integrated in the interaction with the environment. It is all about choosing from several alternatives or options and then taking the action to generate a particular result. (Hongyi Chen; Ho J.C; Kocaoglu 2009)

To facilitate decision making, a large number of decision making models and software tools have been developed. Many Decision Making softwares are based on multi-criteria decision analysis (MCDA) such as: Analytic Hierarchy Process (AHP), Multi-Attribute Value Theory (MAVT), Hierarchical Decision Model (HDM)), Multi-Attribute Utility Theory (MAUT), Multi-Attribute Global Inference of Quality (MAGIQ). Following are some of the softwares that are based on the Multi Criterion Decision Modeling available over the web.

- SuperDecisions (www.superdecisions.com)
  A *SuperDecisions* model consists of clusters of elements (or nodes), rather than elements (or nodes) arranged in levels. The simplest hierarchical model has a goal cluster containing the goal element, a criteria cluster containing the criteria elements and an alternatives cluster containing the alternative elements. When clusters are connected by a line it means nodes in them are connected. The cluster containing the alternatives of the decision must be named A*lternatives.* Nodes and Clusters are organized alphabetically in the calculations, so an easy way to control the order is to preface the names with numbers.

- MindDecider (www.minddecider.com)
  It is effective decision making software for everyday planning and task management. It has a simple and intuitive interface, easy to work and high functionality based on mind mapping concept, multiple criteria decision analysis (MCDA) and analytic hierarchy process (AHP). MindDecider has a universal scope of application. It is equally useful in optimization of your household budgets or making ethical decisions, planning your workday or performing complex measurements. Clearly, mathematics has a lot to do with programs of such kind. MindDecider, however, successfully hides all the math stuff inside so that a common user could easily enjoy the program without any special technical knowledge. MindDecider may even go without numbers just answering your questions and finding right values.

- 1000 Minds (www.1000minds.com)
  1000Minds decision-support software is internationally recognized for its scientific validity and user-friendliness. 1000Minds applies their own patented PAPRIKA (Potentially All Pairwise RanKings of all possible Alternatives) method for Multi-Criteria Decision-Making and Conjoint Analysis. Notwithstanding its mathematical and technical sophistication, 1000Minds is very intuitive and user-friendly. Depending on the mode selected (at the 'start / mode' step in 1000Minds), 1000Minds supports these three (related) main activities: Discover decision-makers' preference values, Rank or prioritize alternatives entered into 1000Minds (and all hypothetically possible alternatives, if desired) and select alternatives, subject perhaps to a budget constraint – including, when appropriate, considering their 'value for money' and allocating resources.

- Decision Lab (www.visualdecision.com)
  Decision Lab 2000 is a multi-criteria analysis and decision-making software. Its advanced features will provide evidence of strengths vs weaknesses, of conflicts and

consensus. By trying your own "What if?" scenarios, you will really improve the quality and the reliability of your decision-making processes.

- Simlab (www.cst.com/simlab)

  SimLab provides a free development framework for Sensitivity and Uncertainty Analysis. SimLab is a professional tool for model developers, scientists and professionals, to learn, use and exploit global uncertainty and sensitivity analysis techniques. It encourages free non-commercial use. SimLab provides a reference implementation of the most recent global sensitivity analysis techniques. SimLab is an ongoing project where new improvements progress on a regular basis since 1985.

Hierarchical Decision Model (HDM) is a method by which the problem is first decomposed into a hierarchy of more easily comprehended sub-problems to be analyzed independently. A hierarchy is organized into different levels with a number of decision elements reside on each level. At each level, the decision elements are connected to other decision elements on the level above or below them. The elements of the hierarchy can relate to any aspect of the decision problem that applies to the decision at hand. Judgment quantification methods are used to derive local contributions which are supplied as intermediate input to the hierarchical model. These local contributions are then calculated from the pairwise comparisons. These local contributions are subject to variations as the environment changes.

The HDM model was developed at the same time when the Analytical Hierarchical Process (AHP) concept was forwarded by Saaty in dealing with multiple levels of decisions. By organizing and assessing alternatives against a hierarchy of multifaceted objectives, these models provide effective means to deal with complex decision making (Saaty 1980). Based on AHP, software is available over the web known as the “Expert Choice”. The software is designed to help decision-makers overcome the limits of the human mind to synthesize qualitative and quantitative inputs from multiple stakeholders. The result is sound analysis and clear thinking that

everyone in the organization agrees with and understands. Some of the main features are that it brings structure and measurement to your decision-making processes which helps you determine strategic priorities and ensure decisions are tightly aligned. It also communicates priorities and builds consensus around decisions and documents and justifies your decisions.

But there is an issue with the AHP model which is known as the Rank Reversal problem. It is an axiom of certain decision theories that, when new alternatives/elements are included to a decision problem, the ranking/rating of the old alternatives/elements must not change or it can be said that "rank reversal" must not occur. The rationality of this axiom for all applications is questionable, since there are real-world examples where adding new alternatives can change the rank of the old ones. There are also situations where it is unacceptable for the rank of existing elements to change when a new element is added for consideration. For this reason, HDM allows a better, easier, and more efficient identification of selection criteria. Another advantage of HDM is its methods of weighing and analysis, which drastically reduces the decision cycle.

The applications of HDM and its variants including AHP to complex decision situations have numbered in thousands, (J.E. de Steiguer 2003) and have produced extensive results in problems involving planning, resource allocation, priority setting and selection among alternatives. It is also in financial applications, risk analysis, signal processing, neural networks, forecasting, total quality management, business process re-engineering and the quality function deployment.

Since the decision support is obtained by the final ranked alternatives based on the local contributions. In order to develop a strategy to meet the various unforeseen events, one needs to conduct a Sensitivity Analysis (SA) on the HDM results. Sensitivity Analysis is the process of analyzing the impact of uncertainty on the output of a model that is subjected to different sources of variation in the input of the model. (Hongyi Chen; Ho J.C; Kocaoglu 2009). It is very helpful

for the decision maker if he/she wants to determine the impact of a particular variable on the actual outcome if it differs from what was previously assumed. By creating a set of scenarios, the impact of the changes(s) on the target variable can be analyzed and determined (Hongyi Chen; Ho J.C; Kocaoglu 2009)

SA has played a significant role in the effective use and the implementation of the different quantitative models (Dantzig 1963). It has also played different roles in different scenarios for the decision-making process (Evans 1984). It can even provide information which is more significant and useful than simply knowing the model solution (Phillips 1976). It has the following benefits (Hongyi Chen; Ho J.C; Kocaoglu 2009)

- To witness the impact at changes at different levels on the decision at the operational level
- To test the robustness of the system
- To Identify the critical elements at different levels of the decision hierarchy
- To generate different possible outcomes under different conditions
- To help a group to reach consensus
- To answer all the 'What if questions'

The HDM SA algorithm was proposed by H. Chen and D.F. Kocaoglu to study a hierarchical decision model's robustness to changes in every local contribution matrix at different levels and their effects on the ranks of the decision alternatives. The algorithm is independent from the pairwise comparison scales and judgment quantification techniques and is applicable to all hierarchical decision models based on an additive relationship in calculating the overall contributions.

The HDM SA also generates other parameters like the tolerance, TSC (Total Sensitivity Coefficient), OPSC (Operating Point Sensitivity Coefficient), the allowable and the feasible

regions for the perturbations of the contribution values to evaluate their impact on the overall contribution vector. “Allowable range of perturbation” determines the thresholds of changes to the contribution values without changing the rank order of technology alternatives while “tolerance” is the range in which the contribution value can change without altering the rank order of decision alternatives. (H. Chen and D.F. Kocaoglu)

As a part of the thesis, a new corollary for HDM SA was also developed to deal with contribution values. The new corollary is only applicable to the last level. Details about the corollary and its implementation are presented later.

The cognitive burden of using and understanding the HDMSA algorithm has been heavy because of the complex mathematical notations in the theorems and corollaries. For this reason, a software application is needed to provide the end-users (decision makers) with easier decision support.

Therefore, the main purpose of this research is to provide user-friendly decision support software that facilitates the decision makers in complex decision making. Apart from just giving them the rankings of the decision alternatives, it also performs the sensitivity analyses, both one-way and M-way, based on the HDM SA algorithm.

The first release of the application is a windows based application that is developed on the Microsoft .Net platform using C# (C-sharp) as the main coding language. Also, in the first release, we have provided the user to either go with the “pair wise comparison” or the “direct input” options through which the data can be input into the system. The application has the function of executing both the kinds of hierarchies, i.e. a complete and an incomplete Hierarchy. The final output is generated on different platforms. The local contributions, individual matrices, global and overall contributions are all generated on an excel file while the results of the sensitivity analyses are generated on a text file (at least in the first release).

1.1 Overview of the method and the application

For any hierarchical decision model, the basic structure of the hierarchy is presented in the MOGSA form. This structure was first used by Cleland and Kocaoglu. This model consists of 5 levels of decision elements namely Mission, Objectives, Goals, Strategies and Actions. It can be seen in the Figure 1.1.

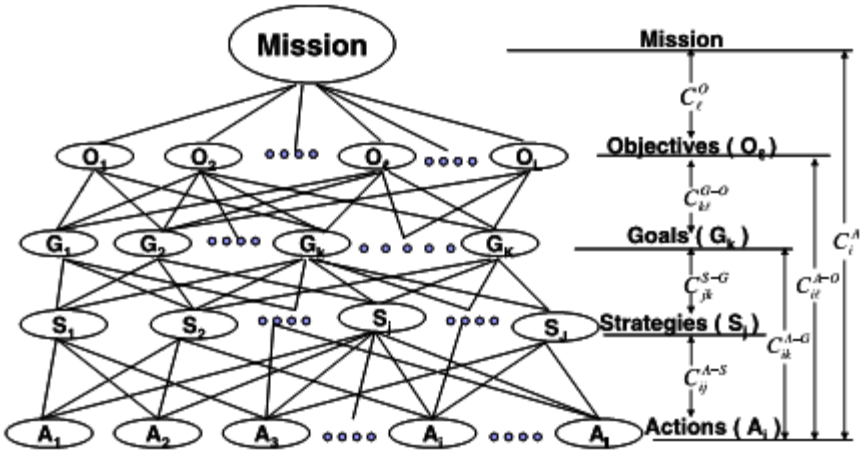


Figure 1.1 MOGSA Model (Ho 2004)

Where O_l is the lth objective and l = 1, 2, 3…10

G_k is the kth goal and k = 1, 2, 3…10

S_j is the jth strategy and j = 1, 2, 3…10

A_i is the ith action and i = 1, 2, 3…10

As it can be seen in Figure 1.1, there are 5 levels that are involved in the decision making. The last level includes all the alternatives that are to be ranked according to the pair wise comparison or the direct input. The C_{xy}^{A-B} is the contribution of the xth decision element A to the yth decision element B. All these contributions are calculated by the judgment quantification methods used in

HDM. The software can handle up to 10 levels with 10 elements on each level to keep up with the performance requirements.

The application was developed in a way that it starts off with creating the model structure in the beginning, then calculating the contributions of the decision elements, then ranking the decision alternatives and finally performing sensitivity analyses in the end. A glimpse of the information flow around the system can be seen in Figure 1.2.

Figure 1.2 Overall Process

1.1.1 Hierarchical Decision Model (HDM)

In HDM, the problem is structured into a hierarchy. All the judgments that reflect ideas or preferences are represented in meaningful numbers which are used to calculate the priorities of the elements of the hierarchy. Then these priorities are synthesized to determine the overall outcome. Those outcomes are further subjected to sensitivity analysis to see their sensitivity to changes in the judgments. The HDM algorithm includes six general steps: (Kocaoglu, 1976 #2)

- Decompose the problem to a hierarchical model
- Compare the decision elements in pairs
- Calculate the local contribution matrices
- Combine group opinions
- Aggregate all local contributions to overall contribution vector
- Rank decision alternatives

There can be two types of hierarchies; a complete hierarchy or an incomplete hierarchy. A hierarchy is said to be complete when all the elements in a level are evaluated in terms of all the elements in the above level. Otherwise, it is said to be an incomplete hierarchy. Examples of a complete and incomplete hierarchy can be seen in Figure 1.3 and Figure 1.4 respectively:

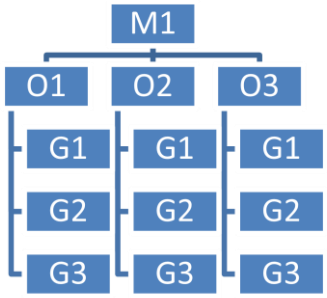


Figure 1.3 Complete Hierarchy (Ho 2004)

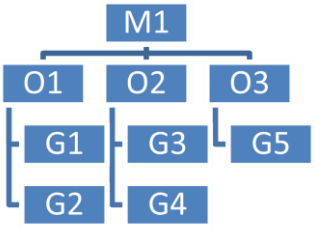


Figure 1.4 Incomplete Hierarchy (Ho 2004)

1.1.1.1 Pair Wise Comparison

This is the second step and it occurs when decision elements at a certain level are being compared with each other. In this case, the end user assigns a certain weight to each decision element based on its relative contribution to an element on the next higher level. There are mainly two approaches used at this step: constant sum measurement or the 1-9 scale with verbal representation. HDM uses the constant sum measurement which refers to the procedure for expressing judgment as a total of 100 points which are divided between the two elements being

compared. For example, when four elements “A”, “B”, “C” and “D” are pairwise compared, the relative weights given to each pair can be:

A : B | 40 : 60 || A : C | 57 : 43 || A : D | 75 : 25

B: C | 38 : 62 || B : D | 20 : 80

C: D | 50 : 50

1.1.1.2 Constant Sum Matrices

The next step is the creation of Matrices. The first Matrix is the Matrix A in which, all the values from the pair wise comparison step are entered in accordance with their pairs. All the values in the diagonal are not considered. The next one is the Matrix B in which all the values in the diagonal are set to 1. All the remaining values are obtained by dividing the values in Matrix A by the values across the diagonal. From Matrix B, we derive Matrix C by dividing the values in a column by values in the next column on right as shown as in the Table 1.3.

Matrix A	A	B	C	D
A		60	75	80
B	40		50	62
C	25	50		57
D	20	38	43	

Table 1.1 Matrix A

Matrix B	A	B	C	D
	1	1.5	3	4
	0.67	1	1	1.63
	0.33	1	1	1.33
	0.25	0.61	0.75	1

Table 1.2 Matrix B

Matrix C	A/B	B/C	C/D
	0.67	0.5	0.75
	0.67	1	0.61
	0.33	1	0.75
	0.41	0.81	0.75

Mean	0.52	0.83	0.72

Table 1.3 Matrix C

1.1.1.3 Normalization

The next process is the normalization process in which the values that are obtained from Matrix C are normalized. Since we calculated the averaged value of A/B, B/C and C/D (as can be seen in Figure 1.5), Assume that D is equal to 1, we can calculate the value of C equaling to 0.72. In the same way, other values are calculated and in the end the sum of all the values is taken. The last step is where all the values obtained are divided by the sum which gives the final result for the normalization process. Figure 1.6 shows the details:

Matrix C	A/B	B/C	C/D
	0.67	0.5	0.75
	0.67	1	0.61
	0.33	1	0.75
	0.41	0.81	0.75

Mean	0.52	0.83	0.72

Suppose D=1,

C = 0.72 (C/D) and B = 0.83 * 0.72 = 0.5976

and A = 0.52 * 0.5976 = 0.3108

Sum = 2.68284

Normalized values are

A = 0.3108/2.6824 = 0.12

B = 0.5976/2.6824 = 0.23

C = 0.72/2.6824 = 0.27

D = 1/2.6824 = 0.38

Figure 1.6 Normalization

1.1.1.4 Orientation

This is the step where different orientations are generated: For example, if we have 4 elements "A", "B", "C" and "D", there can be 24 different orientations of this combination like ABCD, ABDC, and ACDB etc. All the above processes from generating matrix A through normalization are executed for each orientation. In the end, a mean is taken of the entire final A, B, C and D

values to get the final weights or the local contribution of A, B, C and D elements. Figure 1.7 shows a sample table of these orientations:

Orientation	A	B	C	D
ABCD	0.120	0.230	0.270	0.380
ABDC	0.120	0.230	0.270	0.380
ACBD	0.120	0.230	0.270	0.380
ACDB	0.120	0.210	0.280	0.390
ADBC	0.100	0.220	0.270	0.410
ADCB	0.100	0.220	0.270	0.400
BACD	0.110	0.220	0.260	0.380
BADC	0.110	0.230	0.280	0.400
BCAD	0.100	0.210	0.270	0.390
BCDA	0.120	0.210	0.270	0.390
BDCA	0.120	0.220	0.270	0.390
BDAC	0.100	0.230	0.280	0.400
CABD	0.120	0.210	0.260	0.410
CADB	0.110	0.210	0.260	0.400
CBAD	0.120	0.220	0.260	0.400
CBDA	0.100	0.230	0.270	0.410
DACB	0.120	0.230	0.280	0.390
DABC	0.100	0.220	0.270	0.390
DBAC	0.110	0.210	0.260	0.400
DBCA	0.120	0.210	0.260	0.410
DCAB	0.120	0.220	0.270	0.390
DCAB	0.110	0.230	0.280	0.390
DCBA	0.100	0.210	0.280	0.390
Mean	**0.1**	**0.2**	**0.3**	**0.4**

Figure 1.7 Orientations

1.1.1.5 Inconsistency Check

To make sure, the pair-wise comparison judgments obtained are consistent, the inconsistency check is performed by calculating an inconsistency index. The value of the index should be less than 0.04; otherwise the pairwise comparison needs to be performed again. The following formula is followed to calculate the index:

$$\text{Inconsistency Index} = (\text{Sum of Variances} / \text{Number of elements})^{1/2}$$

$$= ((\text{Variance of A} + \text{Variance of B} + \text{Variance of C} + \text{Variance of D}) / 4)^{1/2}$$

1.1.1.6 Global Contributions and Overall Contributions

If the consistency check is passed, at the next step, the local contributions of the elements will be combined using an additive relationship to calculate the global and then the overall contribution values.

The overall contributions are the final output of the HDM by which the decision alternatives are ranked. $C_A^i{}'s$ are the overall contributions and C_{ik}^{A-G} 's and C_{ik}^{A-O} 's are the global contributions.

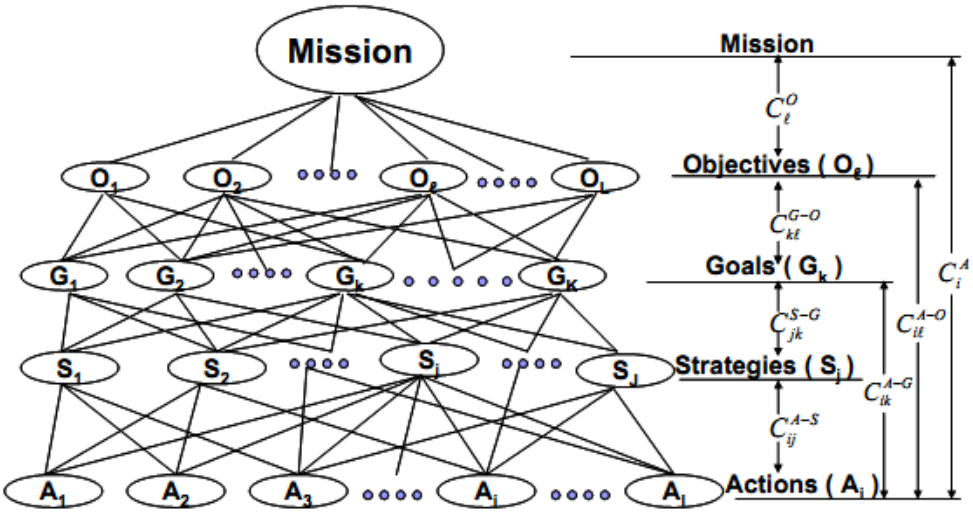


Figure 1.8 MOGSA Model that shows the global and overall contributions

The following formula is followed to calculate the global and the overall contributions:

$$C_A^i = \sum_{l=1}^{L} C_{il}^{A-O} * C_l^O = \sum_{l=1}^{L}\sum_{k=1}^{K} C_{ik}^{A-G} * C_{kl}^{G-O} * C_l^O$$

$$= \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{j=1}^{J} C_{ij}^{A-S} * C_{jk}^{S-G} * C_{kl}^{G-O} * C_l^O$$

1.1.2 Hierarchical Decision Modeling Sensitivity Analysis

As mentioned before, the HDM SA algorithm (Hongyi Chen; Ho J.C; Kocaoglu 2009) calculates parameters. Among them, the tolerance is defined as the allowable range in which a contribution value can vary without changing the rank order of the decision alternatives. It is calculated using

a contribution's base value plus and minus the two threshold values of the allowable range of the perturbation on this contribution.

The main logic regarding the allowable range that has been followed throughout is: suppose originally Ar ranks higher than At indicating $C_r^A > C_t^A$; the rank order of Ar and At will be preserved if the new value of C_r^A is still greater than or equal to the new value of C_t^A. Therefore, the relationship between the perturbation(s) and the contributions can be represented with an expression containing the original contributions and induced perturbations. Several groups of theorems and corollaries have been developed in (Hongyi Chen; Ho J.C; Kocaoglu 2009) and the ones implemented by the software are presented here.

1.1.2.1 Top Level Sensitivity Analysis

This theorem defines an M dimensional allowable region for M perturbations induced on the first level contribution Vector C_l^O. As long as the values are within the allowable region, the rank orders will be preserved (Chen, 2007 #1)

Theorem1 Let $P_{l^*}^O \left(-C_{l^*}^O \leq P_{l^*}^O \leq 1 - C_{l^*}^O\right)$ denote the perturbation induced on one of the $C_l^{o'}s$, which is $C_{l^*}^O$; the original ranking of A_r and A_{r+n} will not reverse if;

$$\lambda \geq P_{l^*}^O \lambda^O$$

Where $\lambda = C_r^A - C_{r+n}^A$

$$\boldsymbol{\lambda = C_{r+n,l^*}^{A-0} - C_{rl^*}^{A-0} - \sum_{l=1,l\neq l^*}^{L} C_{r+n,l^*}^{A-0} \times \frac{C_l^o}{\sum_{l=1,l\neq l^*}^{L} C_l^o} + \sum_{l=1,l\neq l^*}^{L} C_{rl^*}^{A-0} \times \frac{C_l^o}{\sum_{l=1,l\neq l^*}^{L} C_l^o}}$$

Apart from these, we also calculate the tolerance and sensitivity coefficients that include the Total Sensitivity Coefficients and Operating Point Sensitivity Coefficients (TSC and OPSC).

1.1.2.2 Middle Level Sensitivity Analysis

This theorem deals with a situation where a (M+T+Q) dimensional allowable region for the (M+T+Q) perturbations induced in the local contribution matrix $C_{kl^*}^{G-O}$. As long as the values are within the allowable region, the rank orders will be preserved (Chen, 2007 #1)

Theorem 2 Let $P_{k^*l^*}^{G-O}$ $\left(-C_{k^*l^*}^{O} \leq P_{k^*l^*}^{O} \leq 1 - C_{k^*l^*}^{O}\right)$ denote the perturbation induced on one of the $C_{kl}^{G-O}{}'s$, which is $C_{k^*l^*}^{O}$(contribution of a specific goal $G_{k^*}$to a specific objective O_{l^*}); the original ranking of A_r and A_{r+n} will not reverse if;

$$\lambda \geq P_{k^*l^*}^{G-O} \lambda_{kl}{}^{G-O}$$

Where $\lambda = C_r^A - C_{r+n}^A$

$$\boldsymbol{\lambda_{k_1 l}^{G-O} = C_{l^*}^{O} \times \left[C_{r+n,k^*}^{A-G} - C_{rk^*}^{A-G} + \left(\sum_{k=1,k\neq k^*}^{K} C_{rk}^{A-G} - \sum_{k=1,k\neq k^*}^{K} C_{r+n,k}^{A-G} \right) \times \frac{C_{kl^*}^{G-O}}{\sum_{k=1,k\neq k^*}^{K} C_{kl^*}^{G-O}} \right]}$$

1.1.2.3 Bottom Level Sensitivity Analysis

This theorem deals with the perturbations induced in Matrix C_{ij}^{A-S} which is the bottom level of the decision hierarchy. Different theorems and corollaries are developed to address the various situations. (Chen, 2007 #1)

Theorem 3 Let $P_{i_m^* j_\alpha^*}^{A-S}$ (- $C_{i_m^* j_\alpha^*}^{A-S} \leq P_{i_m^* j_\alpha^*}^{A-S} \leq 1$ - $C_{i_m^* j_\alpha^*}^{A-S}$, $\sum_{m=1}^{M} P_{i_m^* j_\alpha^*}^{A-S} \leq 1$ - $\sum_{m=1}^{M} C_{i_m^* j_\alpha^*}^{A-S}$, m=1,2,3…M) denote M perturbations induced in M of the $C_{ij_\alpha^*}^{A-S}$'s (contribution of M actions $A_{i_m^*}$ to the α^{th} changing strategy $S_{j_\alpha^*}$), $P_{i_t^* j_\beta^*}^{A-S}$ (- $C_{i_t^* j_\beta^*}^{A-S} \leq P_{i_t^* j_\beta^*}^{A-S} \leq 1$ - $C_{i_t^* j_\beta^*}^{A-S}$, $\sum_{t=1}^{T} P_{i_t^* j_\beta^*}^{A-S} \leq 1$ - $\sum_{t=1}^{T} C_{i_t^* j_\beta^*}^{A-S}$, t=1,2,3…T) denote T perturbations induced in T of the $C_{ij_\beta^*}^{A-S}$'s (contribution of T actions $A_{i_t^*}$ to the β^{th} changing strategy $S_{j_\beta^*}$), $P_{i_q^* j_Y^*}^{A-S}$ (- $C_{i_q^* j_Y^*}^{A-S} \leq P_{i_q^* j_Y^*}^{A-S} \leq 1$ - $C_{i_q^* j_Y^*}^{A-S}$, $\sum_{q=1}^{Q} P_{i_q^* j_Y^*}^{A-S} \leq 1$ - $\sum_{q=1}^{Q} C_{i_q^* j_Y^*}^{A-S}$, q=1,2,3…Q) denote Q

perturbations induced in Q of the $C_{ij_Y^*}^{A-S}$'s (contribution of Q actions $A_{i_q^*}$ to the Y^{th} changing strategy $S_{j_Y^*}$), the original ranking of A_r and A_{r+n} will not reverse if:

$$C_{r^*}^{A} - C_{r+n}^{A} \geq - C_{j_\alpha^*}^{S} \left(P_{r_m^* j_\alpha^*}^{A-S} + \sum_{m=1}^{M} P_{i_m^* j_\alpha^*}^{A-S} * C_{r+n,j_\alpha^*}^{A-S} / \sum_{i=1.i\neq i_m^*}^{I} C_{i,j_\alpha^*}^{A-S} \right) + C_{j_\beta^*}^{S} * \sum_{t=1}^{T} P_{i_t^* j_\beta^*}^{A-S} * C_{r,j_\beta^*}^{A-S} - C_{r+n,j_\beta^*}^{A-S} / \sum_{i=1.i\neq i_m^*}^{I} C_{i,j_\beta^*}^{A-S} + + C_{j_Y^*}^{S} * \sum_{q=1}^{Q} P_{i_q^* j_Y^*}^{A-S} * C_{r,j_Y^*}^{A-S} - C_{r+n,j_Y^*}^{A-S} / \sum_{i=1.i\neq i_m^*}^{I} C_{i,j_Y^*}^{A-S}$$

1.2 Organization of the thesis

The next sections will present the literature survey that includes all the research work that was conducted over the web to derive the need of making an automated system. It will also shed some light on the literature survey that was conducted in the areas of Hierarchical Decision Models, SA for general MCDM problems including Linear Programming, specific SA for all HDM methods and other related studies in AHP.

Chapter 2 focuses on the Software Development Life Cycle (SDLC). In that chapter, all the phases of the SDLC are discussed with respect to the development of the application.

Apart from that, we have the methodology and software development section which will outline the mathematical theorems that are used in the HDM SA, the pseudo code, class diagrams, data flow diagrams (DFD) and other required UML diagrams.

Chapter 4 is discusses the results that were generated after the execution of the software application on different scenarios including the Ho's model and the implementation of "Technology Development Envelope" to determine the best cooling technologies for a leading computing servers manufacturers. This section also studies all the results generated by the output module as well as the sensitivity analyses modules.

The last section includes the conclusions and recommendations for the future work. The future work will also include the suggestions for future development platforms like web or distributed

environment, for this particular execution. Mathematical details and theorems are included in the Appendix.

2 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

2.1 Introduction

SDLC is the process of creating or altering a system along with the methodologies, practices and procedures used within the cycle to develop a system. The SDLC includes a large number of software development methodologies and processes. All these methodologies form the framework that is used to plan, structure, design, implement and maintain the whole system. An overview of the SDLC can be seen in Figure 2.1:

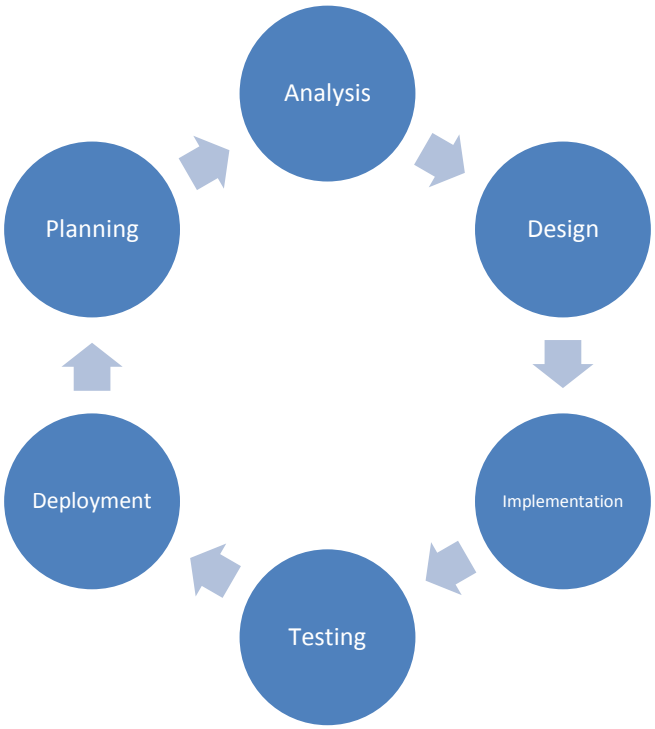


Figure 2.1 The Systems Development Life Cycle

The whole process is used to develop a comprehensive system, which includes requirements gathering, analysis of the problem statement, design of the whole back end and front end, implementation according to the design and then finally, the deployment which further down carries on with the maintenance. The purpose of using an SDLC is to generate a high quality system (product) that meets or exceeds customer expectations, reaches completion within time

and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

2.2 Planning

The goal of this phase was to plan out all the other following phases of the SDLC. The planning phase of the implementation of the HDM SA software started in the summer of 2009. The first thing that was decided during this phase was the development and the deployment platform. It was decided that a web application will be developed and for this reason, three choices were considered; ASP.Net, JSP and PHP. Among them, ASP.Net was selected. The first reason was my comfort level with C#.Net and VB.Net rather than Java (JSP) or PHP. The second reason was the ease of utilizing web services and the integration with other Application Programmers Interfaces (APIs). Lastly, it is easier to migrate from web to windows can be done in .Net.

The next thing was to understand the HDM SA algorithm. For this reason, a thorough reading of the (Chen, 2007 #1) was done. All the complex mathematical notations used in the formulas were reviewed and understood. For better understanding, some sample calculations were done by hand and in Excel. Results were then compared with the ones presented in the (Chen, 2007 #1).

The last thing done in this phase was the investigation of the Windows based application developed in VB6 by a student at Portland State University. All the interfaces were reviewed which were considered as prototypes to get an idea of the data flow throughout the system.

2.3 Systems Analysis

The main purpose of this phase was to state the problem and to analyze the whole problem statement. This phase also includes an in depth study of all the necessary material that was needed to understand HDM, SA and HDM SA. For this reason, a few research papers were studied to see the impact of HDM SA in the real world along with the origin of HDM SA.

Then, the first task was to break down the whole system into manageable pieces to analyze the situation and the project goals in order to engage both of them to the requirements. The whole

system was broken down into modules like, Matrix Decompositions, Pair Wise Comparison, Direct Input, Theorems, Output and Sensitivity Analysis.

Different types of hierarchies with different kinds of criteria and sub criteria relationships were studied. Integration with different Application Programmer Interfaces was also identified in this phase.

Different methodologies of software development such as the Waterfall Model, Spiral Model and Prototyping etc were considered during this phase. In accordance with the length of the project, "Agile Methodology" was considered feasible and the Incremental Method was chosen. Some of the salient features of the method are as follows:

- A series of mini-Waterfalls are performed, where all phases of the Waterfall are completed for a small part of a system, before proceeding to the next increment;
- Overall requirements were defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system;
- The initial software concept, requirements analysis, design of the architecture and system core were defined via Waterfall, followed by iterative Prototyping, which culminates in installing the final prototype, a working system.

All the Object OOP, Service Oriented Architecture and Code Optimization techniques were revisited. Advantages and disadvantages of different platforms were compared while the ease of modifications in the future. Use Cases were also designed during this phase.

2.4 Systems Design

In systems design, the functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage describes the new system as a collection of modules or subsystems.

In this phase, different diagrams were drawn to give a detailed overview of the system to the end user. As there is a very limited involvement of the backend at the runtime, there wasn't any Entity Relationship diagram drawn to depict the main entities of the system. The whole system is depicted in terms of classes and objects which are presented in detail in Chapter 3. Apart from this, data flow diagrams were also drawn which are also presented in Chapter 3.

Once the data flow was figured out, major decisions about the implementation were made. The data flow in the beginning of the phase was not completely defined but with time, it became more and more detailed. Same is the case with the class diagram.

Figure 2.2 gives an overview of how different elements of this phase interact with each other. The whole system is following a similar model.

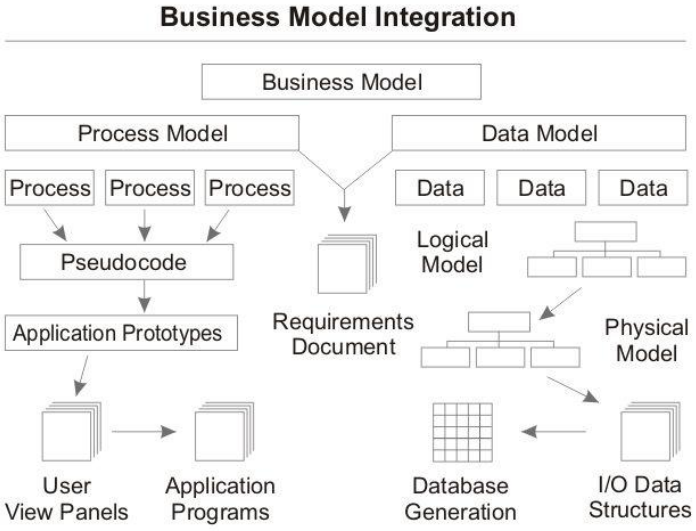


Figure 2.2 The Business Model

The whole Business model is based on the process model and the data model. The process model is based on the pseudocode (i.e. the hand written code on paper for dry run purposes), application programs (i.e. all the console applications that were created to test the algorithm results) and the application prototypes (i.e. the interfaces and the user view panels). The data model is based on

the system architectural diagrams (i.e. the class diagrams and the data flow diagrams) as well as the backend data (i.e. the data stored temporarily in the text and XML files).

2.5 Systems Implementation

In this phase, coding of the system was performed and a detailed overview of the implementation is presented in Chapter 3. The major portion of the implementation took about 1.5 years. As mentioned before, the platform selected for the purpose of implementation is C#.Net. In the beginning, all the efforts were directed towards making a web application. For that reason, ASP.Net was the platform with C# as the coding scheme was selected. Till the HDM output module, the implementation was done for the web part. After that, the platform was changed and the final product at this point is now a windows application. Visual C#.Net was the development platform used in the development of the Windows Application with XML in the backend. The system also uses the "File Stream Readers" and "Writers" to utilize the runtime filing for temporary storage. Some extra classes that are acting as custom libraries are also integrated to perform functions like Microsoft Excel integration, XML Linking and complex string operations in the application.

In the very beginning, for the prototyping and testing purposes, a console based application was also developed. Thorough testing of all the results was conducted before we proceeded to web or windows based implementation. On the console, everything was executed in a sequence according to the data flow diagram that was designed in the design phase. Following the same design, the web and the windows based implementation was proceeded.

There were some issues faced in the web portion as one of the requirements was to get all the results in an excel file. Integration of Microsoft Excel and ASP.Net was taking a long time and hence the development platform was switched to Windows Development.

2.6 Systems Testing

The code was tested at various levels in the testing phase. Unit, integration and user acceptance testing (UAT) were then performed. This is a grey area in the software development life cycle as

many different opinions exist as to what the stages of testing are and how much if any iteration occurs. Iteration is not generally part of the waterfall model, but usually some occur at this stage. Testing of the whole system was conducted in the testing “one by one” and “module by module” fashion.

Following types of tests were performed:

- Defect testing
- Path testing
- Data set testing
- Unit testing
- System testing
- Integration testing
- White box testing
- User acceptance testing
- Performance testing

We have performed unit, integration, black box and white box testing techniques in the whole implementation process thoroughly. Unit testing was performed when individual modules were tested. After integrating all the modules, integration testing was performed. Black Box Testing was performed on the interfaces which led to the inclusion of validation checks on most of the forms. White box testing was performed to correct the flow of the data and the structuring of the loops and conditional statements.

3 SOFTWARE IMPLEMENTATION DETAILS

As mentioned before, the development platform chosen for the implementation is Microsoft C#.Net (2008 and 2010). The Integrated Development Environment was the Microsoft Visual Studio .Net (2005, 2008) and the Microsoft Visual C#.Net Express Edition (2010). An object oriented approach was followed throughout the life cycle of the application. All the three main concepts of Object Oriented Programming (OOP); Encapsulation, Polymorphism and Inheritance have been used to keep up with the standard OOP protocols. OOP provides a clear modular structure for programs and makes it convenient to define abstract data types where implementation details are hidden and the unit has a clearly defined interface. It has also made it easy to maintain and modify existing code as new objects can be created with small differences to existing ones.

The total line of Coding in the business logic file is above 5000. There are 17 window forms in the application which are all interacting with each other during the runtime. To make proper resource utilization, many forms are reused throughout the runtime. There are four main packages in the application including Business Logic, XML, Excel and Permuting which can also be seen in the class diagram. There isn't any particular coding standard that was followed in the coding, but most of the commenting, declaring variables and code optimization techniques are based on the standard .Net Coding Protocols.

At the beginning, for the HDM part, there are two ways in which the user can input the information into the system: Entering the pair-comparison judgments or the contribution values directly. A third option was made available later to the end user where a file can be loaded into the system in which the contribution values were already stored from the previous model building efforts. Output of the HDM part is stored in an excel file and the outputs from the HDM SA are

saved in several text files. The text files created are stored in the root (Bin) Folder while the excel file can be stored in any path.

3.1 Flow Chart

The Flow chart depicts the logical flow for the data in the application. All the conditional statements are depicted by a diamond shape. The rectangular box is the processing unit. The oval shaped triangle (on the left side) is where the outputs are shown. The flow of the data is depicted by arrows. The flow chart can be seen in Figure 3.2

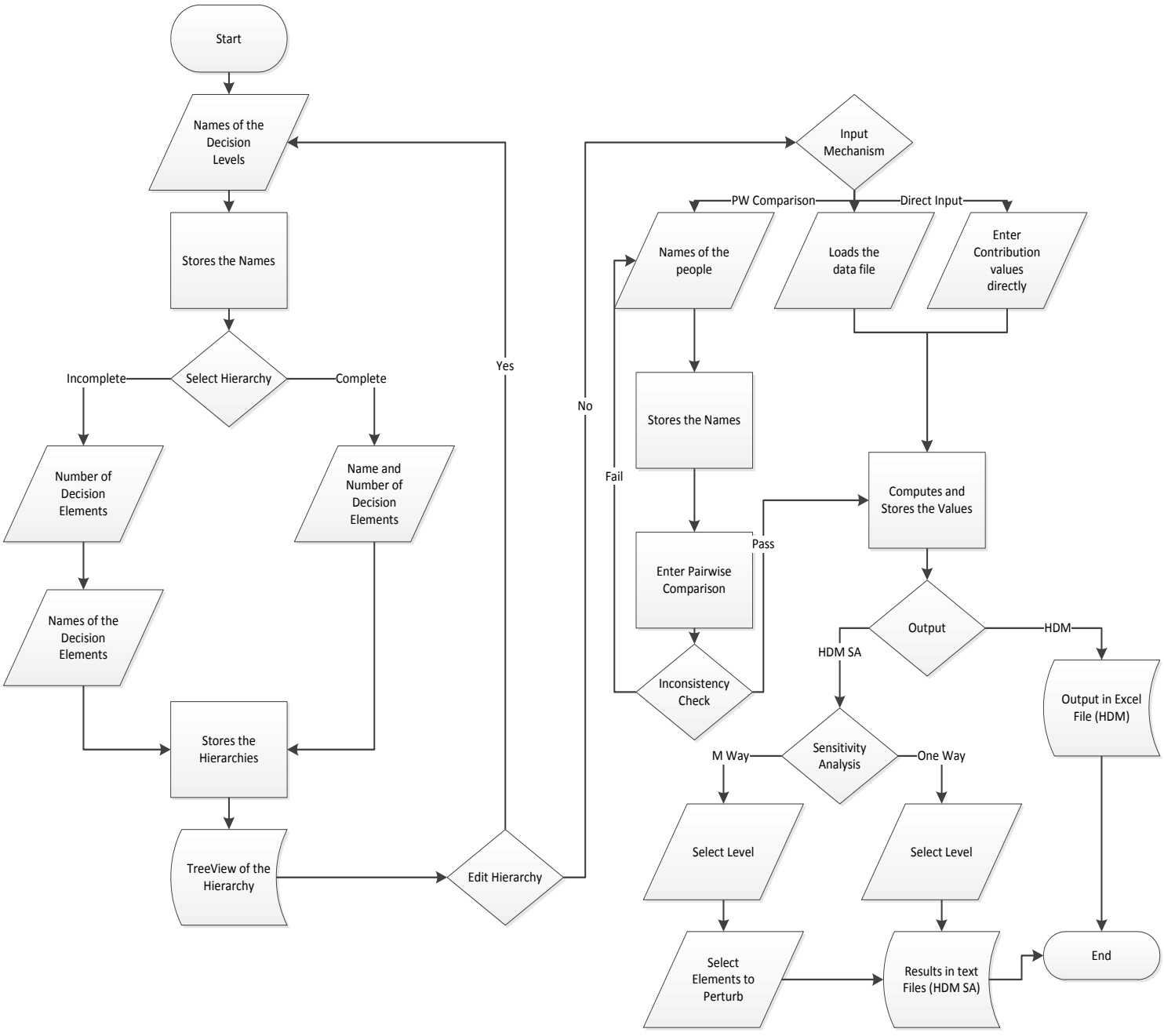


Figure 3.2 Flowchart of the System

3.2 Class Diagram

In Figure 3.3, the class diagram shows the main four packages namely, business logic, ExcelFile, XML and Permute. The main package that is the main processing unit of the algorithm is the business logic package. The significance is quite evident after looking at the diagram where we can see that there are seven classes with hundreds of methods, procedures and variables. In the class diagram, vertically divided triangles represent classes. Each division of the rectangle depicts a certain element of the class. The first division is the name of the class, the second includes all the main attributes (variables) and the last division includes all the main functions that are either static, over loaded or over ridden.

All the methods and attributes are utilized properly and are closely following the HDM and the HDM SA algorithm. A lot of "if then else" and a few "switch case" conditions are used in the program. The looping structure mostly includes "for" loops (A few nested) while the "while" loop is used in a few cases.

Some of the packages are invoked by creating instances of the respective classes (objects) while the static methods are called by using the class variable instance. For example, in the following code snippet Object type and Class type declarations and their usage is shown respectively:

```
static Theorem2 t2 = new Theorem2(); (Object)

MatrixDecompositions.OtherLevelMethods(LvlCnt, LvlNm, Level); (Class)
```

The flow of the algorithm starts with a user interface which collects inputs from the end-user and the input is kept on being processed and it also interacts with the MainNamingIndex class till the business logic is invoked. The business logic package is invoked when a call is made to the MatrixDecomposition class. The MatrixDecomposition class holds the structures and values of all

the Matrices (A, B and C) used for the HDM calculations. The MatrixDecomposition class closely interacts with MainLogic, Inherit and MainNaming Index class.

All the attributes like ObjectiveMatrix, GoalsMatrix and etc. are the Arrays or storage units that hold on the values for the Output module and the Sensitivity Analysis part. The theorems from HDM SA algorithm (Chen, 2007 #1) are implemented in separate modules and have their own respective classes which can be seen in the class diagram as Theorem 2, Theorem 3 and Theorem 4. The attributes (i.e. the variables, methods and instances) which make up a package in the class diagram; in all these classes are almost the same and represents different parts of the equation (in the theorems) that is implemented in the program. The same architectural design was used for all these classes as the methods included in all of them give output in the same format even though they have different procedures.

The Inherit Class is the one holding all the local contribution values as a backend support for the processing. As it can be seen in the class diagram, it has all the attributes that are named after the levels for which they hold values. The MainNamingIndex is the naming unit of the whole application. The tasks assigned to its sub modules is to keep a track of the names that are assigned to either the levels, elements or even the people entering the pairwise comparison.

The Main Logic Class is the one where the algorithm for Direct Input and Pairwise Comparison were implemented. For the pairwise comparison, it includes processing of all the matrices (including Matrix A, Matrix B and Matrix C) as well as the Inconsistency Check. For the Direct Input, it holds the local and global contribution values for all the elements. The treeview structure is also processed in this class.

The other packages are relatively small and can be considered as supporting packages for the main business logic package. The ExcelFile package is used by the classes in the output module to copy the output to Excel File. The Permut package is an algorithm oriented package used by

the classes in the MainLogic class which is used to permut the sequence of a certain string. The XML package works as a storing unit and was used at different instances like helping create the treeview structure and for storing temporary values for the sensitivity analysis.

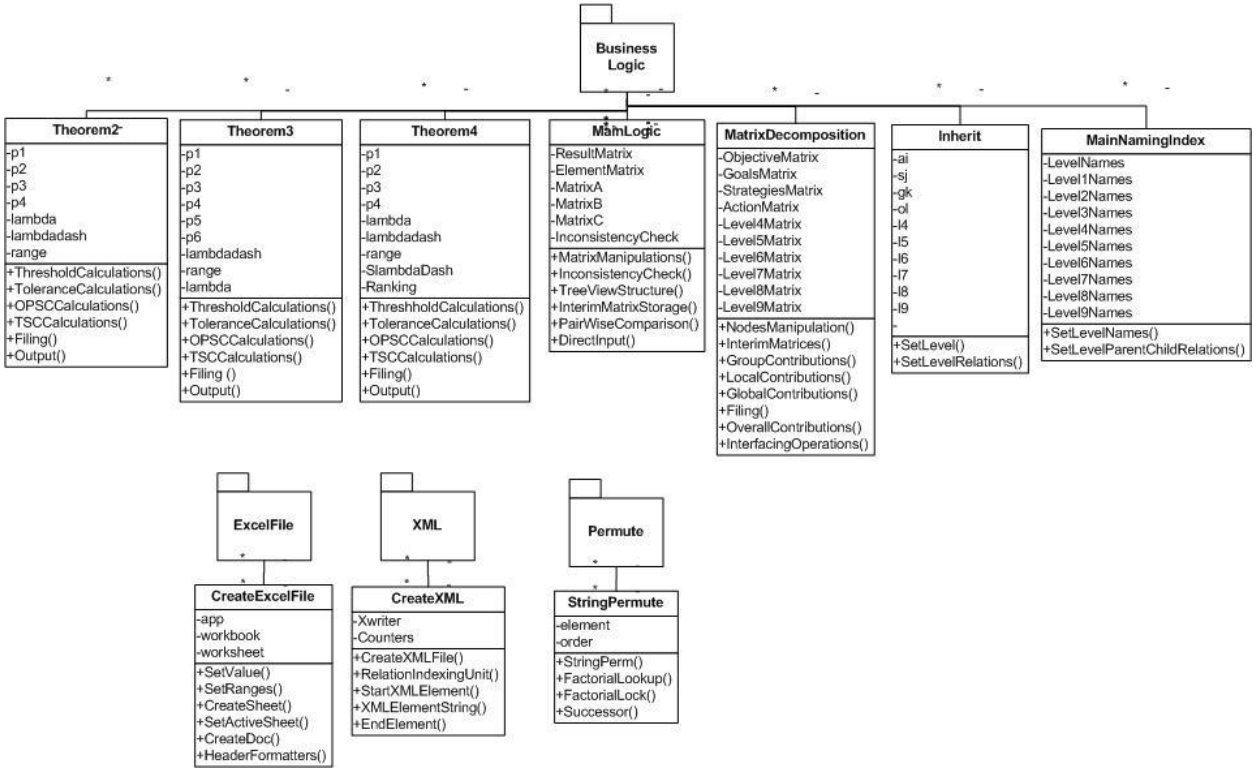

Business Logic
Theorem2
-p1
-p2
-p3
-p4
-lambda
-lambdadash
-range
+ThresholdCalculations()
+ToleranceCalculations()
+OPSCCalculations()
+TSCCalculations()
+Filing()
+Output()
Theorem3
-p1
-p2
-p3
-p4
-p5
-p6
-lambdadash
-range
-lambda
+ThresholdCalculations()
+ToleranceCalculations()
+OPSCCalculations()
+TSCCalculations()
+Filing ()
+Output()
Theorem4
-p1
-p2
-p3
-p4
-lambda
-lambdadash
-range
-SlambdaDash
-Ranking
+ThreshholdCalculations()
+ToleranceCalculations()
+OPSCCalculations()
+TSCCalculations()
+Filing()
+Output()
MainLogic
-ResultMatrix
-ElementMatrix
-MatrixA
-MatrixB
-MatrixC
-InconsistencyCheck
+MatrixManipulations()
+InconsistencyCheck()
+TreeViewStructure()
+InterimMatrixStorage()
+PairWiseComparison()
+DirectInput()
MatrixDecomposition
-ObjectiveMatrix
-GoalsMatrix
-StrategiesMatrix
-ActionMatrix
-Level4Matrix
-Level5Matrix
-Level6Matrix
-Level7Matrix
-Level8Matrix
-Level9Matrix
+NodesManipulation()
+InterimMatrices()
+GroupContributions()
+LocalContributions()
+GlobalContributions()
+Filing()
+OverallContributions()
+InterfacingOperations()
Inherit
-ai
-sj
-gk
-ol
-l4
-l5
-l6
-l7
-l8
-l9
+SetLevel()
+SetLevelRelations()
MainNamingIndex
-LevelNames
-Level1Names
-Level2Names
-Level3Names
-Level4Names
-Level5Names
-Level6Names
-Level7Names
-Level8Names
-Level9Names
+SetLevelNames()
+SetLevelParentChildRelations()
ExcelFile
CreateExcelFile
-app
-workbook
-worksheet
+SetValue()
+SetRanges()
+CreateSheet()
+SetActiveSheet()
+CreateDoc()
+HeaderFormatters()
XML
CreateXML
-Xwriter
-Counters
+CreateXMLFile()
+RelationIndexingUnit()
+StartXMLElement()
+XMLElementString()
+EndElement()
Permute
StringPermute
-element
-order
+StringPerm()
+FactorialLookup()
+FactorialLock()
+Successor()

Figure 3.3 Class Diagram of the software

3.3 Data Flow Diagram

Figure 3.4 shows the data flow diagram. A square represents an interface; a rounded rectangle is the processing unit while the rectangle opened from one side is a data store. The arrows depict the flow of data in the system. There can never be any direct interaction of an interface with the storage unit or another interface itself and vice versa.

The flow of data in this software seems really simple. It starts with the Level Names which are processed and saved in the Level Name Store (Main Naming Index Class) and then the number and the names of the elements at different levels are entered. All the inputs are processed and are stored at their respective storage units in the Main Naming Index Class. Then, all the data from the storage unit goes into XML package for processing and the Tree View is formed which depicts the structure of the model.

The treeview gives an option of either make changes to the model structure or continue forward. If editing is needed, it goes back to the "Level Name" step and shows previous inputs to be modified. If editing is not needed, the program continues forward with the input mechanism which gives three options, Direct Input, Pair Wise Comparison or Load File.

If the option selected is Pair Wise Comparison, the data continues to the group input processing module that accepts the number and the names of the persons participating in the pairwise comparison at a particular level. After the data is recorded, it is saved for the output module via Filing (backend text file support). On the next step, respective inputs by the designated people are supplied which are then processed and via filing is stored in the backend files.

For the Direct Input Module the user enters the local contribution values into the system which goes through the "Levels Processor" and after that, the data is stored in the Level Elements store. All the steps in both the pairwise comparison and direct input mechanisms are almost the same except for the absence of the group input processing in the "Direct Input" case. In this case, the values are entered by the end-user which is stored directly into the backend files after a bit of processing.

For the load file option, there is a condition that should be met which is that the previously created files should be in the root (Bin) Folder and the structure should match the input supplied by the user previously. The structure includes the number of levels, number of decision elements on that level and the name of the elements. Then, a file will be loaded from the Level Element Store (backend files) for the Output Module.

The output module in the end loads the files and processes all the values whether supplied to the store from the Pair Wise Comparison module, Direct Input or the Load File mechanism. After processing the values, the output module generates an excel file that includes all the local, global and overall contribution values of all the elements.

Further on, the values can be used by the Sensitivity Analysis module (both One-way or M-way) to generate the Analysis report on text files.

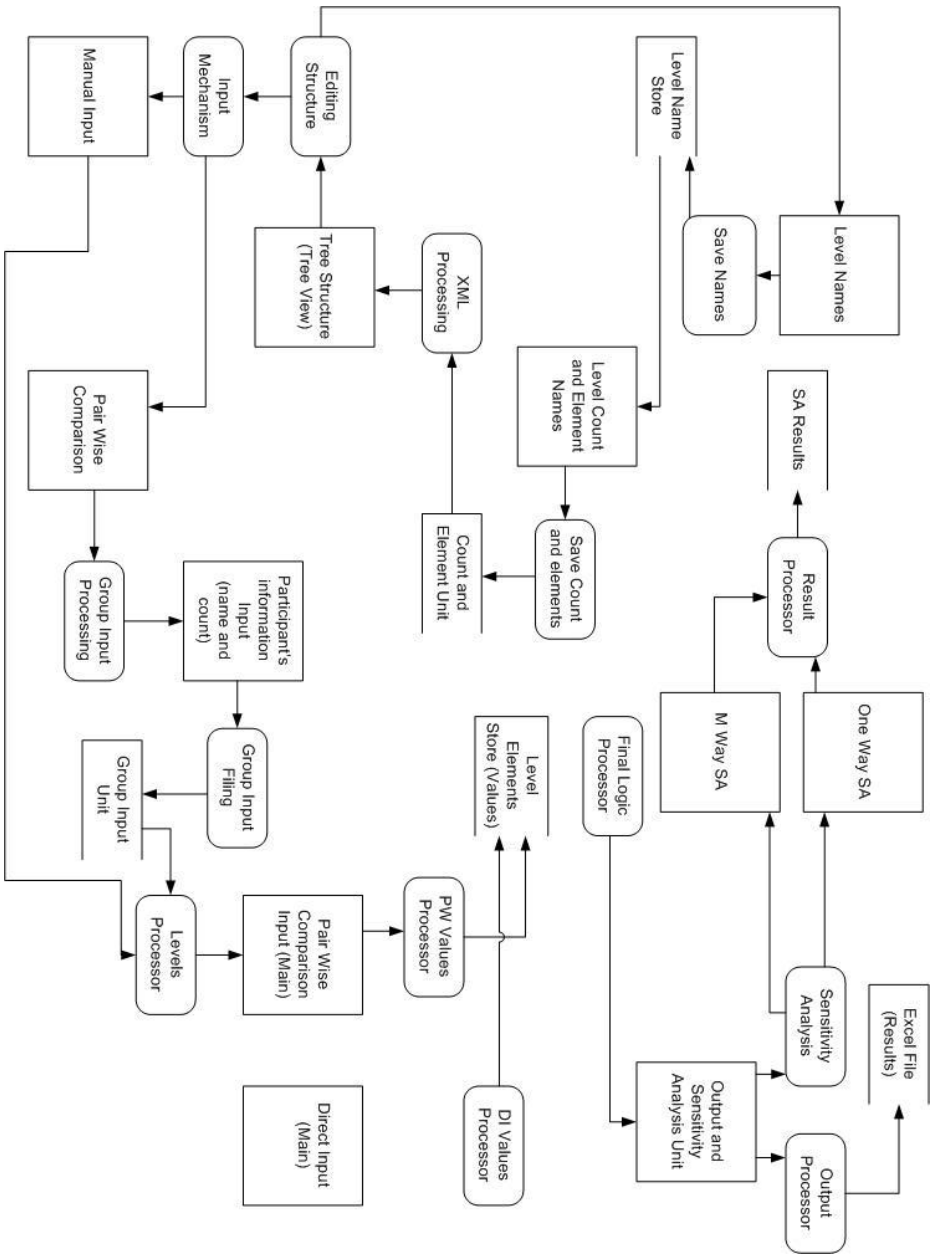


Figure 3.4 Data Flow Diagram of the software

3.4 Use Case Diagrams

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. (Glinz, 2000 #3)

In the Figure 3.4, we can see different modules where different actors are interacting with multiple use cases to perform certain functionality. This use case depicts the Input module which consists of two main actors that are interacting with its use cases. As we can see that the end-user initiates either a load file or Direct Input option which is responded by the system by creating an excel file.

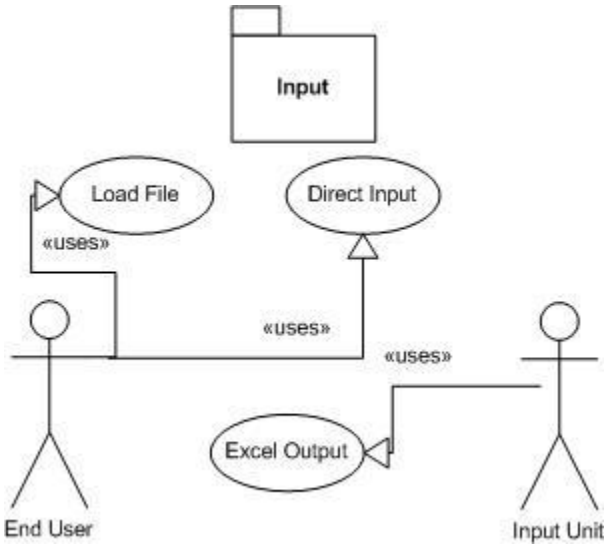


Figure 3.5 Use Case Diagram 1

The next one depicted in Figure 3.6 is the Pair Wise Comparison module which also has two actors and three use cases. The two actors include the end-user and the system (the pairwise comparison unit in this case). The end-user is interacting with the Group Information and the Pair

Wise Comparison Input use cases and the system responds to this by using the excel output use case.

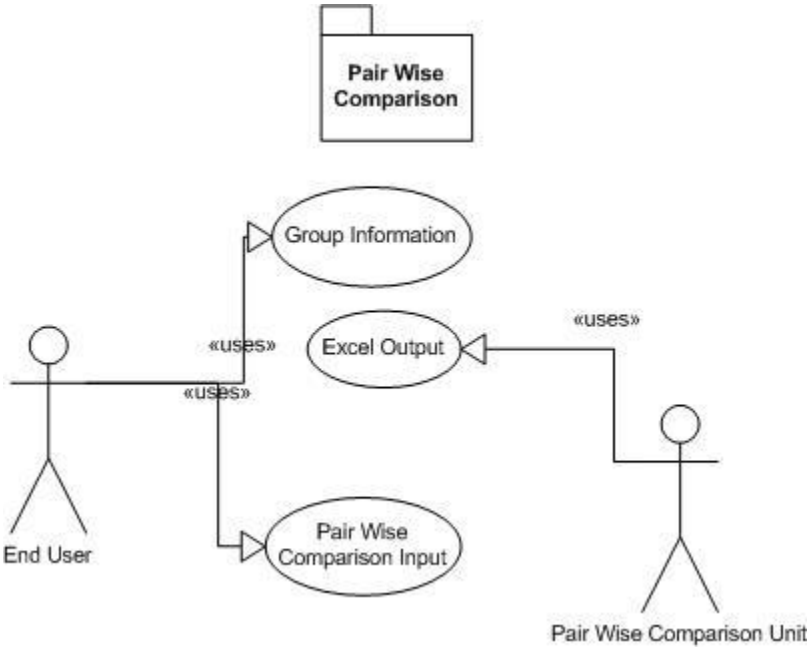


Figure 3.6 Use Case Diagram 2

The Modeling module is the one that performs the indexing and structuring of the HDM. There are two actors and four use cases in this module, as shown in the Figure 3.7. The end-user interacts with the Level Names, Level and Element Number and the Element Names use cases where the information is stored in the system. The system responds by creating a tree view structure which depicts the hierarchical structure of the model.

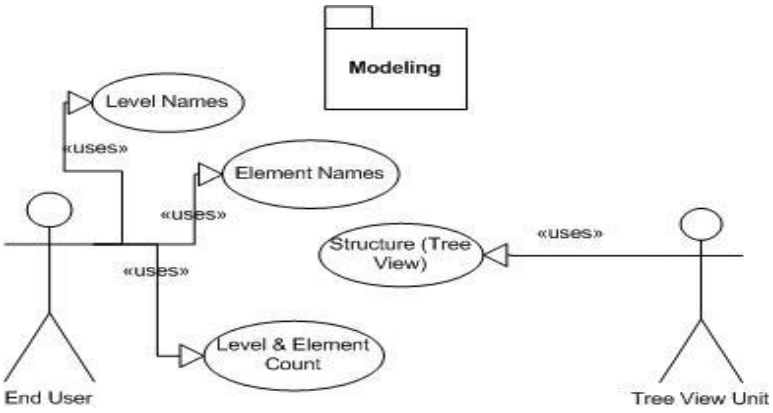


Figure 3.7 Use Case Diagram 3

The last use case is the Sensitivity Analysis module as shown in Figure 3.8. In this case, the end-user chooses the option of either going with the One-way sensitivity analysis or M-way Sensitivity Analysis. The system responds with text files that include the main output of the Sensitivity Analysis.

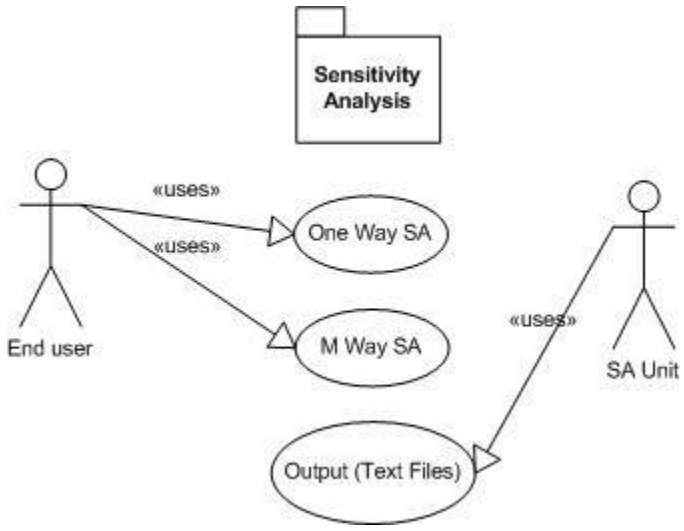


Figure 3.8 Use Case Diagram 4

3.5 Naming Convention Followed

The general naming conventions that are discussed in this text is related to choosing the best names for the elements in the libraries. Guidelines given by {MSDN, 2008 #5} are followed. They include:

- Easily readable identifier names. For example, a property named HorizontalAlignment is more readable in English than AlignmentHorizontal.
- Favoring readability over brevity. The property name CanScrollHorizontally is better than ScrollableX (an obscure reference to the X-axis).
- Not using underscores, hyphens, or any other nonalphanumeric characters while naming the variables or methods.
- Avoiding the use of identifiers that conflict with keywords of widely used programming languages like main, void and static etc.
- All the names of the variables are usually named according to the text that was analyzed during the analysis phase. There might be some variables which are used for holding the values for a short time and their names might not be following any specific naming convention. Names used in this application follow these guidelines. For example, permut() is the name of the method that performs permutations, inconsistcheck()is the name of the method that performs the inconsistency check; ArrayList MatrixA or ArrayList MatrixB are the names of the arraylists that hold the values for Matrix A and B; Theorem2 t2 = new Theorem2 () is the declaration of the object t2 that is the instance for Theorem 2 class.

3.6 Important Code Snippets

This section includes some of the major implementation details (codes) of the mathematical theorems and algorithms that are used from HDM and HDM SA. All the coding part is done using the Microsoft C#.net as the main development platform at the front end while XML, text files and excel files in the back-end.

3.6.1 SetElement Method (Results Store for Pairwise Comparison)

In this method, the Matrices A, B and C are filled with values and it is a part of the Pair Wise Comparison model. This method takes in the pair wise comparison values and starts filling the locally defined multi dimensional arrays by the name Matrix A and MatrixCopy A. Following is the declaration of the method SetElement() with its parameters and their declaration types.

```
public int SetElement(double[] pred, string succ_elemname, string
pred_elemname, int flag, int min_succ, int[] count,int level, int
personcount)
```

After these two matrices are filled up, calculations are performed on this array to generate Matrix B and Matrix C. In the following snippet we can see how the Matrix A is loaded with values:

```
for (int x = 0; x < nodes; x++)
{
   for (int y = x + 1; y < nodes; y++)
     {
     MatrixA[y, x] = SR.ReadLine();
                  MatrixA[x, y] = Convert.ToString(100 -
Convert.ToInt32(MatrixA[y, x]));
     }
 }
```

Further down the method, the Matrix A is sent to the nodesmanipulation method which is a public method where other operations of Pair Wise Comparison are performed. Following is the calling of that method:

```
nodesmanipulation(elemname, MatrixA);
```

3.6.2 Nodes Manipulation

The following is the code snippet shows the declaration of the nodes manipulation method which takes in the names of the elements at a certain level along with Matrix A.

```
public int nodesmanipulation(string[] names, string[,] MatrixA)
{}
```

The following code saves the name of the elements in the namevaluecollection nvc:

```
for (int l = 0; l < nodes; l++)
      {
```

```
                indx = pSub[l].IndexOf(".");
                if (indx != -1)
                {
                    pSub[l] = pSub[l].Substring(indx + 1, 1);
                    atoms[l] = pSub[l];
                    nvc.Add(atoms[l].ToString(), l.ToString());
                }
                else
                {
                    nvc.Add(pSub[l].ToString(), l.ToString());
                }
            }
```

All the names are stored in namevalue collections. In the following code, we can see the calculations that are done to generate the Matrix B and the Matrix C:

```
            for (int x = 0; x < nodes; x++)
                {
                    for (int y = 0; y < nodes; y++)
                    {
                        if (x != y)
                        {
                            double a = Convert.ToDouble(MatrixA[x,
y]);
                            double b = Convert.ToDouble(MatrixA[y,
x]);
                            double temp = a / b;
                            MatrixB[x, y] = Convert.ToString(temp);
                        }
                        else
                        {
                            MatrixB[x, y] = "1";
                        }
                    }
                }

            for (int x = 0; x < nodes - 1; x++)
                {
                    for (int y = 0; y < nodes; y++)
                    {

                        double a = Convert.ToDouble(MatrixB[y, x]);
                        double b = Convert.ToDouble(MatrixB[y, x +
1]);
                        double temp = a / b;
                        MatrixC[y, x] = Convert.ToString(temp);


                    }

                }
```

The results from these matrices are generated in the following code snippets. Before the normalization, all the values are converted to double and then are handled by other methods like normalization and saveresults:

```
                    for (int x = 0; x < nodes - 1; x++)
                      {
                          for (int y = 0; y < nodes; y++)
                          {
                              sum = sum + Convert.ToDouble(MatrixC[y,
x]);
                          }
                          result[x] = sum / Convert.ToDouble(nodes);
                          sum = 0.0;
                      }
                      double[] normalized_values = new double[nodes];
                      normalized_values = normalization(result, permut);

                      SaveResults(normalized_values, loopc);
```

3.6.3 Normalization

This takes place as the last step of the pair wise comparison. The method uses two parameters, a double array and a string. The string is basically a sequence of characters like 'A C B D' which is then used to orient the values in the value store according to the sequence. There are certain counter variables like count and ct along with temporary string variables like temp_st which and arr_st. Both the temporary strings are declared globally. A NameValueCollection (nvc) is also being used in the method to maintain the sequences properly.

```
public double[] normalization(double[] mean, string orient)
{
    double sum = 0.0;
    double[] normalize = new double[mean.Length + 1];
    int count = 0;
    int ct = 0;
    string temp_st = "";
    string orient_st = "";
    string[] arr_st;
    temp_st = orient.Substring(2, (orient.Length - 3));
    temp_st = temp_st.Trim();
    arr_st = temp_st.Split(' ');

    for (int i = 0; i < arr_st.Length; i++)
    {
        if (ct == 0 && arr_st[i] != "")
        {
            orient_st = Convert.ToString(arr_st[i]);
```

```
                    ct++;
                }
                else
                {
                    if (arr_st[i] != "")
                    {
                      orient_st = orient_st
+Convert.ToString(arr_st[i]);
                    }
            }
            }

            int[] arr_index = new int[orient_st.Length];
            string[] str_index = new string[orient_st.Length];
            double[] dbl_index = new double[orient_st.Length];

            for (int i = 0; i < orient_st.Length; i++)
            {
                        arr_index[i] =
        Convert.ToInt32(nvc.Get(orient_st[i].ToString()));
            }

            for (int i = orient_st.Length - 1; i >= 0; i--)
            {
                if (count == 0)
                {
                    dbl_index[i] = 1;
                    count++;
                }
                else
                {
               dbl_index[i] =
              Convert.ToDouble(mean[i])*Convert.ToDouble(dbl_index[
              i + 1]);
                }
            }
            for (int i = 0; i < dbl_index.Length; i++)
            {
                sum = sum + dbl_index[i];
            }

            for (int i = 0; i < dbl_index.Length; i++)
            {
                dbl_index[i] = dbl_index[i] / sum;
            }

            for (int i = 0; i < normalize.Length; i++)
            {
                normalize[arr_index[i]] = dbl_index[i];
            }
            return normalize;
        }
```

3.6.4 String Orientation

The orientation method is a prerequisite of the normalization method that is discussed in the previous section. The Orientation method maintains the table or a large matrix of different sequences of strings which are created by using the Permut Package. The permut package performs permutation on any sequence of strings which in return sends back all the sequences in order for the method to initiate the call. In the following method, pointer calculations are performed to ensure that the character under processing is the right one. For this purpose, a multi dimensional array, two strings and one integer variable are supplied to the method as parameters. The method returns a multidimensional array of string type. In the following snippet, different "if and else" conditions are used to check the sequence and on the basis of that, the values are maintained in the table.

```
    public string[,] Orientation
(string[,] arr_orientation, string permut_str, string main, int
loopcount)
            {
                int i = 2;
                int index_main = 0;
                int index_mposition = 0;
                int index_permut = 0;
                int index_pposition = 0;

                //arr_orientation=new string[loopcount,loopcount];

                while (permut_str[i].ToString() != "}")
                {
                    if (permut_str[i].ToString() != main[i].ToString())
                    {
                        index_main = main.IndexOf(main[i].ToString());
                        index_mposition = index_main / 2;
                     index_permut = IndexOf(permut_str[i].ToString());
                      index_pposition = index_permut / 2;

                        for (int x = 0; x < loopcount; x++)
                        {
                            arr_orientation[x, index_mposition - 1] =

                            MatrixBClone[x, index_pposition - 1];
                        }

                    }
                    i = i + 2;
                }
```

```
            return arr_orientation;
        }
```

3.6.5 Inconsistency Check

Inconsistency Check is performed for the Pair wise comparison to see whether the inputs given by the user or group of users are consistent or not. The inconsistency check is a method that takes in all the result values as parameters (nodes) and returns a double value which should be less than 0.004; else the user needs to enter the pair wise comparison again. The method calculates the standard deviation and variance by the self-made code rather than using the System.Math class.

```
public double InconsistencyCheck(int nodes)
    {
        //Mean Values
        double[] mean = new double[nodes];
        double[] variance = new double[nodes];
        double[] stddev = new double[nodes];
        double sum = 0.0;
        double incons_value = 0.0;
        int j = 0;
        while (j < nodes)
        {
            for (int i = 0; i < nodes_fact; i++)
            {
                sum = sum + ResultMatrix[i, j];
            }

            mean[j] = sum / nodes_fact;
            j = j + 1;
            sum = 0.0;
        }

        //Variance Values
        j = 0;
        sum = 0.0;
        while (j < nodes) //column
        {
            for (int i = 0; i < nodes_fact; i++) //row
            {

              sum = sum + (Math.Round(ResultMatrix[i, j],2) -
              mean[j]) * (Math.Round(ResultMatrix[i, j], 2) -
              mean[j]);

            }

            variance[j] = Math.Round(sum, 5) / nodes;
            j = j + 1;
            sum = 0.0;
```

```
        }

        sum = 0.0;

    for (int i = 0; i < variance.Length; i++)
    {
        sum = sum + variance[i];
    }

              incons_value = Math.Sqrt(sum/nodes);
              return incons_value;

   }//Method
```

3.6.6 SetElementDirect (Direct Input)

The following code snippet shows the declaration for the SetElementDirect() and its parameters with their declaration types:

```
public int SetElementDirect(double[] pred, string succ_elemname,
string pred_elemname, int flag, int min_succ, string s0, string
s1, string s2, string s3,int[] count, int level)
{}
```

Above is the method declaration for the SetElementDirect method that takes in 11 parameters of different types and dimensions. The method is similar to the SetElement method which is discussed in the section 3.4.1. The only difference is that, it does not generate any Matrix A, B or C but directly store all the contribution values which are further saved into the result unit using the SaveElement method as can be seen in the following code line:

```
element = DirectValues(nodes, s0, sone, s2, s3, val);

SaveElement(element, i, arr_startpoint[i], arr_elemcount[i],
                                elemsize);
```

The DirectValue Method can either store the values that are sent by the above code snippet or could read XML Files to maintain the HDM model with their contribution values. (Note: The

reading of values from XML files is disabled for the first release of the application and hence the code is commented). The following code snippet depicts the working of this method:

```
      if (morc == "r")
                {
                    // Compile a standard XPath expression

                    //for (int i = 0; i < elemcount; i++)
                    //{
                          //    xpathexpr = szero + sone + stwo + "/" +
      sthree + (ecount + 1).ToString() + "/value";
                    //    XPathExpression expr;
                    //    expr = nav.Compile(xpathexpr);
                    //    XPathNodeIterator iterator =
nav.Select(expr);

                    //    iterator.MoveNext();
                    //    XPathNavigator nav2 =
iterator.Current.Clone();

                    //    nav2.MoveToFirstChild();

                    //    temp[i] = Convert.ToDouble(nav2.Value);
                    //    nav2.MoveToNext();
                    //    ecount++;
                    //}
                }
                else
                {
                    for (int i = 0; i < elemcount; i++)
                    {
                        temp[i] = arr[i];
                    }
                }
```

3.6.7 Final Normalization

This method is called after all the contributions are calculated to make sure that the sum of all the contribution values at a certain level equals to 1. Following snippet shows its working:

```
        public void finalNormalization(double[] arr)
        {
            double Nsum = 0.0;
            for (int i = 0; i < arr.Length; i++)
            {
                Nsum = Nsum + arr[i];
            }

            for (int i = 0; i < arr.Length; i++)
```

```
            {
                arr[i] = arr[i] / Nsum;
            }

        }//Method
```

3.6.8 Interfacing Methods

Interfacing methods are the middle tier methods between the front end and the back end. These methods are basically overloaded and include the ObjectiveMethod and OtherLevelMethod. The following snippets depict their working and the sequence of callings of other methods:

```
    public static void OtherLevelMethods(int min_e, int[] count, int
Level)
            {
                if (Level == 2)
                {
                    if (min_e == 0)
                    {
                                inconsistindx_flag =
      m1.SetElementDirect(objectivesMatrix, "Goals", "Objectives", 1,
      min_e,  "", "", "", "",count,Level);
                        GoalsMatrix = objectivesMatrix;
                        InterimElements = m1.LocalMatrix();
                        inh.SetLevel2(GoalsMatrix);
                        inh.SetLevel1_2(InterimElements);
                    }
                    else
                    {
                                inconsistindx_flag =
      m1.SetElementDirect(objectivesMatrix, "Goals", "Objectives", 0,
      min_e,  "", "", "", "", count,Level);
                        InterimElements = m1.LocalMatrix();
                                GoalsMatrix =
      m1.GetElement(objectivesMatrix, 2, InterimElements);
                        //m1.finalNormalization(GoalsMatrix);
                        inh.SetLevel2(GoalsMatrix);
                        inh.SetLevel1_2(InterimElements);
                    }
                }


      public static void ObjectiveMethod(string morc, double[] arr)
        {
            elemcount = MainNamingIndex.Level1.Length;
            objectivesMatrix = new double[elemcount];

            if (morc == "m" || morc == "M")
            {
                for (int i = 0; i < objectivesMatrix.Length; i++)
                {
```

```
                objectivesMatrix[i] = arr[i];
            }
        }

        inh.SetLevel1(objectivesMatrix);
    }
```

3.6.9 Theorem 2 (Sensitivity Analysis)

This method is also a part of the MatrixDecomposition class. This method is demonstrating the Theorem 2 or the top level sensitivity analysis. It is an overloaded method but for the demonstration, only one method is illustrated which is shown in the following code snippet. 1). The method takes in two integer variables as parameters to determine whether it is One-way or M way 2). Which level to perform SA. In case of M-way, it would be an array of values in place of just one integer value (inp). The method interacts with the Inherit and the MainLogic classes to perform the analyses. The first line of the method shows the StreamWriters writing path or the name of the file where the resultant values will be stored. P_1 is a globally defined array that holds the values for the allowable region temporarily which is further exported to text files for permanent storage.

The most important method is highlighted in the following code snippet is t2.Pert_1(). The t2 is the object that is instancing for the class Theorem 2.

```
public static void Theorem2(int thrm, int inp)
  {
              TextWriter tw = new StreamWriter("Data_1.txt");
              int indx = 0;
              string s1 = "";
              if (inp == 1)
              {

                  for (int l = 0; l < Inherit.ol.Length; l++)
                  {

                      int size = t2.size() - 1;
                      P_1 = t2.ranges(l);
                      for (int r = 0; r < Inherit.ai.Length - 1;
r++)
                      {
                          r = r + 1;
                          thrsh = t2.Pert_1(inp, l, size, r);
```

```
                                indx =
    NamingIndex.LastLevelIndicatorUnit();
                                s1 =
    MainNamingIndex.LevelNames[indx].Substring(0, 1);
                                tw.WriteLine("------------------------
    --------------------------");
                                tw.WriteLine("The feasible range of the
    peturbation P"+(l+1).ToString()+ " is between {0} and ",
    Convert.ToString(P_1[0]), Convert.ToString(P_1[1]));
                                for (int i = 0; i < thrsh.Length; i++)
                                {
                                    if (thrsh[i] < 0)
                                    {
                                        tw.WriteLine("When P" + (l +
    1).ToString() + " is >= {0}, the rank order between " + s1 +
    (r).ToString() + " and " + s1 + (r + ( i + 1 )).ToString() + "
    remains unchanged", thrsh[i]);
                                    }
                                    else
                                    {
                                        tw.WriteLine("When P" + (l +
    1).ToString() + " is <= {0}, the rank order between " + s1 +
    (r).ToString() + " and " + s1 + (r + (i + 1)).ToString() + "
    remains unchanged", thrsh[i]);
                                    }

                                }
```

…..

3.6.10 Pert_1

This method is used to perform the one-way SA and is included in all the three theorems. It is organized in such a way that it includes different variables (i.e. p1, p2, p3, p4 and p5) according to the number of elements included in the equations of the HDM SA theorems. For example, in the following snippet shows the working of the Theorem 2's Pert_1 method:

```
for (n = 0; n < size; n++)
{
    n = n + 1;
    lambda = calculate_lambda(ai, r, r + n);
    lambda = Math.Round(lambda, 6);
    p1 = p_1(ol, l);
    p1 = Math.Round(p1, 6);
    p2 = values(glbl_cntrbns, r + n, l);
    p2 = Math.Round(p2, 6);
    p3 = values(glbl_cntrbns, r, l);
    p3 = Math.Round(p3, 6);
    p4 = p_one(glbl_cntrbns, p1, ol, r + n, l);
    p4 = Math.Round(p4, 6);
    p5 = p_one(glbl_cntrbns, p1, ol, r, l);
```

```
                p5 = Math.Round(p5, 6);
                LambdaDash = CalLamdaDash(p2, p3, p4, p5);
                LambdaDash = Math.Round(LambdaDash, 6);
                ThreshHold = lambda / LambdaDash;
                ThreshHold = Math.Round(ThreshHold, 6);
                n = n - 1;
                thrsh[n] = ThreshHold;
               }

public double CalLamdaDash(double val2, double val3, double val4,
double val5)

           {
            return val2 - val3 - val4 + val5;
           }
```

The CalLambdaDash in the above snippet is the organization of the elements in an equation. This method is called in the Pert_1 method and it returns the value that is divided by the lambda value that is also calculated as a part of the sequence of Pert_1 to gives the allowable region of the perturbation for a specific element. Both the methods i.e. Pert_1 and CalLambdaDash are different for all the three theorems. The following are the methods for theorem 3 and theorem 4.

```
   //Theorem 3
         for (n = 0; n < size; n++)
            {
                n = n + 1;
                lambda = calculate_lambda(ai, r, r + n);
                lambda = Math.Round(lambda, 6);
                p1 = p_1(parent, l);
                p1 = Math.Round(p1, 6);
                p2 = values(cntrbns_L2, r + n, x);
                p2 = Math.Round(p2, 6);
                p3 = values(cntrbns_L2, r, x);
                p3 = Math.Round(p3, 6);
                p4 = p_one(cntrbns_L1, cntrbns_L2, r, x, l, level);
                p4 = Math.Round(p4, 6);
                p5 = p_one(cntrbns_L1, cntrbns_L2, r + n, x, l, level);
                p5 = Math.Round(p5, 6);
                p6 = MidSum1(cntrbns_L1, x, l, level);
                p6 = Math.Round(p6, 6);
                LambdaDash = CalLamdaDash(p1, p2, p3, p4, p5, p6);
                LambdaDash = Math.Round(LambdaDash, 6);
                ThreshHold = lambda / LambdaDash;
                ThreshHold = Math.Round(ThreshHold, 6);
                n = n - 1;
                thrsh[n] = ThreshHold;
            }
      public double CalLamdaDash(double val1, double val2, double val3,
double val4, double val5, double val6)
```

```
        {
            return val1 * (val2 - val3 + (val4 - val5) / val6);
           }


    //Theorem 4


       for (int n = 0; n < size; n++)
             {
                 n = n + 1;
                 lambda = calculate_lambda(ai, r, r + n);
                 lambda = Math.Round(lambda, 6);
                 p1 = part1(sj, j);
                 p1 = Math.Round(p1, 6);
                 p2 = values(cntrbns_L2, r, j);
                 p2 = Math.Round(p2, 6);
                 p3 = values(cntrbns_L2, r + n, j);
                 p3 = Math.Round(p3, 6);
                 p4 = p_one(cntrbns_L1, i, j);
                 p4 = Math.Round(p4, 6);

                 rank = ranking(i - 1);

                 LambdaDash = CalLamdaDashOne(p1, p2, p3, p4, rank, r, r
+ n, i, j, cntrbns_L1, var, lambda);
                 LambdaDash = Math.Round(LambdaDash, 6);
                 ThreshHold = lambda / LambdaDash;
                 ThreshHold = Math.Round(ThreshHold, 6);
                 n = n - 1;
                 thrsh[n] = ThreshHold;
             }//n

public double CalLamdaDashOne(double val1, double val2, double val3,
double val4, int flag, int r, int rplusn, int i, int j, double[,]
glbl_cntr, string d, double lmb)
        {
            if (d == "i")
            {

                if (flag == rplusn)
                {
                    return val1;
                }

                else if (flag == r)
                {
                    return -val1;
                }

                else
                {
                    return 0.000;
                }
            }
            else
            {
```

```
            if (flag != r && flag != rplusn)
            {
                return val1 * (val2 / val4 - val3 / val4);
            }
            else if (flag == rplusn)
            {
                return val1 * (1 + (val2 / val4));
            }
            else if (flag == r)
            {
                return (-1) * (val1) * (1 + (val3 / val4));
            }
            else
            {
                return 0.0;
            }
        }
    }
```

3.6.11 Pert_M

This method serves the M-way Sensitivity Analysis part of all the three theorems. It is similar to the Pert_1 that is discussed in the previous section 3.4.10. The only difference is the input variables that it takes as input. In case of theorem 4, the calculations are different and are based on many different conditions. The following are code snippets for this method in the sequence of Theorem 2, Theorem 3 and Theorem 4:

```
//Theorem 2
      public double[] Pert_M(int[] inp, int k, int r, int size)
        {

            int n = 0;
            k = k + 1;
            double[] thrsh = new double[size];

            glbl_cntrbns = AssignValues_3();

            for (n = 0; n < size; n++)
            {
                n = n + 1;
                p1 = part1(ol, inp);
                p1 = Math.Round(p1, 6);
                p2 = values(glbl_cntrbns, r + n, k);
                p2 = Math.Round(p2, 6);
                p3 = values(glbl_cntrbns, r, k);
                p3 = Math.Round(p3, 6);
                p4 = p_mul(glbl_cntrbns, p1, ol, r + n, inp);
                p4 = Math.Round(p4, 6);
                p5 = p_mul(glbl_cntrbns, p1, ol, r, inp);
```

```
                p5 = Math.Round(p5, 6);
                LambdaDash = CalLamdaDash(p2, p3, p4, p5);
                LambdaDash = Math.Round(LambdaDash, 6);
                n = n - 1;
                thrsh[n] = LambdaDash;

            }

            return thrsh;
        }


      public double CalLamdaDash(double val2, double val3, double val4,
double val5)
        {
            return val2 - val3 - val4 + val5;
        }


//Theorem 3


for (n = 0; n < size; n++)
            {
                n = n + 1;
                lambda = calculate_lambda(ai, r, r + n);
                lambda = Math.Round(lambda, 6);
                Filing(lambda.ToString());
                p1 = p_1(parent, l);
                p1 = Math.Round(p1, 6);
                p2 = values(cntrbns_L2, r + n, k[inp]);
                p2 = Math.Round(p2, 6);
                p3 = values(cntrbns_L2, r, k[inp]);
                p3 = Math.Round(p3, 6);
                p4 = p_mul(cntrbns_L1, cntrbns_L2, r, k, l, level);
                p4 = Math.Round(p4, 6);
                p5 = p_mul(cntrbns_L1, cntrbns_L2, r + n, k, l, level);
                p5 = Math.Round(p5, 6);
                p6 = MidSumM(cntrbns_L1, k, l, level);
                p6 = Math.Round(p6, 6);
                LambdaDash = CalLamdaDash(p1, p2, p3, p4, p5, p6);
                LambdaDash = Math.Round(LambdaDash, 6);
                n = n - 1;
                thrsh[n] = LambdaDash;
            }

public double CalLamdaDash(double val1, double val2, double val3,
double val4, double val5, double val6)
        {
            return val1 * (val2 - val3 + (val4 - val5) / val6);
        }


//Theorem 4


for (int n = 0; n < size; n++)
```

```
            {
                n = n + 1;
                lambda = calculate_lambda(ai, r, r + n);
                lambda = Math.Round(lambda, 6);
                Filing(lambda.ToString());
                p1 = part1(sj, j);
                p1 = Math.Round(p1, 6);
                p2 = values(cntrbns_L2, r, j);
                p2 = Math.Round(p2, 6);
                p3 = values(cntrbns_L2, r + n, j);
                p3 = Math.Round(p3, 6);
                p4 = p_mul(cntrbns_L2, j, arr);
                p4 = Math.Round(p4, 6);


                SLambdaDash = CalLamdaDashMul(p1, p2, p3, p4, r, r + n,
arr, d);
                n = n - 1;
                thrsh[n] = SLambdaDash;
            }//n


public string CalLamdaDashMul(double val1, double val2, double val3,
double val4, int r, int rplusn, int[] arr, string d)
        {
            string st = "";
            if (d == "i")
            {

                if (ArrExist(arr, rplusn) && !(ArrExist(arr, r)))
                {
                    for (int i = 0; i < arr.Length; i++)
                    {
                        st = st + Convert.ToString(val1) + ",";
                    }

                }

                else if (ArrExist(arr, r) && !(ArrExist(arr, rplusn)))
                {
                    for (int i = 0; i < arr.Length; i++)
                    {
                        st = st + Convert.ToString(-val1) + ",";
                    }
                }

                else
                {
                    return "0.000";
                }
            }

            else
            {
                if (!ArrExist(arr, r) && !(ArrExist(arr, rplusn)))
                {
```

```
            double db = 0.0;
            for (int i = 0; i < arr.Length; i++)
            {
                if (i != arr.Length - 1)
                {
                    db = val1 * (val2 / val4 - val3 / val4);
                    st = st + Convert.ToString(db) + ",";
                }
                else
                {
                    db = val1 * (val2 / val4 - val3 / val4);
                    st = st + Convert.ToString(db);
                }
            }

        }
        else if (ArrExist(arr, rplusn) && !(ArrExist(arr, r)))
        {
            double db = 0.0;
            for (int i = 0; i < arr.Length; i++)
            {
                if (i != arr.Length - 1)
                {
                    if (arr[i] == rplusn)
                    {
                        db = val1 * (1 + (val2 / val4));
                        st = st + Convert.ToString(db) + ",";
                    }
                    else
                    {
                        db = (val1) * (val2 / val4);
                        st = st + Convert.ToString(db) + ",";
                    }
                }
                else
                {
                    if (arr[i] == rplusn)
                    {
                        db = val1 * (1 + (val2 / val4));
                        st = st + Convert.ToString(db);
                    }
                    else
                    {
                        db = (val1) * (val2 / val4);
                        st = st + Convert.ToString(db);
                    }
                }
            }

        }
        else if (ArrExist(arr, r) && !(ArrExist(arr, rplusn)))
        {
            double db = 0.0;
            for (int i = 0; i < arr.Length; i++)
            {
                if (i != arr.Length - 1)
                {
```

```
                            if (arr[i] == r)
                            {
                                db = (-1) * (val1) * (1 + (val3 /
val4));
                                st = st + Convert.ToString(db) + ",";
                            }
                            else
                            {
                                db = (-1) * ((val1) * (val3 / val4));
                                st = st + Convert.ToString(db) + ",";
                            }
                        }
                        else
                        {
                            if (arr[i] == r)
                            {
                                db = (-1) * (val1) * (1 + (val3 /
val4));
                                st = st + Convert.ToString(db);
                            }
                            else
                            {
                                db = (-1) * ((val1) * (val3 / val4));
                                st = st + Convert.ToString(db);
                            }
                        }
                    }


                }
                else if (ArrExist(arr, r) && ArrExist(arr, rplusn))
                {
                    double db = 0.0;
                    for (int i = 0; i < arr.Length; i++)
                    {
                        if (i != arr.Length - 1)
                        {
                            if (arr[i] == r)
                            {
                                db = -val1;
                                st = st + Convert.ToString(db) + ",";
                            }
                            else if (arr[i] == rplusn)
                            {
                                db = val1;
                                st = st + Convert.ToString(db) + ",";
                            }
                            else
                            {
                                db = 0.0;
                                st = st + Convert.ToString(db) + ",";
                            }
                        }
                        else
                        {
                            if (arr[i] == r)
                            {
```

```
                        db = -val1;
                        st = st + Convert.ToString(db);
                    }
                    else if (arr[i] == rplusn)
                    {
                        db = val1;
                        st = st + Convert.ToString(db);
                    }
                    else
                    {
                        db = 0.0;
                        st = st + Convert.ToString(db);
                    }
                }
            }

        }
    }
    return st;
}
```

3.7 User Interface

The user is asked to enter the respective inputs on the Windows Forms that appear in a specific order. The first form to appear is the main form that asks for the number of levels and the names of each level. It can be seen in Figure 3.9:

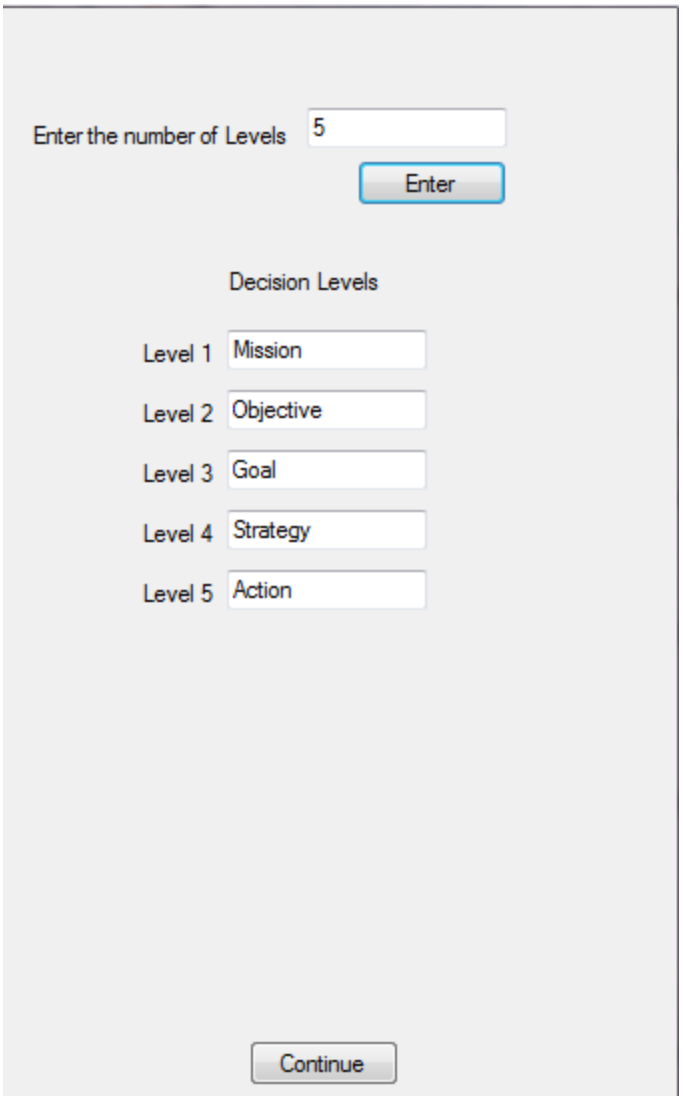


Figure 3.9 Main Form

Default names as shown in Figure 3.9 (for the first five levels) can be modified. Once the user hits "Continue", the data is saved in the storing units of the business logic. Then the user needs to select the type of hierarchy; complete or incomplete, which can be seen in Figure 3.10:

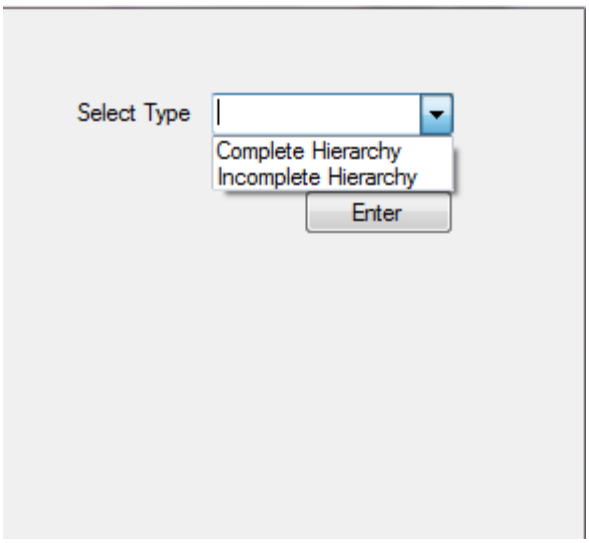


Figure 3.10 Hierarchy Option Form

On the basis of the selection made, the application proceeds further where the user enters the number of elements as well as their names on the first level. It can be seen in the Figure 3.11 and Figure 3.12:

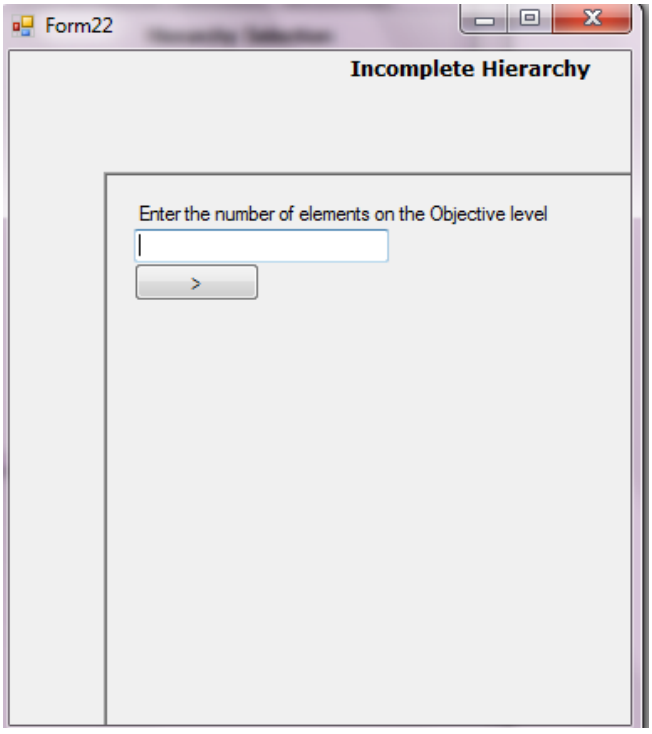


Figure 3.11 Number of the first level elements

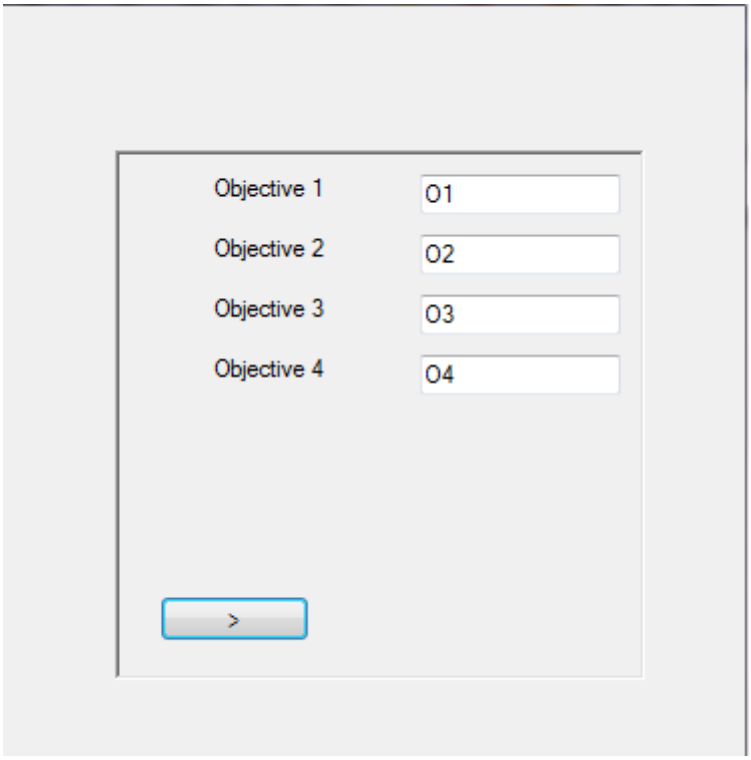


Figure 3.12 Top Level Naming

Default name of the elements are given using the first letter of the element name attached with a number starting from 1. Users are allowed to change them if needed. The user will be asked to enter the number of elements and their names at each level.

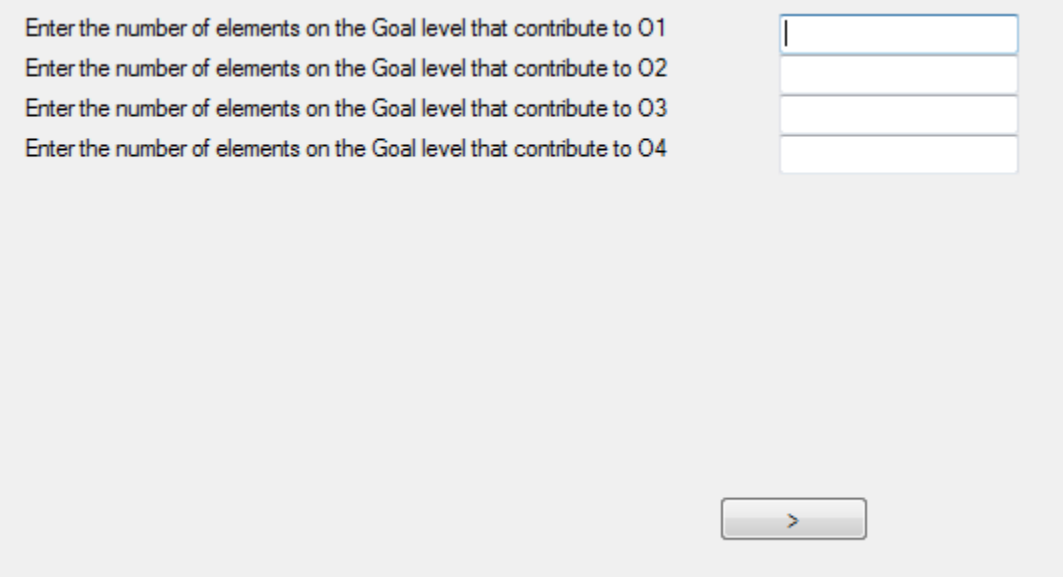


Figure 3.13 Decision Element number associated with the level above

If the model is a complete hierarchy, the same interface will be used for the rest of the levels. Figure 3.14 shows the example of the O and the G levels. However, if the model is an incomplete hierarchy, then the interfaces from 3.11 through 3.13 will be used instead to get the number and the name of the elements n the following levels.

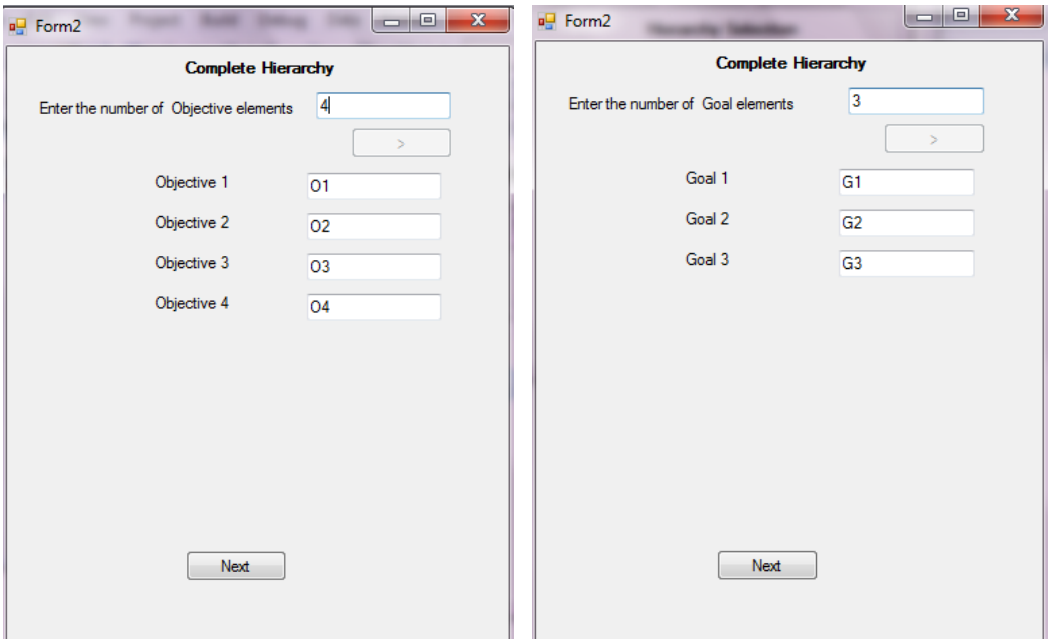


Figure 3.14 Complete Hierarchy Forms

After the user is done with all the levels, the hierarchical structure will be displayed in Tree View (.Net Component) format based on XML. It looks like a Tree whose branches can be collapsed or expanded by single click. It can be seen in Figure 3.15. It should be noted that the last level i.e. A1, A2 and A3 is duplicated in the treeview structure. It will not affect the model result. The model shown in the treeview only contains five levels M, O, G, S and A.

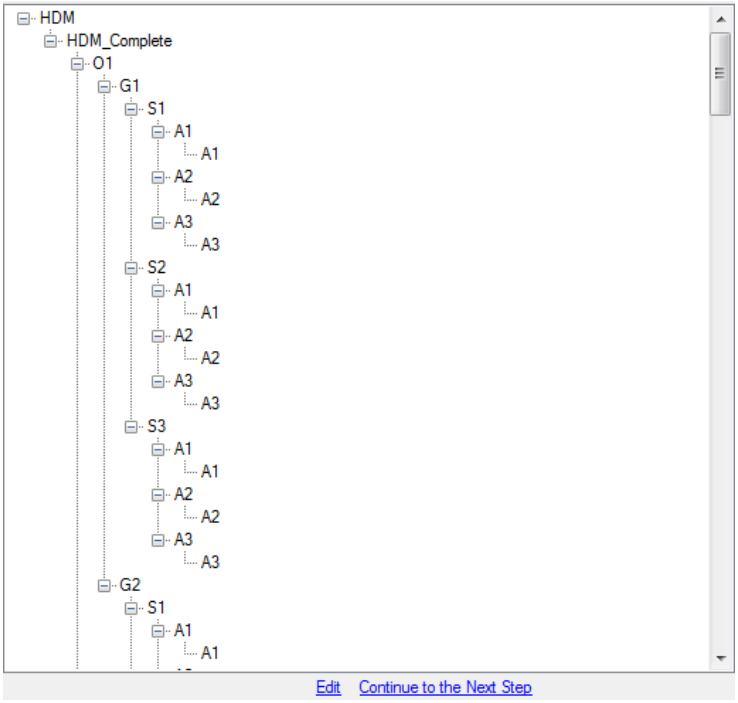


Figure 3.15 Hierarchical Model

After viewing and checking the model in the treeview format, if the user needs to make any changes in the model structure, he/she can click the "edit" link button under the tree view and it will bring the user back to the first window with all the previously input numbers and names of the elements in the text boxes on their respective forms in the editable mode. Otherwise, the user continues to the next step by clicking "Continue to Next Step" button.

The next step asks the user to select one of the options to proceed further. It can be seen in figure 3.16:

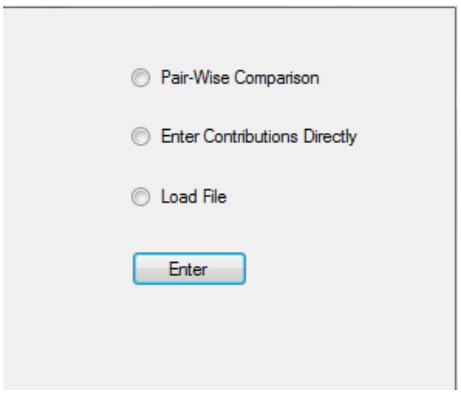


Figure 3.16 HDM Input Options

If the user selects the 'Pair Wise Comparison" option, the application takes the user to the Group Information module where the user is required to enter the number and the names of the persons that will perform the pairwise comparison. This will happen for each level. Figure 3.17 shows the user interface:

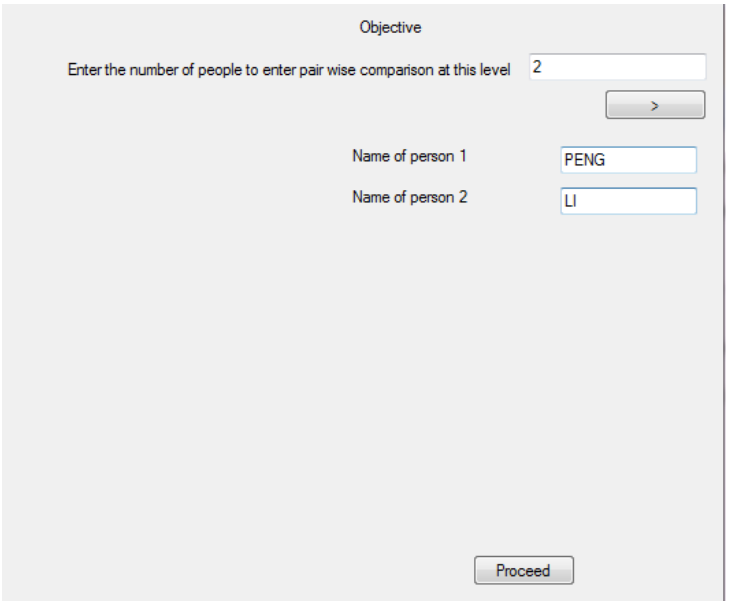


Figure 3.17 Person Naming Unit

After this, the application takes the user to the Pair Wise Comparison module where the user inputs the pairwise comparison with respect to the decision elements at that particular level. If multiple people are performing the pairwise comparison, each person will be asked to perform the

pairwise comparison and the person's name will appear on the top of the form which can be seen in Figure 3.18

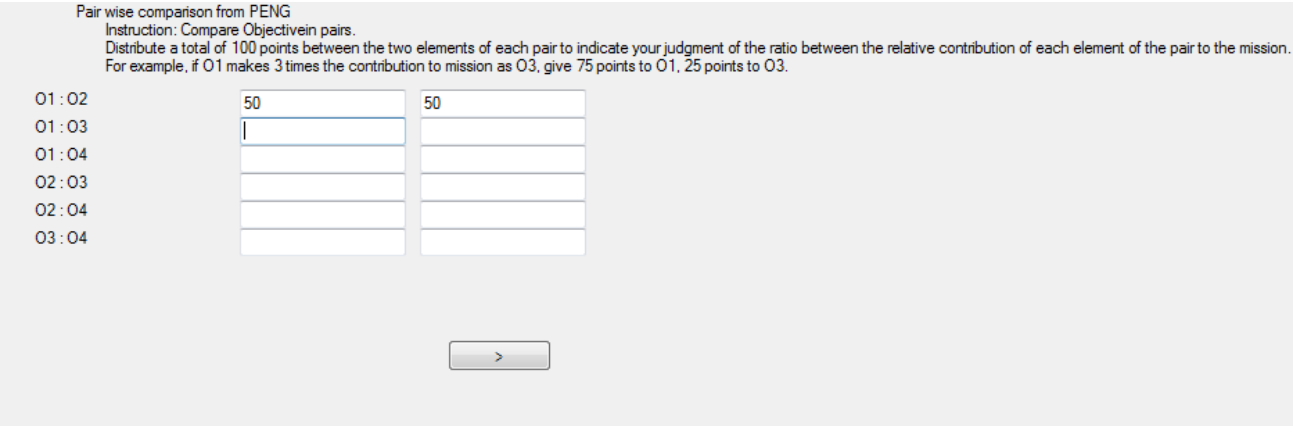


Figure 3.18 Pair Wise Comparison

When all the people are done with the pairwise comparison for all the elements at each level, data will be processed and stored in the storing units of the business logic.

If the user chooses to enter the contributions directly instead of performing the pairwise comparison, a window shown in Figure 3.19 will be shown instead. Hence the local contribution values can be entered directly.

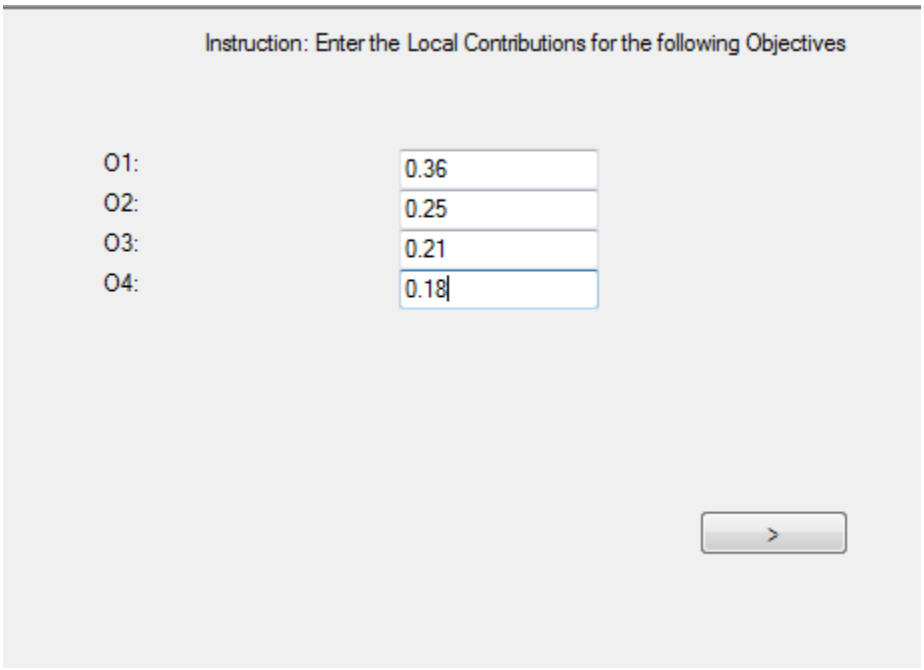


Figure 3.19 Direct Input for the top level

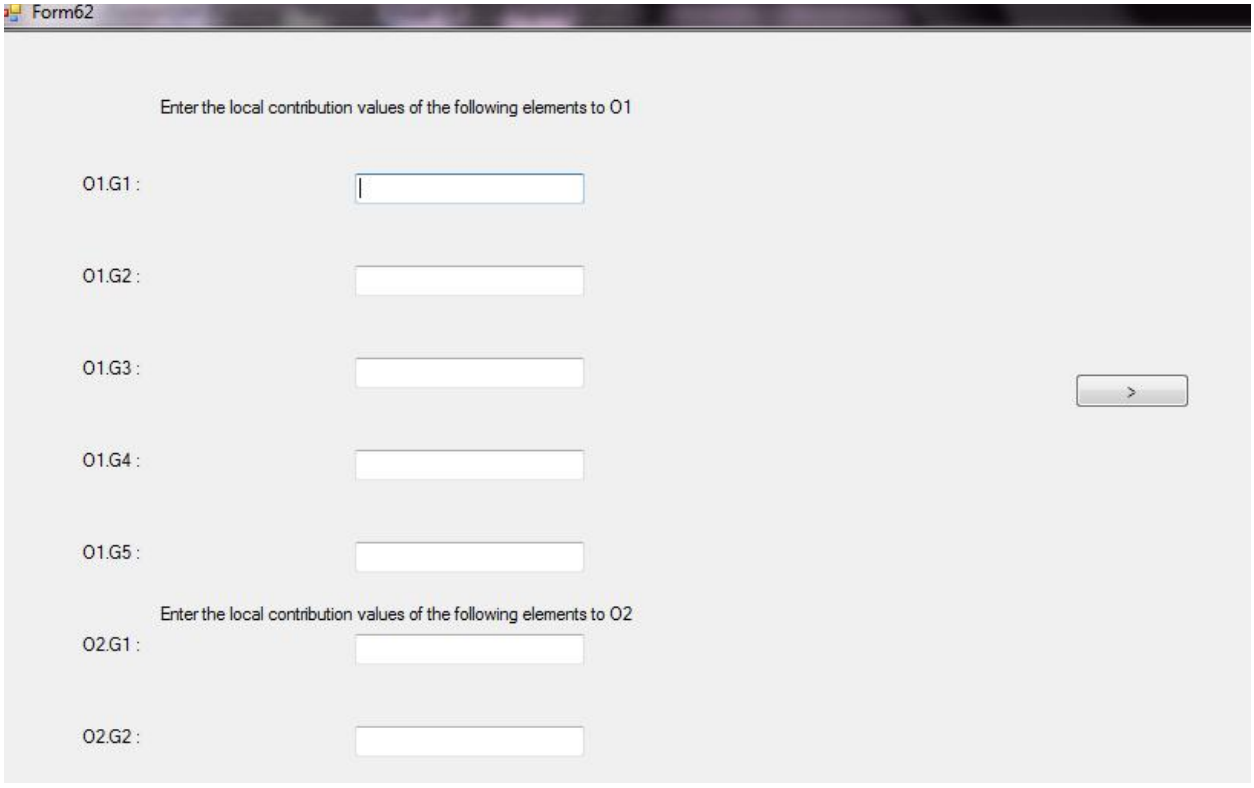


Figure 3.20 Direct Input on the middle levels

When all the inputs are done, the user can choose to see the HDM results in an excel file or continue with the Sensitivity Analysis, in one-way or m-way.

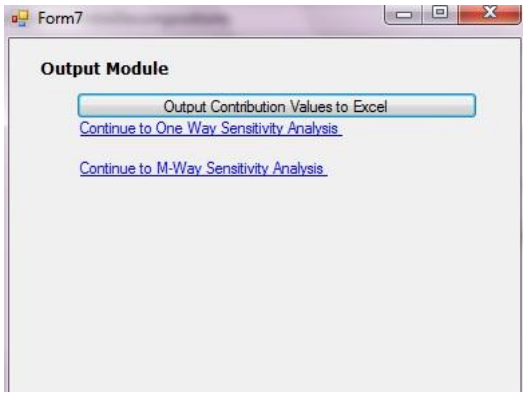
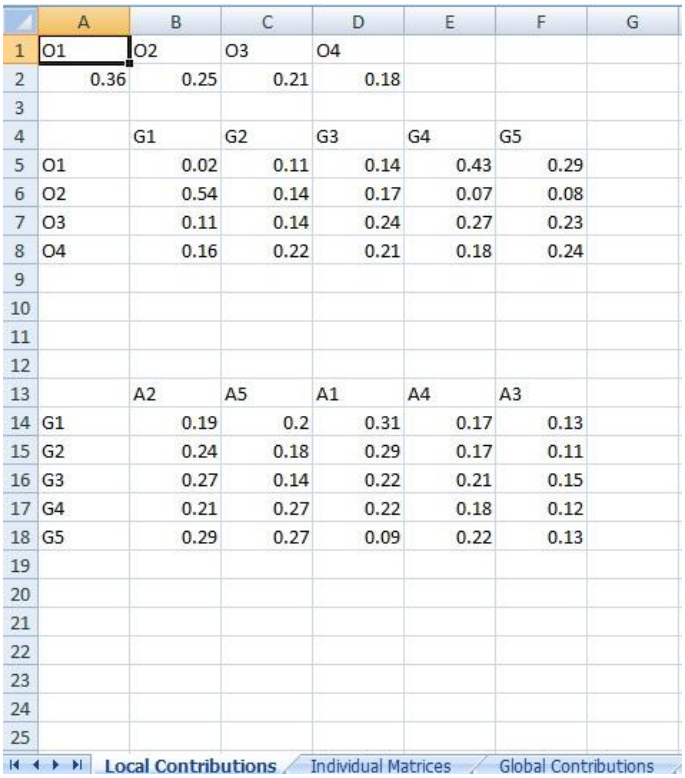


Figure 3.21 Output Module

If "HDM Results" is chosen, the output is generated in an excel file which can be saved for further on analysis. The output is generated in three different categories which are Local Contributions, Individual Matrices and the Global Contribution and they are located in three different sheets of the Excel File.



	A	B	C	D	E	F	G
1	O1	O2	O3	O4			
2	0.36	0.25	0.21	0.18			
3							
4		G1	G2	G3	G4	G5	
5	O1	0.02	0.11	0.14	0.43	0.29	
6	O2	0.54	0.14	0.17	0.07	0.08	
7	O3	0.11	0.14	0.24	0.27	0.23	
8	O4	0.16	0.22	0.21	0.18	0.24	
9							
10							
11							
12							
13		A2	A5	A1	A4	A3	
14	G1	0.19	0.2	0.31	0.17	0.13	
15	G2	0.24	0.18	0.29	0.17	0.11	
16	G3	0.27	0.14	0.22	0.21	0.15	
17	G4	0.21	0.27	0.22	0.18	0.12	
18	G5	0.29	0.27	0.09	0.22	0.13	
19							
20							
21							
22							
23							
24							
25							

Local Contributions | Individual Matrices | Global Contributions

Figure 3.22 HDM Results

If one-way sensitivity analysis is chosen, the interface shown in Figure 3.23 is displayed. After selecting the level to perturb, the user can perform the respective sensitivity analysis. If the middle level is chosen, the drop-down list will have all the middle levels displayed for the user to select a specific one.

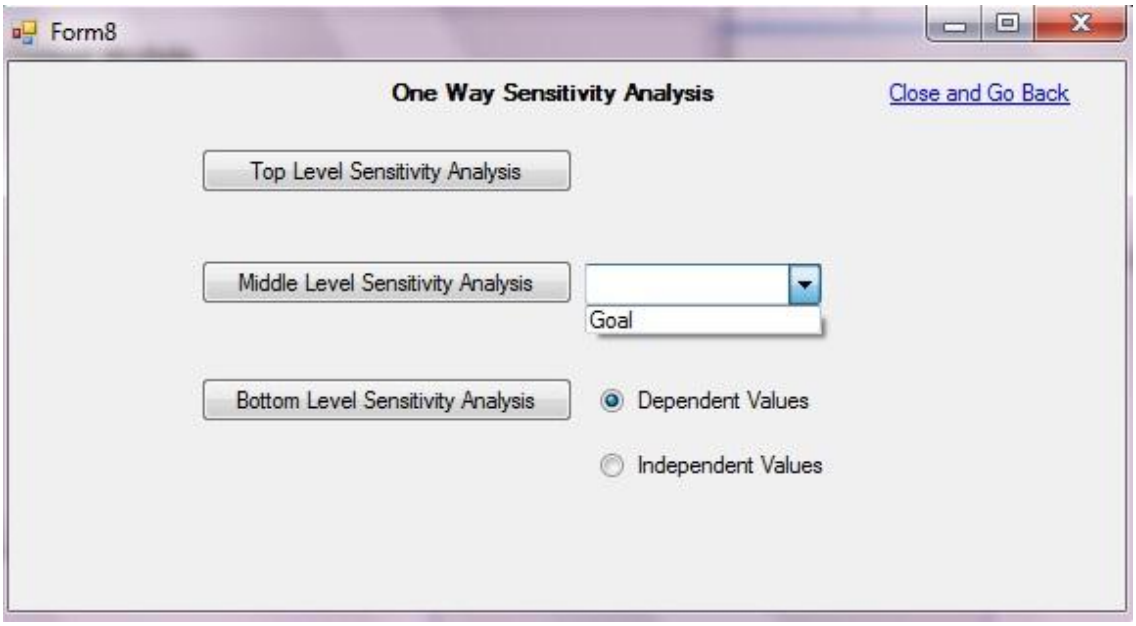


Figure 3.23 One-Way Sensitivity Analysis

If M-way sensitivity analysis is chosen, the interface shown in Figure 3.24 is displayed. After selecting the level and the elements that needs to be perturbed, the user can perform the respective sensitivity analysis.

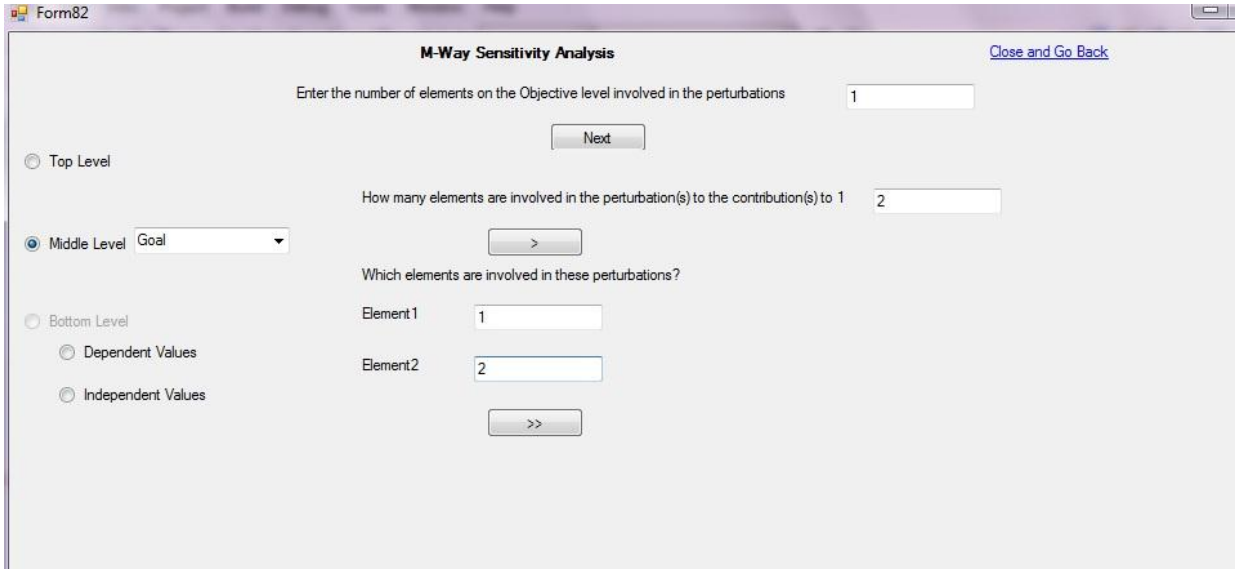


Figure 3.24 M-Way Sensitivity Analysis

The results of the Sensitivity Analysis are generated in text files. A sample text file can be seen in Figure 3.25 which was generated as a result of One-way Sensitivity Analysis of the Top Level. In the Figure 3.26, a sample of M-Way sensitivity Analysis at the Middle Level is shown:

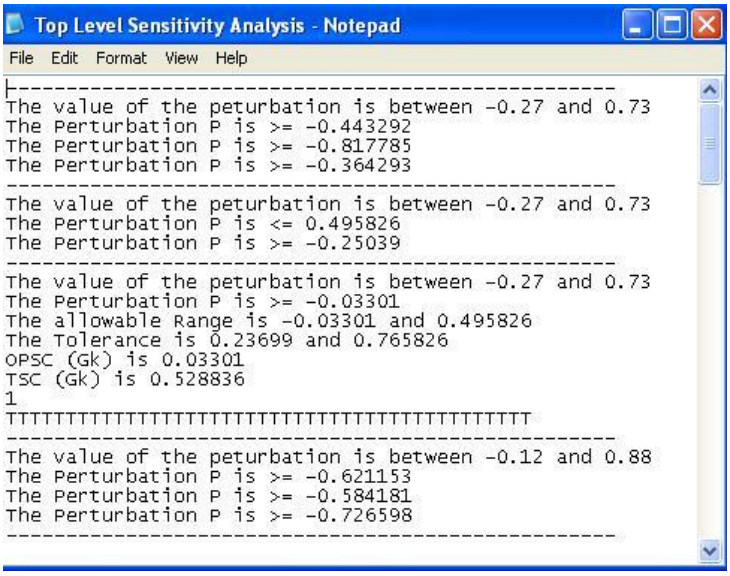


Figure 3.25 One-Way Sensitivity Analysis Results

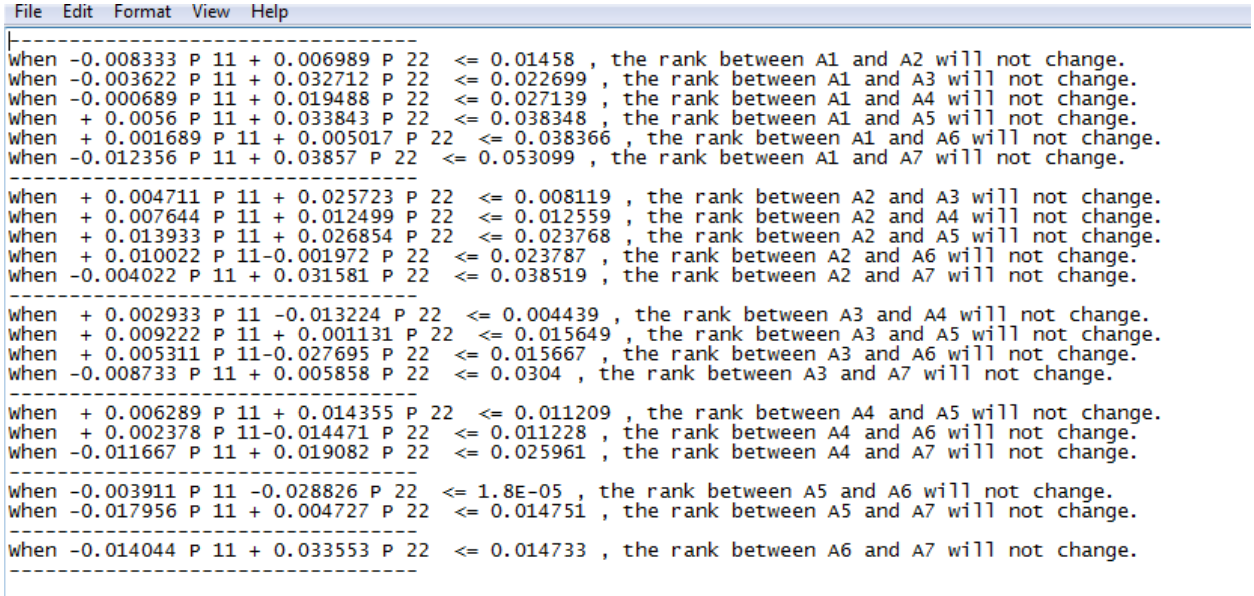


Figure 3.26 M-Way Sensitivity Analysis Results

3.8 Major difficulties met in the Development Phase

This section includes all the major issues and exceptions that were handled during the development phase of the application. There are also some unhandled issues that need to be resolved in future releases of the application.

1. The major issue that was handled was the pixel handling at several points in the application. For dynamic generation of Window Form Components like Text boxes, radio buttons, check boxes, link buttons and buttons, intense testing and calculations were performed to make sure so that the user interface window shows no pixel overlapping in them. There are only three forms in the application that have static components on them while all the others have dynamic generation of these components. (Solved)

2. The next issue was keeping a track of the names of levels, people and elements that are all stored in the Naming Unit of the Business Logic. For this reason ArrayLists and dynamic arrays (of static data types) were used. (Solved)

3. Using the Microsoft.Office.Interlop objects and methods was another issue. By taking reference from MSDN, a separate file was created that deals with all the handling of the Microsoft Excel output module that is used in the Results unit of the business logic. (Solved)

4. At some levels of Sensitivity Analysis, some multi dimensional arrays (Matrices) are converted to their opposite form. For example, from column to row or from row to column. In the beginning a separate file was created that used to handle this issue but later a separate method was included in the business logic that performs this operation. (Solved)

5. To keep a track of the Ranking of the alternatives created another issue. Since the ranking takes place at the bottom level, it affects the Sensitivity Analysis module the most. The issue was resolved when I started keeping both the ranked and unranked matrices in the business logic. (Solved)

6. The inclusion of the new mathematical corollary that was developed to solve the issue of contribution values greater than 1, also started to hinder the proper flow of the application at the

sensitivity analysis of the bottom most level. Using switch cases in the already developed module for the bottom level sensitivity analysis resolved the issue by extensive debugging. (Solved)

7. When the console version of the application was created, it could cater up to only 5 levels of the MOGSA model which had similar methods and structures. When the application switched to the Windows Forms, up to 10 levels were needed to be taken care of. For this reason, all the methods and structures were combined into one that can now accommodate up to 10 levels. (Solved)

8. The most difficult thing occurred when Stream Writers and Readers were including white spaces and carriage returns in the data. This started to create issues in the current methods which were throwing exceptions like in Figure 3.27: (Solved)

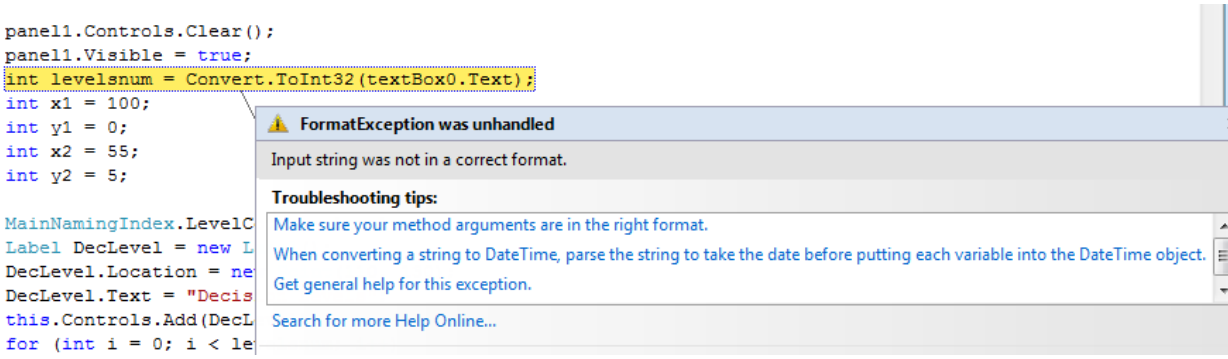


Figure 3.27 Run Time Exception

9. In the beta release of the application, the application doesn't include any validators that can validate the data. (Solved)

10. Implementing Object Oriented concepts at different levels of the development was another issue. Polymorphism and Inheritance is used in the main business logic of the application. Keeping a track of all the inherited objects and methods was an issue as the business logic file has above 7000 lines of code. (Unsolved)

11. Including a database at this level of the application development was being considered during the development. Different options were considered and in the end, XML (extensible markup language) files were used. The XML structuring created an issue as the syntax and the

protocols of XML are very carefully handled. The XML file keeps the structure of HDM model. The Console Application version of this application also includes an option for some people who can use that as an input mechanism rather than entering data manually which is discussed in future sections. This option in the Console Application is also based on XML. (Unsolved)

12. The filing has been used extensively throughout the application. The filing unit creates text files with data that are used by different modules. Keeping a track of all the different files was another issue which was handled by including all of the files in the bin folder of the application. (Unsolved)

13. Configuration of authorization and authentication mechanisms of .Net was also a major issue. (Unsolved)

4. APPLICATION OF HDM SA (SOFTWARE APPLICATION)

To test and also demonstrate the software application, HDM SA was performed on different models. Two of those applications are presented in this chapter. The first model is the Technology Assessment model developed by (Ho 2004) and the second one is the Technology Development Envelope (TDE) model developed by (Kocaoglu, 2007 #4). Both models use the HDM algorithm introduced in Chapter 1 to assess the decision alternative's contribution to the overall mission.

4.1 Ho's Technology Assessment Model

Around 10 people from different domains including technology, government and research organizations formed the expert panel for the Ho's model. In Figure 4.1, the technology evaluation is presented:

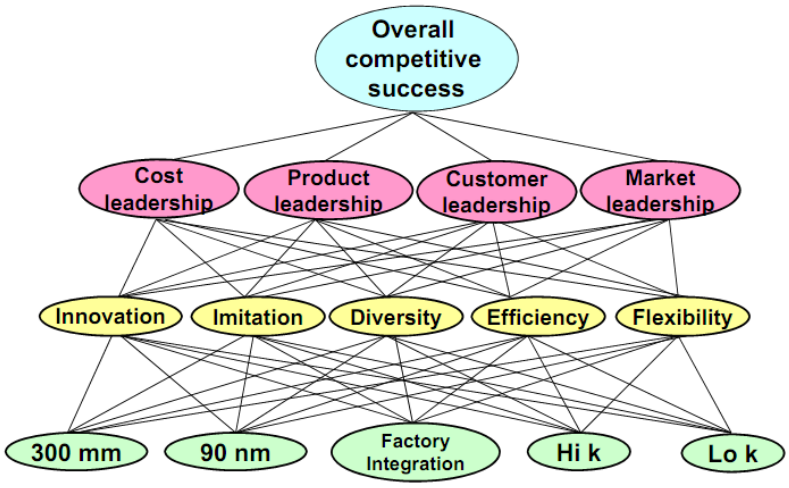


Figure 4.1 Ho's Model (Ho 2004)

The model is based on the MOGSA Model that includes Mission, Objectives, Goals, Strategies and Actions (Alternatives) at successive levels. The first level, the Mission level in this case is the Overall Competitive Success of a company that is measured in terms of the Return on Investment. The next Level is of the Goals which are based on leadership in different dimensions like Cost,

Product, Customer and Market Leadership. The third level includes the strategies that are needed to be undertaken to achieve the goals at the level above. The strategies include innovation, imitation, diversity, efficiency and flexibility. The last level is the different technology alternatives that are under evaluation. There are five of such technologies; 300 mm, 90 nm, Factory Integration, Hi K and Lo K.

4.1.1 HDM Results

The Direct Input module was used to input the local contributions to the software. Results can be seen in the Figure 4.2. The snapshot is taken from the excel file generated as a result of the Result Module. The first part depicts the local contributions of the objectives to the strategies and strategies to the actions. It also shows the global contributions of the objectives to the overall mission. The second part depicts the global contributions of the strategies and the overall contributions of the actions to the overall mission.

	A	B	C	D	E	F
1	O1	O2	O3	O4		
2	0.36	0.25	0.21	0.18		
3						
4		S1	S2	S3	S4	S5
5	O1	0.02	0.11	0.14	0.43	0.29
6	O2	0.54	0.14	0.17	0.07	0.08
7	O3	0.11	0.14	0.24	0.27	0.23
8	O4	0.16	0.22	0.21	0.18	0.24
9						
10						
11						
12						
13		A2	A5	A1	A4	A3
14	S1	0.19	0.2	0.31	0.17	0.13
15	S2	0.24	0.18	0.29	0.17	0.11
16	S3	0.27	0.14	0.22	0.21	0.15
17	S4	0.21	0.27	0.22	0.18	0.12
18	S5	0.29	0.27	0.09	0.22	0.13

	A	B	C	D	E	F
1	S1	S2	S3	S4	S5	
2	0.1941	0.1436	0.1811	0.2614	0.2159	
3						
4	A2	A5	A1	A4	A3	
5	0.237745	0.218893	0.218596	0.18999	0.127629	
6						

Figure 4.2 HDM Output in Excel File

4.1.2 Sensitivity Analysis

After getting the results from the HDM Module, Sensitivity Analysis at different levels, for both One-Way and M-Way SA are executed. Since the excel files were developed by Dr. Hongyi Chen while developing her dissertation to calculate the allowable range/region of the perturbations, result from the software were compared with the results during the development of the HDM SA algorithm.

4.1.2.1 One-Way Sensitivity Analysis

The following section presents all the One-Way Sensitivity Analyses results at the Top, Middle and the Bottom Levels.

The top level includes four different objectives based on leadership in different domains including Cost, Product, Customer and Market. In the following snapshots, comparison is done for the allowable range of the perturbations induced on C_1^O, C_2^O, C_3^O, C_4^O and C_5^O.

Figure 4.3 compares the results of excel file and the software results. The allowable range of the perturbation induced on the local contribution of the first objective can be seen in the highlighted parts, which are the same on both files. Possible rank changes between the indicated pair of the technology alternatives will occur when the perturbation value goes beyond the given threshold value.

Excel Results

			A	B	C	D	A-B+C-D	Єℓ*<=(positive) Єℓ*>=(negative)
	a_{1+n}	a_1-a_{1+n}	$\alpha_{i+n\ell^*}$	$\alpha_{i\ell^*}$	$\sum\alpha_{i\ell}\times O\ell/\sum O\ell$	$\sum\alpha_{i+n\ell}\times O\ell/\sum O\ell$		
n=1	a_2	0.0146	0.24	0.24	0.2321875	0.209375	0.0228125	0.8459
n=2	a_3	0.0154	0.19	0.24	0.2321875	0.23625	-0.0540625	**-0.3642**
n=3	a_4	0.0421	0.2	0.24	0.2321875	0.18890625	0.00328125	-15.712
n=4	a_5	0.1029	0.13	0.24	0.2321875	0.13328125	-0.0110938	-9.670
Conclusion: when -0.285<=Єℓ*<0.64, T2 will remain a1.								

Text File generated by the software

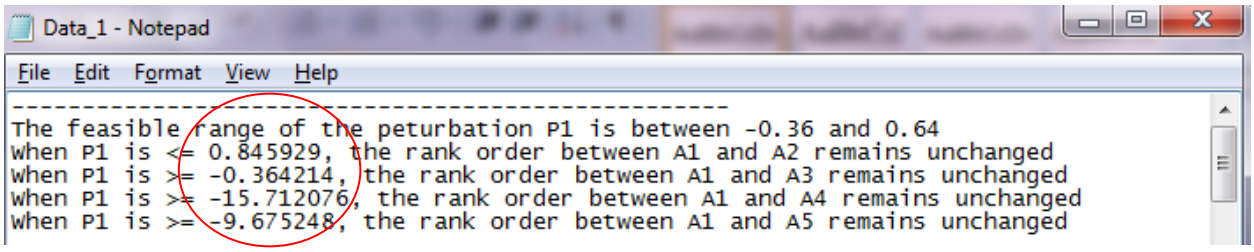


Figure 4.3 Comparison of the software generated results with excel results (Top)

The Middle Level includes five different strategies namely innovation, imitation, diversity, efficiency and flexibility. The Sensitivity Analysis performed at this level indicates the rank changes when the perturbation is applied on the contribution from a middle level element to a top level element. Next, set of inequalities were generated to show the allowable range of perturbations induced on the local contributions of the strategies. In the following snapshots, comparison is done between the excek results and the software generated results:

Figure 4.4 compares the results of Excel file and the software results. The highlighted parts show the perturbation on the contribution value of the first Goal (G1) to the first Objective (O1) which are the same on both files. The snapshot also indicates the possible rank changes if the value of the perturbation goes beyond the threshold value.

Excel Results

27	**Calculating Єk*ℓ***										
28	When r = 1,										
29		Ar	Ar+n	Ar-Ar+n	Oℓ*	a_{r+n,k^*}	a_{rk^*}	$\sum a_{r+n,k}\times g_{kl^*}$	$\sum a_{r,k}\times g_{kl^*}$	$\sum g_{k\ell^*}$	Єk*ℓ*
30	n=1	a1	a2	0.018852	0.36	0.2	0.19	0.2338	0.2386	0.97	3.503149
31	n=2	a1	a3	0.019149	0.36	0.31	0.19	0.1834	0.2386	0.97	0.300676
32	n=3	a1	a4	0.047755	0.36	0.17	0.19	0.1893	0.2386	0.97	4.303451
33	n=4	a1	a5	0.110116	0.36	0.13	0.19	0.1224	0.2386	0.97	5.115542
34	**Conclusion: when -0.02<=Єℓ*<0.3, T2 will remain a1.**										

Text File generated by the software

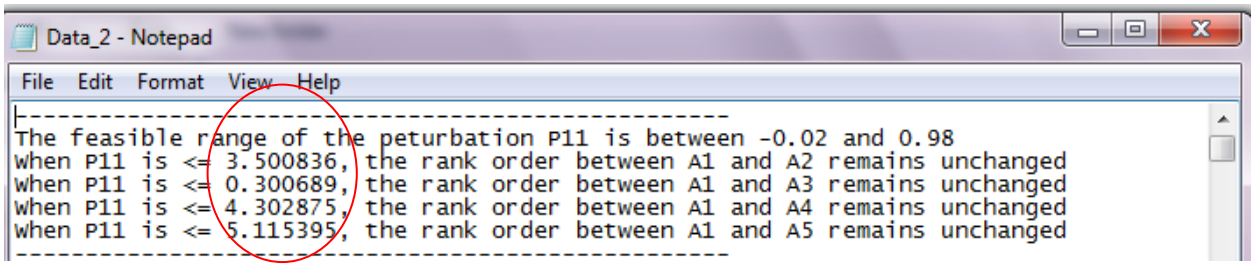


Figure 4.4 Comparison of the software generated results with manual results (Middle)

There are five alternatives at the bottom level. The Sensitivity Analysis performed at this level indicates the rank changes when the perturbation is applied at the contribution values of the bottom level elements to the middle level elements. In the following snapshots, comparison is done between the excel results and the software generated results.

Figure 4.5 compares the results of Excel file and the software results. The allowable range of the perturbation induced can be seen in the highlighted parts The highlighted parts show the perturbation on the contribution value of the second Alternative (A2) to the third strategy (S3) which is the same on both files. The snapshot also indicates the possible rank changes if the value of the perturbation goes beyond the threshold value.

Excel Results

When r = 1,

	Ar	Ar+n	$a_{r}-a_{r+n}$	s_{j^*}	$a_{r+n,\,j^*}$	$a_{r,\,j^*}$	$\sum a_{ij^*}$		$\epsilon_{i^*j^*}$
n=1	a1*	a2	0.019	0.1941	0.200	0.190	0.810	>=	-0.07789
n=2	a1*	a3	0.019	0.1941	0.310	0.190	0.810	<=	-0.07135
n=3	a1*	a4	0.048	0.1941	0.170	0.190	0.810	<=	-0.20335
n=4	a1*	a5	0.110	0.1941	0.130	0.190	0.810	<=	-0.48886

Conclusion: when -0.071 <= $\epsilon_{i^*j^*}$ <= 0.81, T2 will remain a1.

Text File generated by the software

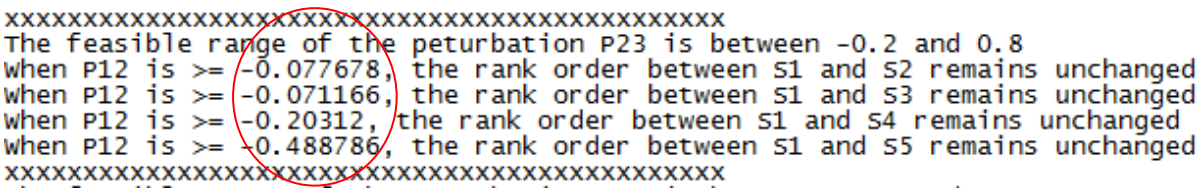


Figure 4.5 Comparison of the software generated results with manual results (Bottom)

4.1.2.2 M-Way Sensitivity Analysis

There are times when more than one contributions at a level of the decision hierarchy change simultaneously. Therefore in this section, M-Way Sensitivity Analysis is performed where the

number of perturbations, indicated by the "M" value at a certain level is greater than one. This analysis is also performed at all the different levels.

The scenario is depicted in Figure 4.6 is one where there are two perturbations induced at the top level. The contribution values of the objectives selected for this purpose are O1 and O2.

Excel Results

When r = 1

$O\ell=\sum\alpha 1\ell\times O\ell/\sum O\ell$ 0.24

		A	B	C	D	E	F	A-B+C-D	A-B+E-F
a_{1+n}	a_{1-a1+n}	$\sum\alpha r\ell\times O\ell/\sum O\ell$	$\sum\alpha r+n\ell\times O\ell/\sum O\ell$	$\alpha r+n\ell 1^*$	$\alpha r\ell 1^*$	$\alpha r+n\ell 2^*$	$\alpha r\ell 2^*$		
a_2	0.0146	0.24	0.2154	0.24	0.24	0.2	0.22	0.0246	0.0072
a_3	0.0154	0.24	0.2146	0.19	0.24	0.27	0.22	-0.0217	0.0791
a_4	0.0421	0.24	0.1946	0.2	0.24	0.18	0.22	0.0022	0.0134
a_5	0.1029	0.24	0.1354	0.13	0.24	0.13	0.22	-0.0012	0.0261

When r = 2

$O\ell=\sum\alpha 2\ell\times O\ell/\sum O\ell$ 0.215384615

		A	B	C	D	E	F	A-B+C-D	A-B+E-F
a_{2+n}	a_{2-a2+n}	$\sum\alpha r\ell\times O\ell/\sum O\ell$	$\sum\alpha r+n\ell\times O\ell/\sum O\ell$	$\alpha r+n\ell 1^*$	$\alpha r\ell 1^*$	$\alpha r+n\ell 2^*$	$\alpha r\ell 2^*$		
a_3	0.0008	0.2154	0.2146	0.19	0.24	0.27	0.2	-0.0492	0.0708
a_4	0.0275	0.2154	0.1946	0.2	0.24	0.18	0.2	-0.0229	0.0062
a_5	0.0883	0.2154	0.1354	0.13	0.24	0.13	0.2	-0.0262	0.0190

Text File generated by the software

Output_SATest - Notepad

File Edit Format View Help

```
-----------------------------------
when  + 0.025069 P 11 + 0.007169 P 12  <= 0.018852 , the rank between A1 and A2 will not change.
when -0.021669 P 11 + 0.079131 P 12  <= 0.019149 , the rank between A1 and A3 will not change.
when  + 0.002192 P 11 + 0.013392 P 12  <= 0.047755 , the rank between A1 and A4 will not change.
when -0.001177 P 11 + 0.026123 P 12  <= 0.110116 , the rank between A1 and A5 will not change.
-----------------------------------
when -0.046738 P 11 + 0.071962 P 12  <= 0 , the rank between A2 and A3 will not change.
when -0.022877 P 11 + 0.006223 P 12  <= 0.018852 , the rank between A2 and A4 will not change.
when -0.026246 P 11 + 0.018954 P 12  <= 0.019149 , the rank between A2 and A5 will not change.
-----------------------------------
when  + 0.023861 P 11 -0.065739 P 12  <= 0.047755 , the rank between A3 and A4 will not change.
when  + 0.020492 P 11 -0.053008 P 12  <= 0.110116 , the rank between A3 and A5 will not change.
```

Figure 4.6 M-Way SA (Top)

Figure 4.6 shows the software output of the allowable region of the two perturbations to keep certain pairs of the decision alternatives unchanged.

At the middle level, two strategy contributions to the first goal are selected to be perturbed with respect to their contribution to the first objective. Following snapshot in Figure 4.7 compares the excel result with the software output.

J	K	L	M	N	O	P	Q	R	S
a_{r+n,k_1}*	a_{rk_1}*	a_{r+n,k_2}*	a_{rk_2}*	A	B				
0.2	0.19	0.18	0.24	0.0028	-0.0224		0.0028*x-0.0224*y<=0.0189	y>=	-0.84375
0.31	0.19	0.29	0.24	0.0686	0.0434		0.0686*x+0.0434*y<=0.0191	y<=	2.11751
0.17	0.19	0.17	0.24	0.0102	-0.0078		0.0102*x-0.0078*y<=0.0478	y>=	-6.12821
0.13	0.19	0.11	0.24	0.0211	-0.0041		0.0211*x-0.0041*y<=0.1101	y>=	-26.8537
a_{r+n,k_1}*	a_{rk_1}*	a_{r+n,k_2}*	a_{rk_2}*	A	B				
0.31	0.2	0.29	0.18	0.0658	0.0658		0.0658*x+0.0658*y<=0.0003	y<=	0.00456
0.17	0.2	0.17	0.18	0.0074	0.0146		0.0074*x+0.0146*y<=0.0289	y<=	1.97945
0.13	0.2	0.11	0.18	0.0182	0.0182		0.0182*x+0.0182*y<=0.0913	y<=	5.01648
a_{r+n,k_1}*	a_{rk_1}*	a_{r+n,k_2}*	a_{rk_2}*	A	B				
0.17	0.31	0.17	0.29	-0.0584	-0.0512		-0.0584*x-0.0512*y<=0.0286	y>=	-0.55859
0.13	0.31	0.11	0.29	-0.0476	-0.0476		-0.0476*x-0.0476*y<=0.091	y>=	-1.91176

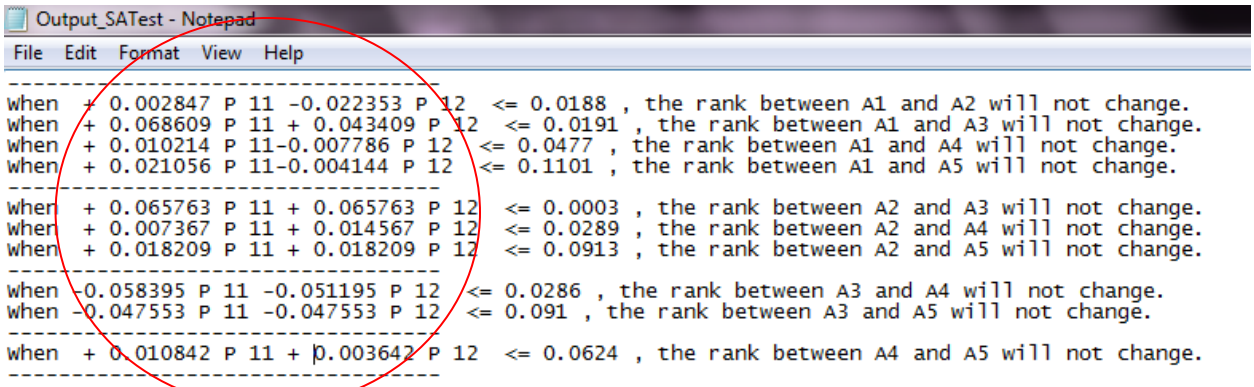


Figure 4.7 Comparison of the software generated results with manual results (M-Way Middle)

The software output indicates the allowable region to keep the certain pair of decision alternatives unchanged.

At the bottom level, Alternative 3 and Alternative 1's contributions to Strategy 1 were selected. Following perturbed are the comparison of the excel results and the results generated by the software.

A	B	C	D	E	F	G	H	I	J
When r = 1,				p1	p3	p2	p4	a3*	a1*
	Ar	Ar+n	a_{r}-a_{r+n}	s_{j}*	$a_{r+n,\ j}$*	$a_{r,\ j}$*	$\sum a_{ij}$*	Pa3	Pa1
n=1	a1*	a2	0.019	0.1941	0.200	0.190	0.500	-0.078	-0.27174
n=2	a1*	a3*	0.019	0.1941	0.310	0.190	0.500	0.1941	-0.1941
n=3	a1*	a4	0.048	0.1941	0.170	0.190	0.500	-0.066	-0.260094
n=4	a1*	a5	0.110	0.1941	0.130	0.190	0.500	-0.050	-0.244566
T2 will remain a1.									
When r = 2,									
	Ar	Ar+n	a_{r}-a_{r+n}	s_{j}*	$a_{r+n,\ j}$*	$a_{r,\ j}$*	$\sum a_{ij}$*		
n=1	a2	a3*	0.000	0.1941	0.310	0.200	0.500	0.272	0.07764
n=2	a2	a4	0.029	0.1941	0.170	0.200	0.500	0.012	0.012
n=3	a2	a5	0.091	0.1941	0.130	0.200	0.500	0.027	0.027
T5 will remain a2.									
When r = 3,									
	Ar	Ar+n	a_{r}-a_{r+n}	s_{j}*	$a_{r+n,\ j}$*	$a_{r,\ j}$*	$\sum a_{ij}$*		
n=1	a3*	a4	0.029	0.1941	0.170	0.310	0.500	-0.260094	-0.066
n=2	a3*	a5	0.091	0.1941	0.130	0.310	0.500	-0.244566	-0.050
T1 will remain a3.									
When r = 4,									
	Ar	Ar+n	a_{r}-a_{r+n}	s_{j}*	$a_{r+n,\ j}$*	$a_{r,\ j}$*	$\sum a_{ij}$*		
n=1	a4	a5	0.062	0.1941	0.130	0.170	0.500	0.016	0.016

MyTextFile_1 - Notepad

File Edit Format View Help

```
When P11 -0.07764 P21 -0.27174 is < 0.0188, then the rank between A1 and A2 will not change
When P11 0.1941 P21 -0.1941 is < 0.0191, then the rank between A1 and A3 will not change
When P11 -0.065994 P21 -0.260094 is < 0.0477, then the rank between A1 and A4 will not change
When P11 -0.050466 P21 -0.244566 is < 0.1101, then the rank between A1 and A5 will not change
--------------------------------------------------
When P11 0.27174 P21 0.07764 is < 0.0188, then the rank between A2 and A3 will not change
When P11 0.011646 P21 0.011646 is < 0.0191, then the rank between A2 and A4 will not change
When P11 0.027174 P21 0.027174 is < 0.0477, then the rank between A2 and A5 will not change
--------------------------------------------------
When P11 -0.260094 P21 -0.065994 is < 0.0188, then the rank between A3 and A4 will not change
When P11 -0.244566 P21 -0.050466 is < 0.0191, then the rank between A3 and A5 will not change
--------------------------------------------------
When P11 0.015528 P21 0.015528 is < 0.0188, then the rank between A4 and A5 will not change
--------------------------------------------------
S1
```

Figure 4.8 Comparison of the software generated results with manual results (M-Way Bottom)

4.1.3 Summary of Results

4.1.3.1 Top Level (One-Way)

Table 4.1 summarizes the results generated by the software application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbation goes beyond the threshold value specified in the first column. Due to large amount of data, only the first objective is selected for the purpose i.e. the Cost Leadership, which when affected by the perturbation, results in the rank changes as listed in the last column.

P_1^G	Rank Change
> 0.84	**(1,2)**
< -0.364214	**(1,3)**
< -15.712076	**(1,4)**
< -9.675248	**(1,5)**
< -0.004168	**(2,3)**
< -1.142366	**(2,4)**
> 2.207971	**(3,5)**
< -7.476025	**(4,5)**

Table 4.1 Rank changes in the top level

4.1.3.2 Middle Level (One-Way)

Table 4.2 summarizes the results generated by the application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbation goes beyond the threshold value specified in the first column. Due to large amount of data, only the changes to the contribution of the first goal to the first objective, are selected for the purpose i.e. the contribution of Innovation to Cost Leadership, which when affected by the perturbation, results in the rank changes as listed in the last column.

P_{11}^{S-G}	**Rank Change**
>3.500836	**(1,2)**
>0.300689	**(1,3)**
>4.302875	**(1,4)**
>5.115395	**(1,5)**
>0.005351	**(2,3)**
>5.058793	**(2,4)**
>5.653865	**(2,5)**
<-0.543811	**(3,4)**
<-2.157539	**(3,5)**
>5.97996	**(4,5)**

Table 4.2 Rank changes at the middle level

4.1.3.3 Bottom Level (One-Way)

Table 4.3 summarizes the results generated by the application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbation goes beyond the threshold value specified in the first column. Due to large amount of data, only the changes to the contribution of the first alternative to the first goal, is selected for the purpose i.e. the contribution of Increasing Wafer Size to Innovation, which when affected by the perturbation, results in the rank changes as listed in the last column.

P_{11}^{T-S}	**Rank Change**
<-6.683256	**(1,2)**
>0.077157	**(1,3)**
>8.478493	**(1,4)**
>6.523285	**(1,5)**
>0.001198	**(2,3)**
>3.424576	**(2,4)**

>4.636636	**(2,5)**
<-0.11822	**(3,4)**
<-0.394503	**(3,5)**
>5.545681	**(4,5)**

Table 4.3 Rank changes at the bottom level

4.1.3.4 Top Level (M-Way)

Table 4.4 summarizes the results generated by the application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbations times the variables (e.g. x and y, 0.0246x and 0.0046y) less than or greater than the lambda value (obtained as a part of the results). For the demonstration, the allowable region for perturbations induced on the contribution values of Cost to Product Leadership is shown.

P_1	**P_2**	**Lambda**	**Rank Change**
0.0246	0.0046	0.0188	**(1,2)**
-0.0246	0.0754	0.0191	**(1,3)**
0.0054	0.0054	0.0477	**(1,4)**
-0.0054	0.0146	0.1101	**(1,5)**
-0.0492	0.0708	0.0003	**(2,3)**
-0.0192	0.0008	0.0289	**(2,4)**
-0.0300	0.0100	0.0913	**(2,5)**
0.0300	-0.0700	0.0286	**(3,4)**
0.0192	-0.0608	0.0910	**(3,5)**
-0.0108	0.0092	0.0624	**(4,5)**

Table 4.4 Rank changes at the top level (M-Way)

4.1.3.5 Middle Level (M-way)

Table 4.5 shows the snapshot of the results generated by the application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbations times the variables (e.g. x and y, 0.0028x and -0.0224y) less than or greater than the lambda value (obtained as a part of the results). For the demonstration, only a glimpse of the results is shown where the contribution values selected are the contributions of Innovation and Imitation to Cost Leadership.

P_1	P_2	Lambda	Rank Change
0.0028	-0.0224	0.018852	**(1,2)**
0.0686	0.0434	0.019149	**(1,3)**
0.0102	-0.0078	0.047755	**(1,4)**
0.0211	-0.0041	0.110116	**(1,5)**
0.0658	0.0658	0	**(2,3)**
0.0074	0.0146	0.018852	**(2,4)**
0.0182	0.0182	0.019149	**(2,5)**
-0.0584	0.0512	0.047755	**(3,4)**
-0.0476	-0.0476	0	**(3,5)**
0.0108	0.0036	0.110116	**(4,5)**

Table 4.5 Rank changes at the middle level (M-Way)

4.1.3.6 Bottom Level (M-Way)

Following is also a snapshot that is also taken from the results generated from the application. It shows the pair of decision alternatives whose rank order will be reversed if the value of the perturbations times the variables (e.g. x and y, -0.078x and -0.27174y) less than or greater than the lambda value (obtained as a part of the results). For the demonstration, only a glimpse of the results is shown where the contribution values selected are the contributions of Increasing Wafer Size and Reducing Line Width to Innovation.

P_1	P_2	Lambda	Rank Change
-0.078	-0.27174	0.049201	**(1,2)**
0.1941	-0.1941	0.051151	**(1,3)**
-0.066	-0.260094	0.112883	**(1,4)**
-0.050	-0.244566	0.001951	**(1,5)**
0.272	0.07764	0.063683	**(2,3)**
0.012	0.012	0.061732	**(2,4)**
0.027	0.027	0.012455	**(2,5)**
-0.260094	-0.066	0.014567	**(3,4)**
-0.244566	-0.050	0.135667	**(3,5)**
0.016	0.016	0.035607	**(4,5)**

Table 4.6 Rank changes at the bottom level (M-Way)

4.2 Applying the software to the test the Sensitivity of a Technology Development Envelope Model

4.2.1 The TDE Model

To test and demonstrate the software, HDM SA was also performed on a model developed by Nathasit Gerdsri and Dundar F. Kocaoglu. The model is known as the Technology Development Envelope (TDE) and AHP is its main method. A hierarchical decision model is presented as a part of the study where Criteria, factors and technology alternatives reside on three levels of the hierarchy.

The TDE framework is structured by obtaining strategic information on the development of technologies and then using this information to evaluate the value of each technology based on the impacts of its characteristics on the organization's objectives in each time period. A technology development envelope is formed by connecting technologies that have the highest value in each period throughout the specified timeframe. (Kocaoglu, 2007 #4)

The AHP method and Delphi method are combined as a part of TDE framework. The Delphi method is a systematic, interactive forecasting method which relies on a panel of experts. It deals with obtaining the strategic information about the development of technologies. AHP is used for the evaluation of the technologies. Results are then further used to form the TDE. Following is the formula that is used to calculate the value from the AHP model to evaluate technologies:

$$\text{TVn} = \sum_{k=1}^{K} * \sum_{jk=1}^{Jk} wk * fjk,k * V(tn,jk,k)$$

Where

T Vn: Technology value of technology (n) determined according to a company's objective

wk : Relative priority of criterion (k) with respect to the company objective

f jk ,k : Relative importance of factor (jk) with respect to criterion (k) PJk

V(tn, jk ,k): Desirability value of the performance and physical characteristics of technology (n) along factor (jk) for criterion (k)

As a result, the value of a technology indicates the degree to which each technology exclusively satisfies an organization's objective. The ideal technology from an organization's point of view would represent the value of 100. (Kocaoglu, 2007 #4)

4.2.2 Data Collection

All the decision elements that include Criteria, Factors and Technology Alternatives which make up the hierarchical decision model are presented in the

Objective	Criterion	Factor
To achieve technical competitiveness from the new thermal platform development for computer servers	C1: Performance	F11: Heat Removal Flux F21: Thermal Resistance F22: Temperature Controllability
	C2: Geometric	F12: Height F22: Pooling Space F32: Weight F42: Distance of Heat

		Transportation
	C3: Reliability	F13: Continuous Operation F23: Durability under Advance environment condition F33: % of performance drop overtime F43: Length of the warming up period at the start F53: Longevity
	C4: Economic	F14: Power Consumption of the cooling system F24: Cost of Fabrication F34: Cost of recharging, servicing and reclamation
	C5: Environmental Compatibility	F15: Toxic control of cooling media and combustion products F25: Temperature control of exhausted coolant (air/gas/liquid)
	C6: Serviceability and maintenance	F16: Installation and maintenance complexity F26: Interchangeability of components
	C7: Flexibility	F17: Physical mold ability F27: Scalability F37: Upgradeability

Table 4.7 Structure of the TDE Model

Table 4.8 shows the local contributions of the criteria and factors to the overall mission. The desirability values or the local contribution values of the technology alternatives to the factors in the year 2004 are shown in Appendix A.

Criterion	Contribution	Factors	Contribution
C1: Performance	0.27	F11: Heat removal flux	0.34
	0	F21: Thermal resistance	0.46
	0	F31: Temperature controllability	0.2
C2: Geometric	0.12	F12: Height	0.33
	0	F22: Footing space	0.35
	0	F32: Weight	0.16
	0	F42: Distance of heat transportation	0.16
C3: Reliability	0.2	F13: Continuous operation	0.22

	0	F23: Durability under adverse environment conditions	0.2
	0	F33: % of performance drop overtime	0.22
	0	F43: Length of the warming up period at start	0.14
	0	F53: Longevity	0.22
C4: Economic	0.15	F14: Power consumption for cooling system	0.52
	0	F24: Cost of fabrication	0.23
	0	F34: Cost for recharging, servicing and reclamation	0.25
C5: Environmental Compatibility	0.09	F15: Toxic control of cooling media and combustion products	0.54
	0	F25: Temperature control of exhaust coolant (air/gas/liquid)	0.46
C6: Serviceability and Maintenance	0.09	F16: Installation & maintenance Complexity	0.45
	0	F26: Interchangeability of components	0.55
C7: Flexibility	0.08	F17: Physical Mold ability	0.24
	0	F27: Scalability	0.4
	0	F37: Upgrade ability	0.36

Table 4.8 Global Contributions

4.2.3 HDM SA

Since the elements at the last level did not come from the pairwise comparison but from the technological curve developed by Gerdsri and Kocaoglu, the contribution values on the last level are not dependent on each other. The values range from 0 to 100 and do not sum up to 1. To cater this issue, a new methodology (theorem) was developed and presented as:

Theorem Let P_{ij}^{A-S} ($-C_{ij}^{A-S} <= P_{ij}^{A-S} <=$ Max Value $- C_{ij}^{A-S}$) denote the perturbation induced on C_{ij}^{A-S}'s, which is C_{i*j*}^{A-S} (contribution of a specific Action A_{i*} to a specific strategy S_{j*}); the original ranking of A_r and A_{r+n} will not reverse if

$\lambda >= P_{ij}^{A-S} \lambda_{ij}^{A-S}$

Where $\lambda_{ij}^{A-S} = - S_{j*}$ (where P_{ij}^{A-S} is induced on r)

And $\lambda_{ij}^{A\text{-}S} = S_{j*}$ (where $P_{ij}^{A\text{-}S}$ is induced on r+n)

To avoid entering the entire data set of data every time HDM SA is to be performed, a module of the direct input is used which is the Console version included in this model. It provides flexibility to the end-user by loading all the contribution values directly via database which is an XML file. The file follows all the protocols of XML and can be modified according to the user requirements. The same file has been used twice with different combinations and values. Figure 4.9 shows the snapshot of the file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Testing>
  - <Criterion>
      <count>7</count>
    - <C1>
        <value>0.27</value>
      - <Factors>
          <count>3</count>
        - <F1>
            <value>0.34</value>
          - <Technology>
              <count>16</count>
            - <T1>
                <value>12</value>
              </T1>
            - <T2>
                <value>1</value>
              </T2>
            - <T3>
                <value>5</value>
              </T3>
            - <T4>
                <value>15</value>
              </T4>
```

Figure 4.9 XML Structure

This data represents the local contributions and the "direct input" option was used to input them into the software to perform the HDM SA.

For future use, the XML file can be created programmatically by loading data from any data source. The file is loaded easily as the console inputs can guide the end-user at each level in a sequential manner and drive the process itself.

4.2.4 Results

After executing the application and running the sensitivity analyses at different levels, the HDM Results and the HDM SA results are presented in this section.

The HDM results include the global contributions and the overall contributions calculated from the local contributions. According to the overall contributions, the technology alternatives are ranked in terms of their desire abilities in the year 2004:

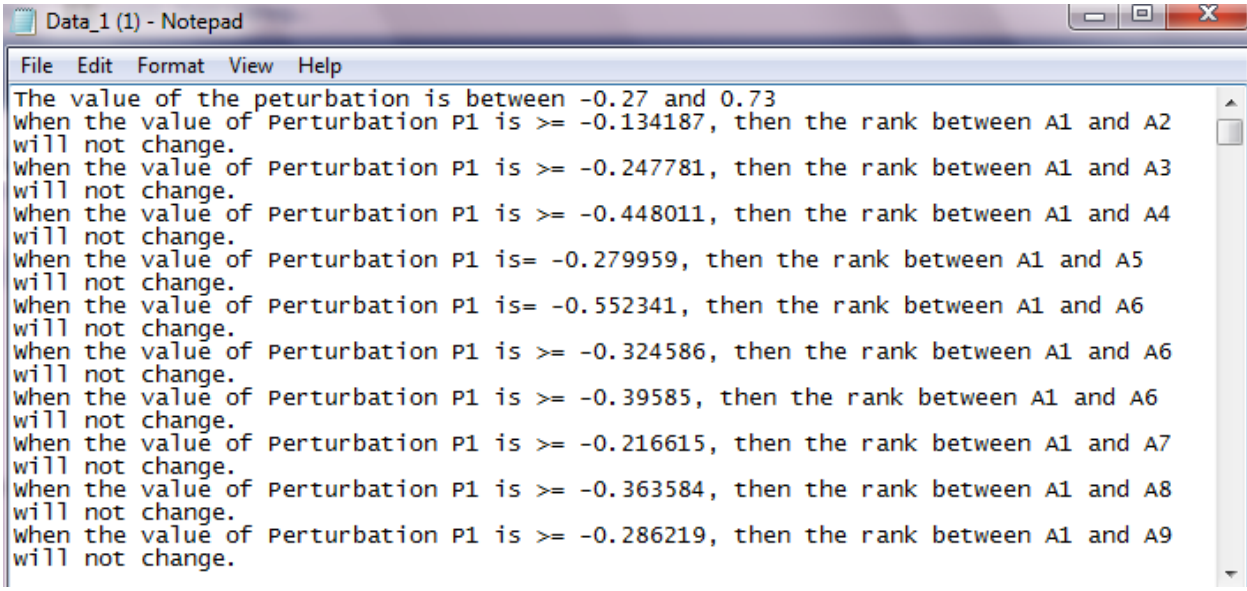
Technologies	**Overall Contributions**	**Ranks**
T6	75.08992466	1
T5	75.08992	2
T10	70.79189818	3
N4	67.46543192	4
T11	66.4163957	5
N3	65.83371717	6
N1	65.03091078	7
T9	64.40841369	8
T4	63.36440508	9
T1	61.77136237	10
T3	59.35151049	11
T8	56.30688592	12
N2	55.06328456	13
T2	49.75966181	14
T7	49.45753475	15
T12	37.56485774	16

Table 4.10 Technology Alternatives rankings in 2004

As it shows the Technology T6 (Jet-impingement and Spray Cooling) is ranked first which is the best alternative to go with taking into consideration, all the criterions and factors.

4.2.4.1 One-Way Sensitivity Analysis

First, the one-way sensitivity analysis was performed on the criteria level. A set of inequalities were generated to show the allowable range of perturbations induced on the local contributions of the criteria.



……..

```
--------------------------------------------------
The value of the peturbation is between -0.08 and 0.92
The Perturbation P is >= -0.927488
The Perturbation P is >= -0.205954
The Perturbation P is >= -0.555471
--------------------------------------------------
The value of the peturbation is between -0.08 and 0.92
The Perturbation P is >= -0.013656
The Perturbation P is >= -0.472868
--------------------------------------------------
The value of the peturbation is between -0.08 and 0.92
The Perturbation P is >= -2.76555
The allowable Range is -0.011066 and 0.008475
The Tolerance is 0.068934 and 0.088475
OPSC (Gk) is 0.008475
TSC (Gk) is 0.019541
7
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

Figure 4.10 One-Way Sensitivity Analysis (Top Level)

Figure 4.10 shows the screenshot of the text files generated as a result of the one-way sensitivity analysis at the top level. We did not show the entire table since it is too long – only the significant attributes of the results were highlighted. The first part shows the feasible region while the second part is at the end of the text file where the tolerance, OPSC and TSC are shown. Following Table 4.11 shows different ranges for the criteria's contribution values and the pair of technology alternatives whose ranking will be changed if the perturbations go beyond the range.

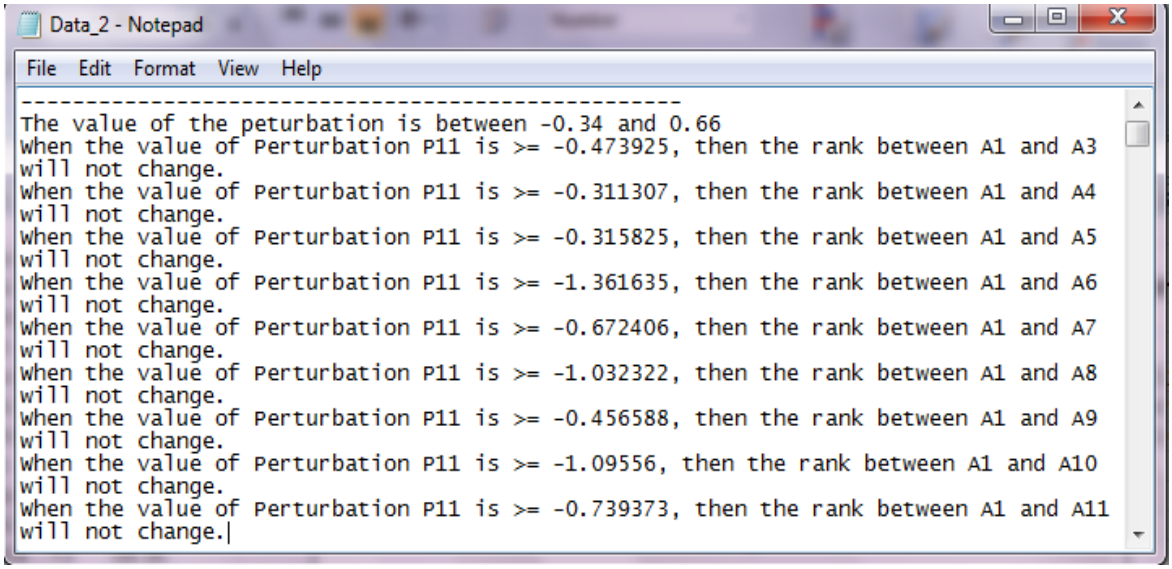
Contribution	Range	Rank Changes
C_1^C	[0,0.24264]	**(12,13)**
	[0.24265,0.274982]	**No**
	[0.274983,1]	**(13,15)**
C_2^C	[0,0.092894]	**(7,8)**
	[0.092895,0.123777]	**No**
	[0.123778,1]	**(10,11)**
C_3^C	[0,0.188897]	**(14,15)**
	[0.188898,0.231564]	**No**
	[0.2315655,1]	**(4,5)**
C_4^C	[0,0.144700]	**(10,11)**
	[0.144701,0.165696]	**No**
	[0.165697,1]	**(12,13)**
C_5^C	[0,0.077490]	**(14,15)**
	[0.077491,0.108784]	**No**

	[0.108785,1]	**(7,8)**
C_6^C	[0,0.08744]	**(9,10)**
	[0.08745,0.109762]	**No**
	[0. 109763,1]	**(4,5)**
C_7^C	[0,0.068933]	**(3,4)**
	[0. 068934,0.088974]	**No**
	[0.088975,1]	**(4,5)**

Table 4.11 Evaluation of the Top Level

As shown in the Table 4.11 where we have 7 criteria and each one has an impact on the rank alternatives. The most unstable alternative seems to be T4 (Channel Flow Boiling). It is because it is involved in most of the rank changes. Next in line are technologies T5, T10, T13 and T15 in terms of the changes made in the top most level. Other technologies like T1, T2 and T6 are very stable and are not impacted by any changes made at the top level.

Next, the one-way SA was performed on the middle level for the local contributions of the factors to criteria. Figure 4.11 shows the screenshots of the results generated by the software:



….

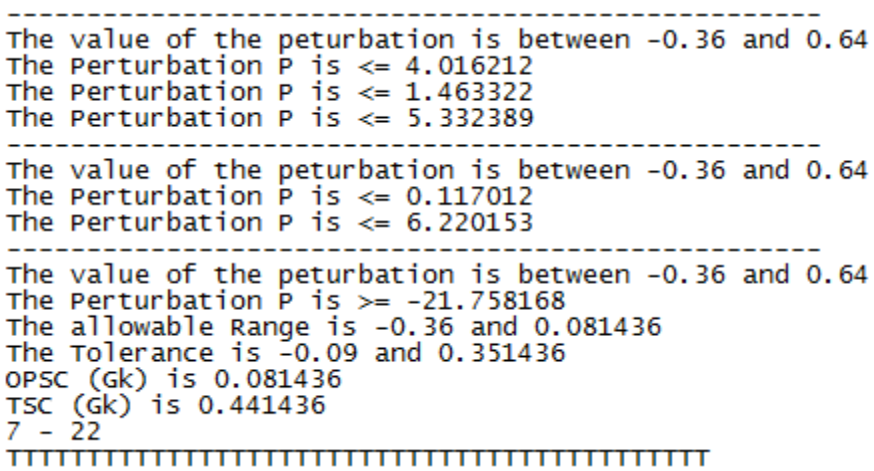


Figure 4.11 One-Way Sensitivity Analysis (Middle Level)

Figure 4.11 shows the screenshot of the text files generated as a result of the one-way sensitivity analysis at the middle level. We did not show the entire table since it is too long – only the significant attributes of the results were highlighted. The first part shows the feasible region while the second part is at the end of the text file where the tolerance, OPSC and TSC are shown. Following Table 4.12 that shows different ranges for local contribution values and the pair of technology alternatives whose ranking will be changed if the perturbations go beyond the range. (Note: only first four technologies are considered for the analysis purposes because the total number of technologies is sixteen).

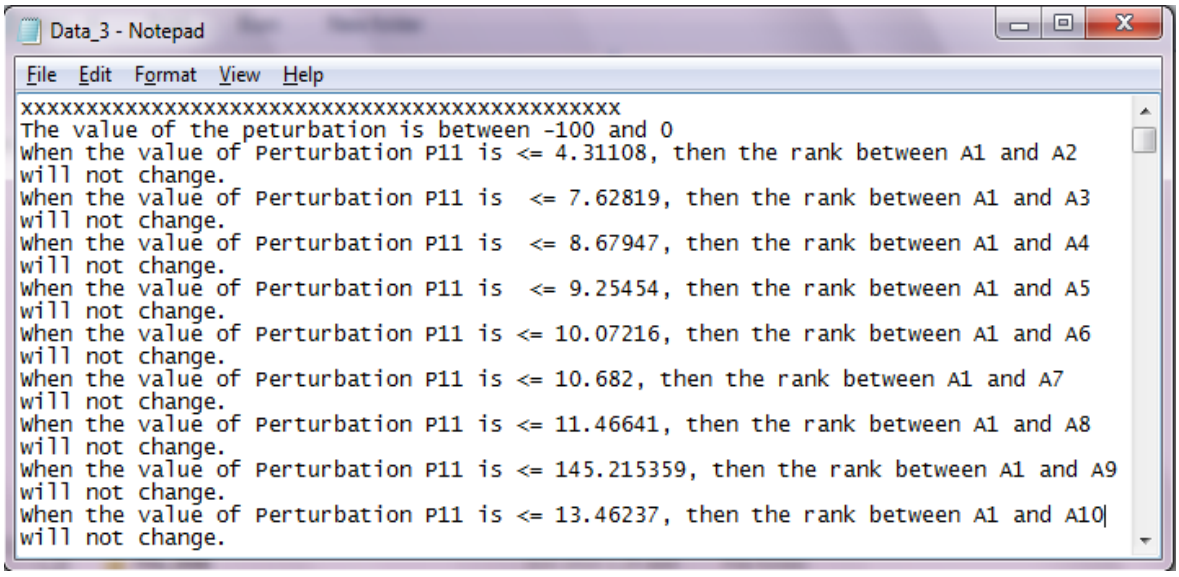
			Technologies			
Contributions	**Range**	**T_6**	**N_1**	**T_{11}**	**T_5**	**Rank Changes**
C_{11}^{F-C}	[0,0.503198]	**1**	**2**	**3**	**4**	**No**
	[0.503198,1]	1	2	4	3	**(3,4)**
C_{21}^{F-C}	[0,0.565659]	1	2	3	4	**No**
	[0.565659,0.7556]	1	2	4	3	**(3,4)**
	[0.7556,0.961216]	1	3	4	2	**(2,4)**
	[0.961216,1]	1	2	4	3	**(2,3)**
C_{31}^{F-C}	[0,0.157033]	1	2	4	3	**(3,4)**
	[0.157033,0.866542]	1	2	3	4	**No**

	[0.867,0.96106]	2	1	4	3	**(1,2)**
	[0.97,1]	2	3	4	1	**(1,3)**
C_{41}^{F-C}	[0,0.113588]	1	2	3	4	**No**
	[0.113588,0.205287]	1	2	4	3	**(3,4)**
	[0.205287,0.251732]	1	4	2	3	**(2,4)**
	[0.251732,1]	1	4	3	2	**(2,3)**
C_{51}^{F-C}	[0,0.214205]	1	2	3	4	**No**
	[0.214205,1]	1	2	4	3	**(3,4)**

Table 4.12 Evaluation of the Middle Level

By only looking at changes to the first five factor's contributions to the first criterion, as summarized in Table 4.12, we can tell that the current top choice (T6) remains stable at the top spot in all the scenarios except C_{13}^{F-C}. The results also reveal that T5 is the most unstable technologies because it is involved in most of the rank changes. The most influential factor at this level to keep T6 as the top choice is the relative impacts of strategy F11 (length of the warm up periods) to almost all the criteria.

Next, the one-way SA was performed on the bottom level for the local contributions of the technology alternatives to the factors level. Figure 4.12 shows the screenshots of the results generated by the software:



….

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
The value of the peturbation is between -32.6 and 67.4
The Perturbation P is >= -1493.959085
The Perturbation P is >= -7873.501805
The Perturbation P is >= -8225.944645
The Perturbation P is <= Infinity
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
The value of the peturbation is between -32.6 and 67.4
The Perturbation P is <= Infinity
The Perturbation P is <= Infinity
The Perturbation P is <= 21027.966306
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
The value of the peturbation is between -32.6 and 67.4
The Perturbation P is <= Infinity
The Perturbation P is <= 14648.423586
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
The value of the peturbation is between -32.6 and 67.4
The Perturbation P is <= 14295.980746
The allowable Range is -32.6 and 67.4
The Tolerance is 0 and 100
OPSC (Gk) is 32.6
TSC (Gk) is 100
22 - 16
```

Figure 4.12 One-Way Sensitivity Analysis (Bottom Level)

The allowable ranges of perturbations induced on C_{ij}^{T-F} contributions (the *i*th technology alternative to the *j*th factor) are generated for the top four technology alternative's contributions to the first three factors. In Table 4.13, the results that were obtained after executing the sensitivity analysis at the bottom level are evaluated with respect to their impact on the rank changes. (Note only four technology alternatives are considered in this scenario).

			Technologies			
Contributions	**Range**	**T_6**	**N_1**	**T_{11}**	**T_5**	**Rank Changes**
C_{11}^{T-F}	[0,100]	**1**	**2**	**3**	**4**	**No**
C_{21}^{T-F}	[0,100]	1	2	3	4	**No**
C_{31}^{T-F}	[0,45.150654]	1	2	3	4	**No**
	[45.2,100]	1	3	2	4	**(2,3)**
C_{41}^{T-F}	[0,14.739216]	1	2	3	4	**No**
	[15,49.8]	1	2	4	3	**(3,4)**
	[50,100]	1	4	2	3	**(2,3)**
C_{12}^{T-F}	[0,7.62254]	2	1	3	4	**(1,2)**
	[8,100]	**1**	**2**	**3**	**4**	**No**

C_{22}^{T-F}	[0,40.7814]	1	4	3	2	**(2,4)**
	[41,48.71908]	1	4	2	3	**(2,3)**
	[49,100]	1	2	3	4	**No**
C_{32}^{T-F}	[0,81.16232]	1	2	4	3	**(3,4)**
	[82,100]	1	2	3	4	**No**
C_{42}^{T-F}	[0,93.93768]	1	2	4	3	**(3,4)**
	[94,100]	**1**	**2**	**3**	**4**	**No**
C_{13}^{T-F}	[0,100]	1	2	3	4	**No**
C_{23}^{T-F}	[0,4.587222]	1	4	3	2	**(2,4)**
	[5,22.84389]	1	4	2	3	**(2,3)**
	[23,100]	**1**	**2**	**3**	**4**	**No**
C_{33}^{T-F}	[0,72.84333]	1	2	4	3	**(3,4)**
	[73,100]	1	2	3	4	**No**
C_{43}^{T-F}	[0,41.456667]	**1**	**2**	**3**	**4**	**No**
	[42,100]	1	2	4	3	**(3,4)**

Table 4.13 Evaluation of the Bottom Level

From the above results, we can infer that the current top choice (T6) is quite stable at the top spot. T5 is once again the most unstable technology. Based on the sensitivity coefficients, we can say that T5 is the most critical element at this level.

4.2.4.2 M Way Sensitivity Analysis

Among the competitive criteria at the top level, an M-Way Top Level Sensitivity Analysis was performed to see the impact when more than one contribution values are perturbed at the same time. Following screenshot shows the results of the M-way Top Level Sensitivity Analysis where the value of M=2, and the criteria selected were C_1 and C_2.

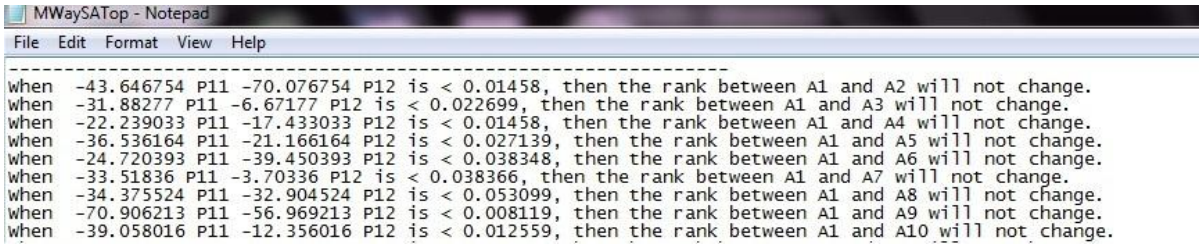


Figure 4.13 M-Way Sensitivity Analysis (Top Level)

Similarly, M-way Sensitivity Analysis can also be performed on the middle and the bottom level to evaluate the impact of the change and to see their effect on the overall contributions. Following is the result obtained after the sensitivity analysis (M-way) was performed on the middle level.

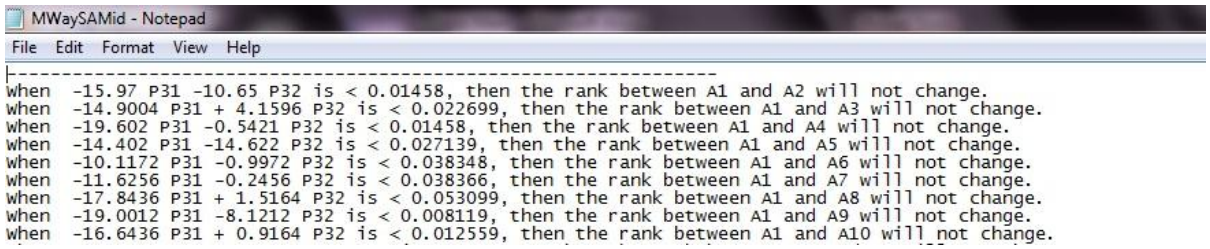


Figure 4.14 M-Way Sensitivity Analysis (Middle Level)

At this stage, we can evaluate the combination of variations that are made at the middle level and the bottom level. In the Figure 4.14, C_{31} and $C_{32,}$ i.e. the impact of T6 and T5 with respect to C3 which is reliability.

4.3 Conclusion

In this section, the software developed was used to test the sensitivity of two hierarchical decision models. For Ho's Technology Assessment Model, all the results that were generated manually on Excel by Dr. Chen while developing her dissertation were validated against the results generated by the software. A modular approach was followed during the implementation of all the theorems. Results from each module were compared with the results generated in Excel. After merging all the modules, an integration test (Test Phase of the SDLC) was performed and the final results were compared with the results in the Excel files.

For TDE, a new theorem was developed to deal with the independent contributions that are not derived from the pairwise comparison at the bottom level. Secondly, initial analysis was performed at different levels of the TDE model. It shows how the HDM SA can help in the following ways:

- How robust the model is with respect to elements at different levels

- How stable is the optimal solution under changes in different parameters
- Which one are the most influential variables with respect to the rank alternatives
- How important a particular input is to the main output
- What affects come into play when assessing assumptions on the validity of the model
- Which ones are the redundant and under-weighted variables

5 CONCLUSION

The purpose of the whole application is to fasten the process of HDM and HDM SA for decision makers. In past, similar attempts have been made to facilitate the process but with limitations in different areas. This application also has the ability to provide the user to enter the values directly, through already loaded file data that were used previously and through pair wise comparison. The application also shows the user the hierarchical model with an option to edit it if needed. All the results are tabulated and are copied into an excel file which can be further saved for future analyses while all the results of the sensitivity analyses are saved in different text files. The pair wise comparison option, offers the flexibility to the end-user by allowing multiple users to enter pair wise comparisons at each level.

The HDM SA analyses can further explore the relationships among different decision elements at different levels and can help in forecasting changes and corresponding solutions with insights for future adaptive redirection (Hongyi Chen, Jonathan C. Ho, and Dundar F. Kocaoglu, Fellow, IEE " A Strategic Technology Planning Framework: A Case of Taiwan's Semiconductor Foundry Industry"). The analysis also helps organizations that have already invested in outdated alternatives which do not favor their current organizational objectives or the strategies due to limited resources.

The model is applicable in different domains where the overall mission can be subdivided into multiple levels of hierarchy. The current application developed can furnish up to ten levels of hierarchy with a maximum of ten elements at each level. The main intent of performing the HDM SA might be different in all these models, but the general purpose remains the same.

6 FUTURE WORK

6.1 Implementation of Algorithms for SA induced simultaneously on multiple levels

Theorems that are implemented so far can only cater either single or multiple perturbations at a specific level. While there exist a possibility of having multiple perturbations that are induced simultaneously on different levels for which the theorems have already been proposed but not implemented in this application.

There can be cases when the perturbations are induced in the top level contribution vector and the second level contribution matrix while keeping the lower levels unchanged. There can be cases when the perturbations are induced on the middle levels and the levels below it while keeping the other ones unchanged.

6.2 Limitations in the application

There are certain things which will be done in the future releases of the application. They are listed as follows:

- Graphs for M-way sensitivity analysis, both 2-d and 3-d to depict the allowable regions for the perturbations on different levels
- Sensitivity Analysis for the Matrix A in the Pair Wise Comparison module. Some research was also conducted during the implementation phase of this application but due to discrepancies in the algorithm, that module is pending
- Web based interface of the implemented modules where the whole application will be hosted on the web server which would be easily accessible from any place. The interfaces for the Web portion are also completed until the HDM part but Sensitivity Analysis part still needed to be implemented
- Interactive interfaces will be implemented to allow users to load their already created excel files into XML files which can be fed into the application easily. This is usually for complex situations. A console snapshot is shown in Figure 6.1:

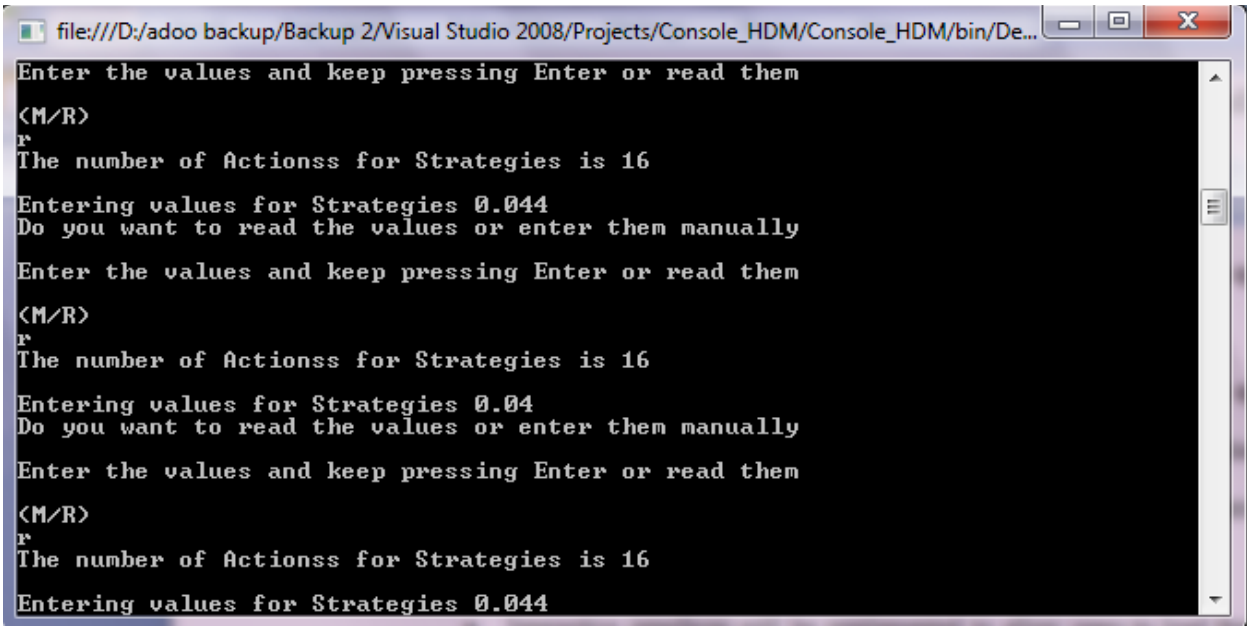


Figure 6.1 Console version of reading file option

- Currently the application is only using XML files. In future, databases can be used which will be either on Oracle or MS SQL Server
- Naming Conventions (Microsoft .Net)
- Critical Points to be checked for future hidden errors
    - All the theorems are implemented in the same manner. The architecture for all the theorems is the same, only the numbers of parameters that are passed are different and this applies to both 1-way and M-way.
    - Output module is another critical point that can throw exceptions if an unusual input is given to the software.
    - The filing module can also create issues because of the authorization rights on a particular system or a drive.
    - Form 221.cs and Form 62.cs are likely to cause an error on the interface as they have been the most dynamic forms throughout the whole application.

7 References

Armacost, R., Hosseini, J. (1994). "Identification of determinant attribute using the analytic hierarchy process." Journal of the Academy of Marketing Science.

Dantzig, G. B. (1963). "Linear Programming and Extensions."

Evans, J. R. (1984). "Sensitivity analysis in decision theory." Decision Sciences.

Ho, C. (2004). "Strategic evaluation of emerging technologies in the semiconductor foundry industry."

Hongyi Chen; Ho J.C; Kocaoglu, D. F. (2009). "Strategic Technology Planning Framework: A Case of Taiwan's Semiconductor Foundry Industry." Engineering Management, IEEE Transactions on **56**(1).

J.E. de Steiguer, J. D., Vicente Lopes (2003). "The Analytic Hierarchy Process as a means for Integrated Watershed Management."

Phillips, D. T. R., A; Solberg, J.J (1976). "Operations Research Principles and Practice."

Saaty, T. L. (1980). "The Analytic Hierarchy Process."

Winebrake, J. J., Creswick, B.P., (2003). "The future of hydrogen fueling systems for transportation." Technological Forecasting and Social Change.

Yeh, J., Kreng, B., Lin, C., (2001). "A consensus approach for synthesizing the elements of comparison matrix in the analytic hierarchy process." International Journal of System Science.