2011

Automated Support for Model Selection Using Analytic Hierarchy Process

Mario Sarkis Missakian
Nova Southeastern University, msm@symetcorp.com

Automated Support for Model Selection Using Analytic Hierarchy Process

By

Mario Sarkis Missakian

A dissertation report submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Information Systems

Graduate School of Computer and Information Sciences

Nova Southeastern University

2011

We hereby certify that this dissertation, submitted by Mario S. Missakian, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Sumitra Mukherjee, Ph.D. Date
Chairperson of Dissertation Committee

Maxine Cohen, Ph.D. Date
Dissertation Committee Member

Junping Sun, Ph.D. Date
Dissertation Committee Member

Approved:

Amon Seagull, Ph.D. Date
Dean

Graduate School of Computer and Information Sciences

Nova Southeastern University

2011

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Automated Support for Model Selection Using Analytic Hierarchy Process

by
Mario S. Missakian

February 2011

Providing automated support for model selection is a significant research challenge in model management. Organizations maintain vast growing repositories of analytical models, typically in the form of spreadsheets. Effective reuse of these models could result in significant cost savings and improvements in productivity. However, in practice, model reuse is severely limited by two main challenges: (1) lack of relevant information about the models maintained in the repository, and (2) lack of end user knowledge that prevents them from selecting appropriate models for a given problem solving task.

This study built on the existing model management literature to address these research challenges. First, this research captured the relevant meta-information about the models. Next, it identified the features based on which models are selected. Finally, it used Analytic Hierarchy Process (AHP) to select the most appropriate model for any specified problem. AHP is an established method for multi-criteria decision-making that is suitable for the model selection task. To evaluate the proposed method for automated model selection, this study developed a simulated prototype system that implemented this method and tested it in two realistic end-user model selection scenarios based on previously benchmarked test problems.

Acknowledgments

Many thanks to my dissertation advisor Dr. Sumitra Mukherjee who continuously provided the support and guidance needed to proceed with this project. Also, many thanks to my dissertation committee members, Dr. Maxine Cohen and Dr. Junping Sun, for the valuable advice and for keeping me in check throughout this effort.

Thanks to my immediate family, Missak, Betty, Carmen, Colette, Raymond, Natasha, and Tatiana, whose presence alone gives me the peace and stability needed to carry on with my projects.

Special thanks to Ruba Moghrabi who, besides being my “bestest friend”, tirelessly proofed this project on many occasions.

Table of Contents

Chapters

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Statement of the Problem to be Investigated

The Model Management (MM) field witnessed a boom throughout the eighties and up to the early nineties. This boom is thoroughly documented by Bharadwaj, Choobineh, Lo, and Shetty (1992). During this period, MM was mostly geared to technical people and highly advanced analysts, who were well seasoned and fluent in topics of management science and operations research. The invention of the personal computer and spreadsheet applications such as VisiCalc, Lotus 123, and Excel opened the decision modeling field to previously excluded non-technical personnel. The advent of the Internet facilitated and favored the sharing of models. Spreadsheet-based packaged models and tools proliferated, and a whole new industry was born.

Nowadays, a significant proportion of decision models are created by non-technical end-users or power-users using common tools like Excel spreadsheets. Organizations have invested huge sums of money in spreadsheet based models. These changes, which moved the model creation process out of the controlled environment of the Information Technology department and into the end user realm, prompted many new problems such as:

1. The void left when the creators/users of such models leave a company or move to different functional areas, taking with them their experience and insights of tweaking and using the models
2. The big costs incurred by companies to manage and maintain various versions of spreadsheet based models mainly residing on individual user computers (time consuming and error prone)
3. Faced with the difficulty of locating, understanding and comparing undocumented models, new users end up favoring the creation of their own new models, leaving behind unused existent valuable intellectual capital

In order to alleviate these problems, research in the area of end user centered model management has witnessed some activity. It prompted the search for methods which automate many of the tasks involved in model management by either 1) completely automating areas which require a high degree of specialized technical knowledge, or 2) creating software agents which provide users with a series of wizards assisting them in completing certain difficult tasks. Such automation can help in the problem areas described above. The characteristics of an automated/assisted end-user friendly model management environment should at least enable a non-technical user to:

1. Visually create, modify, and store spreadsheet models in a centralized area.
2. Allow users to keep creating their own spreadsheet models, following their current preferences, using their preferred software packages such as Excel.
3. Visually inspect and compare the internal makeup of two or more similar models (i.e. alternative reformulations).

4. Visually integrate or compose different models to form new ones.
5. Receive guidance by the model management system to evaluate and select the best model in case of the availability of multiple similar models.
6. Shield the non-technical end user from internal technical details by providing a mapping mechanism which helps in converting technical details and mapping them to easy actionable end user based decision items (for example, shield the user from the fact that the underlying solution could be based on linear programming or genetic algorithm, and rather provide more user-friendly decision parameters)

Solutions concerning items 1 through 3, as well as item 6 above have been addressed by Iyer, Shankaranarayanan, and Lenard (2005). Item 4 (model composition/integration) has been covered in past literature but not in the context of end user spreadsheet based model management environments. This item could represent viable future research work. Item 5, model selection, is one of the main tasks of model management. A good working definition of model selection is provided by Chari and Krishnan: ‘model selection leverages the existence of previously developed models to create a model for a new problem. An advantage of this approach is the ability to reuse debugged and validated models’ (2000, p.2). Although the concept of model selection has received some attention in past literature (Chari & Krishnan, 2000; Iyer, Shankaranarayanan, & Lenard, 2005; Liang & Jones, 1988; Muhanna, 1992; Steiger, 1998), there seems to be a lack of research in assisting non-technical users in the selection of end-user based models such as a spreadsheet. Also, to allow proper selection of models, they must initially be properly stored in a model management system. This dissertation aims to add value in this specific area of model selection for non-technical users (items 5 and 6 above) and also in the model storage area (item 1).

Goal of the Study

The goal of this dissertation was to provide an improved model selection method. It did so by building on the existing literature and by designing a recommender system which would be integrated into an existing spreadsheet based model management system. The described recommender system collects data from various actors such as end users, analysts or builders of a model; stores the data in a specially designed metadata model based on the Relational database model; and based on such data it presents insights to non-technical users to assist them in the task of selecting appropriate models.

More specifically, this dissertation extended the model management environment proposed by Iyer, Shankaranarayanan, and Lenard (2005) by designing, creating and integrating a model selection mechanism, and by disclosing the internal technical details of such a system. In the process, it also improved the model storage mechanism to include and highlight internal model structure information. This work also extended the research by Barkhi, Rolland, Butler, and Fan (2005) by taking the existing examples and by devising a mechanism which maps internal technical insights and presenting them in a suitable manner for non-technical personnel. Finally, the described recommender system presented end-users with insights recorded in previous usage sessions (e.g. during creation, testing, previous usage, etc.). This study showed a proof of concept using the

Analytic Hierarchy Process (AHP) as a front end analysis tool for a decision model recommender system i.e. to help match available alternative models with various model selection requirements (criteria). The rationale for choosing AHP is further elaborated in the next section.

Relevance and Significance

The overall model management field has been dormant since the mid nineties (Dolk, 2000). The end-user based (spreadsheet) model management was not addressed until the work of Iyer et al. (2005) which revived interest in this area. The available literature about spreadsheet based model management mostly addresses model creation and usage techniques and deals with models one spreadsheet at a time. Iyer et al. provided a virtual environment where spreadsheet models can be managed and used. There is still much to be done in this arena including the creation of a model selection facility.

Benefits of the Proposed Solution

The benefits of this study include: 1) an easy to use AHP end-user based front-end decision making tool based on the proposed model evaluation framework which structures the model selection environment and simplifies the task of selecting appropriate models; 2) a metadata model which stores and retrieves selection knowledge about models; and 3) a mapping mechanism which shields users from model internal (technical) information.

Contribution to the Literature in Model Management

The main inspiration for this research came from Iyer et al. (2005) in their work titled “Model management decision environment: a Web service prototype for spreadsheet models” which reignited the research about end user based decision support model management. Based on literature, this study presents a model management system which converts a simple spreadsheet into a visual counterpart, while simplifying its use i.e. understanding a model and making changes to it visually without delving into the internal technical details. The study by Iyer et al. does not concern itself with the model selection aspect, but rather, it only focuses on covering one sample model. It suggests the need for future research to support model selection. There is also a lack of research which examines how model selection can be incorporated into an existing end user based model management environment, automating the selection task or at least assisting a user, while demonstrating the internal technical mechanisms involved.

The second source of inspiration/motivation (and goal) for this study came from insights gained from the article by Barkhi et al. (2005) titled “Decision Support System induced guidance for model formulation and solution”. Although this study provides insights about the process of choosing one model over another, the insights require specialized knowledge, rendering them difficult to non-technical end users. This study too does not include an automation or assistance in the guidance of the selection process.

AHP was chosen as a front end decision aid tool for many reasons. Firstly, because as described by Saaty (1986) the three step method [i.e. i) breaking down a problem into a hierarchy of criteria/alternative, ii) allowing the comparison of similar items, and iii) assigning priorities to each level and calculating final weights] makes it easy for end-users to structure and make complex decisions. This ease is further documented by Forman and Gass (2001) who state that AHP is simpler to use/implement than other Multiple Criteria Decision Making (MCDM) methods and is suitable as a general methodology for a wide variety of situations. They continue to state ‘the prime use of the AHP is the resolution of choice problems in a multi-criteria environment… its methodology includes comparisons of objectives and alternatives in a natural, pairwise manner (Forman & Gass, 2001, p. 469) and they testify to its wide acceptance with ‘the general validity of the AHP, and the confidence placed in its ability to resolve multi-objective decision situations, is based on the many hundreds (now thousands) of diverse applications in which the AHP results were accepted and used by the cognizant DMs’ (Forman & Gass, 2001, p. 469). This prevalent use of AHP is also documented by two seminal studies (Dyer, Fishburn, Steuer, Wallenius, & Zionts, 1992; Wallenius, Dyer, Fishburn, Steuer, & Deb, 2008) which spanned a period of around 15 years. These arguments/justifications suggest that AHP is more suitable for non-technical end-user environments than other techniques since other techniques do not 1) provide a similar structuring and synthesis facility and 2) do not use a pairwise comparison method. Another reason for choosing AHP is that there is no evidence of existing research which use AHP as end user model selection method based on a spreadsheet model environment.

In summary, this study built on the existing model management literature to address the research challenges. First, it built on the work of Iyer et al. (2005) in order to capture the relevant meta-information about the models. Second, it extended the work of Barkhi et al. (2005) to identify and include the features based on which models are selected. Finally, it used Analytic Hierarchy Process (AHP) to structure the model selection process and to select the most appropriate model for any specified problem. AHP is an established method for multi-criteria decision making that is suitable for the model selection task. To evaluate the proposed method for automated model selection the study simulated a prototype system that implements the method and tests it on previously used benchmark test problems.

Barriers and Issues

As documented by Bharadwaj et al. (1992), the model management discipline witnessed a lot of activity in the second part of the 20th century. Dolk (2000) states that the overall model management field became dormant starting in the mid 1990s and attributes this state to the theoretical difficulty of the topic and to the rush of researchers to more pressing and higher visibility internet-related issues. Starting in the early 1980s, spreadsheet based decision modeling experienced democratization with the prevalence of personal computers. As a result, hundreds of millions of spreadsheets were created by tens of millions of professionals (Panko, 1999). Although currently there is an abundance of studies covering spreadsheet related issues, these mostly deal with best practices in the creation of a single spreadsheet model at a time and do not address model management or reuse issues. Iyer et al. (2005) sparked the revival of spreadsheet model management topic but did not address the issue of model selection. This research study focused on this specific issue.

Chapter 2

Review of the Literature

Introduction

This chapter presents a literature review relevant to the research and is organized as follows: 1) general decision model management, 2) spreadsheet based modeling, 3) Analytical Hierarchy Process (AHP) based literature, and 4) model selection specific review.

General Decision Model Management

During the eighties and up to early nineties, the model management movement experienced very heavy research activity. This is documented in the survey conducted by Bharadwaj et al. (1992) However, the research activity did not progress much into the end user based DSS environments. Later, Dolk (2000) characterized the state of the research in this area as dormant. Dolk provides many reasons for this halt such as 1) lack of demand, 2) huge software development effort, 3) theoretical difficulties and 4) the emergence of the Internet. The last reason was cited by many other researchers as the breaking point that exacerbated this research area and all research seems to have moved towards internet related topics. Many other fields of research witnessed the exodus of research toward internet based topics.

Bharadwaj et al. (1992) provide a rather comprehensive survey of the model management field. They categorize the model management research topic into five areas:

1) algebraic modeling languages such as GAMS and AMPLE; 2) database oriented; 3) graph-based; 4) knowledge-based which is further subdivided into a) semantic nets and frame, b) first order predicate calculus, c) rules; and 5) specialized systems.

In the graph based area, Geoffrion (1987, 1989, 1991, 1992) is a very influential contributor. His work was adopted by many researchers and is used as the basis for their work. Also, the work of Jones (1990, 1991, 1992; Jones & Schocken, 1993) is the basis for many graphical oriented modeling efforts. Basu and Blanning (1995, 1997, 1998) present an alternative graphical centered approach based on Metagraphs.

In the data management oriented area, Dolk (1986) suggests data as a model approach. Lenard (1986) provides solutions based on the relational database model. Bhargava, Krishnan, and Mukherjee (1992) provide an insightful combination of data oriented and mathematical model.

In the knowledge based area, Liang (1988a, 1988b; Liang & Jones, 1988) presents very comprehensive frameworks. Muhanna (1992) also provides a comprehensive model management framework which is based on systems theory. This work is fully expanded in earlier studies of the same author (Muhanna, 1987, 1990; Muhanna & Pick, 1988). It was also was based on an earlier working paper which was also published at a later date as Muhanna and Pick (1994).

Gagliardi and Spera (1995) provide a formal theory about model integration which is motivated by three sources: 1) increasing productivity, 2) reducing errors, and 3) saving time and money. This theory requires the same constraints as stipulated by Geoffrion’s (1987, 1989, 1991, 1992) structured modeling paradigm.

Research in model management has seen some sporadic activity in the various areas since the Bharadwaj et al. (1992) survey. For example, Chari and co-authors kept some activity lately (Chari, 2002, 2003; Chari & Krishnan, 2000; Chari & Sen, 1998). Their activity is mostly in the knowledge based modeling systems.

This section covered general decision model management topics. The following section focuses the review on spreadsheet based modeling topics.

Spreadsheet Based Modeling

This section addresses two types of literature which are relevant to the research: 1) spreadsheet modeling and 2) solvers.

Spreadsheet Modeling

Nowadays, organizations are littered with spreadsheet based decision models created by end-users. Panko (1999) confirm this fact by stating, 'tens of millions of managers and professionals around the world create hundreds of millions of spreadsheets each year' (p.159). Even with this over-abundance of models, end users end up recreating models that already exist because 1) they have no way of locating appropriate models hidden in personal computers and 2) once located, it is hard for them to really understand the logic behind these models (Iyer et al., 2005). Users end up creating their own models from scratch, instead of taking advantage of existing models. The creation of these models requires huge amounts of time and effort, and therefore incurs costs (Panko, 1998, 1999, 2006). Ronen, Palley, and Henry (1989) also corroborate on the errors/cost issue and suggest the usage of structured analysis and design modeling technique as a foundation for spreadsheet modeling.

The most relevant study to this dissertation is by Iyer et al. (2005). It provides an end user based model management framework, which includes a spreadsheet-based working example. It describes a model management paradigm which covers all phases of modeling and using a Decision Support System. Within this paradigm, an important phase is the Model Content Management (MCM), which covers the actual techniques used for the creation and modification of models. Iyer et al. illustrate a technique of MCM which combines three different areas of research: 1) spreadsheet based user oriented modeling; 2) Structured Model Language (Geoffrion, 1987, 1989, 1991, 1992); and 3) graphical oriented representations based on Attributed Graph-Grammar (Jones, 1990, 1991, 1992).

Iyer et al. (2005) borrow a spreadsheet model example from existing literature (Isakowitz, Schocken, & Lucas, 1995) and provide a description of the steps involved in maintaining it. They first suggest converting a spreadsheet into its model schema in a "*factoring-like process*". An extended version of the Structured Modeling Language (SML) (Geoffrion, 1987, 1989, 1991, 1992) is then used to document the schema of the model. And finally the schema is represented in a format using a Generic Structure Diagram. Figure 1 shows these model conversion steps.

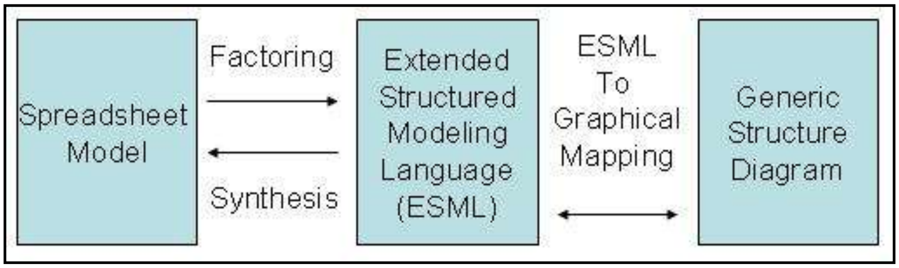
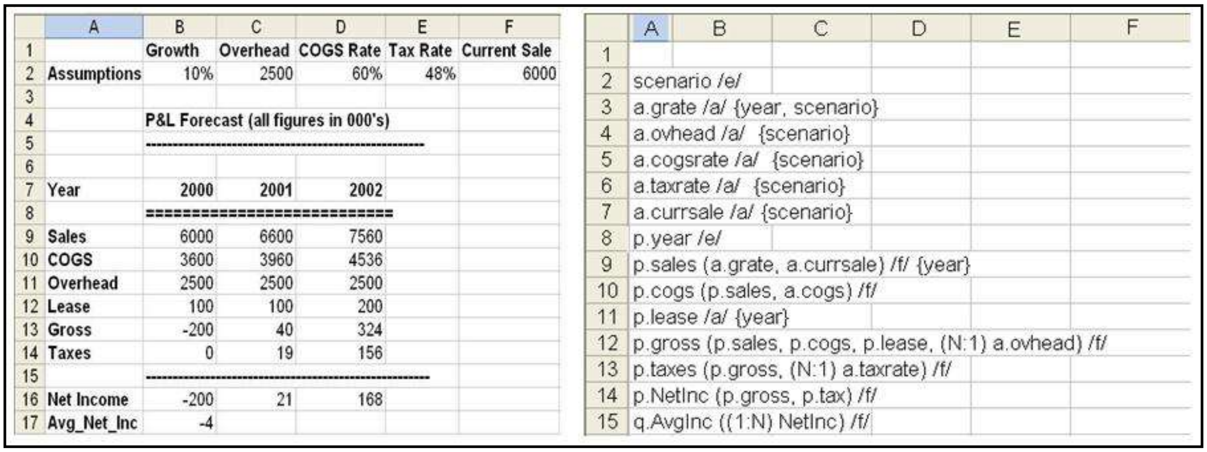

Spreadsheet Model
Factoring
Synthesis
Extended Structured Modeling Language (ESML)
ESML To Graphical Mapping
Generic Structure Diagram

Figure 1. The Model Conversion Process as Described in Iyer et al. (2005)

The extended version of SML is called the Extended Structured Modeling Language (ESML), which proves to be more suitable for visually rendering a model in its graphical representation. Figure 2 shows a sample of the textual version of ESML.



	A	B	C	D	E	F
1		Growth	Overhead	COGS Rate	Tax Rate	Current Sale
2	Assumptions	10%	2500	60%	48%	6000
3						
4		P&L Forecast (all figures in 000's)				
5		--				
6						
7	Year	2000	2001	2002		
8		==========================				
9	Sales	6000	6600	7560		
10	COGS	3600	3960	4536		
11	Overhead	2500	2500	2500		
12	Lease	100	100	200		
13	Gross	-200	40	324		
14	Taxes	0	19	156		
15		---				
16	Net Income	-200	21	168		
17	Avg_Net_Inc	-4				

	A	B	C	D	E	F
1						
2	scenario /e/					
3	a.grate /a/ {year, scenario}					
4	a.ovhead /a/ {scenario}					
5	a.cogsrate /a/ {scenario}					
6	a.taxrate /a/ {scenario}					
7	a.currsale /a/ {scenario}					
8	p.year /e/					
9	p.sales (a.grate, a.currsale) /f/ {year}					
10	p.cogs (p.sales, a.cogs) /f/					
11	p.lease /a/ {year}					
12	p.gross (p.sales, p.cogs, p.lease, (N:1) a.ovhead) /f/					
13	p.taxes (p.gross, (N:1) a.taxrate) /f/					
14	p.NetInc (p.gross, p.tax) /f/					
15	q.AvgInc ((1:N) NetInc) /f/					

Figure 2. Spreadsheet Model (left) and Corresponding Text-Based ESML Schema (right)

The schema model notated in ESML is then mapped into its graphical representation using the Generic Structure Diagram based on Jones (1990, 1991, 1992; Jones & Schocken, 1993). Figure 3 shows a graphical version of the ESML schema.

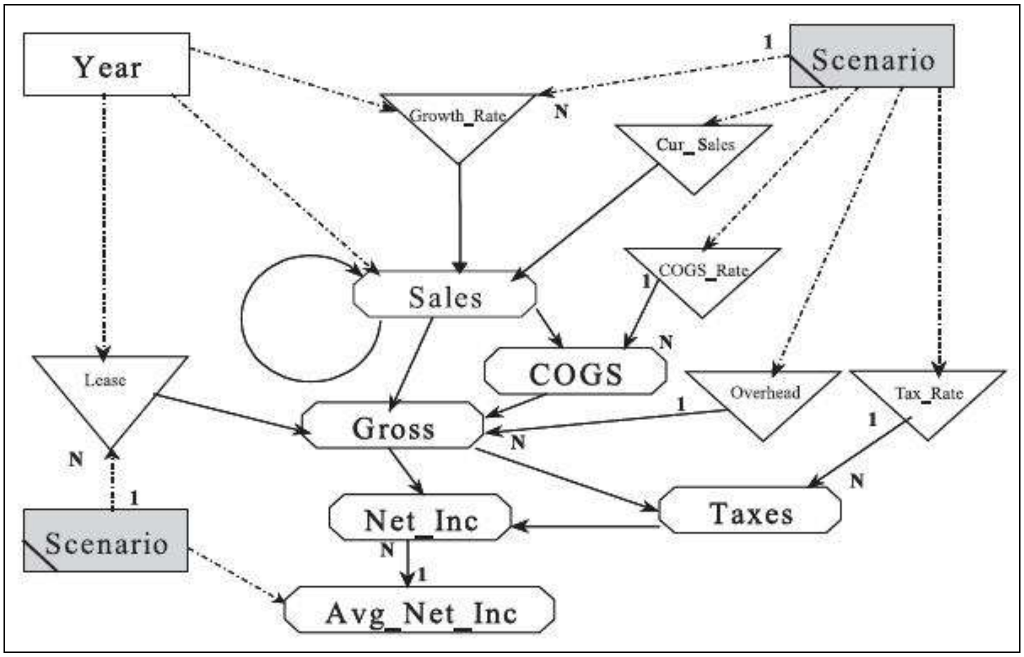
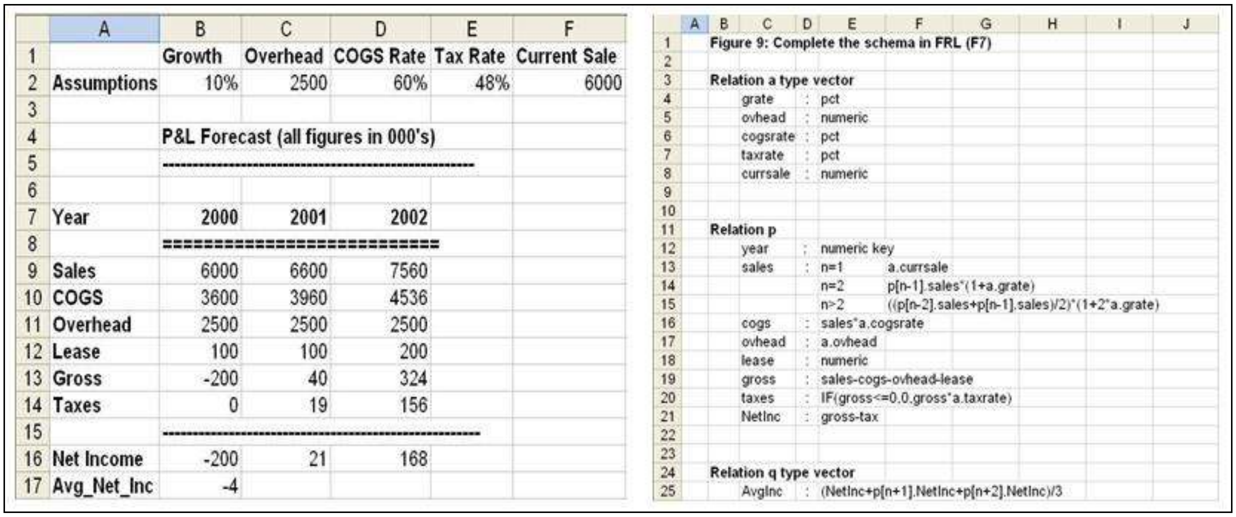


Figure 3. The Graphical Version of the ESML Schema

Isakowitz et al. (1995) conceived the method by which a spreadsheet model can be converted from its visual (physical) format into its logical schema (*factorization*), using a notation called Functional / Relational Language (FRL). They also provide the reverse method, which re-creates a spreadsheet starting with an FRL schema (*synthesis*). Figure 4 shows a spreadsheet model and its corresponding FRL. Appendix A explains this conversion process in a more detailed and step-by-step manner.



	A	B	C	D	E	F
1		Growth	Overhead	COGS Rate	Tax Rate	Current Sale
2	Assumptions	10%	2500	60%	48%	6000
3						
4		P&L Forecast (all figures in 000's)				
5		--				
6						
7	Year	2000	2001	2002		
8		==========================				
9	Sales	6000	6600	7560		
10	COGS	3600	3960	4536		
11	Overhead	2500	2500	2500		
12	Lease	100	100	200		
13	Gross	-200	40	324		
14	Taxes	0	19	156		
15		--				
16	Net Income	-200	21	168		
17	Avg_Net_Inc	-4				

	A	B	C	D	E	F	G	H	I	J
1		Figure 9: Complete the schema in FRL (F7)								
2										
3		Relation a type vector								
4			grate	:	pct					
5			ovhead	:	numeric					
6			cogsrate	:	pct					
7			taxrate	:	pct					
8			currsale	:	numeric					
9										
10										
11		Relation p								
12			year	:	numeric key					
13			sales	:	n=1	a.currsale				
14					n=2	p[n-1].sales*(1+a.grate)				
15					n>2	((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)				
16			cogs	:	sales*a.cogsrate					
17			ovhead	:	a.ovhead					
18			lease	:	numeric					
19			gross	:	sales-cogs-ovhead-lease					
20			taxes	:	IF(gross<=0,0,gross*a.taxrate)					
21			NetInc	:	gross-tax					
22										
23										
24		Relation q type vector								
25			AvgInc	:	(NetInc+p[n+1].NetInc+p[n+2].NetInc)/3					

Figure 4. Spreadsheet Model (left) and Corresponding FRL Schema (right)

The benefit of such a spreadsheet to FRL conversion is that an FRL can be easily manipulated programmatically to instantiate different cases, whereas a spreadsheet model needs to be manually changed by a person. For example, in order to expand the spreadsheet model shown in Figure 4 to include more years (i.e. beyond the year 2002), a person needs to manually create a column for each of the additional years. With the FRL schema format, the model management system executing the model can prompt the user for a range of years, re-create the appropriate model, and then execute it. A feature such as this can by itself save a lot of time and money by promoting reuse of existing models. One of the main reasons model builders are attracted to spreadsheet modeling systems is that these environments give them complete flexibility in model creation, due in large part to their visual orientation. However, this same aspect makes it very hard to programmatically control spreadsheet models since they contain many formatting information (“*editorial information”* according to Isakowitz (1995)) that are not relevant to the actual execution of a model. In order to be able to, for example, automate model comparison or evaluation tasks, many of the visual aspects of a spreadsheet model need to be neutralized, thus allowing clear access to the underlying logic of the spreadsheet.

Solvers

Often, solving problems require the use of some optimization algorithm. Spreadsheet development environments package solution engines and refer to them as *solver*s. Solvers implement various solution search techniques and make them accessible to end users through simplified user interfaces. Examples of traditional solution search algorithms include linear and integer programming. Genetic algorithms (GA) and simulated annealing (SA) are examples of intelligent search techniques. Each type of solver/algorithm is appropriate for certain classes of problems as reminded by Fazlollahi and Vahidov (2001), ‘GA can deal with problems that incorporate nonlinearity, discontinuity, uncertainty, complexity, and other demanding features. These features make the application of traditional search and optimization methods inappropriate.’ (p. 232)

The inner mechanisms of solvers and their underlying search algorithms are beyond the scope of this study. However, the choice of an appropriate solver is of great importance to this research since, as shown by Barkhi et al. (2005), such a choice (i.e. the matching of problem characteristics/formulation with appropriate search techniques) can have a drastic impact on the quality of the solution as well as on the time required to find a solution. For example, Barkhi et al. show that combining different formulations of a same problem with different solvers can sometimes prevent finding solutions:

"GA could not solve the problems formulated with linear objective function and constraints shown in strong and weak formulation while the traditional search techniques find the solutions in reasonable time." (Barkhi et al., 2005, p. 276)

Thus, a decision maker should be careful in selecting a *solver* appropriate to the manner in which the problem at hand is formulated. Zigurs and Buckland (1998) in their task/technology fit theory, explain the process of matching technology with specific types of problems. The theory elaborates on the different types of tasks (simple, problem, decision, judgment, and fuzzy), the types of support a technology offers (communication support, process structuring, and information processing), and matches task requirements with the support provided by a technology. Even though the theory is mostly concerned with team decision making, it provides significant insights and could be beneficial to individual decision support situations. An automated model selection system should provide facilities which help the decision maker in structuring a problem in a manner which facilitates fitting the task (problem formulation) with technology (solvers and underlying algorithms).

In summary, this section discussed spreadsheet based model management and solver related concepts. The following section covers literature on decision making based on the Analytical Hierarchy Process (AHP).

Analytical Hierarchy Process (AHP) Based Literature

Over time, AHP has proven to be a very versatile decision making method. Saaty (1990) is the originator of the AHP method. Forman and Gass (2001) provide a good primer on AHP along with a wide range of projects where AHP was successfully implemented. Partovi, Burton, and Banerjee (1990) detail the application of AHP on various problems. For example, Stannard, Zahir, and Rosenbloom (2006) and Lee and Hsu (2004) show examples of AHP in capacity planning related problems. There are many applications of AHP in various fields and discipline such as in Customer Relationship Management (Barbarosoglu & Yazgac, 2000; Colombo & Francalanci, 2004), manufacturing (Singh, Choudhury, Tiwari, & Maull, 2006), evaluation of financial statements (Uzoka, 2005), task assignment to suppliers (Yuan-Jye & Yu-Hua, 2005), Information Technology decisions (Sarkis & Sundarraj, 2001), software development (Ruhe, Eberlein, & Pfahl, 2003), and managing creativity in advertising (Davies, 2000). Foulds and Partovi (1998) show the application of AHP to the facilities layout problem. Forgionne (1999) and later Phillips-Wren, Mora, Forgionne, and Gupta (2009) show the use of AHP to determine the overall effectiveness of a DSS. There is no evidence of using AHP in model management studies aimed at helping users select an appropriate decision model.

Model Selection Specific Review

The topic of model comparison and selection in DSS has not witnessed a high level of activity. Model selection is mostly mentioned as an overview or as a starting point in more general studies about DSS. Following is a list of such representative research and the selection criteria concepts included.

Concerning model selection, Liang and Jones (1988) list five evaluation criteria and specify that these could be used as a starting point on the subject: 1) accuracy of the model; 2) measure of user preferences e.g. trust in the model and the credibility of the previous users; 3) distance from goal in number of stages i.e. the number of steps required to complete the decision; 4) the number of components involved; and 5) total cost of the model.

Iyer et al. (2005) classify knowledge about modeling into five categories: workflow, evaluative, operational, content, and process knowledge. Evaluative knowledge is one which contains information about a model's overall value and any metrics associated with it. This type of knowledge provides responses to questions posed by analysts and decision-makers on issues such as the reliability, robustness, and usefulness of the model in decision-tasks.

In the area of evaluating an entire DSS (not just models within it), Phillips-Wren et al. (2009) provide a framework which breaks down the process into four levels where each level shows the process from the perspective (worldview) of different stakeholders (organization, user, designer, and builder). These four levels are as follows: 1) decision-making level (organization and user) considers the impact of using the DSS on the process of decision-making and on the outcome of a decision; 2) decisional service-task level (user and designer) focuses on the support of analysis and synthesis services; 3) architectural-capability level (user, designer and builder) examines the user interface, the data and knowledge component, and processing; and 4) computational/program/symbol level (designer and builder) elaborates on the impact of the specific AI algorithm used in the DSS. This evaluation method uses AHP in breaking down the components and solving the overall DSS effectiveness, and is built upon the previous work one of the authors (Forgionne, 1999) where further detail and examples of using AHP are shown.

Muhanna (1992) advocates the use of 1) forward reasoning or 2) a backward reasoning search mechanisms where the former method eliminates models based on the model data input required by a model and provided by the user, and the latter method eliminates models based on the user's required output.

Steiger (1998) proposes the generation and usage of *evaluative arguments* 'which may take the form of comparing the advantages and disadvantages of two competing models, to determine which model … is the better model with respect to accuracy, simplicity, conceptual validity' (p. 207). Other aspects include model 1) sufficiency i.e. whether or not the model is sufficient by itself to represent the problem domain; 2) necessity i.e. whether or not all the model's components are necessary to solve the problem; and 3) consistency deals with whether or not a model's components are all consistent, for example in the usage of units of measurements.

Chari and Krishnan (2000) view model selection from three different perspectives: 1) *organizational issues* where the focus is on organizing existing models in a manner which makes it easy for users to spot the similarities/differences between models and to communicate them; 2) *representation issues* which is concerned with two different areas i) the features of the models themselves for which users select them i.e. rationale of the model, assumption, performance, robustness, difficulty of solvability and ii) the methods or standards to be followed when dealing with a model library representation i.e.

modeling adaptiveness to the modeler usage patterns or domain; and 3) *processing issues* which focus on three aspects i) the types of operations needed by the user in order to find candidate models such as browsing existing models or providing search mechanisms ii) the expressivity of the querying language which helps users in identifying candidate models, and iii) the computational complexity of the previous two items. Table 1 compiles and summarizes some of the variables just discussed. This list of variables serves as criteria in the selection of a model.

Table 1. List of Criteria Based on Literature Review

Criteria Variable	Source Literature
Accuracy of model	Liang and Jones (1988)
Computational complexity	Chari and Krishnan (2000)
Trust in model	Liang and Jones (1988)
Sufficiency	Steiger (1998)
Input/Output needs	Muhanna (1992)
Distance from goal	Liang and Jones (1988)
Performance/ difficulty of solvability	Chari and Krishnan (2000)
Number of components	Liang and Jones (1988)
Cost of model	Liang and Jones (1988)
Robustness of model	Chari and Krishnan (2000); Iyer et al. (2005)
Reliability of model	Iyer et al. (2005)
Architecture/structure of model	Phillips-Wren et al. (2009)
Simplicity of model structure	Steiger (1998)
Availability of designer comments	Phillips-Wren et al. (2009)

Summary

This chapter presented a literature review relevant to this research and it organized the material into four different areas: 1) general model management literature, 2) spreadsheet modeling specific literature review, 3) AHP based literature, and 4) model selection specific literature review. The following chapter discusses the methodology to be employed for the research.

Chapter 3

Methodology

Introduction

This chapter starts by covering the guidelines and characteristics required for *design-science research*, and establishes conformance of the employed methodology to them. It then discusses the methodology employed for the research, which includes the design objectives, the steps involved in accomplishing the design objectives and the validation process.

Characteristics of the Research Methodology

van Aken (2004, 2005) makes the distinction between *description-driven* sciences which attempt to explain and describe a problem domain, and *prescription-driven* sciences which attempt to find methods which help guide the finding of solutions for a problem domain. van Aken calls the former *explanatory sciences* and the latter *design sciences*. She states 'the mission of a design science is to develop knowledge for the design and realization of artifacts … or to be used in the improvement of the performance of existing entities' (van Aken, 2004, p.224).

Similarly, Hevner et al. (2004) characterize research in information systems as being based on two distinct paradigms each having its roots in different disciplines: 1) behavioral science which is influenced by natural science methods and 2) design science which 'has its roots in engineering and the sciences of the artificial' (p. 76). They state that design science is a problem solving process which results in utility to the research/practitioner community, whereas the behavioral science research is mostly concerned with truth finding. The study goes on to provide a framework which shows the role of each type of research. It also lists seven guidelines to which design science research should comply. The following passage by Hevner et al. (2004) summarizes these guidelines:

> ‘Design-science research requires the creation of an innovative, purposeful artifact (Guideline 1) for a specified problem domain (Guideline 2). Because the artifact is purposeful, it must yield utility for the specified problem. Hence, thorough evaluation of the artifact is crucial (Guideline 3). Novelty is similarly crucial since the artifact must be innovative, solving a heretofore unsolved problem or solving a known problem in a more effective or efficient manner (Guideline 4). In this way, design-science research is differentiated from the practice of design. The artifact itself must be rigorously defined, formally represented, coherent, and internally consistent (Guideline 5). The process by which it is created, and often the artifact itself, incorporates or enables a search process whereby a problem space is constructed and a mechanism posed or enacted to find an effective solution (Guideline 6). Finally, the results of the design-science research must be communicated effectively (Guideline 7) both to a technical audience (researchers who will extend them and practitioners who will implement them) and to a managerial audience (researchers who will study them in context and practitioners who will decide if they should be implemented within their organizations)’ (p. 82)

This study prescribes a method to facilitate solving the model selection problem and therefore falls in the design-science research domain. The research method is therefore guided by and complies with Hevner’s (2004) seven guidelines. As the first guideline stipulates, the artifacts of this study are *innovative* (i.e. has not been created before) and *purposeful* (i.e. yields utility to the user of the model selection process). The study is specific for the model management domain (Guideline 2). The artifacts were evaluated using the two different end-user model selection scenarios (Guideline 3). The study applied AHP to the model selection process in a way that has not been done before (Guideline 4). The solution was clearly documented using well established software engineering techniques such as Relational Database Models in order to document the internal storage model. The filtering of models based on specified criteria was carried out using formal language such as SQL (Guideline 5). The model criteria comparison and selection which was based on AHP has been widely used and validated in countless multi-criteria decision making problems (Guideline 6). And finally, the result of this study is valuable for academic researchers who would be interested in the method used as well as to the practitioner community who would be interested in applying it to their specialization domains.

Design Objectives

The goal of this study is to provide an improved model selection method for spreadsheet-based models. As such, the recommended model selection method complied with a specific set of design objectives, as follows:

1. Easy-to-use: Its features are accessible through an easy-to-use graphical user interface which do not require the user to memorize data and commands
2. Sufficient Information: It presents users with all information necessary to make informed decisions about models
3. Model Inspection: It allows users to inspect the internal makeup of models as well as provide external information such as historical execution information (time to complete, quality of results, etc.)

4. Adaptable: It adapts to the sophistication level of users i.e. it shields users from (or allows access to) model internal technical details depending on their (users') level of sophistication
5. Model Comparison: It facilitates the comparison of multiple models without requiring the memorization of information about each model
6. Model Ranking: It finally ranks the available models based on feedback from users

The design of a model selection method would largely depend on the way existing models were initially stored. Therefore, prior to the model selection process, the following design requirements were also added to a spreadsheet model selection system:

7. Model Creation: Allow users to keep creating their own spreadsheet models by following their current modeling preferences, while using their preferred software packages such as Microsoft Excel or competing products.
8. Model Modification: Allow users to modify and store spreadsheet models visually in a centralized area where other users can have access and can re-use them.

The following section presents the steps followed in order to meet these design objectives.

Solution Approach

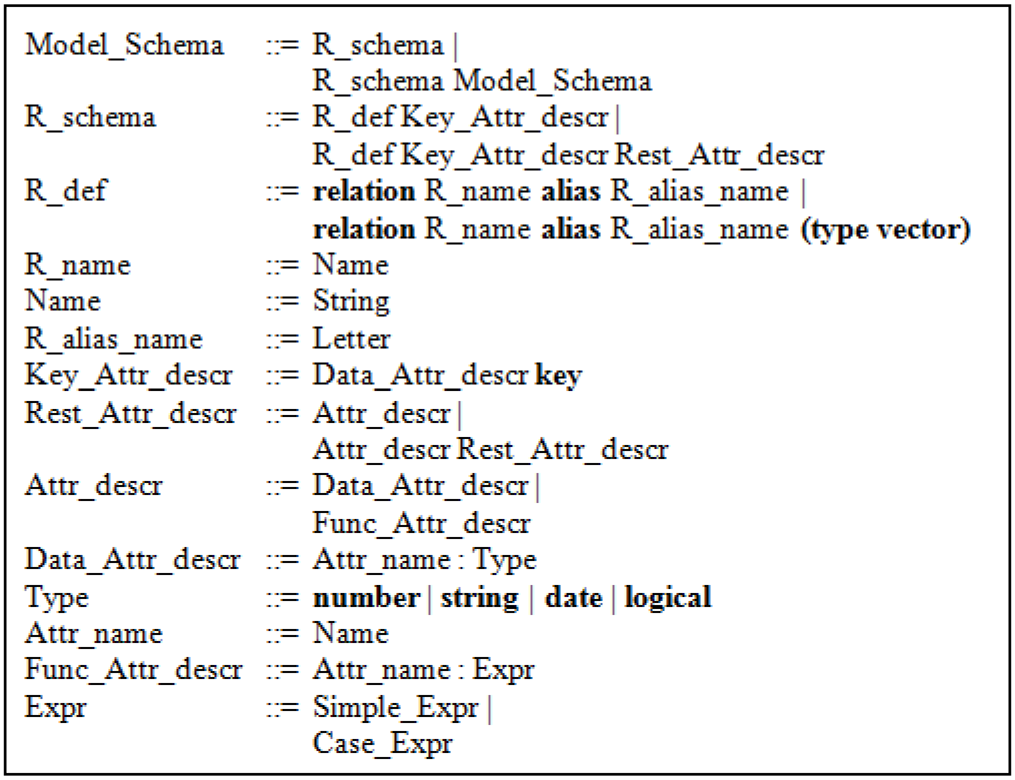
This section shows the detailed sequence of events which were followed and the corresponding output in order to achieve the goals of this dissertation. These research goals were achieved by providing a prototype system which adheres to the design objectives. In order to meet the stated design objectives, the main tasks of designing the model selection method included (1) selecting an appropriate way to represent and store spreadsheet models in a format which facilitates automated inter-model comparisons; (2) identifying the appropriate metadata about the spreadsheet models that need to be maintained; and (3) developing a suitable spreadsheet model selection strategy. The following section describes the mechanisms and artifacts for each phase

Phase I – Selecting an Appropriate Way to Represent and Store Spreadsheet Models

Model Internal Structure: As described in the literature review chapter, there are two schema notation methods (FRL or ESML) either of which would be suitable for the purposes of this study. However, since this study is not concerned with the graphical aspects of models, the Functional/Relational Language (FRL) was adopted for its simplicity. Figure 4 shows an example of a spreadsheet model and its counterpart in FRL. FRL represents a model as a series of *relations* very similar to a relational database model except that FRL incorporates relationships and dependencies among attributes. This addition is very important since most spreadsheets are modeled by having the values of cells depend on values of other cells. Appendix B shows a full description of FRL in terms of Backus-Naur Form (BNF).

In essence, an FRL model consists of two types of *relations* which store data about a model: 1) vector relations which hold one tuple only, and 2) non-vector (no name is given in the original literature) which holds data in tabular format with various columns and only one index column.

Figure 5 shows the top section of BNF based FRL as described by Isakowitz et al (1995). This section is included for reference and changes to it are discussed next. The remainder of this FRL can be examined in Appendix B.

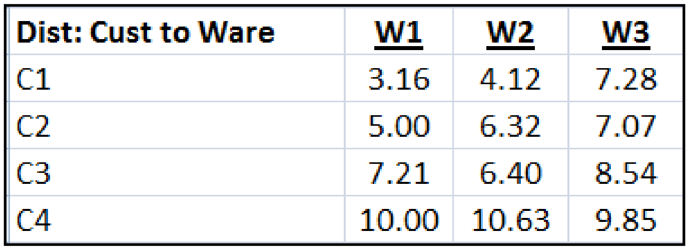


```
Model_Schema     ::= R_schema |
                     R_schema Model_Schema
R_schema         ::= R_def Key_Attr_descr |
                     R_def Key_Attr_descr Rest_Attr_descr
R_def            ::= relation R_name alias R_alias_name |
                     relation R_name alias R_alias_name (type vector)
R_name           ::= Name
Name             ::= String
R_alias_name     ::= Letter
Key_Attr_descr   ::= Data_Attr_descr key
Rest_Attr_descr  ::= Attr_descr |
                     Attr_descr Rest_Attr_descr
Attr_descr       ::= Data_Attr_descr |
                     Func_Attr_descr
Data_Attr_descr  ::= Attr_name : Type
Type             ::= number | string | date | logical
Attr_name        ::= Name
Func_Attr_descr  ::= Attr_name : Expr
Expr             ::= Simple_Expr |
                     Case_Expr
```

Figure 5. Top Portion of the FRL in BNF Format

This BNF based FRL indicates that a model schema can be made up of two distinct types of *relations* (R_def), a standard relation (no name provided) and a vector relation (type vector). A *relation* is identified by a name (R_name) and optionally an alias (R_alias_name). Each *relation* can have two types of *attributes*: 1) key (Key_Attr_descr), and 2) non-key (Attr_descr). This latter can be either data (numeric, string, date, or logical) or function (simple or case).

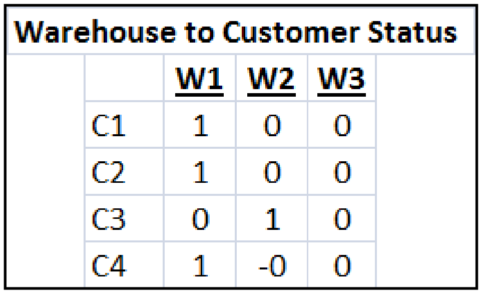
These constructs are adequate for the purposes of the original example by Isakowitz et al (1995), however, more specialized constructs are necessary for more complex examples, such as when representing a two dimensional matrix indexed at both the column and row levels. Figure 6 shows an example of such a matrix where on the vertical line the table is indexed by Customers, on the horizontal line it is indexed by Warehouses, and the intersection represents the distance between them.



Dist: Cust to Ware	W1	W2	W3
C1	3.16	4.12	7.28
C2	5.00	6.32	7.07
C3	7.21	6.40	8.54
C4	10.00	10.63	9.85

Figure 6. Two Dimensional Matrix of Customers and Warehouses

When representing a binary integer problem in spreadsheets, a duplicate matrix with binary entries is required. Since a binary integer optimization example is used in the prototype of this study, FRL needed to be expanded to accommodate them. Figure 7 shows a duplicate of the matrix shown in Figure 6 except that it includes binary typed intersection cells.



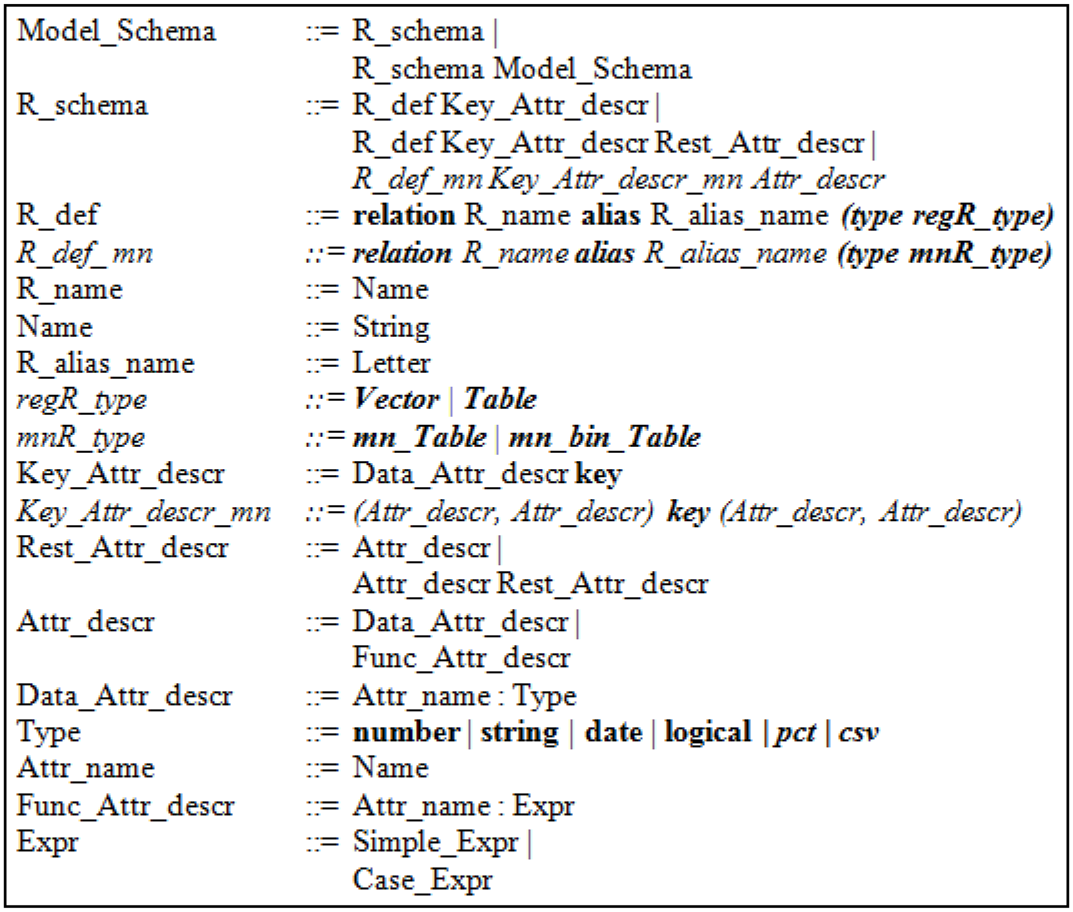
Warehouse to Customer Status

	W1	W2	W3
C1	1	0	0
C2	1	0	0
C3	0	1	0
C4	1	-0	0

Figure 7. Matrix With Binary Cells

Figure 8 shows the expanded FRL schema. In the original FRL, there are two types of relations: Vector and non-vector (the latter was not named meaning that a relation that has no Vector keyword would be considered a non-vector relation. For this dissertation, the FRL definition would be expanded to include four types of relations: 1) *vector* which is a relation that contains only one tuple; 2) *table* relation which is a regular relation with a series of attributes and only one key attribute; 3) *mn_table* relation which is a relation with two key attributes in order to represent two dimensional matrices; and 4) *mn_bin_table* relation is a duplicate of an *mn_table* relation except that its only attribute contains binary values: this is a common requirement for modeling binary integer problems using spreadsheets tools.

In addition, the expanded FRL shows two additional types of relations (R_def_ mn): 1) *mn_table* i.e. a two dimensional matrix indexed by *m* and *n* and 2) *mn_bin_table*, a duplicate of the *mn_table* except that it holds binary entries. These relations defer from regular relations in that they are indexed by two attributes (Key_Attr_descr_mn). The changed or added sections of the expanded FRL in Figure 8 are illustrated in *italic*. The complete original FRL is shown in Appendix B.



Model_Schema	::= R_schema | R_schema Model_Schema
R_schema	::= R_def Key_Attr_descr | R_def Key_Attr_descr Rest_Attr_descr | *R_def_mn Key_Attr_descr_mn Attr_descr*
R_def	::= **relation** R_name **alias** R_alias_name ***(type regR_type)***
R_def_mn	*::= **relation** R_name **alias** R_alias_name **(type mnR_type)***
R_name	::= Name
Name	::= String
R_alias_name	::= Letter
regR_type	*::=* ***Vector*** | ***Table***
mnR_type	*::=* ***mn_Table*** | ***mn_bin_Table***
Key_Attr_descr	::= Data_Attr_descr **key**
Key_Attr_descr_mn	*::= (Attr_descr, Attr_descr)* ***key*** *(Attr_descr, Attr_descr)*
Rest_Attr_descr	::= Attr_descr | Attr_descr Rest_Attr_descr
Attr_descr	::= Data_Attr_descr | Func_Attr_descr
Data_Attr_descr	::= Attr_name : Type
Type	::= **number** | **string** | **date** | **logical** | ***pct*** | ***csv***
Attr_name	::= Name
Func_Attr_descr	::= Attr_name : Expr
Expr	::= Simple_Expr | Case_Expr

Figure 8. Expanded FRL to Accommodate Prototype

An example showing the use of the expanded FRL is later shown in the prototype. Next, the model storage requirements are discussed.

<u>Model Storage Requirements:</u> Figure 9 shows a high level conceptual model which describes the business objects needed to implement the model storage and selection tasks. This figure includes the Model entity at its center which designates the model schema without its data. The Instance entity stores the data portion of the model. The Solver entity stores solver related parameters. These three entities combined can be executed to create solutions for the model. User or system generated feedback can be attached to the

Feedback entity. The Mapper entity's function is to group related models instances along with their corresponding feedback.

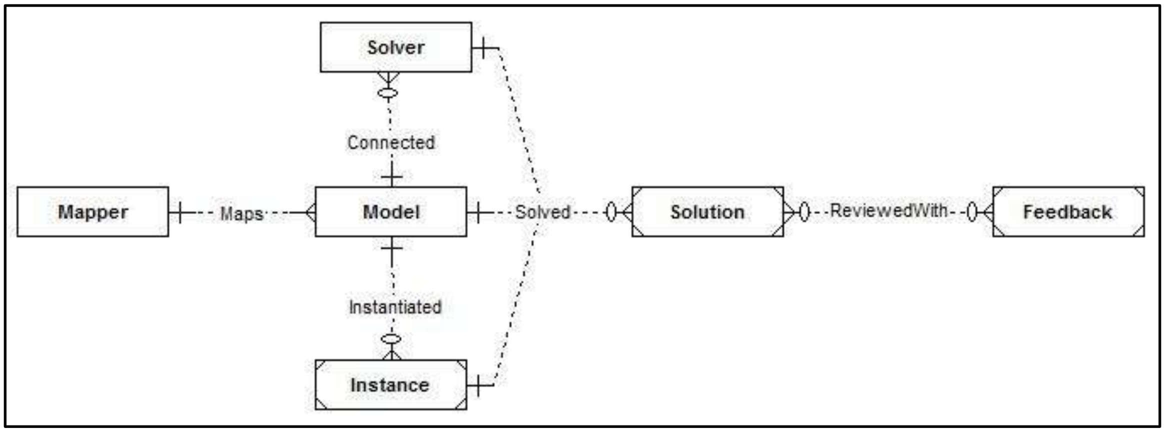


Figure 9. High Level Conceptual Model

The conceptual model shown in Figure 10 decomposes the Model entity of Figure 9 in order to accommodate the storage of the constructs specified in the expanded FRL (see Figure 8). This decomposed conceptual model is made up of three entities: Model, Relation, and Attribute. The Model entity holds one tuple per spreadsheet model. The Relation entity holds many tuples per model. Depending on the type of Relation, the Attribute entity holds one or many attributes per Relation. The Primary Key of each entity is designated with an underline. These entities all have many attributes in common which give a general description of each tuple such as: Name, Description.

Some attributes specific to certain entities would store information specific to those entities. For example, Type in Relation would be 1) vector, 2) table, 3) mn_table, or 4) mn_bin_table, whereas Type in Attribute would contain either 1) data or 2) function. The *data* type attribute is relevant only when Type equals 'data' and would contain one of the following: number, string, date, logical, pct, and csv.

Note: The following attributes, CreatedOn, CreatedBy, ModifiedOn, and ModifiedBy are included in all subsequent relations and provide information about dates when tuples were added or modified along with information about concerned users.

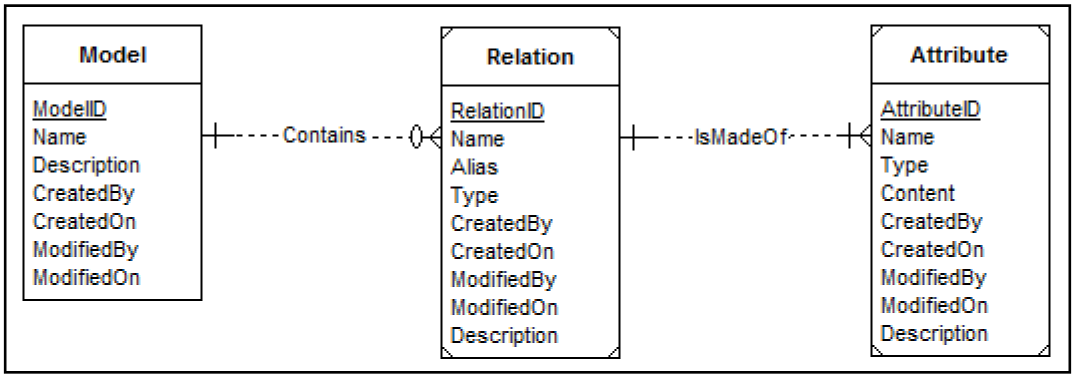


Figure 10. Model Schema Storage Conceptual Model

Implementing the conceptual model of Figure 10 using a Relational DBMS such as Microsoft Access yields the Relational model shown in Figure 11. The conceptual model and relational model each use different ways to model the association of data from the different entities or relations: the conceptual model (ER model) uses the relationship constructor, while the relational model uses referential integrity constraints in the form of a pair of Primary Key and Foreign Key.

The Foreign Key ModelID in table Relation references the Primary Key of the Model table and was created to establish the relationship between the two tables. Similarly, the table Attribute shows a new column called RelationID which references the Primary Key of the Relation table.

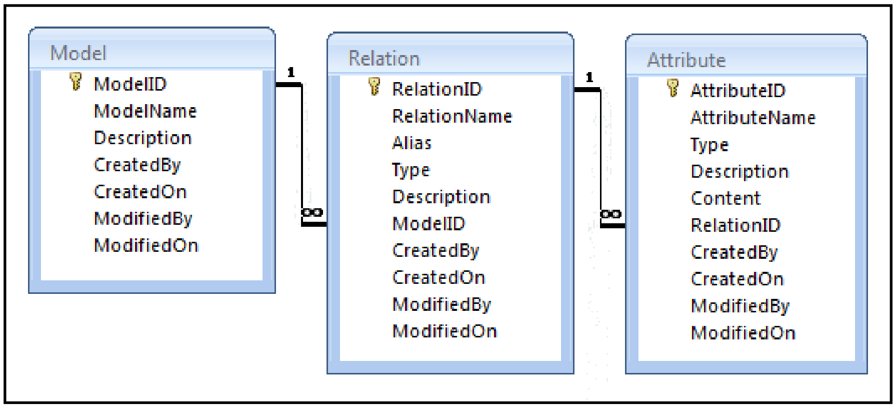
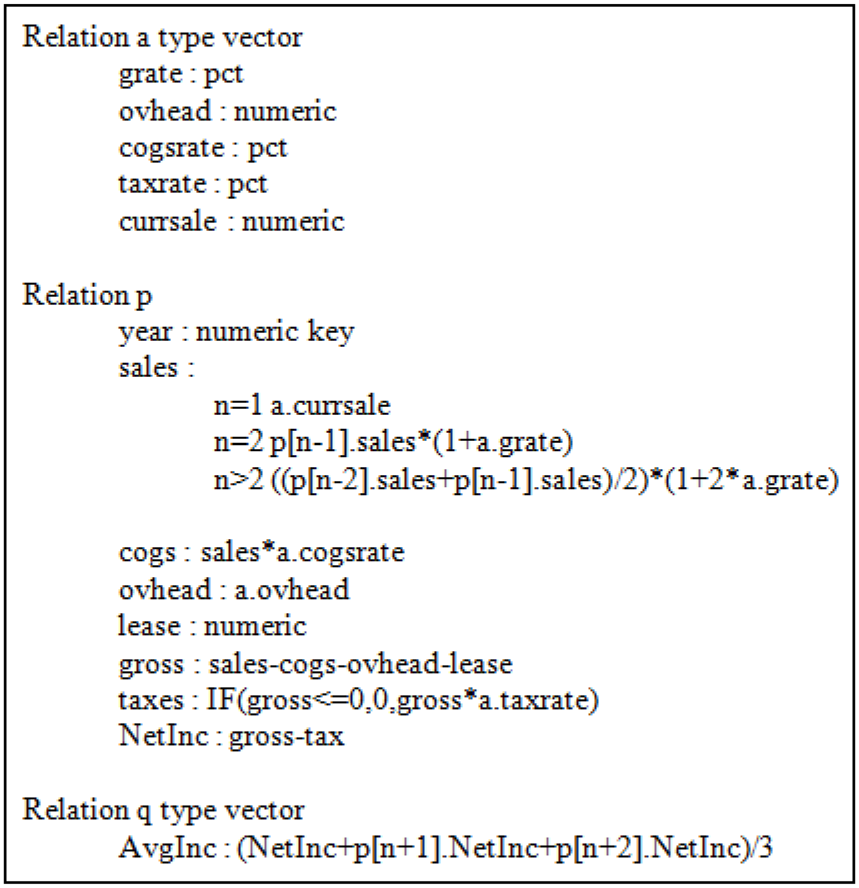


Figure 11. Model Storage Relational Model.

The schema of the spreadsheet model used in Appendix A is reproduced and shown in Figure 12 as an illustration. A more elaborate spreadsheet model is shown in the prototype section.

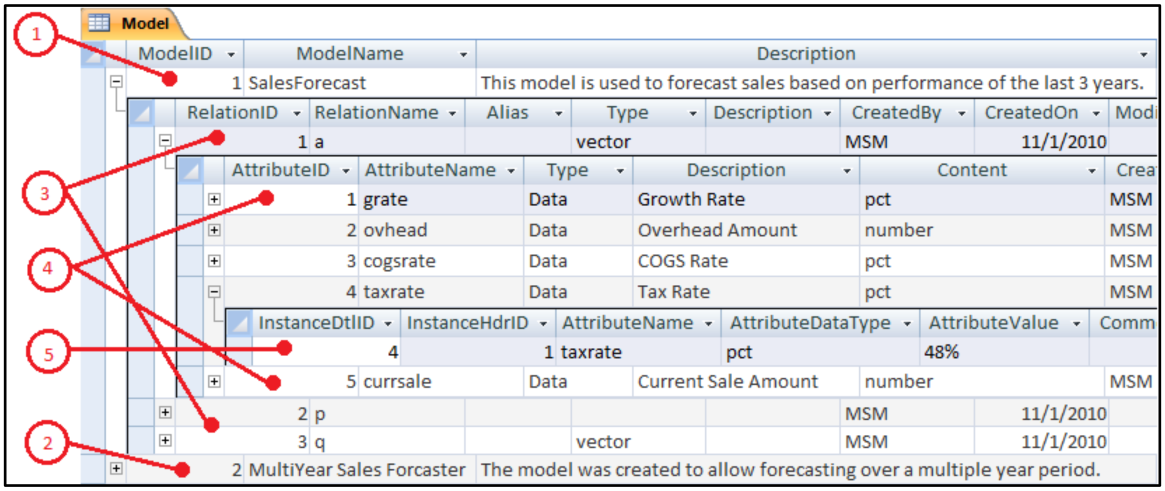


```
Relation a type vector
        grate : pct
        ovhead : numeric
        cogsrate : pct
        taxrate : pct
        currsale  : numeric

Relation p
        year : numeric key
        sales :
                n=1 a.currsale
                n=2 p[n-1].sales*(1+a.grate)
                n>2 ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)

        cogs : sales*a.cogsrate
        ovhead : a.ovhead
        lease : numeric
        gross : sales-cogs-ovhead-lease
        taxes : IF(gross<=0,0,gross*a.taxrate)
        NetInc : gross-tax

Relation q type vector
        AvgInc : (NetInc+p[n+1].NetInc+p[n+2].NetInc)/3
```

Figure 12. FRL Schema of Model to be Used as Sample

Figure 13 shows the actual tuples which make up the model discussed in Figure 12. This view is taken from Microsoft Access, which allows displaying records related by Primary/Foreign Keys in a hierarchical manner. This figure shows tuples for two models where ModelID = 1 and 2. The model with ModelID equal to one is expanded to show its components (note [1] within the figure), while ModelID equal to two is not expanded i.e. it contains a plus sign (note [2] within the figure). The model with ModelID equal to one is made up of three records (RelationID = 1, 2, and 3) as pointed out in note [3] within the figure. The relation with RelationID equal to one is expanded to show five attributes

(AttributeID = 1…5) as shown in note [4] within the figure. Note: this figure is showing an attribute instance (note [5]) which will be covered in the following section.



ModelID	ModelName	Description
1	SalesForecast	This model is used to forecast sales based on performance of the last 3 years.

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	Modi
1	a		vector		MSM	11/1/2010	

AttributeID	AttributeName	Type	Description	Content	Crea
1	grate	Data	Growth Rate	pct	MSM
2	ovhead	Data	Overhead Amount	number	MSM
3	cogsrate	Data	COGS Rate	pct	MSM
4	taxrate	Data	Tax Rate	pct	MSM

InstanceDtlID	InstanceHdrID	AttributeName	AttributeDataType	AttributeValue	Comm
4	1	taxrate	pct	48%	

AttributeID	AttributeName	Type	Description	Content	Crea
5	cursale	Data	Current Sale Amount	number	MSM

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	Modi
2	p				MSM	11/1/2010	
3	q		vector		MSM	11/1/2010	

ModelID	ModelName	Description
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.

Figure 13. Model Schema Stored in Access Database

Next, a storage area is required to keep the different instances (data) for each model schema in the database. Figure 14 shows the conceptual entity, Instance, which accommodates such data. This entity is broken down into a master/detail structure where the InstanceHdr contains one tuple for each instance of a model, and InstanceDtl contains the list of data items.

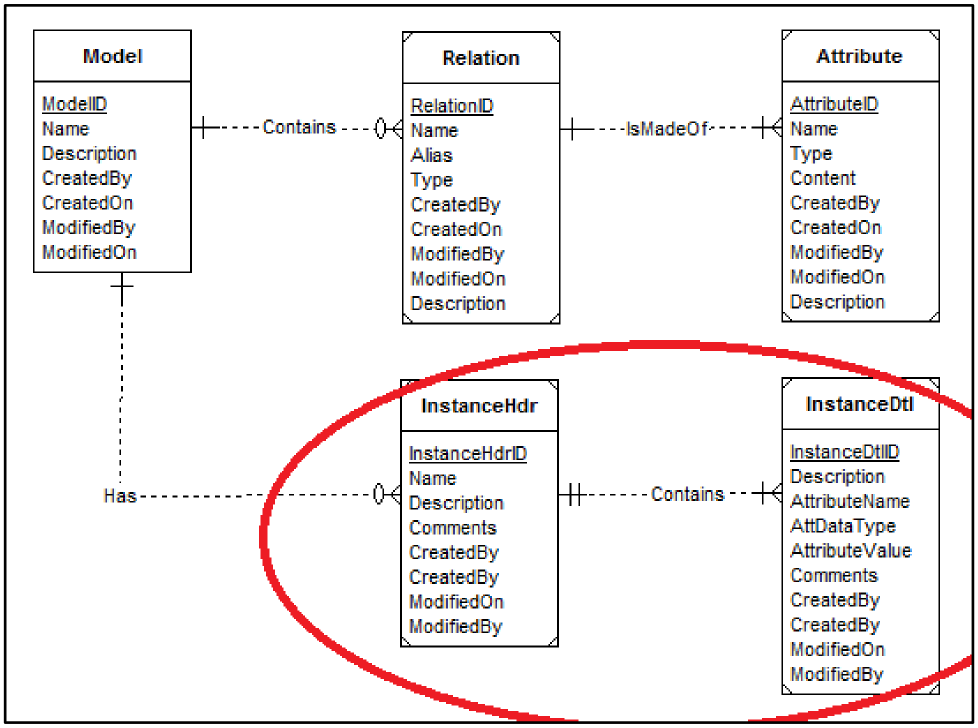


Figure 14. Instance Related Entities Added to the Conceptual Model

The attributes of InstanceHdr are: InstanceID which uniquely identifies each data instance of a model, Name and Description. Within InstanceDtl, AttributeName contains the names of the column for which this instance holds information; AttDataType contains the data type of the attribute, and AttributeValue contains the actual value for the attribute. The Comments attribute holds user comments, and the remaining attributes hold user information, and date of instance creation and/or modification. Mapping these new entities to the relational model yields the relational tables shown in Figure 15.

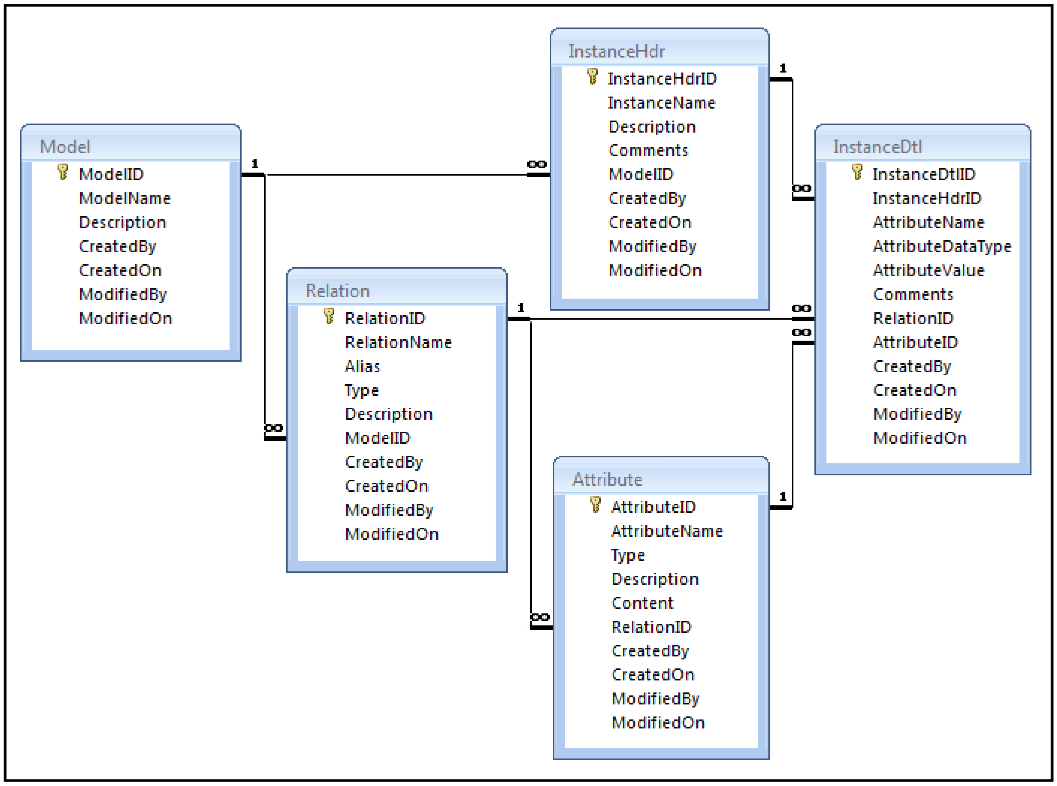
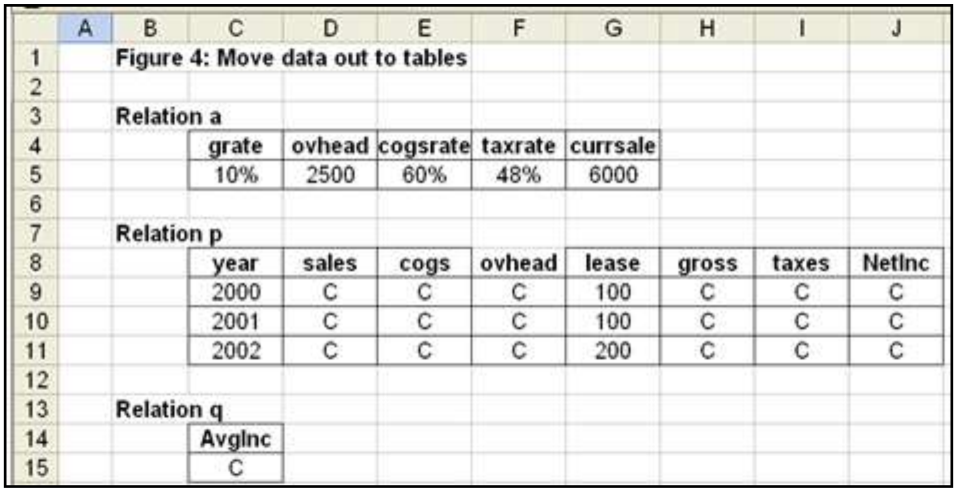


Figure 15. The Instance Tables Added to the Model

Note that these two new tables (InstanceHdr and InstanceDtl) are related to existing tables by Primary Key/Foreign Key relationships. Namely, InstanceHdr is related to the Model table since it contains general Instance information at the model level. While InstanceDtl is related to Relation and Attribute tables since these are at the lowest levels of a model.

To illustrate the usage of the instance related tables, let us use the sample model shown in Appendix A. Figure 16 shows the data portion of the spreadsheet model used in Appendix A reproduced here as a reference.

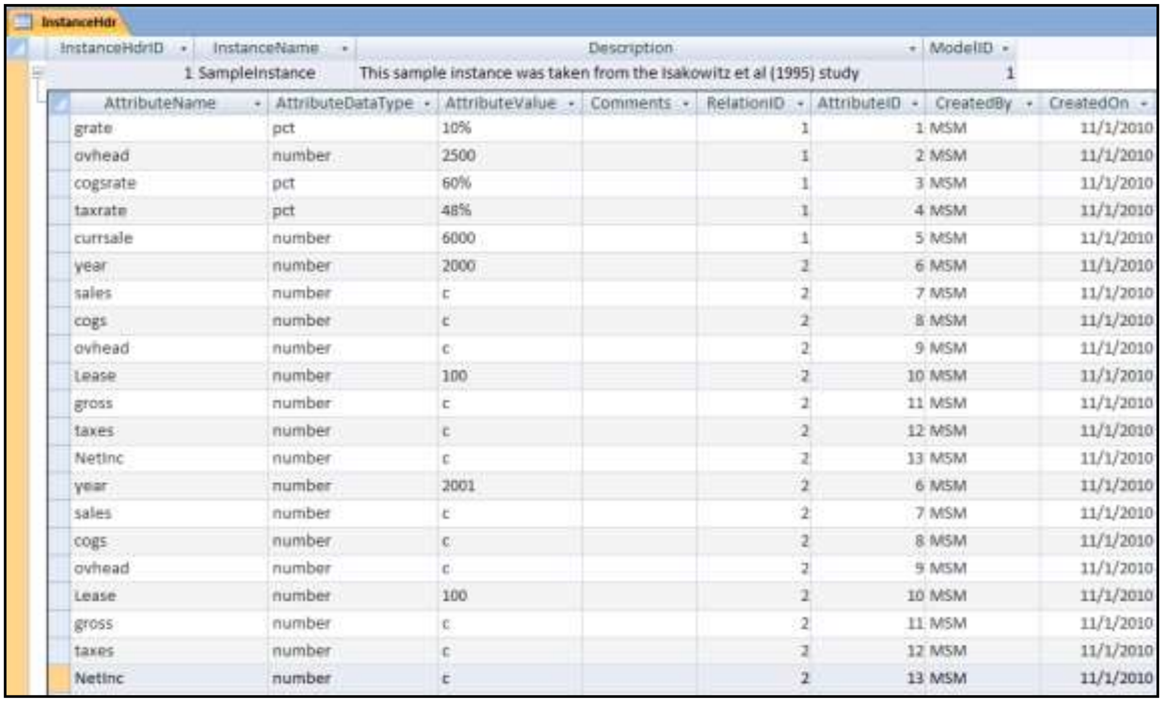


	A	B	C	D	E	F	G	H	I	J
1		Figure 4: Move data out to tables								
2										
3		Relation a								
4			grate	ovhead	cogsrate	taxrate	currsale			
5			10%	2500	60%	48%	6000			
6										
7		Relation p								
8			year	sales	cogs	ovhead	lease	gross	taxes	NetInc
9			2000	C	C	C	100	C	C	C
10			2001	C	C	C	100	C	C	C
11			2002	C	C	C	200	C	C	C
12										
13		Relation q								
14			AvgInc							
15			C							

Figure 16. The Data Portion of the Decision Model

Figure 17 shows the storage of the data portion of the decision model within Access. Note that this figure shows one record of InstanceHdr which is expanded to show the InstanceDtl records. Each record in InstanceDtl contains the attribute name, its data type, its value (note that where these values are equal to 'C' means that these attributes are calculated using some formulae which is stored in the model schema). Each record also contains the RelationID to which it is related as well as to the AttributeID. These two columns are references to storage structures within the model schema.

Figure 13 shows the data of an actual Instance within the context of the model schema. This example is pointed out by note [5] within Figure 13 and shows only one instance (InstanceDtlID = 4).



InstanceHdrID	InstanceName	Description	ModelID
1	SampleInstance	This sample instance was taken from the Isakowitz et al (1995) study	1

AttributeName	AttributeDataType	AttributeValue	Comments	RelationID	AttributeID	CreatedBy	CreatedOn
grate	pct	10%		1	1	MSM	11/1/2010
ovhead	number	2500		1	2	MSM	11/1/2010
cogsrate	pct	60%		1	3	MSM	11/1/2010
taxrate	pct	48%		1	4	MSM	11/1/2010
currsale	number	6000		1	5	MSM	11/1/2010
year	number	2000		2	6	MSM	11/1/2010
sales	number	c		2	7	MSM	11/1/2010
cogs	number	c		2	8	MSM	11/1/2010
ovhead	number	c		2	9	MSM	11/1/2010
Lease	number	100		2	10	MSM	11/1/2010
gross	number	c		2	11	MSM	11/1/2010
taxes	number	c		2	12	MSM	11/1/2010
NetInc	number	c		2	13	MSM	11/1/2010
year	number	2001		2	6	MSM	11/1/2010
sales	number	c		2	7	MSM	11/1/2010
cogs	number	c		2	8	MSM	11/1/2010
ovhead	number	c		2	9	MSM	11/1/2010
Lease	number	100		2	10	MSM	11/1/2010
gross	number	c		2	11	MSM	11/1/2010
taxes	number	c		2	12	MSM	11/1/2010
NetInc	number	c		2	13	MSM	11/1/2010

Figure 17. The Data Portion of the Model Stored in Access

When models contain too many instances, it would be productive to organize them into some groups in order to facilitate managing them. For example, when a model is instantiated for different solvers, each solver would have its own requirements and characteristics. Grouping instances by solver would prove very helpful. To accommodate such a grouping feature, the InstanceHdr table should be in a many-to-many relationship with a new table named Group. Figure 18 shows such a table. The relation GroupInstances acts as an intersection table and holds the Primary Keys of both related tables (Group and InstanceHdr).

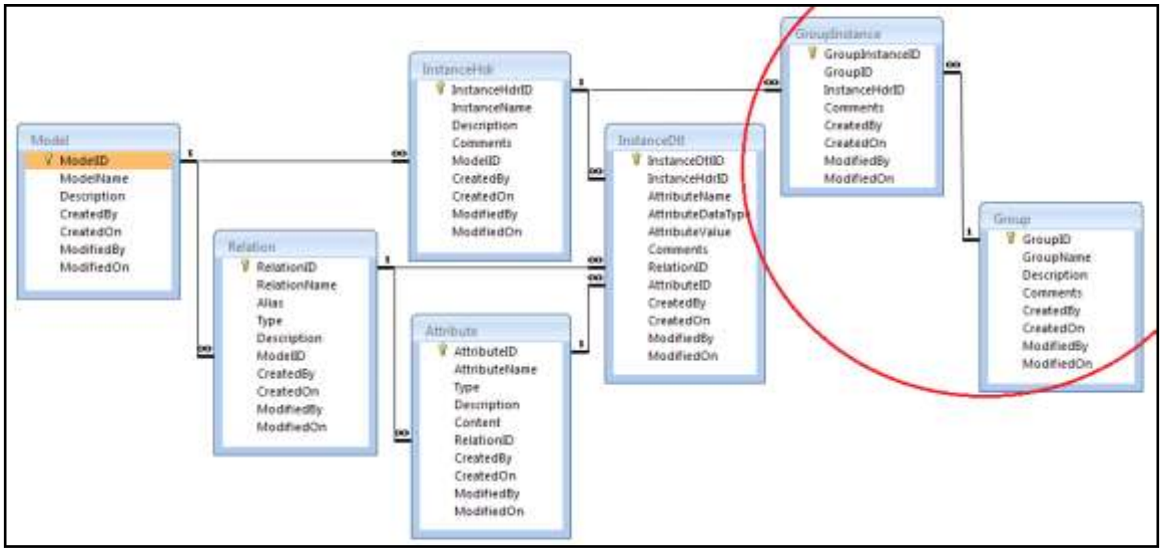


Figure 18. Grouping Feature Added

When the schema notation methods (FRL and ESML) were initially applied to spreadsheet modeling, the main focus was to represent one model at a time (Isakowitz et al., 1995; Iyer et al., 2005). Inter-model comparisons and evaluations were not a concern. Spreadsheet software products like Microsoft Excel provide a tool called *scenario manager* which also addresses one model at a time. A tool which allows a user to execute different models using a same set of input parameters would be a very useful feature.
In order to automate the comparison of different models, created by different developers, there needs to be a dictionary of terms employed within models, and a mapping mechanism which relates the terms/fields used within different models. For example, let us consider a new spreadsheet model that performs the exact same calculations as the model shown in Figure 4, with the following differences: 1) The new model is spatially laid out in a completely different format, and 2) the field names/labels are different (e.g. *avg_net_inc* is called *avi*). Without a mechanism which maps the field *avg_net_inc* in one

model to a field called *avi* in another model, it would not be possible to automatically compare the two models. Figure 19 shows a sample of mapping terms between two models A and B.

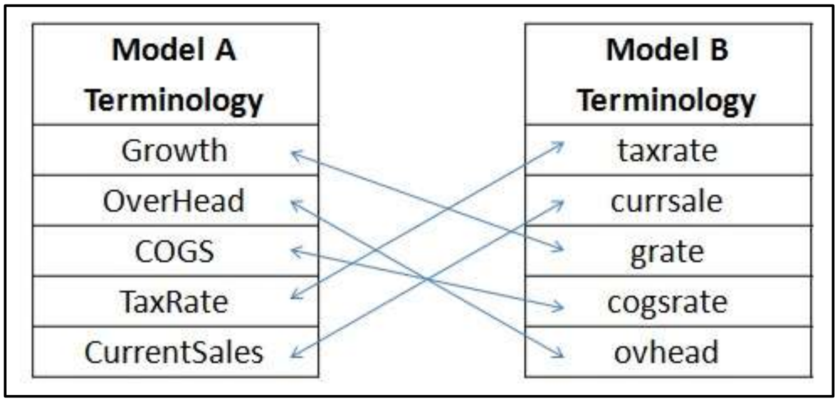


Figure 19. Sample Terminology Mapping Between Two Models A and B

Once such a mapping mechanism is created, a Virtual Business Environment (VBE) such as the one described by Iyer et al. (2005) can be extended to include features which automatically instantiate different models and perform model comparison tasks (Scenario One of the prototype in Chapter 4 provides a complete example), thus providing suitable model selection insights and guidance to end users. Figure 20 shows a conceptual model for the dictionary mapping and its implementation in the relational database model is shown in Figure 21.

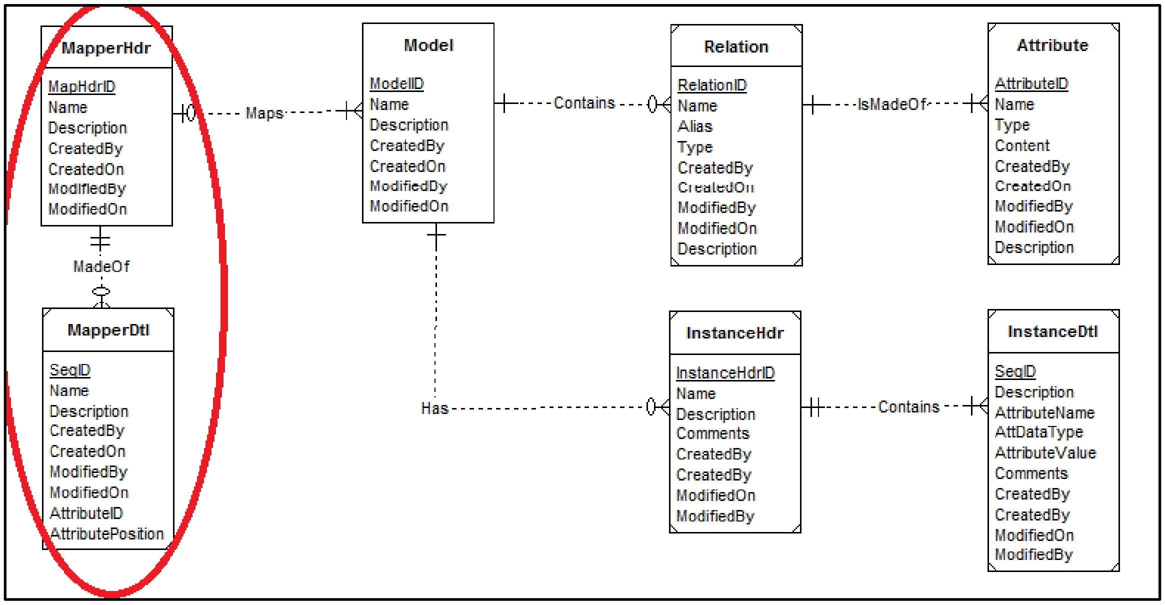


Figure 20. Model Mapper Portion Added

The conceptual Mapper entity requires two entities: 1) a mapper header and a 2) mapper detail. The header portion (MapperHdr) holds one tuple per mapping (i.e. for a set of models which are mapped to each other) and it contains the name of the mapping and a user entered description. The detail portion (MapperDtl) holds one tuple for each attribute of models mapped. For example, if three models are mapped, and each model contains five attributes, the total number of tuples within MapperDtl for such a mapping is 15 (i.e. three times five). AttributeID holds the ID of the attribute to be mapped taken from the Attribute table. AttributePosition holds a number which designates the internal position of attributes. For example, when mapping say the attributes of three different models, attributes to be mapped to each other have the same AttributePosition. From the example shown in Figure 19, attributes *growth* and *grate* have a same number. Figure 21 shows the implementation of the Mapper entity into the relational model.

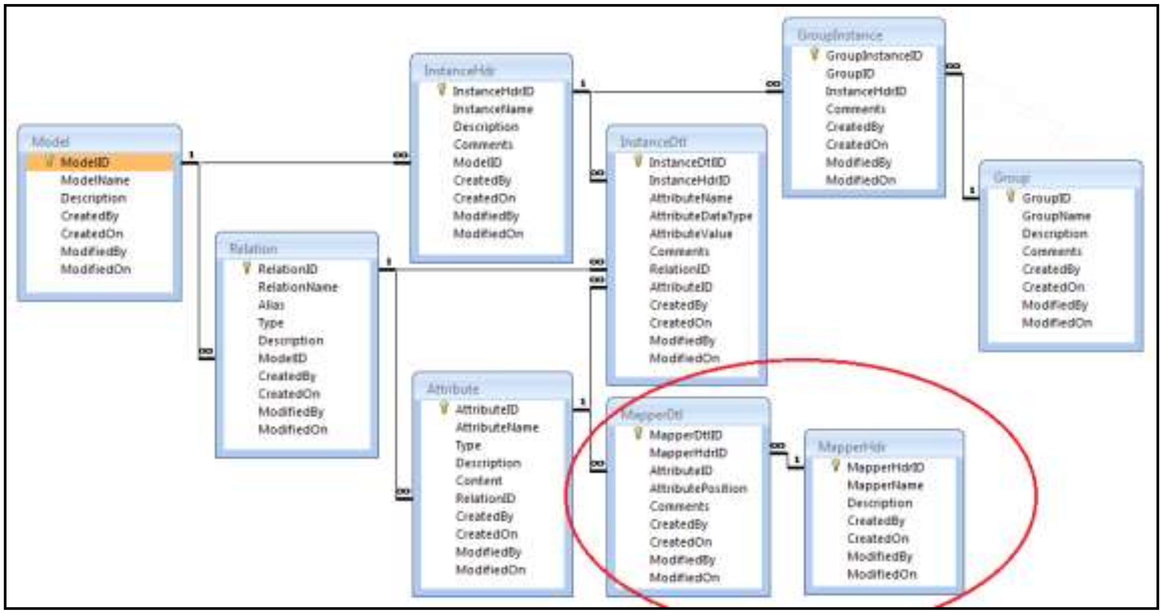
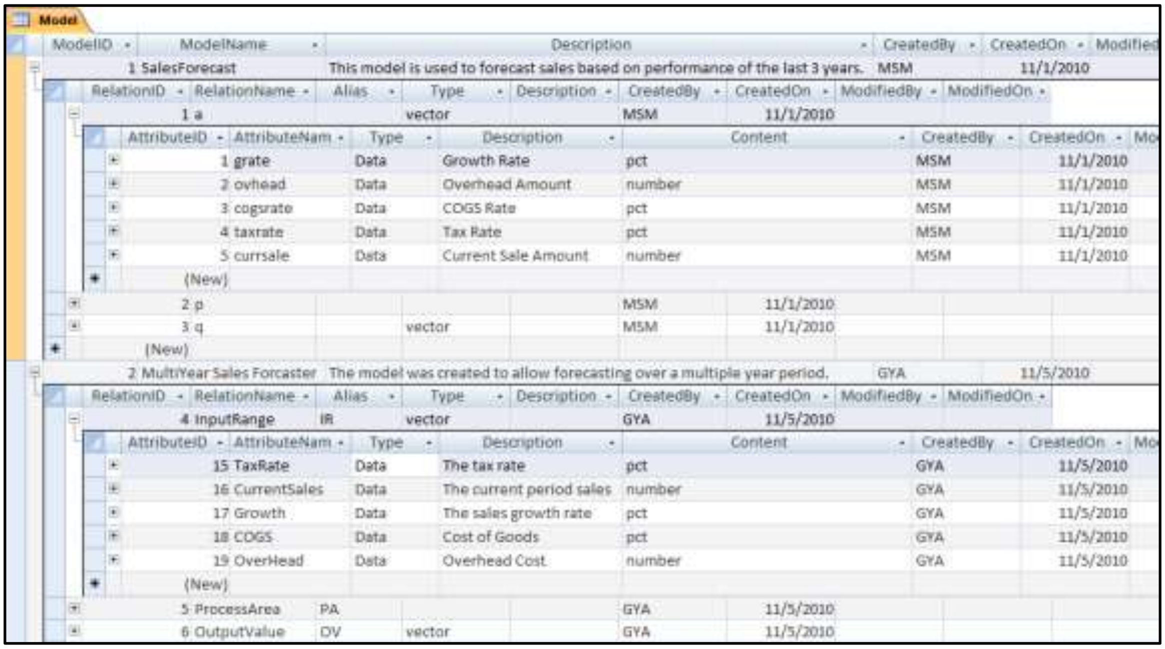


Figure 21. Mapper Related Table Added

Note: the MapperDtl table does not contain the ModelID information since it can be derived from the AttributeID which is a unique identifier linked to only one model. In order to illustrate the mapping feature, the mapping example used in Figure 19 is implemented in the relational model. Figure 22 shows the two different models: ModelID = 1 and 2.



ModelID	ModelName	Description	CreatedBy	CreatedOn	Modified
1	SalesForecast	This model is used to forecast sales based on performance of the last 3 years.	MSM	11/1/2010	

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
1	a		vector		MSM	11/1/2010		

AttributeID	AttributeNam	Type	Description	Content	CreatedBy	CreatedOn	Mo
1	grate	Data	Growth Rate	pct	MSM	11/1/2010	
2	ovhead	Data	Overhead Amount	number	MSM	11/1/2010	
3	cogsrate	Data	COGS Rate	pct	MSM	11/1/2010	
4	taxrate	Data	Tax Rate	pct	MSM	11/1/2010	
5	currsale	Data	Current Sale Amount	number	MSM	11/1/2010	
(New)							

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
2	p				MSM	11/1/2010		
3	q		vector		MSM	11/1/2010		
(New)								

ModelID	ModelName	Description	CreatedBy	CreatedOn	Modified
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.	GYA	11/5/2010	

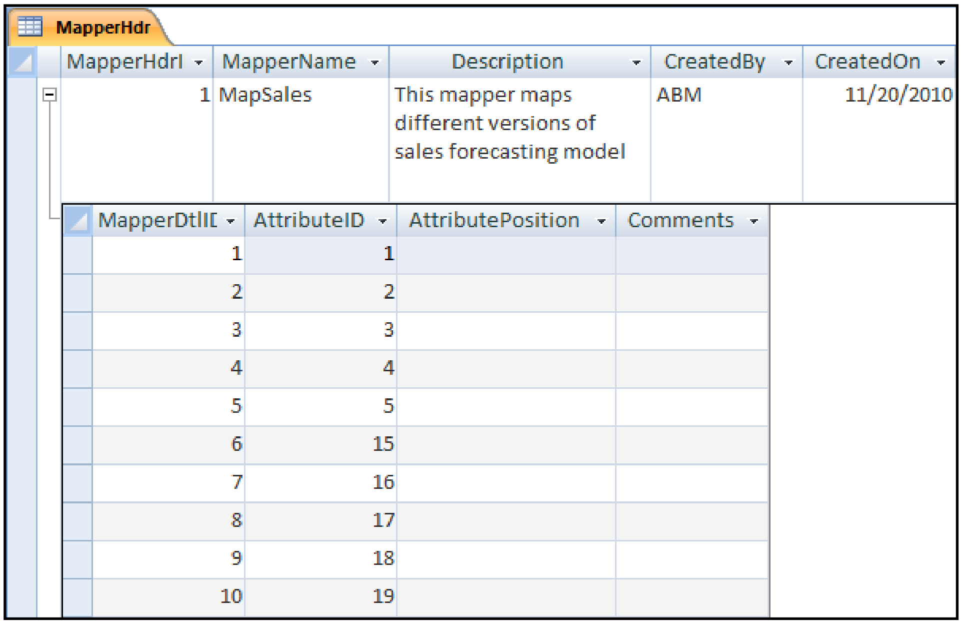
RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
4	InputRange	IR	vector		GYA	11/5/2010		

AttributeID	AttributeNam	Type	Description	Content	CreatedBy	CreatedOn	Mo
15	TaxRate	Data	The tax rate	pct	GYA	11/5/2010	
16	CurrentSales	Data	The current period sales	number	GYA	11/5/2010	
17	Growth	Data	The sales growth rate	pct	GYA	11/5/2010	
18	COGS	Data	Cost of Goods	pct	GYA	11/5/2010	
19	OverHead	Data	Overhead Cost	number	GYA	11/5/2010	
(New)							

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
5	ProcessArea	PA			GYA	11/5/2010		
6	OutputValue	OV	vector		GYA	11/5/2010		

Figure 22. Two Sample Models in SQL Database

Figure 23 shows MapperHdr with a mapping named MapSales along with a list of attributes from different models. These attributes are not yet mapped since the AttributePosition column for all attributes is blank.

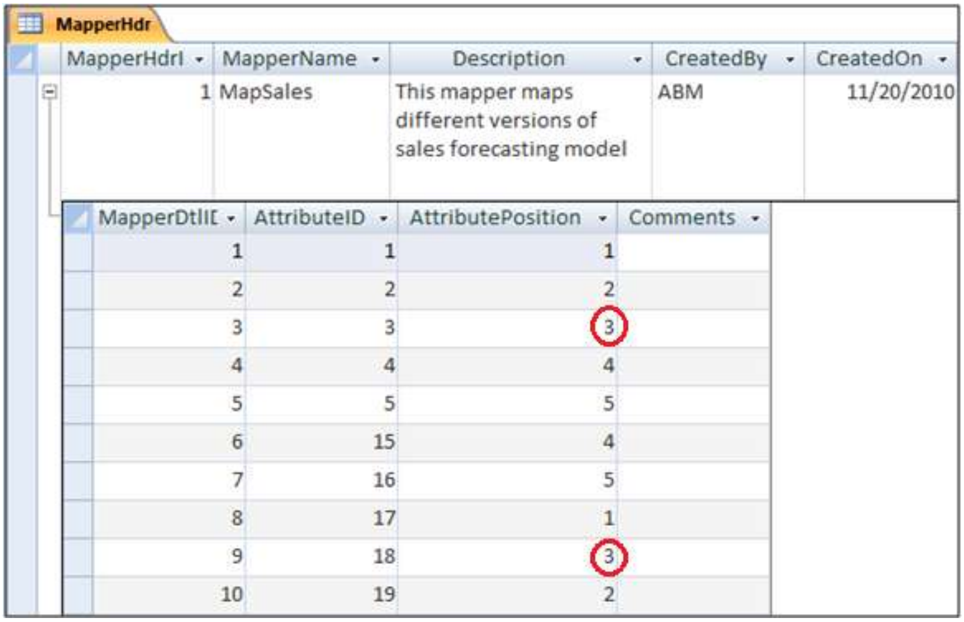


MapperHdrI	MapperName	Description	CreatedBy	CreatedOn
1	MapSales	This mapper maps different versions of sales forecasting model	ABM	11/20/2010

MapperDtlII	AttributeID	AttributePosition	Comments
1	1		
2	2		
3	3		
4	4		
5	5		
6	15		
7	16		
8	17		
9	18		
10	19		

Figure 23. Listing of Unmapped Attributes

Figure 24 shows mapped attributes since the attribute positions have data. In this example, AttributeIDs three and 18 are mapped to each other since they both have the same position entry of three (shown in red circle in the figure).



MapperHdr

MapperHdrI	MapperName	Description	CreatedBy	CreatedOn
1	MapSales	This mapper maps different versions of sales forecasting model	ABM	11/20/2010

MapperDtlI	AttributeID	AttributePosition	Comments
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	15	4	
7	16	5	
8	17	1	
9	18	3	
10	19	2	

Figure 24. Listing of Mapped Attributes With Positions

Phase Outputs: The following list enumerates the key outputs from this phase:

- Description of the mechanisms involved in converting a spreadsheet model into its FRL format and back to spreadsheet format
- Description of the mechanisms involved in storing and managing an FRL schema
- Conceptual data model and a mapping to data structures for a relational database model to accommodate FRL schemata
- Conceptual data model and a mapping to data structures for a relational database model to accommodate an inter-model field mapping mechanism and instance grouping

- Description of the mechanisms involved in storing and managing model mappings which serve to relate various models along with instance grouping.

Phase II – Identifying the Appropriate Metadata About the Spreadsheet Models

In order to facilitate the task of comparing various decision models and to select the most appropriate model for a specific task, 1) a method is needed which facilitates entering comments/insights, storing, categorizing and retrieving them based on some criteria, and 2) a metadata is required which stores information about models.

Since a spreadsheet environment is open and flexible, users can place comments in any cells they wish. Some of these comments may be relevant to the entire model, while others may be applicable to only some parts. This fragmented method of including comments without attaching them to specific cells is not always helpful in comparing different model. A new feature should be added to spreadsheet tools whereby users are able to attach their comments and where these can be consolidated and later presented to users for review. When users enter comments, each should be attached to an attribute instance or group, an attribute, a relation, or a model. Attaching comments to attributes and tuples in instances, relations or models is trivial since all that is needed is to keep track of the ID for each of these.

Figure 25 shows two tables which implement the feedback portion of the model selection system. There are two tables added: 1) Feedback and FeedbackCategory. The Feedback table contains one record for each feedback entered. Each feedback can be attached to: 1) an attribute 2) an attribute instance, 3) a relation, or 4) a model. A logical attribute instance is made up of two physical tables: InstanceHdr and InstanceDtl. Also, the Grouping feature discussed earlier allows the grouping of instances together for further analysis.

Note that Figure 25 does not show a relationship line between Feedback and the tables it is related to. It would have been possible to create one Feedback table for each of the six tables to which Feedback is related: InstanceDtl, InstanceHdr, Group (of instances), Attribute, Relation, or Model. However, for compactness, this dissertation uses only one table (Feedback), and records are linked to the appropriate tables based on the value of the Scope column. Depending on the value of the Scope column of the Feedback table, a record can be connected (related) to any of InstanceDtl, InstanceHdr, Group (grouped instances), Attribute, Relation, or Model. The domain of the *scope* column is (Id, Ih, Ig, A, R, M) respectively to the name of the tables to which each record is related. Each RelationID column of the Feedback table accordingly contains the Primary Keys of one of the following relations: InstanceDtl, InstanceHdr, Group, Attribute, Relation, or Model, thus associating a feedback to one of the tables.

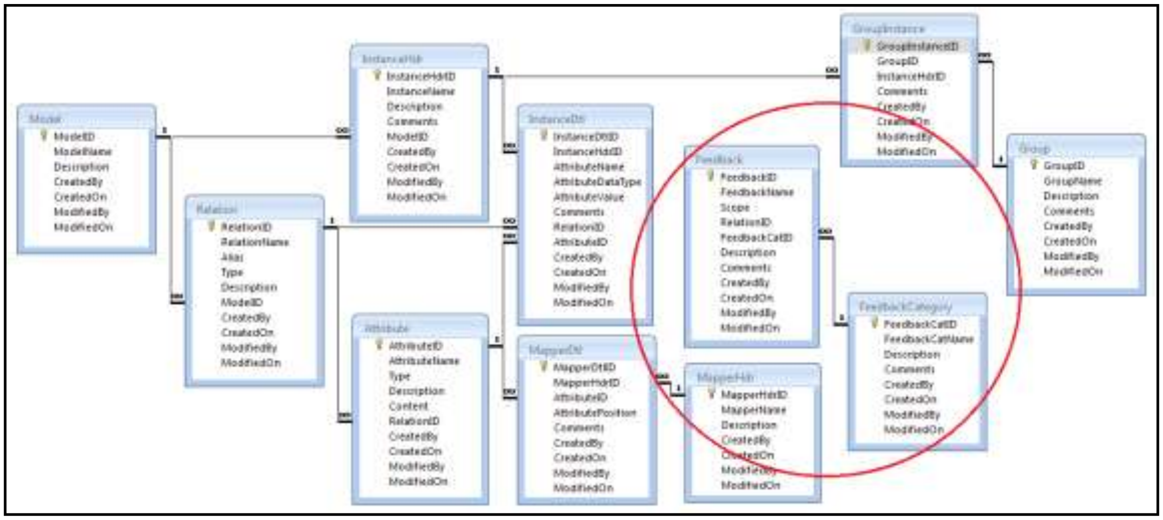


Figure 25. Feedback Portion of Meta-Model Shown

The FeedbackID attribute of the Feedback table contains the category of the comment. These categories are instrumental in grouping all comments generated for each model, and presenting them to potential users in order to facilitate model comparison and selection. These categories are discussed next.

Based on the existing literature reviewed in the literature review chapter, this dissertation proposes a framework which organizes a preliminary criteria list for model selection as shown in Table 1 (found in the literature review chapter), i.e. accuracy of model, computational complexity, trust in model, sufficiency, input/output needs, distance from goal, performance/ difficulty of solvability, number of components, cost of model, robustness of model, reliability of model, architecture/structure of model, simplicity of model structure, and availability of designer comments. This list is preliminary in the sense that it is not the main focus of this study, but rather a small step leading to the overall goal of spreadsheet based model comparison and selection. This list is intended as a starting point for future research. It can be expanded and refined to make it applicable to specific problem domains.

Following the tradition of the testing sub-discipline in software engineering, this list of criteria can be divided into two major categories: 1) black box metrics and 2) white box metrics. Black box metrics deal with variables which do not disclose internal technical information about a model, thus making it easy for a basic user to select a model based on external characteristics. White box metrics on the other hand include internal technical information which requires more technically knowledgeable users.

The breakdown of criteria gives users the opportunity to assign more weight values, based on whether they prefer to inspect the internal makeup of a model, or whether they attach more importance to information external to a model. Figure 26 shows this dissertation's proposed preliminary decision model evaluation framework.

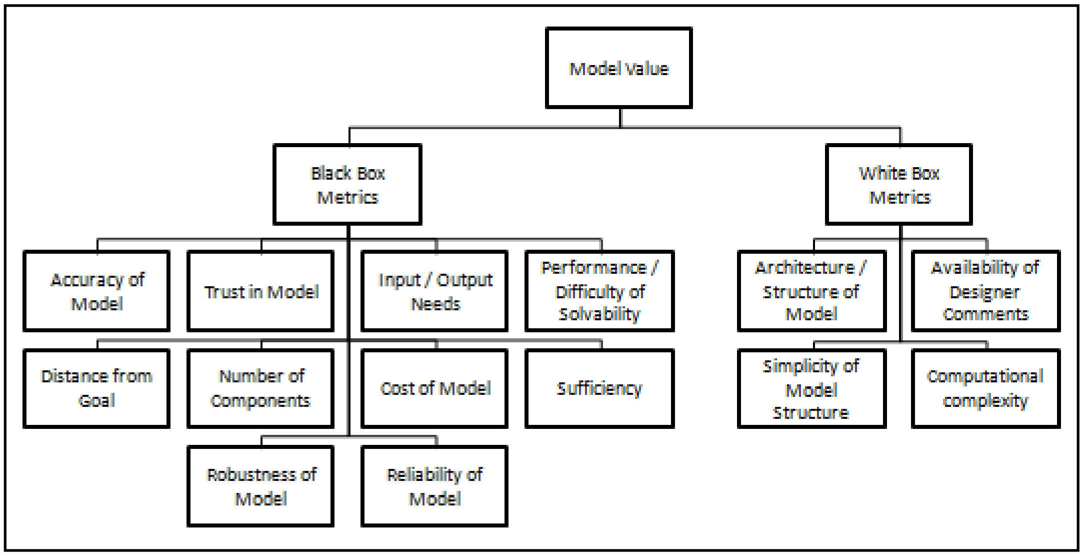
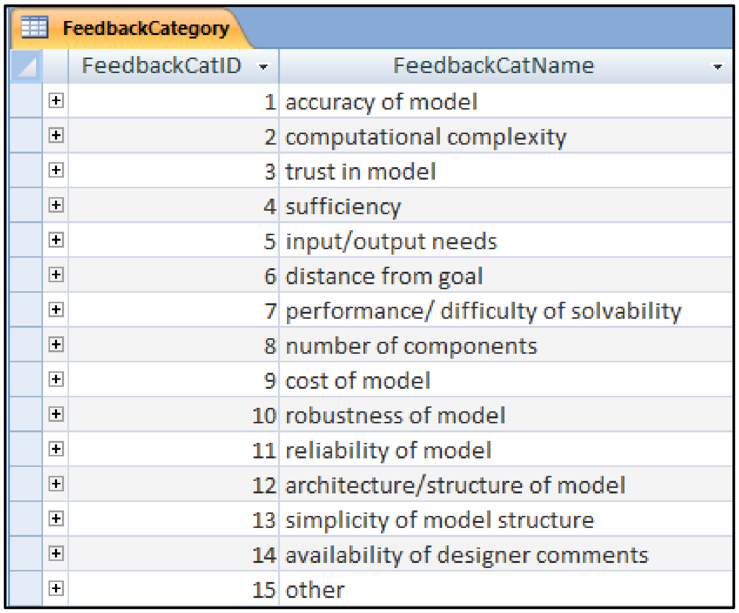


Figure 26. Proposed AHP Model Evaluation Framework

Figure 27 shows a list of sample tuples for the FeedbackCategory table. These feedback categories are based on the prior research. One additional record named *other* is also added in order to allow for cases which do not fit one of the existing categories. This list can be expanded as new categories are needed.

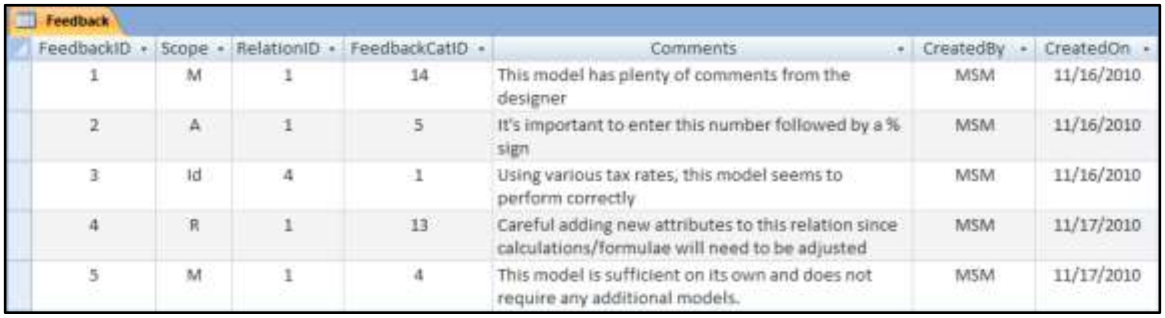


FeedbackCategory

FeedbackCatID	FeedbackCatName
1	accuracy of model
2	computational complexity
3	trust in model
4	sufficiency
5	input/output needs
6	distance from goal
7	performance/ difficulty of solvability
8	number of components
9	cost of model
10	robustness of model
11	reliability of model
12	architecture/structure of model
13	simplicity of model structure
14	availability of designer comments
15	other

Figure 27. A Sample List of Tuples in Table FeedbackCategory

Figure 28 shows a sample feedback for a one model. The fact that it is for just one model is not visible from the figure since some of the feedback records are not referring to a model, but rather to components of a model i.e. relation, attribute, instance etc. The first line (FeedbackID = 1) has the Scope = ‘M’, meaning that this comment is attached to the model as a whole. Therefore, the following attribute, RelationID = 1 is the Primary Key of the table Model. The FeedbackCatID = 14 states that this comment is related to the “availability of designer comments” (see Figure 27 for list of feedback categories). The second line in the figure (FeedbackID = 2) has the Scope = ‘A’, meaning that this comment is attached to a specific attribute within the model. Therefore, the following attribute, RelationID = 1 is the Primary Key of the table Attribute. The FeedbackCatID = 5 means that this comment is related to the “input/output needs” (see Figure 27).

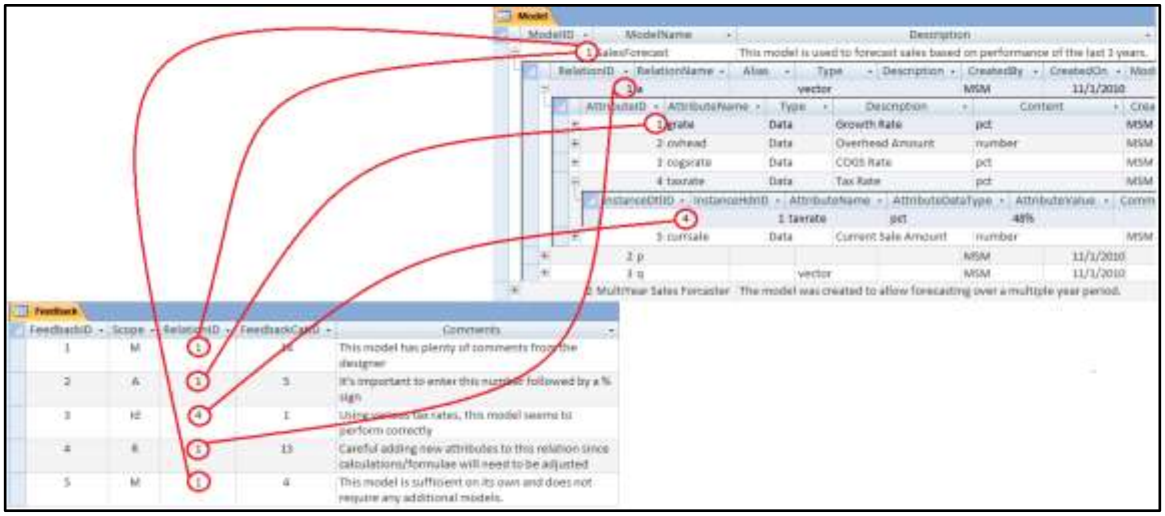


FeedbackID	Scope	RelationID	FeedbackCatID	Comments	CreatedBy	CreatedOn
1	M	1	14	This model has plenty of comments from the designer	MSM	11/16/2010
2	A	1	5	It's important to enter this number followed by a % sign	MSM	11/16/2010
3	Id	4	1	Using various tax rates, this model seems to perform correctly	MSM	11/16/2010
4	R	1	13	Careful adding new attributes to this relation since calculations/formulae will need to be adjusted	MSM	11/17/2010
5	M	1	4	This model is sufficient on its own and does not require any additional models.	MSM	11/17/2010

Figure 28. A Sample Tuple in Table Feedback

The third line in Figure 28 (FeedbackID = 3) has Scope = ‘Id’, meaning that this comment is attached to a specific instance detail (i.e. specific cell). Therefore, the following attribute, RelationID = 4, is the Primary Key of the table InstanceDtl. The FeedbackCatID = 1 means that this comment is related to the “accuracy of model”. The fourth line in Figure 28 (FeedbackID = 4) has Scope = ‘R’, meaning that this comment is attached to a specific relation. Therefore, the following attribute, RelationID = 1, is the Primary Key of the table Relation. The FeedbackCatID = 13 states that this comment is related to the “simplicity of model structure”. The fifth line in the figure (FeedbackID = 5) has the same scope as line 1, i.e. Scope = ‘R’, meaning that this comment is attached to a specific relation in Relation table. The difference with the first record is that line five is related to a different feedback category i.e. FeedbackCatID = 4 which means that this comment is related to the “sufficiency” feedback category of the model.

Figure 29 brings together the Figure 13 and Figure 28 and visually shows the relationship of each tuple of the table in the bottom left corner with those of the tables of the upper right corner. For example, the third tuple from the bottom left (i.e. FeedbackID = 3) is related to the InstanceDtlID Primary Key of the InstanceDtl relation (i.e. InstanceDtlID = 4). The other lines in the figure show the rest of the relationships.



Model

ModelID	ModelName	Description
1	SalesForecast	This model is used to forecast sales based on performance of the last 3 years.
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	Mod
1	a		vector		MSM	11/1/2010	
2	p				MSM	11/1/2010	
3	q		vector		MSM	11/1/2010	

AttributeID	AttributeName	Type	Description	Content	Crea
1	grate	Data	Growth Rate	pct	MSM
2	ovhead	Data	Overhead Amount	number	MSM
3	cogsrate	Data	COGS Rate	pct	MSM
4	taxrate	Data	Tax Rate	pct	MSM
5	cursale	Data	Current Sale Amount	number	MSM

InstanceDtlID	InstanceHdrID	AttributeName	AttributeDataType	AttributeValue	Comm
4	1	taxrate	pct	48%	

Feedback

FeedbackID	Scope	RelationID	FeedbackCatID	Comments
1	M	1	12	This model has plenty of comments from the designer
2	A	1	5	It's important to enter this number followed by a % sign
3	IE	4	1	Using various tax rates, this model seems to perform correctly
4	R	1	13	Careful adding new attributes to this relation since calculations/formulae will need to be adjusted
5	M	1	4	This model is sufficient on its own and does not require any additional models.

Figure 29. Sample of Feedback to Model Components Relation Creation

The data structures shown in this section enable the storage of all feedback in a structured manner conducive to be regrouped and presented in different views in order to facilitate model comparison and selection.

Phase Outputs: The following list enumerates the key outputs realized in this phase:

- A list of criteria which is used to evaluate different decision models. A tentative list is provided in Figure 26
- Conceptual data model and its mapping to data structures for a relational database model to hold various model characteristics, such as time to find a solution, quality of the output, etc. and a feedback maintenance system.

Phase III – Developing a Suitable Spreadsheet Model Selection Strategy

The previous phases showed how to prepare, store and execute models along with their metadata. This phase shows the usage of AHP to compare models and select an appropriate one, based on model-related values (feedback criteria) as discussed in the previous phase. In this phase, users are presented with the list of criteria as shown in the framework (see Figure 26). Using this framework, users select which criteria are most important and relevant for their particular situation, and assign weights to each criterion by performing *pairwise* comparisons as described by Saaty (1986). Users then enter score values for each criterion/alternative based on the information presented to them, or based on their own investigations of each model. The system then calculates the total weighted score for each alternative (model) and presents the models, ordered from best to least fitting to the background and needs of the user.

Let us illustrate how two different types of users differ in their ways of using the AHP model, based on their own personal backgrounds and based on their problem situation and needs. Let us refer to the first type of user as *basic* while referring to the second as *advanced.*

1. *Basic User:* an example of such a user would be a manager without any specialized technical knowledge about internal model structures, and who wishes to select and use a decision model. Once this user is presented with the *Proposed AHP Model Evaluation Framework* model shown in Figure 26, the user has the option to eliminate some criteria by placing an ‘X’ as shown in the following Figure 30.

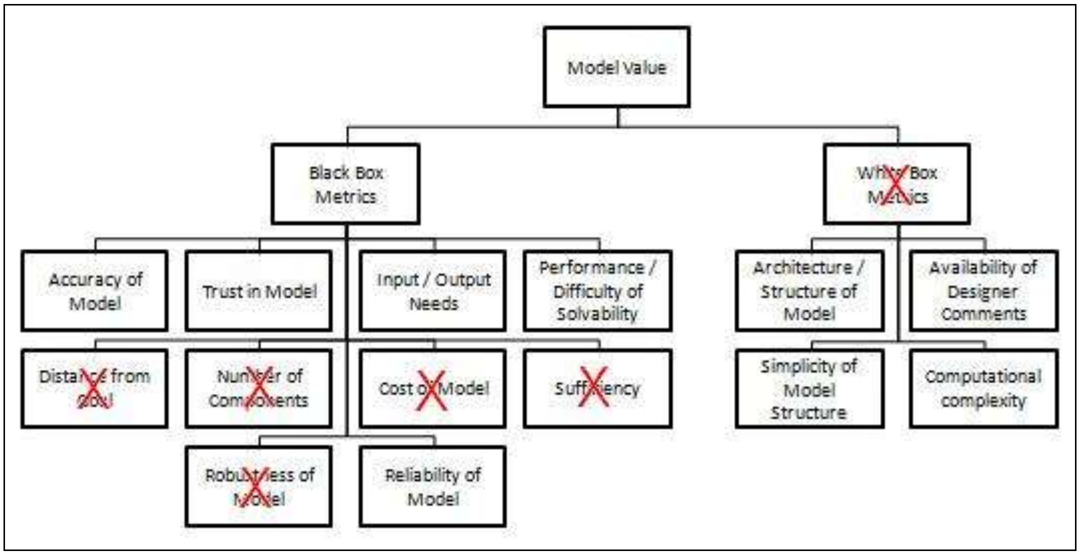


Figure 30. Sample Decision Model Chosen by a Non-Technical User

The system then re-arranges the remaining criteria and assigns default, equally distributed weights as shown in the following Figure 31.

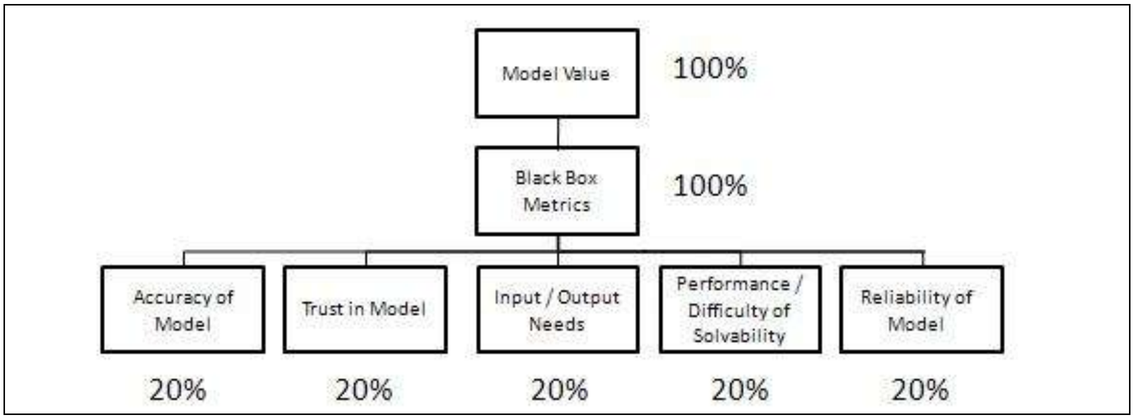


Figure 31. Sample Decision Model With Default Weights Chosen by a Non-Technical User

The system then allows the user to perform *pairwise* comparisons of the selected criteria shown in Figure 31, which further readjusts the weights according to the user's preferences. Figure 32 shows an example of a user-entered weighted criteria list:

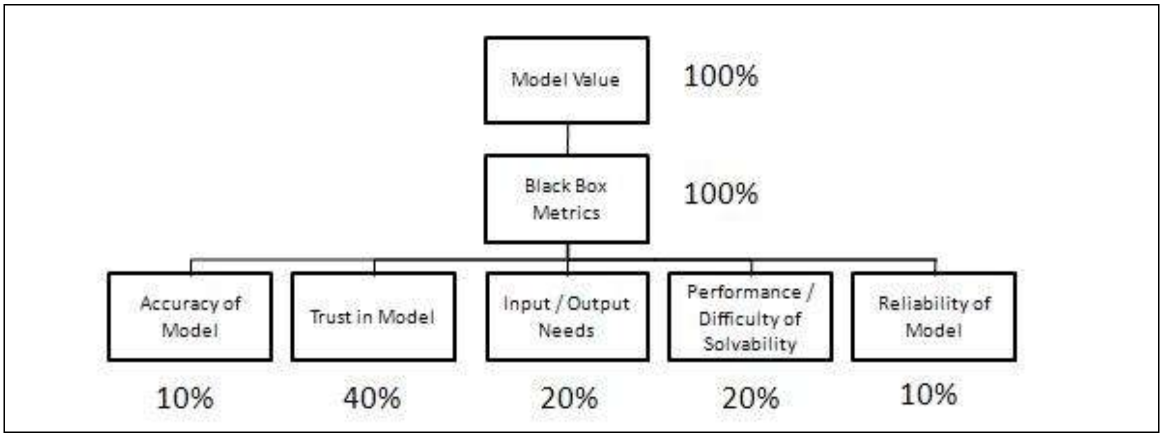


Figure 32. Sample Decision Model With Weights Chosen by a Non-Technical User

Note: the sample decision model displayed in Figure 32 shows that this user attaches the most importance on the criteria *trust in model* (40% weight) i.e. based on what other users have said about each model and how much this user trusts the opinion of others.

The user then examines each of the models presented by the system along with metadata information, and assigns a score based on each criterion. For example, the user might determine that some models are not appropriate for the problem. Such insight about models results in pushing them to the bottom of the prioritized list of models. The system then calculates the weighted scores for each model and displays them ordered from best score to worst.

2. *Advanced User:* another scenario is an example of a manager with more technical knowledge than the previous scenario and one that would be interested in knowing more about the internal makeup of models before selecting the appropriate one. This kind of user looks into and examines the *white box* list of criteria as well as the *black box* ones (see Figure 26). Figure 33 shows one sample of such user choices.

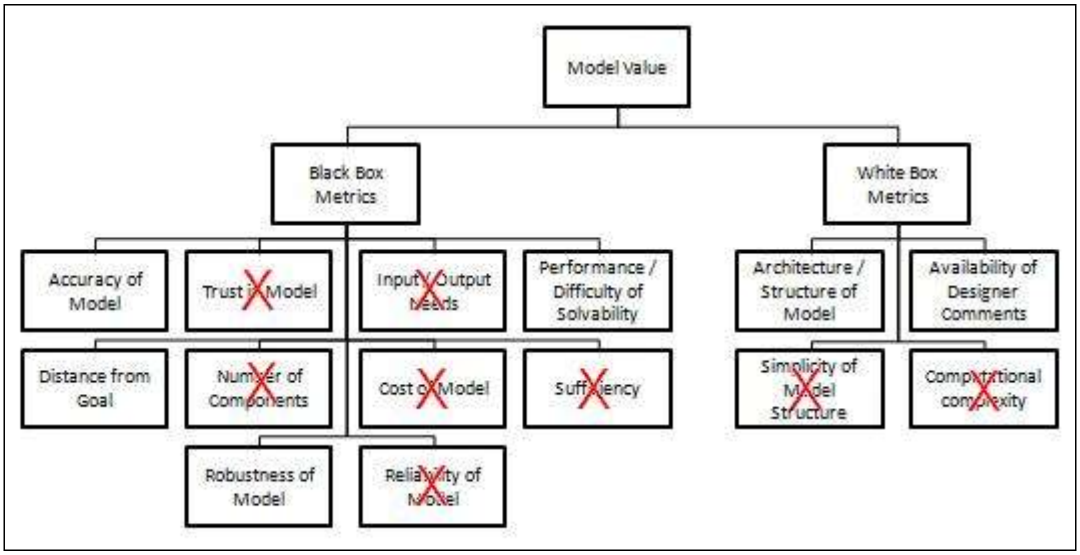


Figure 33. Sample Decision Model Chosen by a Technically-Savvy User

The system then re-arranges the remaining criteria and assigns default weights (equally distributed) as shown in Figure 34.

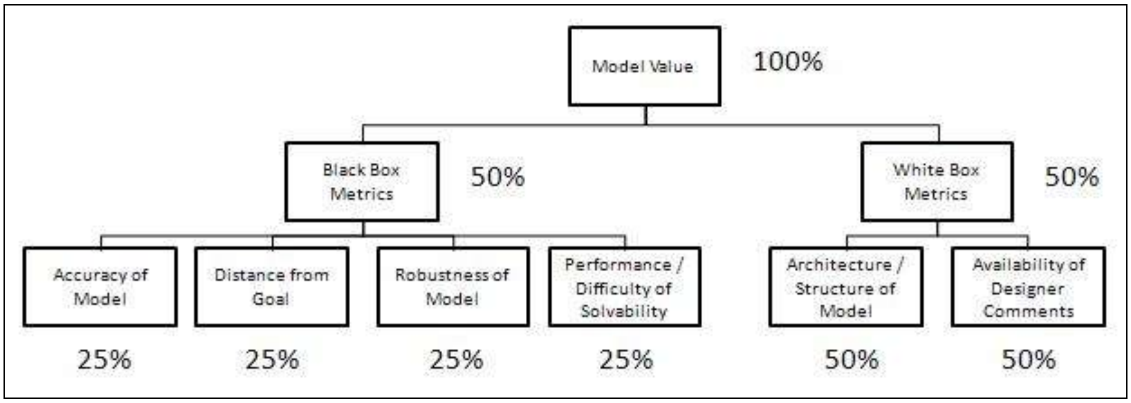


Figure 34. Decision Model With Default Weights Chosen by a Non-Technical User

As in the previous scenario, the system then allows the user to perform *pairwise* comparisons of the selected criteria shown in Figure 34, which in turn readjusts the weights according to the user’s preferences. Figure 35 shows one possible user redistributed weights:

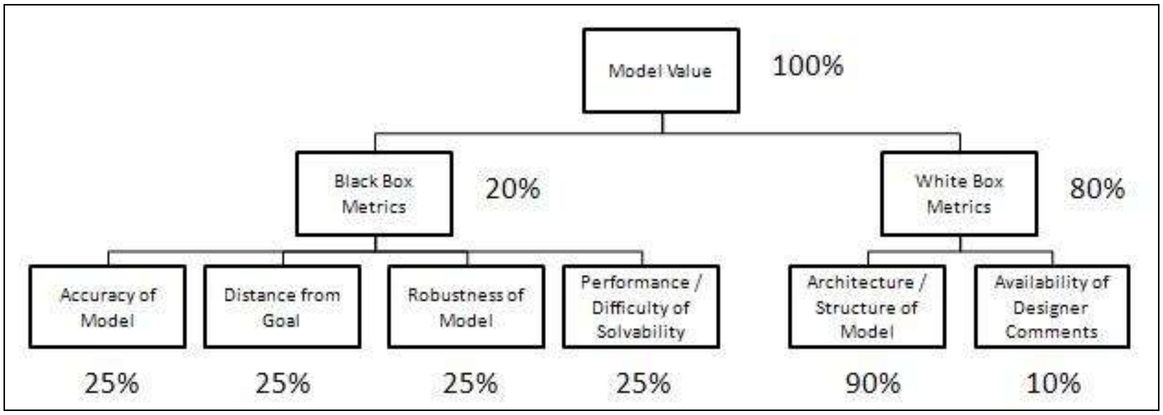


Figure 35. Sample Decision Model With Weights Chosen by a Technically-Savvy User

Note: the sample decision model displayed in Figure 35 shows that this user attaches the most importance to *white box* criteria (80% weight) i.e. based on information extracted from within the models, and resulting from close model investigations conducted by the user performing the model selection. Further, as seen in Figure 35, this criteria is broken down into two sub-criteria with high importance attached to *Architecture/Structure of Model*, which designates that this user examines the internal details of a model (layout, constraints, objective function, etc.) and the score for this examination has the biggest impact on model selections (90% weight).

The user then examines each of the models presented by the system and assigns a score to each alternative (model) based on each criterion. The system then calculates the weighted scores for each model and displays them ordered from highest score to lowest.

In summary, the three phases delineated above describe a process by which spreadsheet models can be stored in a manner conducive to automated comparison, and facilitate the process of selecting a model based on user needs. The following section elaborates the process by which the above method is validated.

Model Validation Process

The validity of the proposed model selection method was demonstrated by developing a simulated prototype system which met the design objectives. Chapter 4 provides two different scenarios/uses of the model comparison and selection system. This section 1) provides a description of the prototype environment, and 2) shows a mapping and linkage between the design phases and the individual design objectives.

Description of Prototype

The prototype for this dissertation consisted of three major components: 1) the prototype environment, 2) the models used i.e. actual examples used within the prototype environment, 3) the process used for the prototype, which finally led to model selection.

1. *The Prototype Environment:* Iyer et al. (2005) describe a Virtual Business Environment (VBE) in which decision models can be instantiated and executed. According to the same source, such a VBE would consist of three major logical components: 1) *Domain Resources* where models, data, and related information are stored, 2) *Engine Manager* which facilitates combining models and associated data and 3) *Dialog Manager* which presents the model in a manner appropriate for the decision maker's needs. Figure 36 shows the components of a VBE.

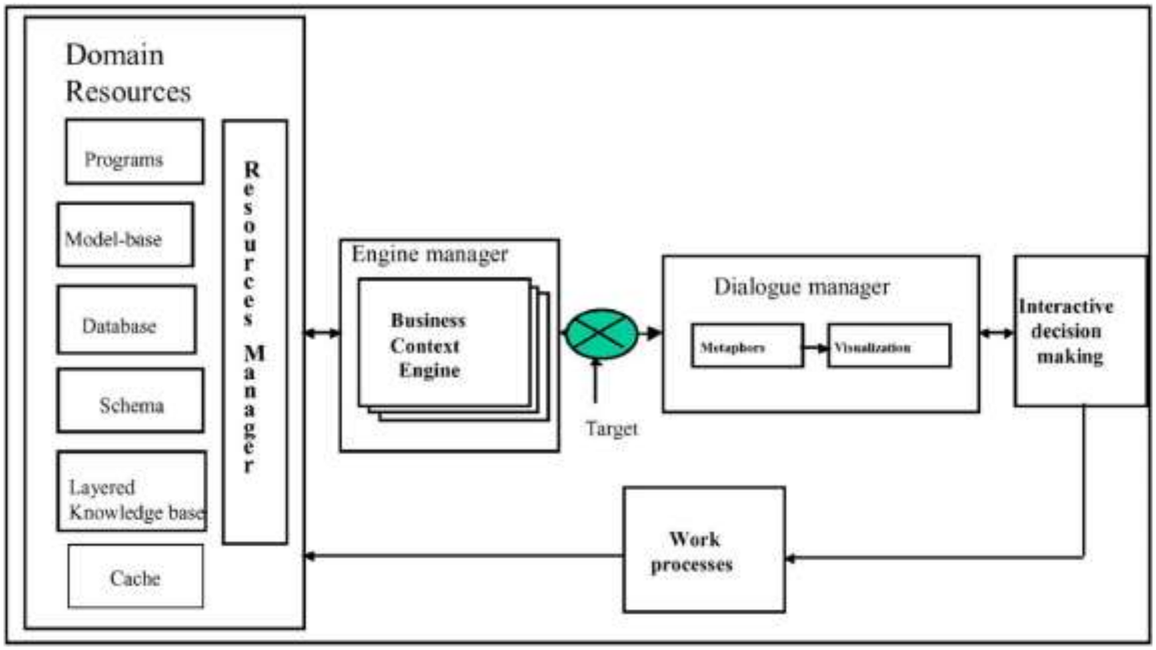
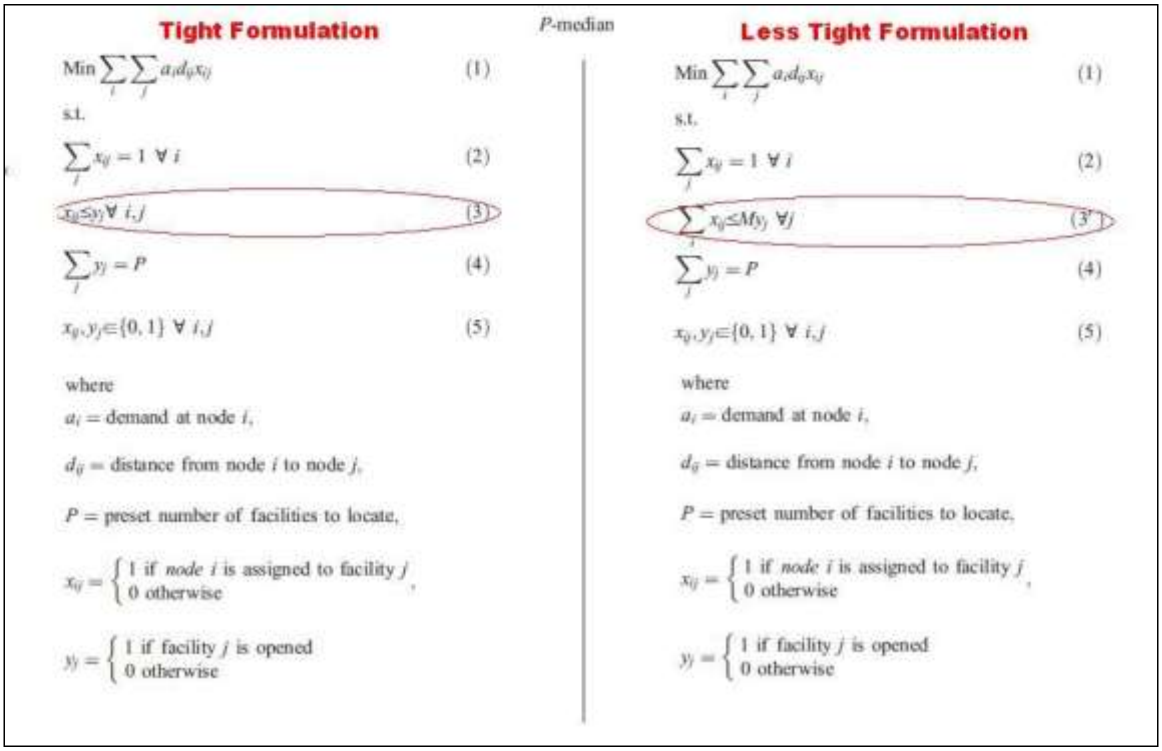


Figure 36. Conceptual View of the Virtual Business Environment (Iyer et al., 2005)

This dissertation simulated the VBE described by Iyer et al. (2005) and showed the instantiation and implementation of the features needed to resolve the model selection task. It showed all the steps necessary to implement the actual models described in the following section.

2. *Prototype Decision Models:* For the purposes of the prototype, let us suppose that a manager needs to make a decision concerning the location and the number of warehouses, which serves different geographically located clients. Such a problem can be analyzed using variations of the standard p-median model. Barkhi et al. (2005) show two different mathematical programming formulations of the p-median problem as reproduced in Figure 37. The only difference between the left and right model in Figure 37 is in the third constraint; constraint 3 (in the left model) is tighter than constraint 3’ (in the right model) because M is a large positive constant.



P-median

Tight Formulation

$$\text{Min} \sum_i \sum_j a_i d_{ij} x_{ij} \quad (1)$$

s.t.

$$\sum_j x_{ij} = 1 \ \forall\, i \quad (2)$$

$$x_{ij} \leq y_j \forall\, i, j \quad (3)$$

$$\sum_j y_j = P \quad (4)$$

$$x_{ij}, y_j \in \{0, 1\} \ \forall\, i, j \quad (5)$$

where

a_i = demand at node i,

d_{ij} = distance from node i to node j,

P = preset number of facilities to locate,

$$x_{ij} = \begin{cases} 1 \text{ if } \textit{node } i \text{ is assigned to facility } j \\ 0 \text{ otherwise} \end{cases},$$

$$y_j = \begin{cases} 1 \text{ if facility } j \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$$

Less Tight Formulation

$$\text{Min} \sum_i \sum_j a_i d_{ij} x_{ij} \quad (1)$$

s.t.

$$\sum_j x_{ij} = 1 \ \forall\, i \quad (2)$$

$$\sum_i x_{ij} \leq M y_j \ \forall j \quad (3')$$

$$\sum_j y_j = P \quad (4)$$

$$x_{ij}, y_j \in \{0, 1\} \ \forall\, i, j \quad (5)$$

where

a_i = demand at node i,

d_{ij} = distance from node i to node j,

P = preset number of facilities to locate,

$$x_{ij} = \begin{cases} 1 \text{ if } \textit{node } i \text{ is assigned to facility } j \\ 0 \text{ otherwise} \end{cases},$$

$$y_j = \begin{cases} 1 \text{ if facility } j \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$$

Figure 37. Two Formulations of the P-Median Problem Reproduced From Barkhi et al. (2005)

These formulations were the basis for creating the prototype models for this dissertation. Each formulation was represented in various spreadsheet models. Also, each formulation was solved using three different solvers/algorithms: 1) traditional linear/integer programming (Excel Solver and Lindo What's Best) and 2) genetic algorithm (Palisade Evolver). Appendix C and Appendix D show the models in spreadsheet format.

3. *The Process of the Prototype:* This section describes the steps that were necessary to implement the three phases described earlier and shows the mechanisms and output required using the models discussed in the previous section. In summary, it shows: the

creation of spreadsheet models based on the two formulations as discussed in the previous section; conversion of these models to their FRL schema model and storing them in a VBE; showing the mapping/linking of various models' terminology; automatically instantiate these models and execute them within the VBE, storing relevant model metadata information; using the model metadata information in conjunction with an AHP based model selection module; and finally allowing analysis of models by end users and selecting the most appropriate one. The details of using the prototype and its findings are discussed in Chapter 4.

In summary, this section described the three components of the prototype which were created for this dissertation and which validated the prescribed model selection method.

Meeting Design Objectives

This dissertation ensured meeting the design objectives stated earlier in Chapter 3. Table 2 shows a mapping between the design phases and the design objectives. The numbers following each design objective in Table 2 denote each objective's sequence number as listed earlier in this chapter.

Table 2. Design Phases and Corresponding Objectives Addressed

Design Phase	Design Phase Description	Design Objectives Addressed
1	Select an appropriate way to represent and store spreadsheet models	Easy-to-use (1) Model Creation (8) Model Modification (7)
2	Identify the appropriate metadata about the models	Sufficient Information (2) Model Inspection (3)
3	Develop a suitable model selection strategy	Easy-to-use (1) Sufficient Information (2) Model Inspection (3) Adaptable (4) Model Comparison (5) Model Ranking (6)

By showing a prototype of the proposed model selection method and its adherence to the design objectives, the validity of the prescribed model selection method was established.

Summary

Carrying out the steps delineated in this chapter, producing the deliverables, and showing a simulated prototype which validates the design objectives achieved the goal of this dissertation. Namely, it facilitated the comparison of decision models by users with differing levels of knowledge, and helped with the selection of an appropriate model.

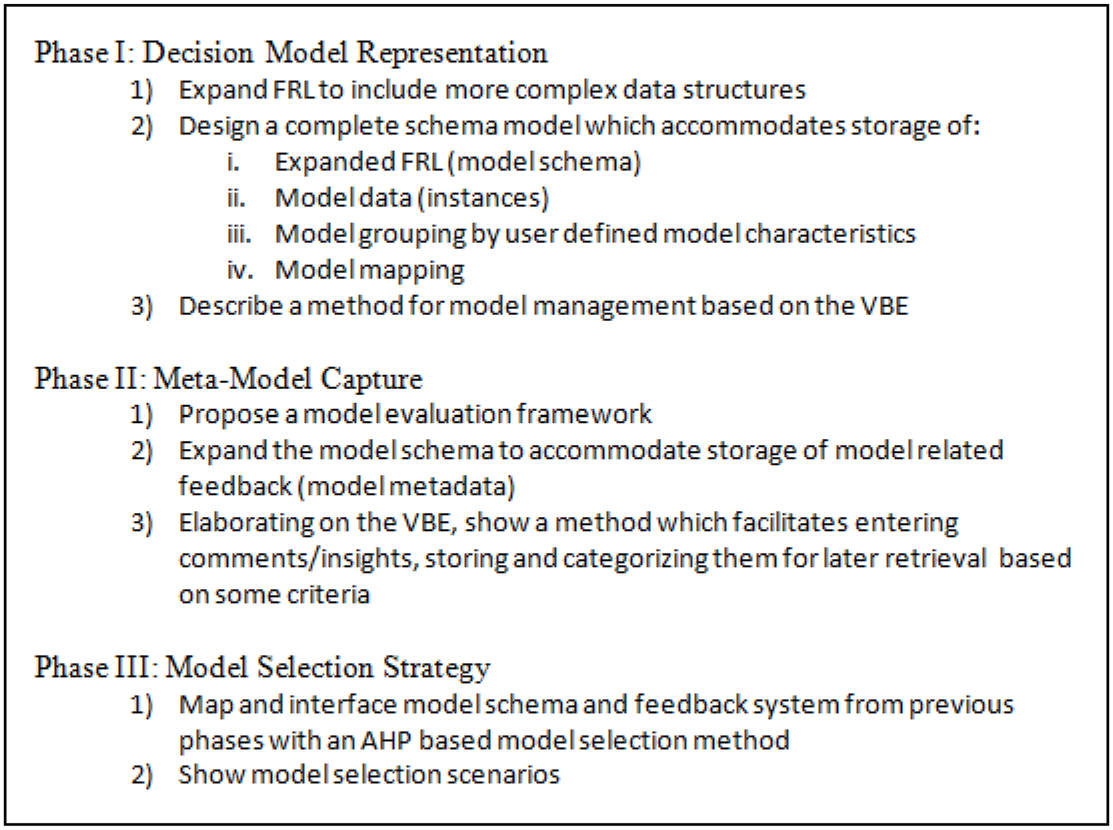
Chapter 4

Discussion of Research Results

Introduction

The main goal of this dissertation was to provide an improved model selection method which caters to environments where spreadsheet-based models are used. Chapter 3 laid out eight general design objectives which guided the creation of such a model selection method. These design objectives stipulated that the prescribed method should include 1) model creation, 2) modification, 3) inspection, 4) comparison and 5) ranking facilities which accommodate spreadsheet based models. From a user experience perspective, the design objectives stipulated that the model selection method should be 6) adaptable to users' knowledge level (i.e. allows model selection even if the user is not an expert or has limited amount of knowledge), 7) one that provides sufficient information to facilitate and aid in model comparison and selection tasks, and 8) easy-to-use i.e. does not require the user to memorize commands or information, therefore employs a graphical user interface.

In order to achieve the research goals and to comply with the design objectives, the model selection method developed in this dissertation 1) adapted the overall architecture of the Virtual Business Environment (VBE) as described by Iyer et al. (2005) (see Figure 36) and 2) designed additional features to the VBE and implemented their internal representations by following a three phase solution approach: I) representing models, II) capturing the relevant metadata about the models, and III) supporting model selection. Figure 38 provides a summary of the features implemented within each phase. Chapter 3 described the concepts behind the steps of each phase in detail and illustrated them with various examples.



Phase I: Decision Model Representation

1) Expand FRL to include more complex data structures
2) Design a complete schema model which accommodates storage of:
    i. Expanded FRL (model schema)
    ii. Model data (instances)
    iii. Model grouping by user defined model characteristics
    iv. Model mapping
3) Describe a method for model management based on the VBE

Phase II: Meta-Model Capture

1) Propose a model evaluation framework
2) Expand the model schema to accommodate storage of model related feedback (model metadata)
3) Elaborating on the VBE, show a method which facilitates entering comments/insights, storing and categorizing them for later retrieval based on some criteria

Phase III: Model Selection Strategy

1) Map and interface model schema and feedback system from previous phases with an AHP based model selection method
2) Show model selection scenarios

Figure 38. Model Selection Solution Approach

In order to demonstrate and validate that the proposed solution meets the objectives of this study, this chapter presents two typical usage scenarios for the model selection process. The first scenario instantiates the features listed within phases I and II as shown in Figure 38, while the second scenario instantiates the features of phase III.

The remainder of this chapter describes these two scenarios, instantiates them, and analyzes the findings from each, while showing linkage between the proposed model selection solution approach and the design objectives.

Scenario One

Scenario One illustrates the instantiation and implementation of the features of phases I (decision model representation) and II (meta-model capture) of the solution approach shown in Figure 38.

Scenario Description

Microsoft Excel based binary integer optimization models seem to yield different results based on the values which exist in the binary tables at the start. A user (analyst) would like to investigate the impact of initial values in these binary tables and would like to experiment with different possible solvers. After some search in the Virtual Business Environment (VBE) for a potential model, two are found; one based on the tight formulation of the p-median problem, and the other based on the loose formulation. These formulations were described in Chapter 3 (see Figure 37). Appendix C and Appendix D show these two formulations as represented in Excel spreadsheets and set up to solve the well known warehouse location problem.

When these two models were initially submitted to the VBE, the factorization algorithm (see Appendix A) converted them to the FRL format before storing them in the SQL based database of the VBE. The FRL converted formats of both models are shown in Appendix E and Appendix F. These models use the expanded FRL format as required in step 1 of the solution approach shown in Figure 38. Appendix M shows a report retrieved from the SQL database of the prototype containing the data portion (instance) of the models as required in step *ii* of the solution approach shown in Figure 38.

Let us assume that neither of the Scenario One spreadsheet models contains any additional information other than just being submitted to the VBE by their initial creators i.e. no comments or metadata were added. Therefore, from the perspective of the VBE, these two models are completely independent and in order to compare them, they first need to be manually associated to each other. This task is addressed by step *iv* of the solution approach shown in Figure 38 i.e. the analyst must invoke the *model mapper* feature of the VBE to map the inputs/outputs of both models so that these two models can be driven by a same set of starting data values and solver parameters. The mechanics of the model mapping feature are described in Chapter 3 and Appendix I shows an instantiation for the Scenario One models.

For this experiment, let us further assume that the analyst decides to set up the two models for three warehouses, four customers, and sets the value of P to two (i.e. one of the warehouses will be shut down leaving only two open). In order to allow the execution of models with different sets of data, the VBE provides the option to populate some of the data such as the customer locations, customer demands, and warehouse locations with randomly generated numbers.

The user directs the VBE to run the two optimization problem formulations three times as follows: 1) the first time by setting all starting values in the binary tables to zero, 2) the second time setting them all to one, and 3) the third time setting these binary tables randomly to either ones or zeroes. The user also stipulates that each of these batches be run using the three already available solvers on the VBE: Excel Solver, Lindo's What's Best, and Palisade Evolver.

The VBE detects that one additional parameter needs to be provided: the loosely formulated model of the problem requires setting a value for M, which is a constant and its use makes a model less constrained. The user asks the VBE to try all values in the range of two and five for each of the batch/solver combinations.

Most solvers include a feature where users can control the amount of interaction and feedback shown (silent or verbose mode). In the silent mode, the user only sees the final results of the experiments i.e. a list of available tabular output generated as a result of running the experiment which may include information such as execution time, optimal values, solver feedback etc. In the verbose mode, the VBE allows the user to interact with each solver/execution one at a time, depending on the available features provided by each solver. This interaction could involve setting/adjusting initial solver parameters; controlling solver execution suspension/interruption as availability of features is provided by each solver; and viewing execution progress status and final reports as generated by each. The VBE can be programmed to communicate with the different solvers and to set the initial parameters for each.

Once all the VBE setup steps are performed, the underlying *synthesis* algorithm reconstitutes the spreadsheet model from internal FRL format as described in Appendix A (see Appendix E for the tight formulation and Appendix F for the loose formulation; Appendix G and Appendix H show these formulations as stored within the SQL database of the prototype), passes all the required parameters for each model, runs each experiment one at a time, allows the user to view the internal execution of each model/solver combination, and allows viewing of final results.

One major benefit of the *factorization/synthesis* algorithm (see Appendix A) provided by Isakowitz et al. (1995) is that even if the models were originally submitted to the VBE with each having a different set of input data requirements, the VBE can recreate them so that they require the same set of data. For example, if a model was originally created to accept nine warehouses and 22 customers as input, using the model factorization/synthesis process, the VBE can automatically reconfigure such a model to run for any number of warehouses and customers, without the manual intervention of a modeler. This feature alone can save the user countless hours of model adjustment and debugging.

Experiment Setup

Simulating the execution of experiments required for Scenario One necessitated the creation of 45 different Excel spreadsheet files each representing one case of the user requirements i.e. nine spreadsheet files for the tight formulation and 36 for the loose formulation. The tight formulation files were equally divided into the three solvers (Excel Solver, Lindo What's Best, and Palisade Evolver), and for each solver three files to represent the following: 1) the starting values of the binary tables all set to zeroes, 2) all ones, and 3) randomly assigned with zeroes and ones. The loose formulation models were also divided into the same groupings, with the addition of four spreadsheet models for each group where M is set to different values ranging from two to five.

The execution related metadata about these 45 individual experiments were manually organized and inserted into the prototype's model instance SQL based tables (InstanceHdr and InstanceDtl) as described in Chapter 3, simulating the log which would be generated from a fully developed and functional VBE. Appendix M shows the internal storage format of such instance data. However, since such internal instance data representation is not suitable for intuitive and quick analyses, this data was tabulated in Excel format and it is shown in Appendix N. Appendix O shows an actual instance of grouping the models (feature *iii* of the solution approach shown in Figure 38).

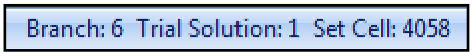
All these simulated experiments of the VBE environment were conducted using an Acer Aspire One computer with Intel Atom 1.2GHz CPU, 2GB RAM, running the Windows7 operating system, and using Microsoft Excel 2007. All solvers (Excel Solver, Lindo What's Best, and Palisade Evolver) were installed and executed with standard default parameters.

Findings and Analysis of Scenario One

The wealth of information generated during the execution of Scenario One represents the source for the feedback required in future model evaluations and comparisons. Therefore, the descriptive and analytic information generated and collected in this section will address the needs of the phase II features of the solution approach as shown in Figure 38.

Experiment Execution Descriptive Details

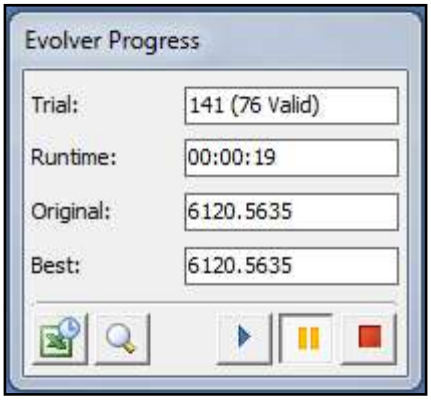
The experiments showed that the Excel Solver provides the least amount of control once execution starts. The progress display is limited to a very small section (see Figure 39) and shows the bare minimum. There is no provision to temporarily suspend execution once execution of the solver starts. By the same token, the final report contains minimal information (see Appendix I) and does not even provide basic information such as the total run time of the solver execution.



```
Branch: 6  Trial Solution: 1  Set Cell: 4058
```

Figure 39. Excel Solver Runtime Output Display

The Palisade Evolver solver provides the most runtime information and control. When execution starts, a minimized window is displayed (see Figure 40) which when maximized contains multiple tabs each showing a wealth of graphical and tabular runtime statistics (see Figure 41). The user is given the control to suspend/continue the execution of the experiments at will.



Evolver Progress

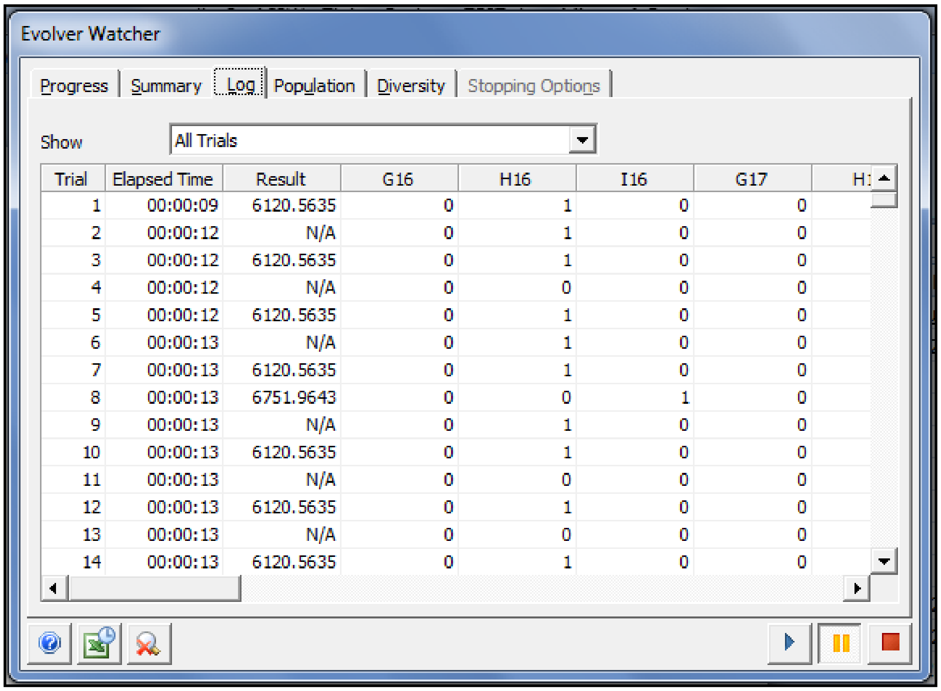
Trial: 141 (76 Valid)

Runtime: 00:00:19

Original: 6120.5635

Best: 6120.5635

Figure 40. Evolver Solver Runtime Minimized Output Display



Evolver Watcher

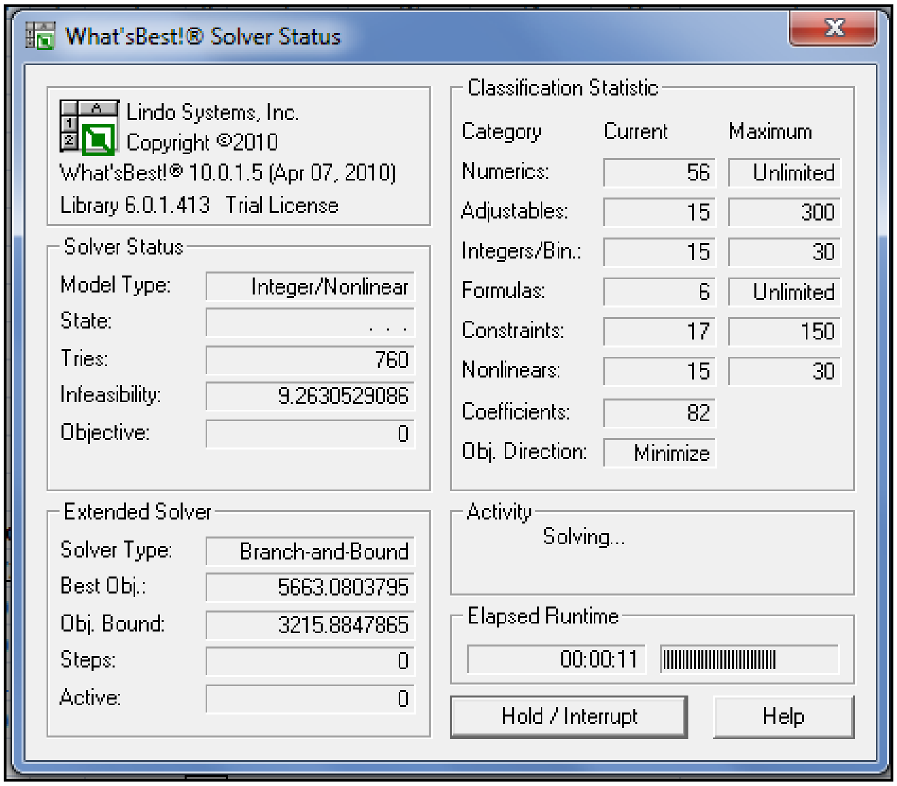
Show: All Trials

Trial	Elapsed Time	Result	G16	H16	I16	G17	H1
1	00:00:09	6120.5635	0	1	0	0	
2	00:00:12	N/A	0	1	0	0	
3	00:00:12	6120.5635	0	1	0	0	
4	00:00:12	N/A	0	0	0	0	
5	00:00:12	6120.5635	0	1	0	0	
6	00:00:13	N/A	0	1	0	0	
7	00:00:13	6120.5635	0	1	0	0	
8	00:00:13	6751.9643	0	0	1	0	
9	00:00:13	N/A	0	1	0	0	
10	00:00:13	6120.5635	0	1	0	0	
11	00:00:13	N/A	0	0	0	0	
12	00:00:13	6120.5635	0	1	0	0	
13	00:00:13	N/A	0	0	0	0	
14	00:00:13	6120.5635	0	1	0	0	

Figure 41. Evolver Solver Runtime Maximized Output Display

The final report generated by Evolver contains the same details as seen on the execution progress monitors. Appendix K provides a sample of the final report.

The Lindo What's Best (WB) solver provides only one window which shows details of execution progress (see Figure 42) but does not provide zoom capabilities in the form of tabs like Evolver. It allows suspension and continuation of execution. The final report generated by WB is the most readable and provides textual feedback in plain language. Appendix L provides a sample output of the report generated by WB.



Lindo Systems, Inc.
Copyright ©2010
What'sBest!® 10.0.1.5 (Apr 07, 2010)
Library 6.0.1.413 Trial License

Solver Status

Model Type: Integer/Nonlinear
State: . . .
Tries: 760
Infeasibility: 9.2630529086
Objective: 0

Extended Solver

Solver Type: Branch-and-Bound
Best Obj.: 5663.0803795
Obj. Bound: 3215.8847865
Steps: 0
Active: 0

Classification Statistic

Category	Current	Maximum
Numerics:	56	Unlimited
Adjustables:	15	300
Integers/Bin.:	15	30
Formulas:	6	Unlimited
Constraints:	17	150
Nonlinears:	15	30
Coefficients:	82	
Obj. Direction:	Minimize	

Activity

Solving...

Elapsed Runtime

00:00:11

Figure 42. What's Best Solver Runtime Output Display

Table 3 shows a summary of the findings concerning the runtime environment, progress status, execution control and final reports for each solver.

Table 3. Summary of Solver Features

Solver	Progress Status	Execution Control	Final Report
Excel Solver	Minimal	No control	Minimal
Lindo What's Best	Intermediate	Suspend/Continue	Comprehensive with Plain English Descriptions
Palisade Evolver	Detailed	Suspend/Continue	Comprehensive without Plain English Descriptions

Analysis of the Experiment Computational Results

As described in the experiment setup section, the execution-related metadata about the 45 individual spreadsheet experiments were manually organized and inserted into the prototype's model instance SQL based tables (InstanceHdr and InstanceDtl), simulating the log which would be generated from a fully developed and functional VBE. Appendix M shows the internal storage format of such instance data and Appendix N shows the same in a more intuitive Excel format, suitable for intuitive and quick analyses. An analysis of this execution-related data follows.

Table 4 shows a summary of the duration related information for the various model/solver executions. This table shows that the lowest execution time was 2.1 seconds and that was for the tight formulation using the Excel Solver. However, upon further investigation, it can be found that although this figure is the lowest, its optimal value of 4495 is below the acceptable optimal value which should be 5663. This figure was obtained by violating some of the binary constraints. It would be more appropriate to discard this value and keep the correct optimal value as provided by another instance i.e. from Appendix N, the row with ID= 11, which is the Loose formulation using Excel Solver, running within 4.3 seconds.

In general, the Excel Solver shows the lowest execution times with means of 7.2 seconds for the loose formulation and 6.9 seconds for the tight formulation.

Table 4. Solver Execution Duration Comparisons

Model Name	**Version**	**Solver**	**Min Exec Duration**	**Max Exec Duration**	**Avg Exec Duration**	**Duration Spread**
p-median3 - 4C3W	Loose	Evolver	9	19	13.8	10.0
p-median3 - 4C3W	Loose	Excel Solver	4.3	12.8	7.2	8.5
p-median3 - 4C3W	Loose	WB	51	648	360.6	597.0
p-median3 - 4C3W	Tight	Evolver	10	14	12.7	4.0
p-median3 - 4C3W	Tight	Excel Solver	2.1	10.1	6.9	8.0
p-median3 - 4C3W	Tight	WB	97	417	209.7	320.0

* all duration figures are in seconds

Table 4 also shows that the WB has the slowest execution time ranging from a minimum of 97 seconds for the tight formulation and up to 648 seconds for the maximum runtime for the loose formulation. These numbers represent roughly a 10 to 65 times in orders of magnitude compared to the lowest running times. As the models' input count increases, this performance could represent a serious drawback for big size problems.

Table 5 shows a comparison of the optimal values found by each set of models. Namely, it shows the minimum, maximum, and average optimal values found. The column which shows the variability in optimal values (Optimal Value Spread) contains the most insight. It shows that the tight formulation using WB is the most consistent with a variability of zero. From the standpoint of reliability, this would be the most reliable. Evolver seems to be the least reliable since the *tight* formulated problem never found the optimal value of 5663. Also, the variability of the loose formulation using Evolver is the highest with 1640 points. The tight formulation problem using Evolver has the second highest variability with 1168 points of difference between the lowest and highest optimal values.

Table 5. Solver Generated Optimal Value Comparisons

Model Name	**Version**	**Solver**	**Min Optimal Value**	**Max Optimal Value**	**Avg Optimal Value**	**Optimal Value Spread**
p-median3 - 4C3W	Loose	Evolver	5663	7303	6386	1640.0
p-median3 - 4C3W	Loose	Excel Solver	5663	6121	5794	458.0
p-median3 - 4C3W	Loose	WB	5663	5852	5710	189.0
p-median3 - 4C3W	Tight	Evolver	6091	7303	6505	1212.0
p-median3 - 4C3W	Tight	Excel Solver	4495	5663	5274	1168.0
p-median3 - 4C3W	Tight	WB	5663	5663	5663	0.0

Table 6 shows that What's Best found the correct optimal value 80% of the times, while Evolver found it only 7% of the experiments. Although Excel Solver found the optimal value 53% of the times, it is important to note that sometimes examining the binary tables showed that they contain values other than binary. In case Excel Solver is used, the user will need to examine and verify that these values remain binary, therefore no constraints are violated.

Table 6. Optimal Values Found per Solver

Model Name	**Solver**	**Count Optimal Found**	**% Optimal Found**
p-median3 - 4C3W	Evolver	1	7%
p-median3 - 4C3W	Excel Solver	8	53%
p-median3 - 4C3W	WB	12	80%

The loose formulation of the p-median problem requires a value for the constant M. In the literature review about the value of this constant, it is often stated that it should be set to be *large*, but there is no indication as to what constitutes *large*. The experiments show that low values of M could drastically impact the probability of finding optimal values. Table 7 shows that no optimal values were found when M was set to 2. From observation it seems that the closer M gets to the total number of warehouses, the higher the count of optimal values found. Therefore, based on this experiment and as a general rule, M should be set as close to the number of warehouses as possible.

Table 7. The Impact of the Value of M and the Optimal Values Found

Model Name	Value of M	Count of Found Optimal	% Optimal Found
p-median3 - 4C3W	2	0	0%
p-median3 - 4C3W	3	4	44%
p-median3 - 4C3W	4	6	67%
p-median3 - 4C3W	5	6	67%

Table 8 shows the count of found optimal solutions while setting the values of the starting values in binary tables to all zeros, or all ones, or a combination of both zeroes and ones. Although these numbers are not conclusive since the counts are too close to each other, it would be wise to start the binary values with zeros as opposed to setting them all to ones or to randomly setting them to zeros or ones.

Table 8. The Impact on the Optimal Values of the Starting Values in Binary Tables

Starting Values of Bin Tables	Count of Found Optimal
Zeros	8
Ones	7
Randomly Zeros and Ones	6

Storage of Experiment Insights

At the conclusion of the experiments in Scenario One as just described, the insights generated must be recorded in the VBE in order to make them available to future users.

Appendix O shows the conversion of these insights into the internal format of the prototype system as represented by the VBE. This step satisfies the features of Phase II of the solution approach as required in Figure 38. These same insights are later made available for consultation by other users. More specifically, these insights are used in Scenario Two to show how a user can rely on them to aid in the model selection decision.

Reconciling Solution Approach and Scenario One

Scenario One showed the instantiation of two spreadsheet models based on the p-median problem. Through the series of experiments and while varying the input parameters, all the non-trivial aspects of the features required for phases I and II were shown. The following re-iterates these linkages for each of these phases:

Phase I Features: Represent Models

The expanded FRL specification was used to factorize/synthesize the spreadsheet models shown in Appendix E and Appendix F. The model instance internal structure and data was shown in Appendix M and a simpler format version was shown in Appendix N. Appendix O showed the model grouping process. Appendix I showed the model mapping process.

Phase II Features: Capture Meta-Model

The insights generated during the execution of Scenario One were organized and the summary was shown in Appendix P. This process elaborated on the phase II requirements of the solution approach as shown in figure 38.

Scenario Two

Scenario Two illustrates the instantiation and implementation of the features of phase III of the solution approach shown in Figure 38 i.e. “devise model selection strategy”. This section provides a description of Scenario Two, an analysis of the process, and shows linkages to the solution approach.

Scenario Description

A manager who oversees the distribution of products from three islands (warehouses) to four different islands (customers) needs to shut down one of the warehouses, thus keeping only two open. He searches the corporate intranet for spreadsheet models which could help him in his decision. He finds access to the VBE and after a few searches, he finds the two model formulations as described in Scenario One, along with the insights as entered in the same scenario. He also finds out that these formulations can be executed using three different solvers (Excel, What’s Best, and Evolver). The combination of formulations and solvers presents six different alternatives from which the manager needs to select the appropriate decision model. The AHP based model selection component of the VBE will help in this decision situation. The following section describes this AHP based model selection process.

Process and Outcome of Scenario Two

As described in Chapter 3, the VBE facilitates the model selection process by presenting the user with an AHP based interface which organizes prior user or system generated insights into different categories in order to simplify the decision process. It walks the user through each category, allows analysis of content, and prompts for various feedback concerning 1) the importance given to each criteria grouping and 2) specific point values assigned to each alternative considered.

Selection Criteria and Preference Weights

The VBE first presents the full list of feedback categories as shown in Figure 27 and requests the user to eliminate those categories that are not relevant to the task. Alternatively, the VBE could guide the decision making process based on those categories for which prior insights already exist within the system. In this case, as shown in Appendix P, the categories for which prior insight exists are: 1) accuracy of model, 2) trust in model, 3) input needs, 4) performance, 5) cost, 6) robustness of model, 7) designer comments, and 8) a last category that the previous user has created: ‘other: control and feedback’.

First, this list of criteria must be assigned importance weights according to the user’s preferences. One method is to just assign weight percentages. For example, after an initial examination, the user may decide to prioritize the criteria categories as shown in Table 9. This table shows that the user assigns the most importance to the accuracy of the model giving this category a 50% weight of the overall point distribution. Since the user is not concerned with the internal makeup of the models, “designer comments” and controlling the model execution are unimportant, giving them zero weights. The cost of the model is not an issue either, giving it a zero weight. The “robustness of model” is given 20%. The remaining categories have 10% weight each, designating that they are all equally important to the user.

Alternatively, the user may decide to use the pairwise comparison method recommended by the AHP method which facilitates preference weight discovery. This pairwise comparison method is shown in Appendix Q as it applies to Scenario Two. The resulting weights for the problem selection criteria are pretty close to those in Table 9. For simplicity, the rounded values of Table 9 will be used throughout this scenario.

Table 9. Priorities/Importance of Each Criteria Category

Criteria Category	**Preference Weight in %**
Accuracy of Model	50%
Trust in Model	10%
Input Needs	10%
Performance	10%
Cost	0%
Robustness of Model	20%
Designer Comments	0%
Other: Control and feedback	0%

Alternatives and Preference Weights

After completing the calculations of preference weights for each model selection criterion, the same should be performed for each available alternative. Pairwise comparisons could be performed for each pair of alternatives with respect to each criterion just as shown in Appendix P. For this scenario, a simpler scoring method will be used: the user will have to enter scores (one to five, where *one* designates least desirable and *five* designates the most desirable) for each combination of criteria/alternative as shown in Table 10. In this case, the choices are the two formulations (loose and tight), each using one of the three solvers (Excel, WB, and Evolver), yielding six possible alternatives in total. The 'Weight in Total %' column is copied from Table 9 and it represents the importance given to each criteria. The last row of Table 10 titled 'Total

Weighted Points' contains the weighted sum for each alternative's scores i.e. the sum of the product of the scores of each alternative with the weight assigned to each criterion. For example, the total 'Total Weighted Points' for the 'Tight Excel' alternative is equal to three and is calculated as follows:

(50%*3)+(10%*3)+(10%*5)+(10%*5)+(0%*0)+(20%*1)+(0%*0)+(0%*0) = 3

Based on the weights and points assigned to each criteria/alternative combination the alternative which collects the highest score in the 'Total Weighted Points' row of Table 10 is considered to be the best option.

Table 10. Points Assigned to Each Model per Decision Criteria

Criteria Category	**Weight in Total %**	**Tight Excel**	**Tight WB**	**Tight Evolver**	**Loose Excel**	**Loose WB**	**Loose Evolver**
Accuracy of Model	50%	3	5	1	3	5	1
Trust in Model	10%	3	5	0	3	5	0
Input Needs	10%	5	5	5	0	0	0
Performance	10%	5	1	5	5	1	5
Cost	0%	0	0	0	0	0	0
Robustness of Model	20%	1	5	5	1	5	5
Designer Comments	0%	0	0	0	0	0	0
Other: Control and Feedback	0%	0	0	0	0	0	0
Total Weighted Points	**100%**	**3**	**4.6**	**2.5**	**2.5**	**4.1**	**2**

Model Selection Based on Criteria and Alternatives

When the total points assigned to each alternative (i.e. model formulation/solver) are sorted, Table 11 shows that the user should select the tight formulation of the p-median model coupled with Lindo's What's Best (WB) solver since it has the highest ranking with 4.6 points.

Table 11. Model/Solver Ranking

Model/Solver	Ranking Highest to Lowest
Tight WB	4.60
Loose WB	4.10
Tight Excel	3.00
Tight Evolver	2.50
Loose Excel	2.50
Loose Evolver	2.00

The loose formulation of WB earns the second place in ranking with a score of 4.1. The third ranking solution (Tight Excel) is 1.1 points away from the top second (Loose WB) leaving the user with the impression that WB (What's Best) is the best solver given the particular user requirements.

Reconciling Solution Approach and Scenario Two

Scenario Two showed an example where a user has to select a decision model to solve a practical business problem. By relying on prior user insights, and using the AHP based decision making process, the model comparison and selection process was facilitated.

Phase III Features: Model Selection Strategy

As stated in phase III of the solution approach shown in Figure 38, Scenario Two showed 1) a concrete model selection scenario and 2) a mapping of the insights stored within the proposed VBE system to an AHP based multi-criteria decision making process.

Summary

This chapter along with the referenced appendixes presented two typical user scenarios which, combined, addressed and elaborated on all of the steps delineated in the solution approach shown in Figure 38.

The instantiation of the first scenario addressed two different portions of the solution approach: 1) the issues involved in the internal representation of the proposed model i.e. the conversion of spreadsheet models into a format which facilitates searching, inspection, assessment and storage; and 2) the process of capturing and storing metadata which support the comparison and thus the selection of an appropriate decision model.

The instantiation of the second scenario illustrated the application of AHP as a model selection strategy by using all the data and insights generated and stored in the first scenario.

The details revealed in these two scenarios and the analyses of the findings served to prove the viability of the proposed solution (Figure 38) as a sound, end-user based model selection platform that is specifically geared for spreadsheet based models.

Next, the final chapter provides some concluding remarks for this study.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Introduction

This chapter concludes the research study. It starts with some concluding remarks, it discusses the implications of this work, it suggests a few recommendations for future directions, and provides a final summary

Conclusions

As shown in the experiments of Scenario One, when an analyst compares different models, many useful insights are generated. Unless these insights are codified, stored and shared, future potential users of the models will need to start from a blank slate. In most cases, such users will prefer to create their own models rather than to understand what others have created. The proposed solution presents a method where such insights can be stored and presented to potential future users, simplifying the model comparison and selection process. The major limitation of this work was the lack of a fully functioning prototype implementing all suggested features. Such a project would have required the expertise of many highly skilled programmers which was out of scope for this dissertation work. However, this can be remedied with future work.

Implications

Organizational repositories are littered with spreadsheet based decision models created by end users. Lack of reuse of such models and recreating them represent significant waste of time and resources. The elaboration and implementation of the described system can result in considerable gains in productivity. This study represents a good starting point for academic research in the automation of spreadsheet based model selection.

Recommendations

Creating a working prototype which implements the features described in the prototype of this dissertation would require significant programming expertise and resources, and such an undertaking is beyond its scope. A logical extension of this work would be to assemble a team of researchers with the required programming background in order to create a complete working prototype. Such a prototype will undoubtedly uncover areas of work to be addressed.

In another direction, the scope of this project can be expanded to include spreadsheet based model composition/integration. This topic has been covered in past literature but not in the context of end user spreadsheet based model management environments. Based on prior work, this research presented a preliminary framework which addresses the criteria used when comparing and selecting a model. As stated in the literature review section, this list of criteria is preliminary in the sense that it is not the main focus of this study, but rather a small step leading to the overall goal of spreadsheet based model comparison and selection. This framework also presents a good starting point for future research.

Summary

Organizations need to make sound decisions on a continual basis. In many cases these decisions are more quantitative than intuition based. To support such decisions, managers most often create analytical models using spreadsheet tools, they use them for their particular situation, and once done, they store them away along with other files.

Over the years, such models have proliferated and have been archived in organizational secondary storage systems. Without an effective mechanism to locating, using or managing these models, new users end up creating their own models from scratch. This process could be time consuming and error prone.

Providing automated support for model selection resulting in effective reuse of these models could result in significant cost savings and improvements in productivity. However, in practice, model reuse is severely limited by two main challenges: (1) lack of relevant information about the models maintained in the repository, and (2) lack of end user knowledge which prevents them from selecting appropriate models for a given problem solving task.

This study built on the existing model management literature to address these research challenges. It showed a simple spreadsheet model taken from literature and walked through decomposing and converting it to its internal structure using the Functional Relational Language (FRL) based model schema notated in BNF. It expanded the FRL in order to accommodate more complex data structures. It then designed data models and implemented them in a relational database model, and showed the storage mechanism of the internal structures of the data model. It created additional features like mapping and linking models for automatic comparison. It showed the mechanisms for automatic instantiation of stored models and executing them. It devised a method for organizing and storing relevant model runtime metadata information. The retrieval and usage of such metadata information was shown in conjunction with an Analytical Hierarchy Process (AHP) based selection method allowing analysis of models by end users, and selection of the most appropriate one. AHP is an established method for multi-criteria decision-making that is suitable for the model selection task.

To evaluate and validate the proposed method for automated model selection, this study simulated a prototype system that implemented the described method and tested it with two realistic end-user model comparison and selection scenarios based on previously benchmarked test problems. The first scenario involved the task of comparing two existing spreadsheet models based on two formulations of the p-median problem (tight and loose). Each formulation was executed with three different solvers (Excel Solver, Lindo What's Best, and Palisade Evolver) by varying many parameters. The insights generated from these experiments were described and analyzed. They were formatted and stored using the method described in this dissertation. The second scenario built on the first one. It showed the process of using the AHP method in conjunction with retrieved insights from the first case. The second scenario showed the logic used when deciding on the choice of the appropriate model.

Appendixes

Appendix A. Roundtrip Conversion of Spreadsheet to FRL - Factoring and Synthesis

Summary of Figures in Appendix A

1.0 – Introduction

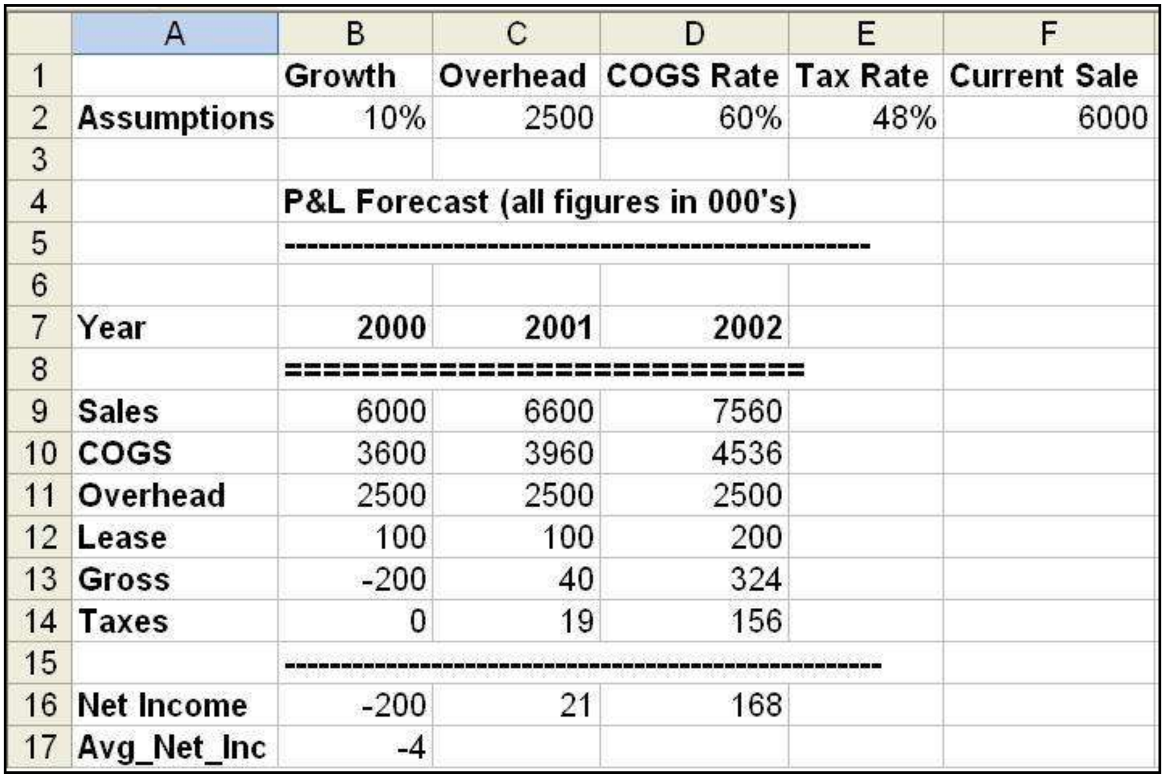
This appendix walks the reader through the steps which convert a spreadsheet model from the spreadsheet format, called its physical representation since it is bound to a specific spreadsheet, to its logical counterpart which is stored in a schema definition language created by Isakowitz et al. (1995) and called Functional / Relational Language (FRL). This step is called *Factoring*. The process of converting the FRL schema to the original spreadsheet model is called *Synthesis*.

The profit and loss spreadsheet that will be used is shown in Figure A-1. This model consists of input values listed in B2:F2. These input values are used in B9:D16 to project sales and income over the next two years, and to estimate the *Net Income*. These intermediary values are then condensed to yield the *Average Net Income* stored in B17. Note: The most complicated and noteworthy values for this model are the calculations of the *sales* values for each *Year*. The *Sales* figure for *Year 2000* is taken from the input value stored in F2, *Current Sale*. The *Year 2001 Sales* figure is obtained by increasing the previous year's sales figure with the *Growth Rate* listed in B2. i.e.

$$Year2001Sales = Year200Sales * (1 + GrowthRate)$$

The *Year 2002 Sales* figure is obtained by averaging the two previous years' sales figures and by increasing this average with double the *Growth Rate* listed in B2. i.e.

$$Year2002Sales = ((Year200Sales + Year2001Sales) / 2) * (1 + (2 * GrowthRate))$$



	A	B	C	D	E	F
1		Growth	Overhead	COGS Rate	Tax Rate	Current Sale
2	Assumptions	10%	2500	60%	48%	6000
3						
4		P&L Forecast (all figures in 000's)				
5		--				
6						
7	Year	2000	2001	2002		
8		============================				
9	Sales	6000	6600	7560		
10	COGS	3600	3960	4536		
11	Overhead	2500	2500	2500		
12	Lease	100	100	200		
13	Gross	-200	40	324		
14	Taxes	0	19	156		
15		--				
16	Net Income	-200	21	168		
17	Avg_Net_Inc	-4				

Figure A-1. Simplified Spreadsheet Model Based on Isakowitz et al. (1995)

1.1 – Spreadsheet to FRL: The Journey Forward (Factoring)

According to Isakowitz (1995), a spreadsheet model is composed of four distinct types of information: 1) *editorial* which refers to any labels or comments; 2) *data* which refers to the actual data values stored in spreadsheet; 3) *schema* which refers to the embedded logic; and 4) *binding* which refers to the physical location of *data* and *schema*.

The following sections present the steps which must be followed when converting a spreadsheet model to its logical representation i.e. the logical model schema in FRL. In the process, the spreadsheet model is decomposed into its four components.

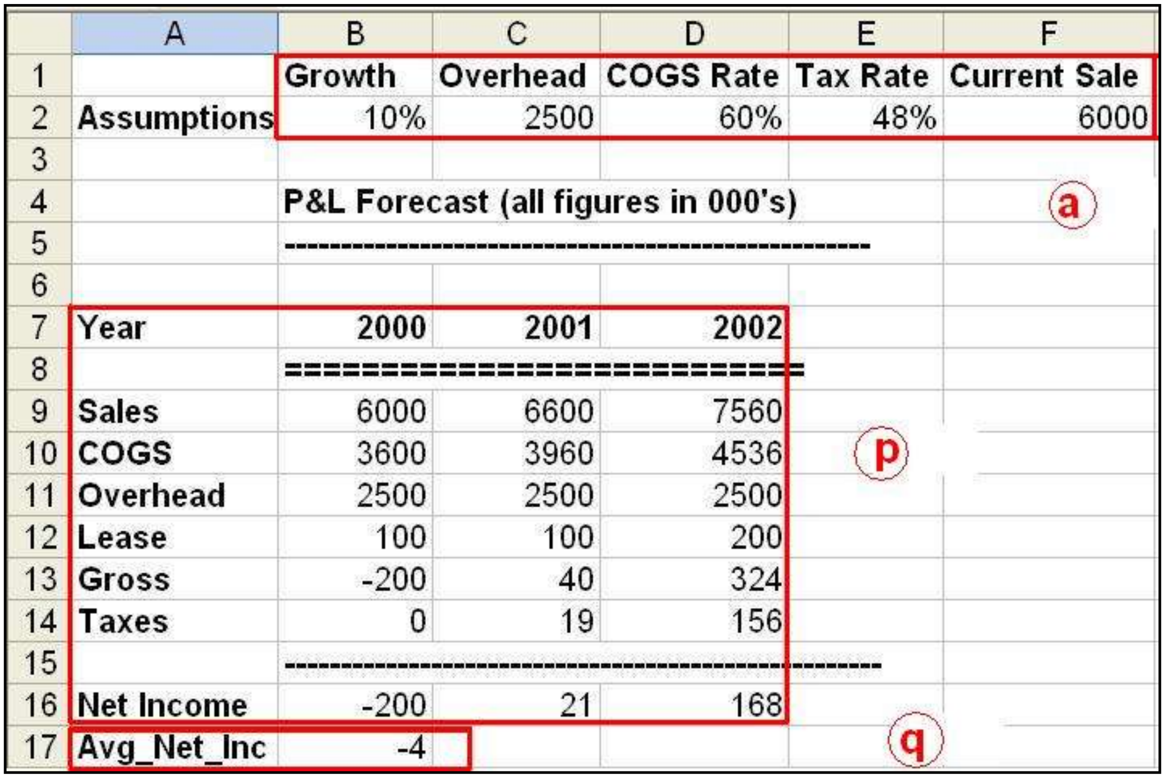
1.1.1 – Outline Groupings of Data

The first step consists of providing users with a tool which helps them in delineating different portions of the spreadsheet model. Basically, each logical grouping of data should be outlined and a name must be assigned to it. As shown in Figure A-2, the user has created three different logical groupings of data. These logical data groupings are referred to as *relations,* a term borrowed from database design.

The first grouping of data is at the top of Figure A-2 and it is named *relation a*. This *relation* is considered a record structure which contains individual fields. A relation with only one record is called a *vector relation*.

The second grouping of data is in the middle of Figure A-2 and it is *relation p*. This *relation* is akin to a table in a relational database and it is indexed by the field *Year* i.e. the additional information for a particular year is obtained just by knowing its index value, in this case the *Year*. It is a noteworthy fact that many fields within this *relation* (table) are dependent on other fields within the same *relation* (table): a clear violation of relational database design.

The last grouping of data is at the bottom of Figure A-2 and it is named *relation q.* This relation is a *vector relation* which is similar in structure to *relation a*.



	A	B	C	D	E	F
1		Growth	Overhead	COGS Rate	Tax Rate	Current Sale
2	Assumptions	10%	2500	60%	48%	6000
3						
4		P&L Forecast (all figures in 000's)				
5		--				
6						
7	Year	2000	2001	2002		
8		=============================				
9	Sales	6000	6600	7560		
10	COGS	3600	3960	4536		
11	Overhead	2500	2500	2500		
12	Lease	100	100	200		
13	Gross	-200	40	324		
14	Taxes	0	19	156		
15		---				
16	Net Income	-200	21	168		
17	Avg_Net_Inc	-4				

Figure A-2. Spreadsheet Model With Logical Sections Delineated

1.1.2 – Create the Annotated Map for the Spreadsheet

Creating the annotated map for the spreadsheet as shown in Figure A-3 (shown in two columns separated by a black line for space convenience) consists of reading each cell from within the spreadsheet and writing it to a sequential file in the following format:

Relation[i].Attribute, CellAddress, Value

Where,

Relation is the name given to the cells when the user outlined each data grouping in step 1.1.1.

[i] is the index which designates the absolute record position of a particular record in the data grouping from step 1.1.1. For example, the attributes grouping (i.e. record) for *Year 2001* are in the second position after *Year 2000*, therefore *i* = 2 for *Year 2001*.

Attribute is the name given to each cell within a *relation*

CellAddress is the physical address of a cell within the spreadsheet

Value refers to the content of the cell at position *CellAddress*

<u>Example 1:</u> See cell range B10:F10 of Figure A-3 which contains the following term:

a[1].ovhead, C2, 2500

Where,

Relation = a;

i = 1; (note: there is only one record in *relation a*, therefore *i* can only be = 1)

Attribute = ovhead;

CellAddress = C2;

Value = 2500

<u>Example 2:</u> See cells E4:F4 of Figure A-3. This example shows editorial information i.e. label or comment. Therefore, it has no *Relation*, *Index* or *Attribute* values.

C1,Overhead

Where,

Relation = none;

i = none;

Attribute = none;

CellAddress = C1;

Value = Overhead

Note: This entry designates a label with ‘Overhead’ stored within it.

Example 3: See cell range K12:O12 of Figure A-3 which contains:

p[3].ovhead, D11, C2

Where,

Relation = p;

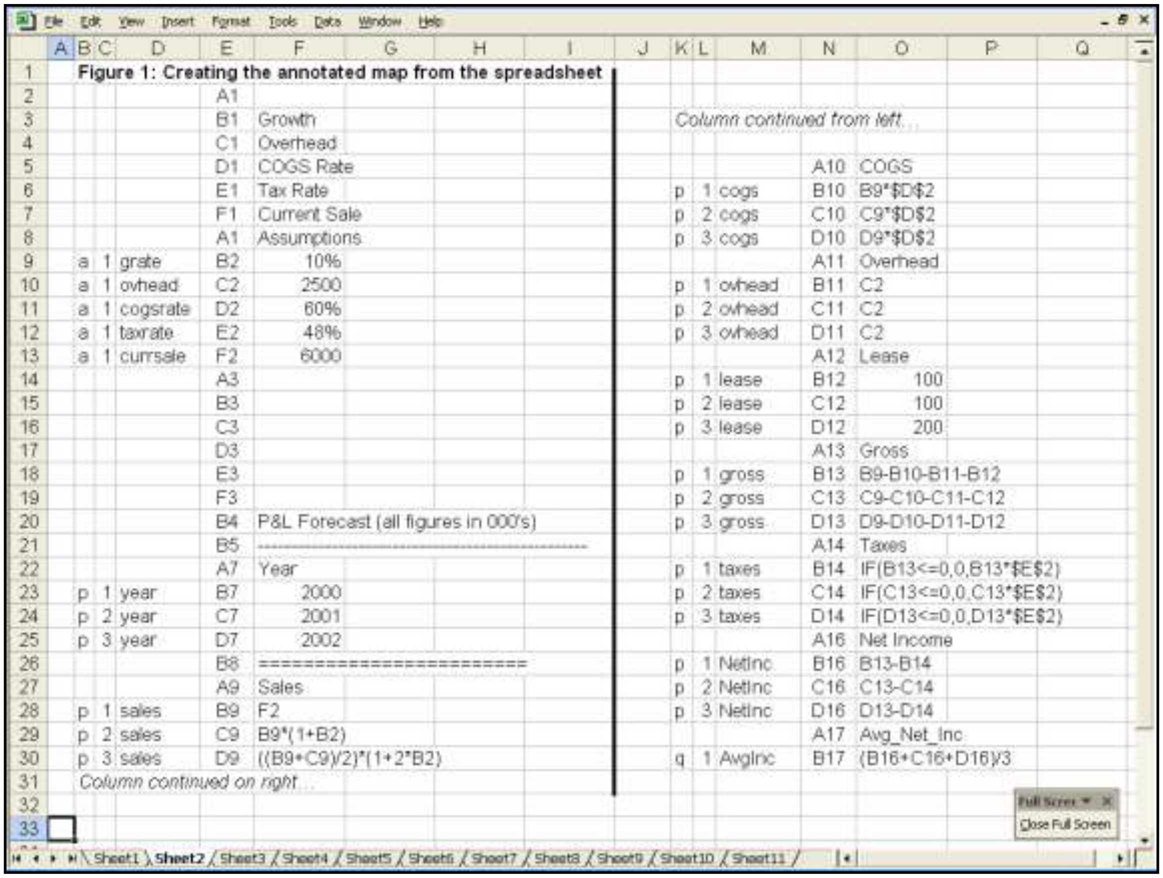
i = 3;

Attribute = ovhead

CellAddress = D11;

Value = C2

Note: This entry designates a field which contains information from another cell, in this case from cell C2.


File Edit View Insert Format Tools Data Window Help
Full Screen
Close Full Screen
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6 Sheet7 Sheet8 Sheet9 Sheet10 Sheet11

	A	B	C	D	E	F	K	L	M	N	O
1		**Figure 1: Creating the annotated map from the spreadsheet**									
2					A1						
3					B1	Growth	*Column continued from left…*				
4					C1	Overhead					
5					D1	COGS Rate				A10	COGS
6					E1	Tax Rate	p	1	cogs	B10	B9*$D$2
7					F1	Current Sale	p	2	cogs	C10	C9*$D$2
8					A1	Assumptions	p	3	cogs	D10	D9*$D$2
9		a	1	grate	B2	10%				A11	Overhead
10		a	1	ovhead	C2	2500	p	1	ovhead	B11	C2
11		a	1	cogsrate	D2	60%	p	2	ovhead	C11	C2
12		a	1	taxrate	E2	48%	p	3	ovhead	D11	C2
13		a	1	currsale	F2	6000				A12	Lease
14					A3		p	1	lease	B12	100
15					B3		p	2	lease	C12	100
16					C3		p	3	lease	D12	200
17					D3					A13	Gross
18					E3		p	1	gross	B13	B9-B10-B11-B12
19					F3		p	2	gross	C13	C9-C10-C11-C12
20					B4	P&L Forecast (all figures in 000's)	p	3	gross	D13	D9-D10-D11-D12
21					B5	--				A14	Taxes
22					A7	Year	p	1	taxes	B14	IF(B13<=0,0,B13*$E$2)
23		p	1	year	B7	2000	p	2	taxes	C14	IF(C13<=0,0,C13*$E$2)
24		p	2	year	C7	2001	p	3	taxes	D14	IF(D13<=0,0,D13*$E$2)
25		p	3	year	D7	2002				A16	Net Income
26					B8	=========================	p	1	NetInc	B16	B13-B14
27					A9	Sales	p	2	NetInc	C16	C13-C14
28		p	1	sales	B9	F2	p	3	NetInc	D16	D13-D14
29		p	2	sales	C9	B9*(1+B2)				A17	Avg_Net_Inc
30		p	3	sales	D9	((B9+C9)/2)*(1+2*B2)	q	1	AvgInc	B17	(B16+C16+D16)/3
31		*Column continued on right…*									
32											
33											

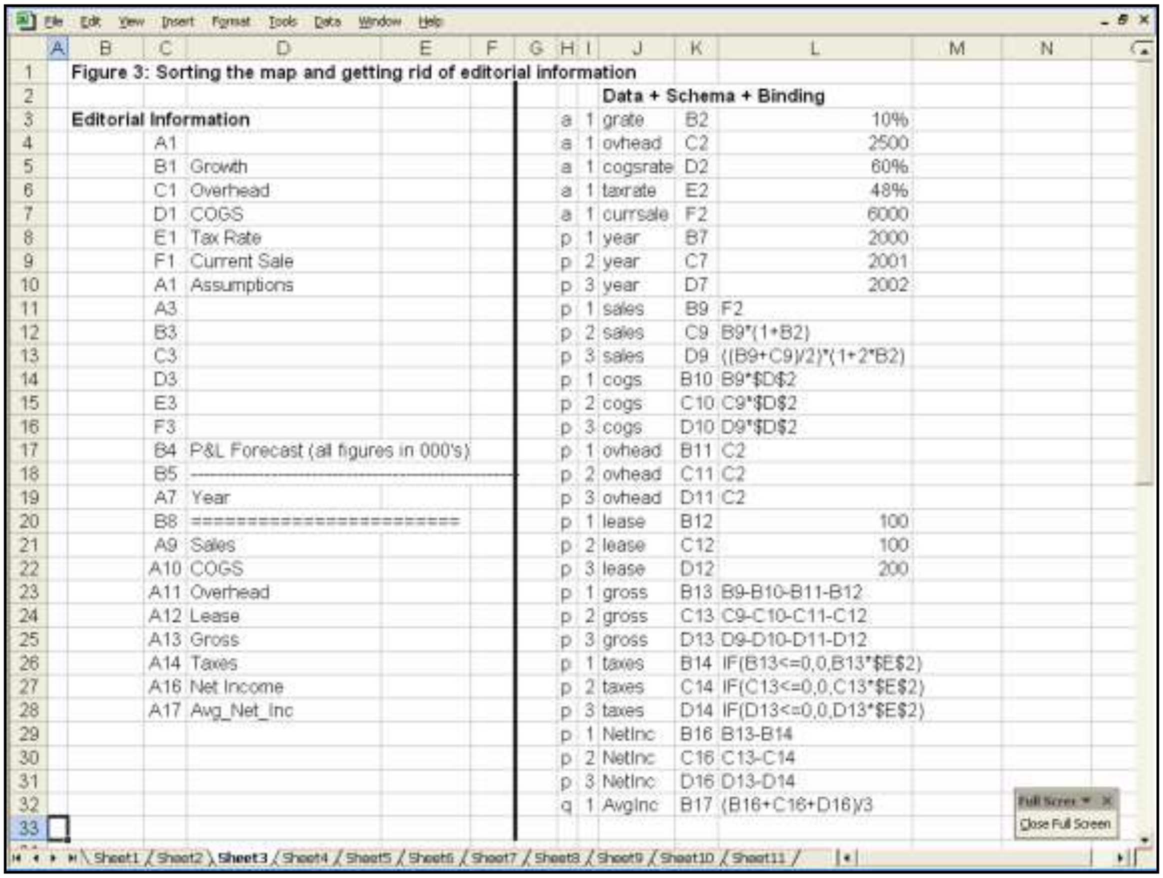
Figure A-3. Annotated Spreadsheet Map

1.1.3 – Separate Editorial Information

The next step involves separating and removing all *editorial* information from the annotated map. Editorial information refers to comments and labels used within a model which enhance the readability of a model, but are not necessary to its workings.

Since *editorial* information do not have values in their *Relation[i].Attribute* column, sorting the annotated alphabetically in ascending order will push these entries to the top of the annotated map list, while the ones with values in these columns will go to the bottom of the list. Figure A-4 shows the annotated map already sorted into two columns:

the column on the left contains editorial information, while the column on the right contains *schema*, *data*, and *binding* information. These two lists are then stored in separate files.



	B	C	D	H	I	J	K	L
1	**Figure 3: Sorting the map and getting rid of editorial information**							
2						**Data + Schema + Binding**		
3	**Editorial Information**			a	1	grate	B2	10%
4		A1		a	1	ovhead	C2	2500
5		B1	Growth	a	1	cogsrate	D2	60%
6		C1	Overhead	a	1	taxrate	E2	48%
7		D1	COGS	a	1	currsale	F2	6000
8		E1	Tax Rate	p	1	year	B7	2000
9		F1	Current Sale	p	2	year	C7	2001
10		A1	Assumptions	p	3	year	D7	2002
11		A3		p	1	sales	B9	F2
12		B3		p	2	sales	C9	B9*(1+B2)
13		C3		p	3	sales	D9	((B9+C9)/2)*(1+2*B2)
14		D3		p	1	cogs	B10	B9*$D$2
15		E3		p	2	cogs	C10	C9*$D$2
16		F3		p	3	cogs	D10	D9*$D$2
17		B4	P&L Forecast (all figures in 000's)	p	1	ovhead	B11	C2
18		B5	--	p	2	ovhead	C11	C2
19		A7	Year	p	3	ovhead	D11	C2
20		B8	========================	p	1	lease	B12	100
21		A9	Sales	p	2	lease	C12	100
22		A10	COGS	p	3	lease	D12	200
23		A11	Overhead	p	1	gross	B13	B9-B10-B11-B12
24		A12	Lease	p	2	gross	C13	C9-C10-C11-C12
25		A13	Gross	p	3	gross	D13	D9-D10-D11-D12
26		A14	Taxes	p	1	taxes	B14	IF(B13<=0,0,B13*$E$2)
27		A16	Net Income	p	2	taxes	C14	IF(C13<=0,0,C13*$E$2)
28		A17	Avg_Net_Inc	p	3	taxes	D14	IF(D13<=0,0,D13*$E$2)
29				p	1	NetInc	B16	B13-B14
30				p	2	NetInc	C16	C13-C14
31				p	3	NetInc	D16	D13-D14
32				q	1	AvgInc	B17	(B16+C16+D16)/3
33								

Figure A-4. Data Separated From Editorial Information

1.1.4 – Separate Data From Schema (Logic or Formulae)

After removing the editorial information as shown in step 1.1.3, from the remaining information (right column of Figure A-4), the actual *data* must be separated from the *schema* and *binding* information. Any entry in this list which has its *value* equal to actual data such as number, date, or text (other than cell addresses and formulae) is considered *data*.

1.1.4.1 – Removing and Storing Data Values

Data values are moved to a separate file designated for holding the actual *data*. This could be a text file in a specific layout format or it can be a DBMS such as RDBMS or OODBMS. Figure A-5 visually shows one possible way for storing the *data* values of the decomposed model. In place of the attributes which contain formulae, the fields are replaced by the letter ‘C’ which stands for *calculated*.

For example, cells D4:D5 of Figure A-5 show the attribute *ovhead* which is located in *relation a* and it is equal to 2500. Whereas, cell I11 of Figure A-5 shows that *taxes* for the 3rd *year* (2002) are calculated and are based on some other values in the model.

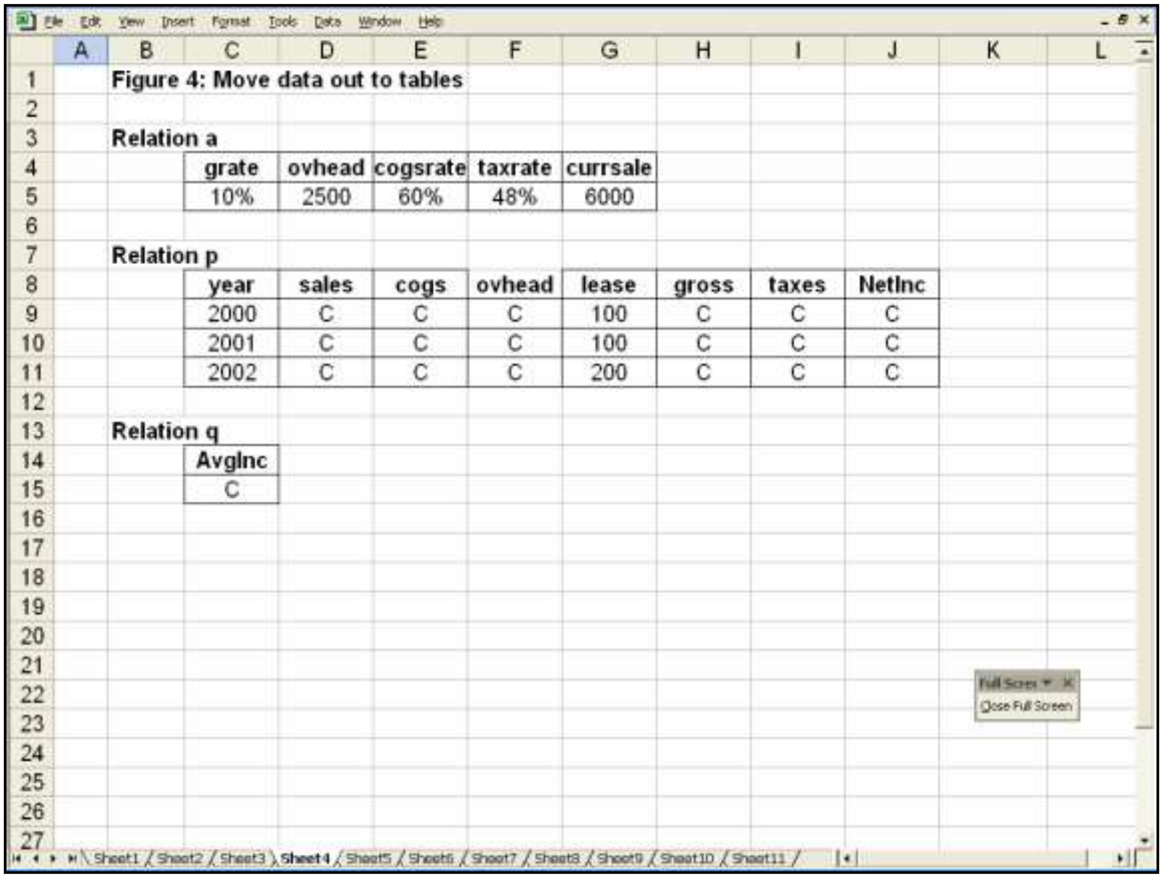

File Edit View Insert Format Tools Data Window Help
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6 Sheet7 Sheet8 Sheet9 Sheet10 Sheet11
Full Screen
Close Full Screen

Figure 4: Move data out to tables

Relation a

grate	ovhead	cogsrate	taxrate	currsale
10%	2500	60%	48%	6000

Relation p

year	sales	cogs	ovhead	lease	gross	taxes	NetInc
2000	C	C	C	100	C	C	C
2001	C	C	C	100	C	C	C
2002	C	C	C	200	C	C	C

Relation q

AvgInc
C

Figure A-5. Data Separated From Other Information of the Model

1.1.4.2 – Processing the Formulae

After moving out the data from the right column of Figure A-4, all data items which were moved out must be replaced with their actual data types.

For example:

a[1].grate, B2, 10%

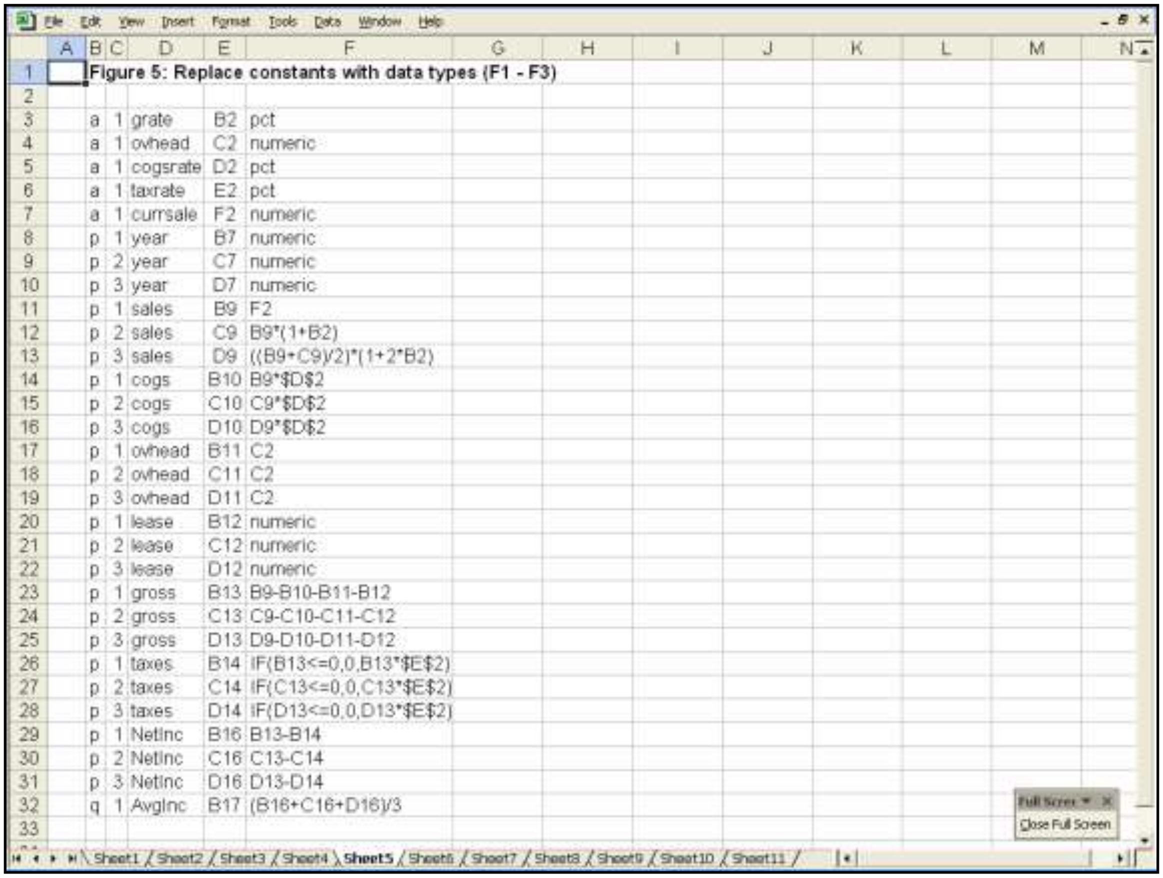
a[1].ovhead, C2, 2500

becomes:

a[1].grate, B2, pct

a[1].ovhead, C2, numeric

Figure A-6 shows the list of formulae stripped of their actual data values (the data which was stored separately in step 1.1.4.1).



	A	B	C	D	E	F
1		Figure 5: Replace constants with data types (F1 - F3)				
2						
3		a	1	grate	B2	pct
4		a	1	ovhead	C2	numeric
5		a	1	cogsrate	D2	pct
6		a	1	taxrate	E2	pct
7		a	1	currsale	F2	numeric
8		p	1	year	B7	numeric
9		p	2	year	C7	numeric
10		p	3	year	D7	numeric
11		p	1	sales	B9	F2
12		p	2	sales	C9	B9*(1+B2)
13		p	3	sales	D9	((B9+C9)/2)*(1+2*B2)
14		p	1	cogs	B10	B9*$D$2
15		p	2	cogs	C10	C9*$D$2
16		p	3	cogs	D10	D9*$D$2
17		p	1	ovhead	B11	C2
18		p	2	ovhead	C11	C2
19		p	3	ovhead	D11	C2
20		p	1	lease	B12	numeric
21		p	2	lease	C12	numeric
22		p	3	lease	D12	numeric
23		p	1	gross	B13	B9-B10-B11-B12
24		p	2	gross	C13	C9-C10-C11-C12
25		p	3	gross	D13	D9-D10-D11-D12
26		p	1	taxes	B14	IF(B13<=0,0,B13*$E$2)
27		p	2	taxes	C14	IF(C13<=0,0,C13*$E$2)
28		p	3	taxes	D14	IF(D13<=0,0,D13*$E$2)
29		p	1	NetInc	B16	B13-B14
30		p	2	NetInc	C16	C13-C14
31		p	3	NetInc	D16	D13-D14
32		q	1	AvgInc	B17	(B16+C16+D16)/3
33						

Figure A-6. Data Replaced by its Data Type

1.1.5 – Obtain the Logical Map for the Spreadsheet Model

Starting with the list in Figure A-6, the *schema* and *binding* list, all references to physical cell addresses (such as ‘A7’ or ‘B21’) which are located within the *value* section must be removed, i.e. range F3:F32 of Figure A-6 (Note: This step should not impact the range E3:E32, which is the *binding* information). These addresses in the *value* information point to physical cell positions in the original spreadsheet. These need to be replaced with relative address positions within the formulae list (*schema*). The relative position consists of the *Relation[i].Attribute* information within each entry in Figure A-6 i.e. range B3:D32.

For example, let us start with cell F11 of Figure A-6: this cell contains F2, which means that its content is derived from cell F2. F2 needs to be located within range E3:E32 of Figure A-6, and once found, F11 will be replaced with the relative position of F2 (the information on its left side) which in this case is *a[1].currsale*. So, B11:F11 in Figure A-6 becomes as shown in B11:F11 of Figure A-7

i.e.

p[1].sales, B9, F2

becomes:

p[1].sales, B9, a[1].currsale

Let us take another example. Let us suppose cell F14 in Figure A-6 needs to be processed/converted to its relative address. This cell currently contains the formula B9*$D$2. B9 and D2 (all $ signs are ignored) need to be located within range E3:E32. When B9 is found, it is noted that its relative position is p[1].sales (i.e. B11:D11 of Figure A-6). D2's relative position is *a[1].cogsrate* (i.e. B5:D5 of Figure A-6). Therefore, cell F14 will be replaced by its relative address: *p[1].sales*a[1].cogsrate*

i.e.

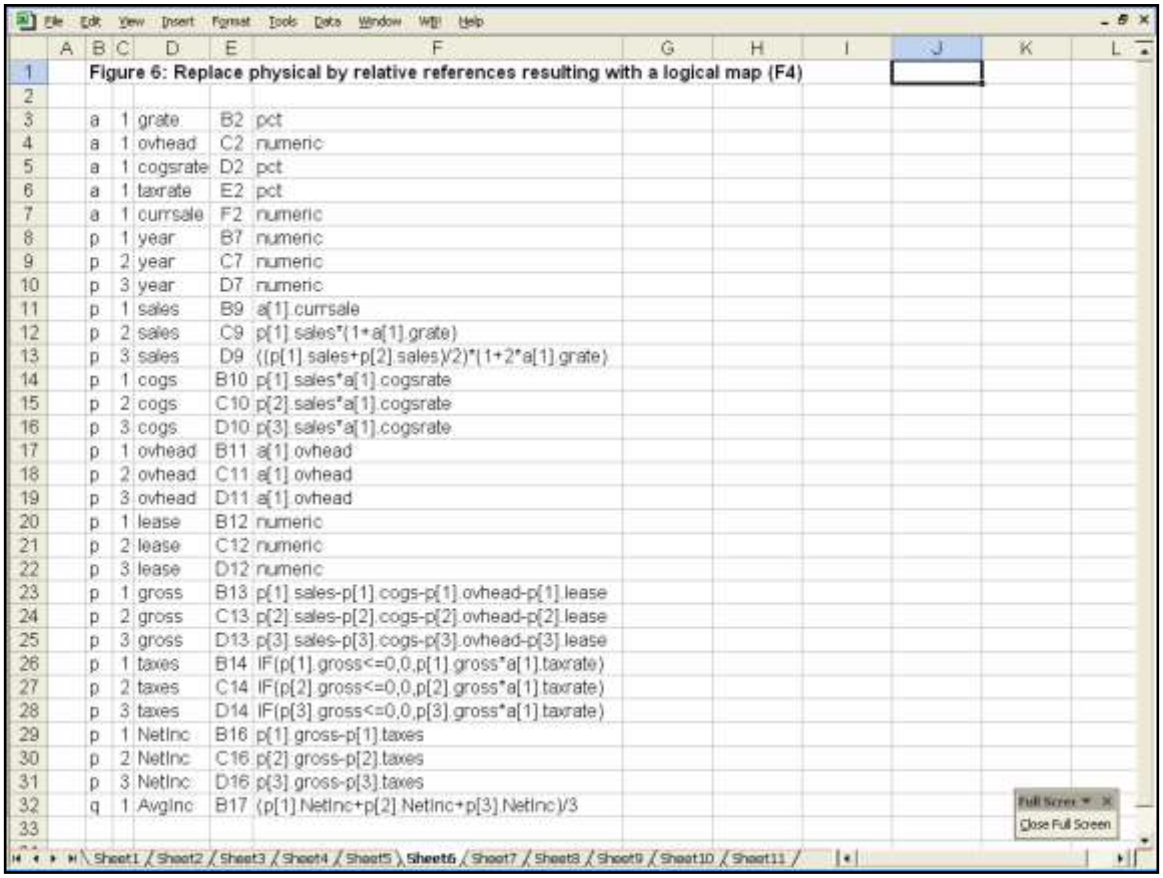
*p[1].cogs, B10, B9*$D$2*

becomes:

*p[1].cogs, B10, p[1].sales*a[1].cogsrate*

Finally, all values in the range F3:F32 which designate data types will remain unchanged. For example, range F3:F10 in Figure A-7 remained the same as is in Figure A-6, since they represent data types (i.e. numeric and pct) and not formulae.

After completing the conversion of all physical addresses in range F3:F32 of Figure A-6 to their logical (relative) address counterparts, the end result is the list shown in Figure A-7. All information within F3:F32 of Figure A-7 is either data types or references other addresses within the range B3:D32 of the same figure. That is what is meant by relative addressing.


File Edit View Insert Format Tools Data Window WB! Help
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6 Sheet7 Sheet8 Sheet9 Sheet10 Sheet11
Full Screen
Close Full Screen

	A	B	C	D	E	F
1		Figure 6: Replace physical by relative references resulting with a logical map (F4)				
2						
3		a	1	grate	B2	pct
4		a	1	ovhead	C2	numeric
5		a	1	cogsrate	D2	pct
6		a	1	taxrate	E2	pct
7		a	1	currsale	F2	numeric
8		p	1	year	B7	numeric
9		p	2	year	C7	numeric
10		p	3	year	D7	numeric
11		p	1	sales	B9	a[1].currsale
12		p	2	sales	C9	p[1].sales*(1+a[1].grate)
13		p	3	sales	D9	((p[1].sales+p[2].sales)/2)*(1+2*a[1].grate)
14		p	1	cogs	B10	p[1].sales*a[1].cogsrate
15		p	2	cogs	C10	p[2].sales*a[1].cogsrate
16		p	3	cogs	D10	p[3].sales*a[1].cogsrate
17		p	1	ovhead	B11	a[1].ovhead
18		p	2	ovhead	C11	a[1].ovhead
19		p	3	ovhead	D11	a[1].ovhead
20		p	1	lease	B12	numeric
21		p	2	lease	C12	numeric
22		p	3	lease	D12	numeric
23		p	1	gross	B13	p[1].sales-p[1].cogs-p[1].ovhead-p[1].lease
24		p	2	gross	C13	p[2].sales-p[2].cogs-p[2].ovhead-p[2].lease
25		p	3	gross	D13	p[3].sales-p[3].cogs-p[3].ovhead-p[3].lease
26		p	1	taxes	B14	IF(p[1].gross<=0,0,p[1].gross*a[1].taxrate)
27		p	2	taxes	C14	IF(p[2].gross<=0,0,p[2].gross*a[1].taxrate)
28		p	3	taxes	D14	IF(p[3].gross<=0,0,p[3].gross*a[1].taxrate)
29		p	1	NetInc	B16	p[1].gross-p[1].taxes
30		p	2	NetInc	C16	p[2].gross-p[2].taxes
31		p	3	NetInc	D16	p[3].gross-p[3].taxes
32		q	1	AvgInc	B17	(p[1].NetInc+p[2].NetInc+p[3].NetInc)/3
33						

Figure A-7. Physical References Replaced With Relative References

1.1.6 – Convert Record References From Absolute to Relative

At this point, except for *binding* information, all physical pointers to the original spreadsheet are removed from the *value* column of Figure A-7 i.e. from the range F3:F32. However, many entries in this column are still indexed in absolute terms i.e. in case there are many entries within a same *relation*, such records are indexed sequentially 1, 2, 3 and so on. These absolute indexes need to be converted into relative ones, meaning later records within a relation must be referred to in terms of the earlier ones. *Vector relations* (i.e. relations which contain only one record) are not impacted by this process. For example, let us consider range B14:F16 in Figure A-7 which is as follows:

*p[1].cogs, B10, p[1].sales*a[1].cogsrate*

*p[2].cogs, C10, p[2].sales*a[1].cogsrate*

*p[3].cogs, D10, p[3].sales*a[1].cogsrate*

The third term in each of these entries, the *value*, must be re-written in terms of the first term i.e. in terms of *relation[i].attribute*. For example, these three entries become

*p[1].cogs, B10, p[n].sales*a[1].cogsrate*

*p[2].cogs, C10, p[n].sales*a[1].cogsrate*

*p[3].cogs, D10, p[n].sales*a[1].cogsrate*

where, in each case, n = the index of the first term, i.e. *p[i].cogs* of the first term.

Let us examine a more complex example and look at B11:F13 of Figure A-7 which is as follows:

p[1].sales, B9, a[1].currsale

p[2].sales, C9, p[1].sales(1+a[1].grate)*

p[3].sales, D9, ((p[1].sales+p[2].sales)/2)(1+2*a[1].grate)*

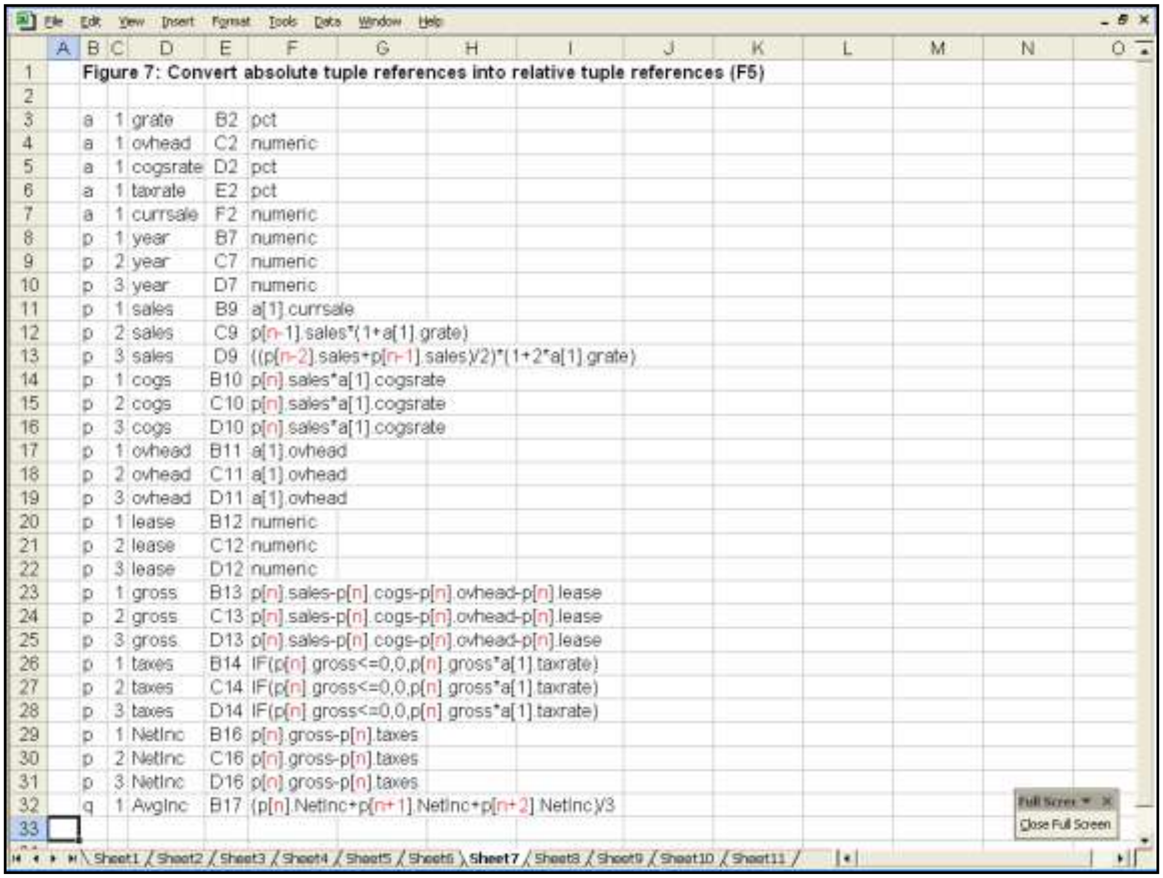
it becomes:

p[1].sales, B9, a[1].currsale

p[2].sales, C9, p[n-1].sales(1+a[1].grate)* *p[3].sales, D9, ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a[1].grate)*

Note: only non-vector *relations* are changed. *Vector relations* such as *relations a* and *q* remain unchanged.

When all these entries are converted from absolute to relative record indexing, the list will look as shown in Figure A-8. All changes are highlighted in red for better visibility.



	A	B	C	D	E	F
1		Figure 7: Convert absolute tuple references into relative tuple references (F5)				
2						
3		a	1	grate	B2	pct
4		a	1	ovhead	C2	numeric
5		a	1	cogsrate	D2	pct
6		a	1	taxrate	E2	pct
7		a	1	currsale	F2	numeric
8		p	1	year	B7	numeric
9		p	2	year	C7	numeric
10		p	3	year	D7	numeric
11		p	1	sales	B9	a[1].currsale
12		p	2	sales	C9	p[n-1].sales*(1+a[1].grate)
13		p	3	sales	D9	((p[n-2].sales+p[n-1].sales)/2)*(1+2*a[1].grate)
14		p	1	cogs	B10	p[n].sales*a[1].cogsrate
15		p	2	cogs	C10	p[n].sales*a[1].cogsrate
16		p	3	cogs	D10	p[n].sales*a[1].cogsrate
17		p	1	ovhead	B11	a[1].ovhead
18		p	2	ovhead	C11	a[1].ovhead
19		p	3	ovhead	D11	a[1].ovhead
20		p	1	lease	B12	numeric
21		p	2	lease	C12	numeric
22		p	3	lease	D12	numeric
23		p	1	gross	B13	p[n].sales-p[n].cogs-p[n].ovhead-p[n].lease
24		p	2	gross	C13	p[n].sales-p[n].cogs-p[n].ovhead-p[n].lease
25		p	3	gross	D13	p[n].sales-p[n].cogs-p[n].ovhead-p[n].lease
26		p	1	taxes	B14	IF(p[n].gross<=0,0,p[n].gross*a[1].taxrate)
27		p	2	taxes	C14	IF(p[n].gross<=0,0,p[n].gross*a[1].taxrate)
28		p	3	taxes	D14	IF(p[n].gross<=0,0,p[n].gross*a[1].taxrate)
29		p	1	NetInc	B16	p[n].gross-p[n].taxes
30		p	2	NetInc	C16	p[n].gross-p[n].taxes
31		p	3	NetInc	D16	p[n].gross-p[n].taxes
32		q	1	AvgInc	B17	(p[n].NetInc+p[n+1].NetInc+p[n+2].NetInc)/3
33						

Figure A-8. Absolute Record References Converted to Relative References

1.1.7 – Contract Repetitive Entries

All *values*, i.e. cells in range F3:F32 of Figure A-8, which have an index of 1 or *n*, can be contracted, because such information will not be lost and can be re-instated if needed. For example the *value* in F11 of Figure A-8,

a[1].currsale

becomes:

a.currsale

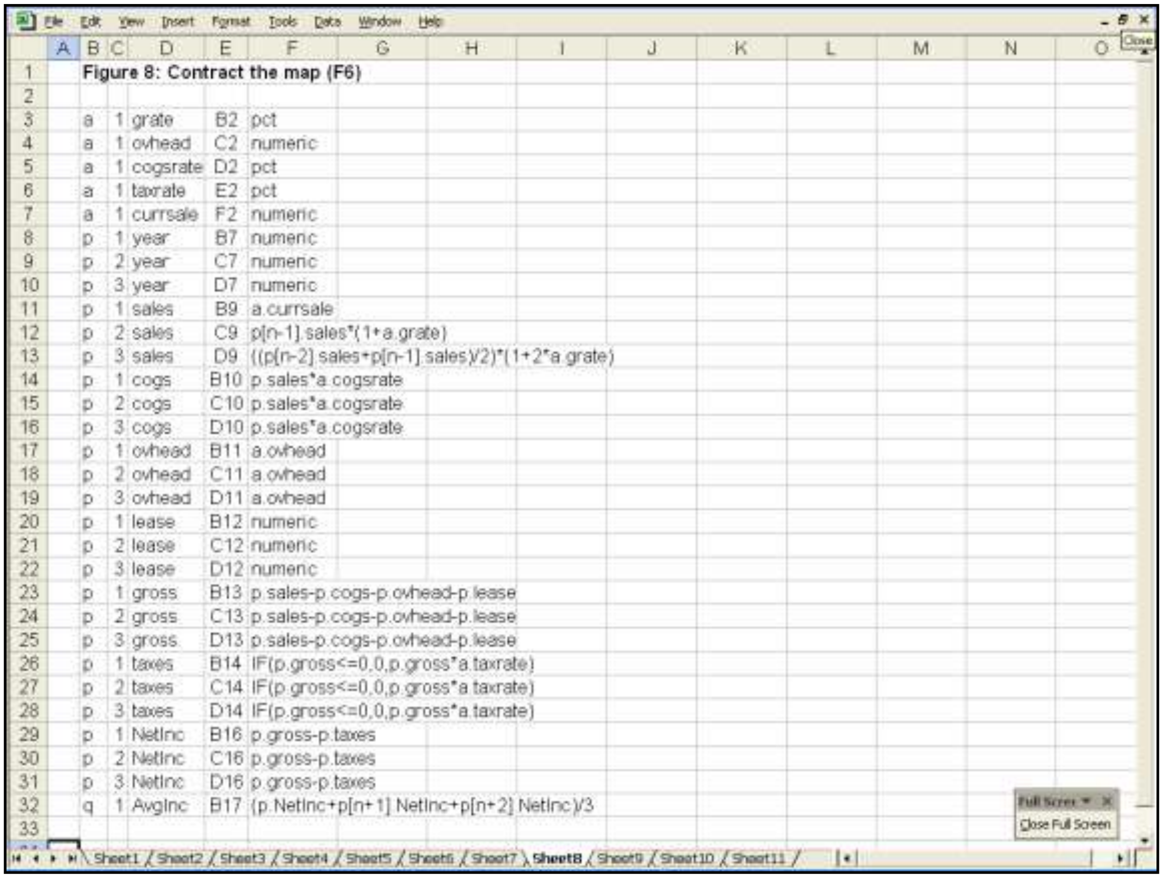
and the *value* in F23 of Figure A-8,

p[1].sales-p[1].cogs-p[1].ovhead-p[1].lease

becomes:

p.sales-p.cogs-p.ovhead-p.lease

At the completion of this step, the *schema* list will look as shown in Figure A-9.

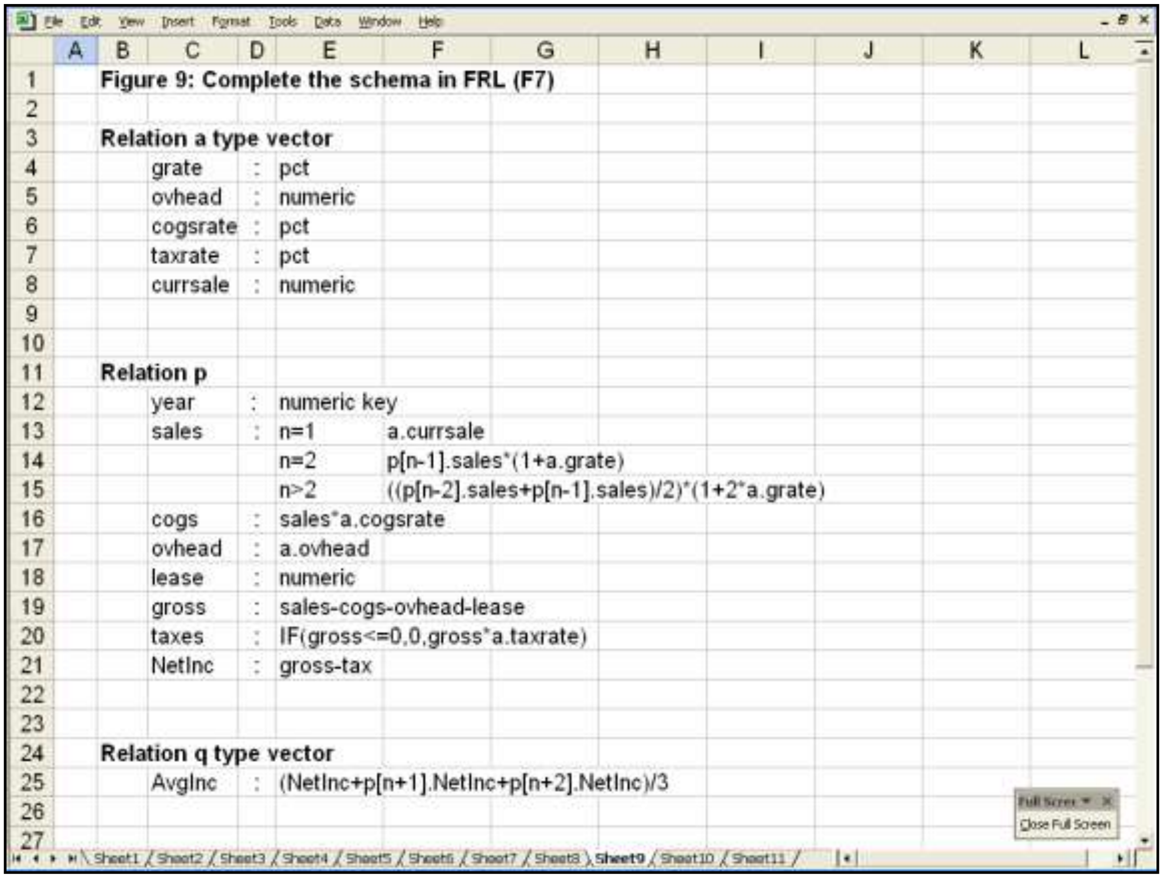


	A	B	C	D	E	F
1		**Figure 8: Contract the map (F6)**				
2						
3		a	1	grate	B2	pct
4		a	1	ovhead	C2	numeric
5		a	1	cogsrate	D2	pct
6		a	1	taxrate	E2	pct
7		a	1	currsale	F2	numeric
8		p	1	year	B7	numeric
9		p	2	year	C7	numeric
10		p	3	year	D7	numeric
11		p	1	sales	B9	a.currsale
12		p	2	sales	C9	p[n-1].sales*(1+a.grate)
13		p	3	sales	D9	((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)
14		p	1	cogs	B10	p.sales*a.cogsrate
15		p	2	cogs	C10	p.sales*a.cogsrate
16		p	3	cogs	D10	p.sales*a.cogsrate
17		p	1	ovhead	B11	a.ovhead
18		p	2	ovhead	C11	a.ovhead
19		p	3	ovhead	D11	a.ovhead
20		p	1	lease	B12	numeric
21		p	2	lease	C12	numeric
22		p	3	lease	D12	numeric
23		p	1	gross	B13	p.sales-p.cogs-p.ovhead-p.lease
24		p	2	gross	C13	p.sales-p.cogs-p.ovhead-p.lease
25		p	3	gross	D13	p.sales-p.cogs-p.ovhead-p.lease
26		p	1	taxes	B14	IF(p.gross<=0,0,p.gross*a.taxrate)
27		p	2	taxes	C14	IF(p.gross<=0,0,p.gross*a.taxrate)
28		p	3	taxes	D14	IF(p.gross<=0,0,p.gross*a.taxrate)
29		p	1	NetInc	B16	p.gross-p.taxes
30		p	2	NetInc	C16	p.gross-p.taxes
31		p	3	NetInc	D16	p.gross-p.taxes
32		q	1	AvgInc	B17	(p.NetInc+p[n+1].NetInc+p[n+2].NetInc)/3
33						

Figure A-9. Contraction of Terms With Index of 1 or *n*

1.1.8 – Complete the Schema in FRL

The last remaining step is to convert the list in Figure A-9 into the FRL format. FRL removes all repetitive information and places all formulae in a condensed format, from which it is possible to reconstitute the original format i.e. back to as shown in Figure A-9. To do this, the list from Figure A-9 needs to be scanned and the information needs to be broken down into *relations*. Since the information is already sorted by the *relation* name (*a*, *p*, or *q*) and within each of these groups, each attribute is sorted by index (1, 2, 3…), it is easy to scan through the list and compact it as shown in Figure A-10.


File Edit View Insert Format Tools Data Window Help
A B C D E F G H I J K L
Full Screen
Close Full Screen
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6 Sheet7 Sheet8 Sheet9 Sheet10 Sheet11

```
Figure 9: Complete the schema in FRL (F7)

Relation a type vector
    grate     :  pct
    ovhead    :  numeric
    cogsrate  :  pct
    taxrate   :  pct
    currsale  :  numeric


Relation p
    year      :  numeric key
    sales     :  n=1        a.currsale
                 n=2        p[n-1].sales*(1+a.grate)
                 n>2        ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)
    cogs      :  sales*a.cogsrate
    ovhead    :  a.ovhead
    lease     :  numeric
    gross     :  sales-cogs-ovhead-lease
    taxes     :  IF(gross<=0,0,gross*a.taxrate)
    NetInc    :  gross-tax


Relation q type vector
    AvgInc    :  (NetInc+p[n+1].NetInc+p[n+2].NetInc)/3
```

Figure A-10. The Schema in FRL Format

The information concerning *relation a* is considered a *vector relation*, because it consists of only one record. Therefore, the left column of the following table will be converted to the format in the right column.

	Relation a type vector
a[1].grate, B2, pct	*grate : pct*
a[1].ovhead, C2, numeric	*ovhead : numeric*
a[1].cogsrate, D2, pct	*cogsrate : pct*
a[1].taxrate, E2, pct	*taxrate : pct*
a[1].currsale, F2, numeric	*currsale : numeric*

The conversion of *relation p* is a bit trickier since it contains multiple records. Therefore, the information stored in multiple records must be condensed to a diminutive format from which the original records can be reconstructed. The following table groups each attribute within the relation and shows target conversion format for each attribute.

Let us take the first *attribute year* of the *relation p* as an example. It is originally stated as follows:

p[1].year, B7, numeric

p[2].year, C7, numeric

p[3].year, D7, numeric

after the conversion it becomes:

year : numeric key

Note: This process removed the *relation*'s name *p* and its index value *n*. Instead of repeating the *relation* name for each record, it will be stated once at the beginning of the *relation*'s definition and will apply to all its subsequent records. The index value *n*, which refers to the number of records in the *relation*, is easily obtainable from counting the number of records in the *data* which was separated in section 1.1.4.1 and shown in Figure A-5. This figure shows that the *relation p* contains three records.

Note: This process also stripped the *binding* information, which will be discussed separately in the next section.

Let us take a more complex example as in the case of the *attribute sales* which was originally shown as

p[1].sales, B9, a.currsale

p[2].sales, C9, p[n-1].sales(1+a.grate)*

p[3].sales, D9, ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)* after the conversion it becomes:

sales : n=1 a.currsale

n=2 p[n-1].sales(1+a.grate)*

n>2 ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)*

In addition to the notes for the previous example, this case needs further processing and explanation. In this case, in order to reconstruct the original records, different formulae for each distinct case is needed. Since records within the *relation* are organized by their *index* number *n*, subsequent reconstruction of the original records is based on this same *index* value. Therefore, the *relation* name and *index* are stripped as explained in the previous example. Then, for each value of *n*, the corresponding formula is noted.

The remainder of the changes is listed in the following table:

The Terms	Will Become
	Relation p
p[1].year, B7, numeric p[2].year, C7, numeric p[3].year, D7, numeric	year : numeric key

p[1].sales, B9, a.currsale
p[2].sales, C9, p[n-1].sales*(1+a.grate)
p[3].sales, D9, ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)
	Sales :
	n=1 a.currsale
	n=2 p[n-1].sales*(1+a.grate)
	n>2 ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)

p[1].cogs, B10, p.sales*a.cogsrate p[2].cogs, C10, p.sales*a.cogsrate p[3].cogs, D10, p.sales*a.cogsrate	cogs : sales*a.cogsrate
p[1].ovhead, B11, a.ovhead p[2].ovhead, C11, a.ovhead p[3].ovhead, D11, a.ovhead	ovhead : a.ovhead
p[1].lease, B12, numeric p[2].lease, C12, numeric p[3].lease, D12, numeric	lease : numeric
p[1].gross, B13, p.sales-p.cogs-p.ovhead-p.lease p[2].gross, C13, p.sales-p.cogs-p.ovhead-p.lease p[3].gross, D13, p.sales-p.cogs-p.ovhead-p.lease	gross : sales-cogs-ovhead-lease
p[1].taxes, B14, IF(p.gross<=0,0,p.gross*a.taxrate) p[2].taxes, C14, IF(p.gross<=0,0,p.gross*a.taxrate) p[3].taxes, D14, IF(p.gross<=0,0,p.gross*a.taxrate)	taxes : IF(gross<=0,0,gross*a.taxrate)
p[1].NetInc, B16, p.gross-p.taxes p[2].NetInc, C16, p.gross-p.taxes p[3].NetInc, D16, p.gross-p.taxes	NetInc : gross-tax

The transformation for *relation q* is similar to *relation a* as described above. The following table shows this transformation.

	Relation q type vector
q[1].AvgInc, B17, (p.NetInc+p[n+1].NetInc+p[n+2].NetInc)/3	AvgInc : (NetInc+p[n+1].NetInc+p[n+2].NetInc)/3

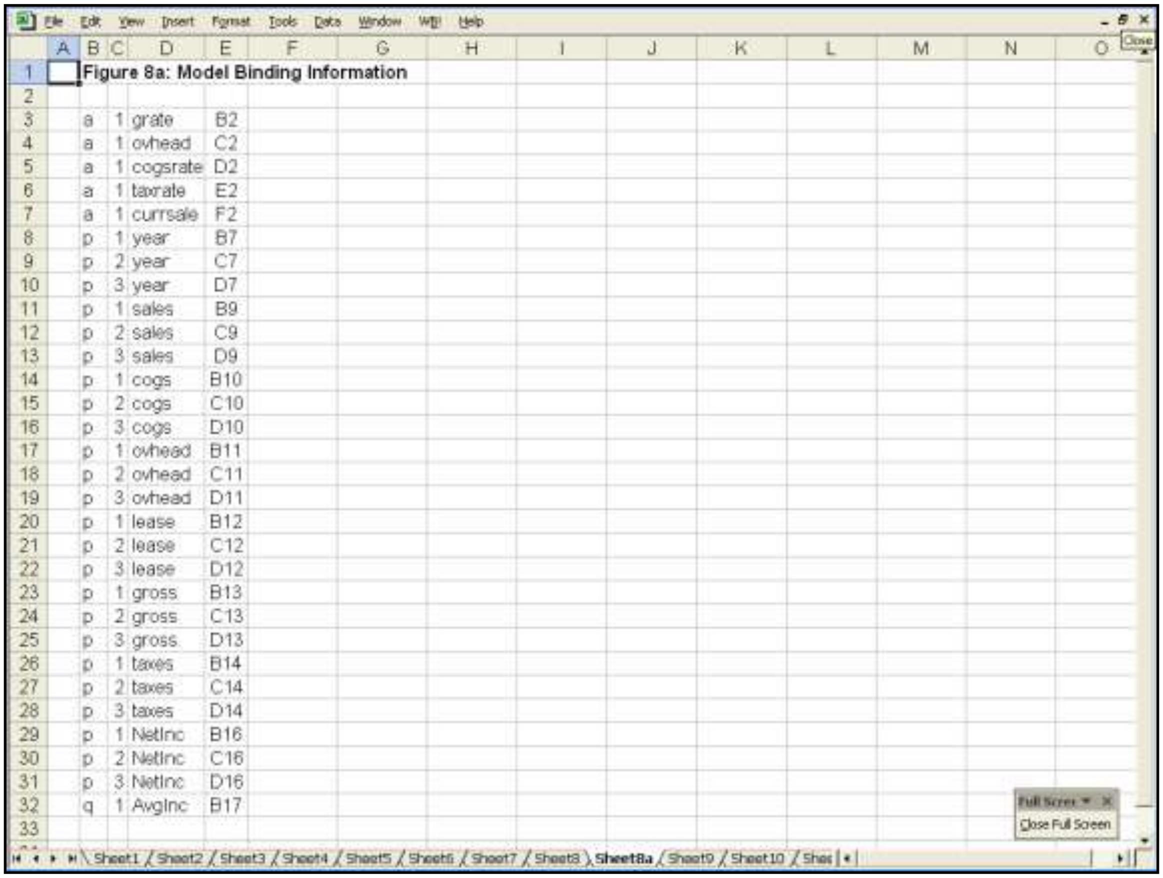
1.1.9 – Save the Schema Binding Information

All *binding* information was removed in the previous step (1.1.8) and their importance was not discussed. Let us examine the left hand side columns of the tables above. Each of these terms includes *binding* information, which is the physical cell position (in the form of a letter followed by a number e.g. B17) into which each term should be redeployed. This information is for reference only, since the logical model does not really have to be put back in the same exact cell positions as was in the original model. This is normal since the FRL contains a logical model, whereas, *bindings* are information which tie a model to a specific physical format.

However, this *binding* information is very important in one respect: to combine the model back with the *editorial* information removed in step 1.1.3 above. The *editorial* information comprises all labels and comments about the model and it is important if the model needs to be reconstructed back (synthesized) as it was before decomposing it (factoring). Otherwise, it would not be possible to re-synchronize the model back with all its comments and labels.

This turns out to be a trivial issue in case the model needs to be recreated exactly as it was before factoring it. A file containing a copy of the range B3:D32 of Figure A-9 is kept. This model *binding* information is shown in Figure A-11. This *binding* list maps each record/attribute of each *relation* to its exact original position. However, the FRL does impose a limit to go back exactly to the same format or content (data composition). For example, the sample model had columns for only three years of sales. A fourth year can be easily created based on the logical information stored within the FRL. But, the newly created year would not have the additional *binding* or the *editorial* information of the original model. This *editorial* information must be re-entered by prompting the user for it.

Note: the *binding* information can be inferred from the existing *binding* i.e. the newly created fourth year will be placed right next to the third year. However, this could create a new problem: the information for the fourth year could be overlaying some other existing *editorial* information. This is a shortcoming of the existing algorithm which should be further elaborated in future research.



	A	B	C	D	E
1		Figure 8a: Model Binding Information			
2					
3		a	1	grate	B2
4		a	1	ovhead	C2
5		a	1	cogsrate	D2
6		a	1	taxrate	E2
7		a	1	currsale	F2
8		p	1	year	B7
9		p	2	year	C7
10		p	3	year	D7
11		p	1	sales	B9
12		p	2	sales	C9
13		p	3	sales	D9
14		p	1	cogs	B10
15		p	2	cogs	C10
16		p	3	cogs	D10
17		p	1	ovhead	B11
18		p	2	ovhead	C11
19		p	3	ovhead	D11
20		p	1	lease	B12
21		p	2	lease	C12
22		p	3	lease	D12
23		p	1	gross	B13
24		p	2	gross	C13
25		p	3	gross	D13
26		p	1	taxes	B14
27		p	2	taxes	C14
28		p	3	taxes	D14
29		p	1	NetInc	B16
30		p	2	NetInc	C16
31		p	3	NetInc	D16
32		q	1	AvgInc	B17
33					

Figure A-11. Model Binding Information

1.1.10 – The Output of the Factorization Process (Model Meta-Knowledge)

In summary, by the end of the Factorization process, four different files are created, each containing a portion of the original model. These are:

1) The *editorial* information - see left column of Figure A-4;

2) The *actual* data used in the model – see Figure A-5;

3) The model FRL *schema* – see Figure A-10; and

4) *Binding* information – see Figure A-11. Starting with these four files, the original spreadsheet model can be recreated.

1.2 – FRL to Spreadsheet: The Trip Back (Synthesis)

The previous section walked through each of the steps in decomposing a spreadsheet based model into four different files which contain all the pieces necessary not only to recreate the original model, but to also expand it if necessary. The recreation of the original model can be obtained from reversing the process described in section 1.1, starting with the last step (1.1.10) and moving backwards up to the first section (1.1.1). However, if the user desires to change or expand the original model starting with the FRL, then other issues must be considered. This section elaborates on these concepts.

1.2.1 – Recreating the Original Model without Making Changes

Recreating the original model without any changes basically puts a limit to the number of records each *relation* can have, as it was in the original model. Therefore, this does not impact any vector *relation*, i.e. made up of only one record. As explained in section 1.1.7, since all *vector relations* have one record, their absolute *index i* (record position within the *relation*) is set to 1. Therefore, *relation a* is converted as shown in the following table. *Relation q* goes through a similar process since it is a *vector relation*. Note: instead of showing the *relation* name once for all the attributes, the *relation* name is placed before each *attribute* and *index*.

From	To
Relation a type vector	
Grate : pct	*a[1].grate, pct*
ovhead : numeric	*a[1].ovhead, numeric*
cogsrate : pct	*a[1].cogsrate, pct*
taxrate : pct	*a[1].taxrate, pct*
currsale : numeric	*a[1].currsale, numeric*
Relation q type vector	
AvgInc : *(NetInc+p[n+1].NetInc+p[n+2].NetInc)/3*	*q[1].AvgInc,* *(p.NetInc+p[n+1].NetInc+p[n+2].NetInc)/3*

For the *schema* model shown in Figure A-10, a decision needs to be made concerning how many records *relation p* will have i.e. how many years of information would the user like to recreate? This information can be obtained from the *binding* information shown in Figure A-11 or from the original *data* information shown in Figure A-5. In either case, it is easy to determine that *relation p* is being repeated three times for each attribute. Therefore, the maximum value for the relative *index n* can reach will be equal to 3. Armed with this information, the FRL representation of *relation p* can be expanded up to three records each as follows. Each *attribute* will be preceded by the *relation* name followed by the absolute *index i*.

The Terms	Will Become
year: numeric key	p[1].year, numeric p[2].year, numeric p[3].year, numeric

sales:
n=1 a.currsale
n=2 p[n-1].sales*(1+a.grate)
n>2 ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)
	p[1].sales, a.currsale
	p[2].sales, p[n-1].sales*(1+a.grate)
	p[3].sales, ((p[n-2].sales+p[n-1].sales)/2)*(1+2*a.grate)

cogs: sales*a.cogsrate	p[1].cogs, p.sales*a.cogsrate p[2].cogs, p.sales*a.cogsrate p[3].cogs, p.sales*a.cogsrate
ovhead: a.ovhead	p[1].ovhead, a.ovhead p[2].ovhead, a.ovhead p[3].ovhead, a.ovhead
lease: numeric	p[1].lease, numeric p[2].lease, numeric p[3].lease, numeric
gross: sales-cogs-ovhead-lease	p[1].gross, p.sales-p.cogs-p.ovhead-p.lease p[2].gross, p.sales-p.cogs-p.ovhead-p.lease p[3].gross, p.sales-p.cogs-p.ovhead-p.lease
taxes: IF(gross<=0,0,gross*a.taxrate)	p[1].taxes, IF(p.gross<=0,0,p.gross*a.taxrate) p[2].taxes, IF(p.gross<=0,0,p.gross*a.taxrate) p[3].taxes, IF(p.gross<=0,0,p.gross*a.taxrate)
NetInc: gross-tax	p[1].NetInc, p.gross-p.taxes p[2].NetInc, p.gross-p.taxes p[3].NetInc, p.gross-p.taxes

The right columns of the previous table contain the contracted map of the model as shown in Figure A-9, except for the binding information which is shown in Figure A-11. These two lists need to be merged based on the *Relation[i].Attribute* and Figure A-9 is the result, including the binding information. Up to this point, steps 1.1.7 to 1.1.10 have been performed in reverse order i.e. starting from 1.1.10.

From step 1.1.6 back up to step 1.1.5 is a mechanical conversion process repeating the original steps in reverse. At this point, the logical map of the model is obtained as shown in Figure A-7. The logical map will need to be merged with the *data* shown in

Figure A-5 which completes step 1.1.4, and the result will be what is shown in the right column of Figure A-4.

Next, the *editorial* information needs to be merged back and placed it in the left column of Figure A-4. From Figure A-4, each line can easily be transferred to its absolute address as shown in E2:E30 and also shown in N5:N30, at which point the original model will be reconstructed (synthesized).

1.2.2 – Recreating the Original Model With Changes

Recreating the original model with change basically means allowing the user to expand beyond the number of records shown in the original model. Again, this does not impact any *vector relation*, i.e. *relations* made up of only one record (*relations a* and *q*). This new change will impact those relations which originally had more than one record, i.e. *relation p*

In the model shown in Figure A-10, a decision has to be made concerning how many records *relation p* will have i.e. how many years of information the new model will have? In this case, the user can be prompted for a number which would designate the number of years. Let us suppose the user enters 5. Armed with this information, the FRL representation of *relation p* can be expanded up to five records. All *attributes* of *p* will be re-generated five times without change, except for the sales attribute which must be determined based on the following formula:

sales:

n=1 a.currsale

n=2 p[n-1].sales(1+a.grate)*

n>2 ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)*

The expanded version will be as follows with the addition of records for years four and five:

p[1].sales, a.currsale

p[2].sales, p[n-1].sales(1+a.grate)*

p[3].sales, ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)*

p[4].sales, ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)*

p[5].sales, ((p[n-2].sales+p[n-1].sales)/2)(1+2*a.grate)*

This type of change will apply to all *non-vector relations*, i.e. relations which contain more than one record.

Note: In case the original model had some *editorial* information for the attribute *year*, with this newly changed model, a user will have to later enter this additional *editorial* information. At worst, the newly created records may not have any additional *editorial* information (i.e. labels or comments), which would not impact the semantic of the model.

1.3 – Conclusion

This appendix showed the steps which convert a spreadsheet model from the spreadsheet format, called its physical representation (i.e. bound to a specific spreadsheet), to its logical counterpart which is stored in a schema definition language created by Isakowitz et al. (1995) and called Functional / Relational Language (FRL). This process is called *Factoring*. The reverse process of going back from FRL to spreadsheet model is called *Synthesis*.

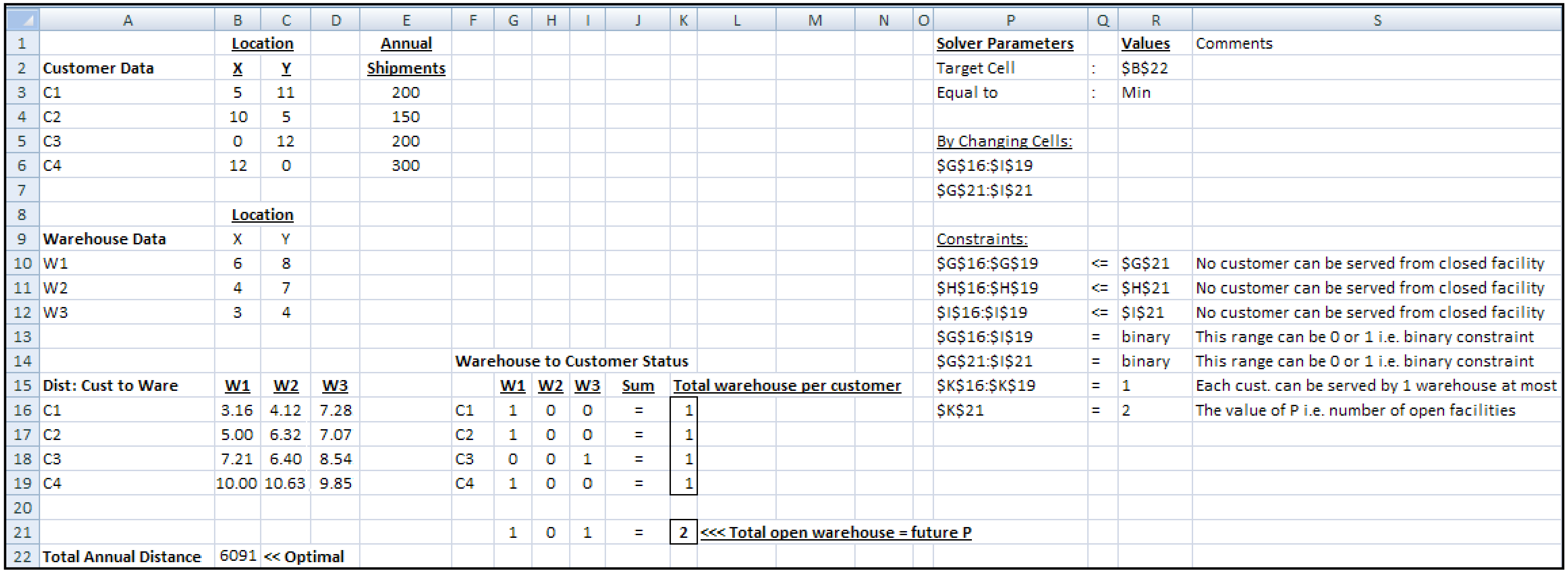
Appendix B. Functional Relational Language (FRL) in Backus-Naur Form (BNF)

This appendix shows the specification of the Functional Relational Language (FRL) as in Backus-Naur Form (BNF) as shown Isakowitz et al. (1995). It is included as a reference and it is referred to it in Chapter 3.

```
Model_Schema      ::= R_schema |
                      R_schema Model_Schema
R_schema          ::= R_def Key_Attr_descr |
                      R_def Key_Attr_descr Rest_Attr_descr
R_def             ::= relation R_Name alias R_alias_name |
                      R_Name alias R_alias_name (type vector)
R_Name            ::= Name
Name              ::= String
R_alias_name      ::= Letter
Key_Attr_descr    ::= Data_Attr_descr key
Rest_Attr_descr   ::= Attr_descr |
                      Attr_descr Rest Attr_descr
Attr_descr        ::= Data_Attr_descr |
                      Func_Attr_descr
Data_Attr_descr   ::= Attr_name : Type
Type              ::= number | string | date | logical
Attr_name         ::= Name
Func_Attr_descr   ::= Attr_name : Expr
Expr              ::= Simple_Expr
Expr              ::= Case_Expr
Case_Expr         ::= Boolean_Cond ↦ Simple_Expr |
                      Boolean_Cond ↦ Simple_Expr Case_Expr
Boolean_Cond      ::= nComparator NUM |
                      NUM ≤ n < NUM
Simple_Expr       ::= Type |
                      Constant |
                      Reference |
                      If_Expr
Constant          ::= NUM | STRING | DATE | LOGICAL
Reference         ::= R_alias_name[key = Ref].Attr_name |
                      R_alias_name[Ref].Attr_name
Ref               ::= Num_expr | Attr_expr
Num_expr          ::= n | Num_expr + NAT | Num_expr − NAT
Attr_expr         ::= Attr_name | FUNC(Attr_exp)
FUNC              ::= next | prev | glb | lub
If_Expr           ::= IF (Bool_Cond, Simple_Expr, Simple_Expr)
Bool_Cond         ::= Reference Comparator Reference
Comparator        ::= < | ≤ | = | > | ≥
NUM               ::= numeric constants, (any rational number)
NAT               ::= natural number constants, 1, 2, 3, . . .
STRING            ::= string constants
DATE              ::= date constants
LOGICAL           ::= logical constants
```

Appendix C. Spreadsheet Model of Tight Formulated P-Median Problem

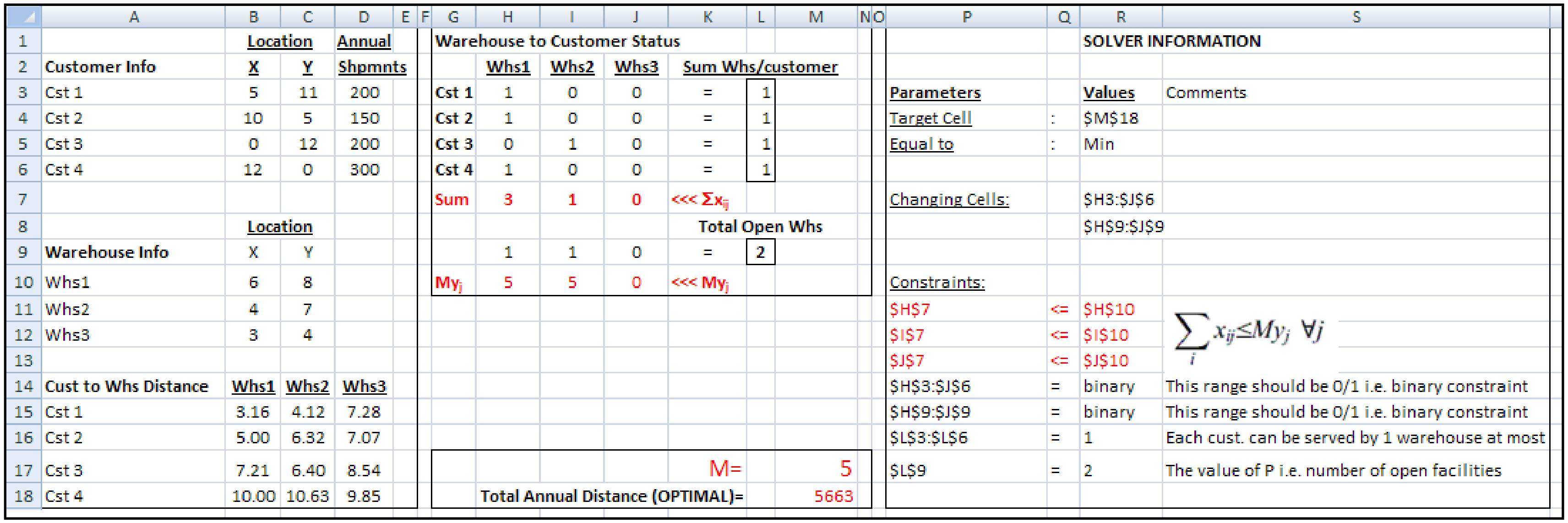
This appendix shows a spreadsheet model which represents the p-median problem's tight formulation. It is implemented for the warehouse location problem to handle four customers and three warehouses. The optimal value is stored in cell B22. The value of P is stored in K21 and is shown to be set to 2. The constraints are shown in the cell range P1 to S16.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		Location			Annual											Solver Parameters		Values	Comments
2	Customer Data	X	Y		Shipments											Target Cell	:	$B$22	
3	C1	5	11		200											Equal to	:	Min	
4	C2	10	5		150														
5	C3	0	12		200											By Changing Cells:			
6	C4	12	0		300											$G$16:$I$19			
7																$G$21:$I$21			
8		Location																	
9	Warehouse Data	X	Y													Constraints:			
10	W1	6	8													$G$16:$G$19	<=	$G$21	No customer can be served from closed facility
11	W2	4	7													$H$16:$H$19	<=	$H$21	No customer can be served from closed facility
12	W3	3	4													$I$16:$I$19	<=	$I$21	No customer can be served from closed facility
13																$G$16:$I$19	=	binary	This range can be 0 or 1 i.e. binary constraint
14						Warehouse to Customer Status										$G$21:$I$21	=	binary	This range can be 0 or 1 i.e. binary constraint
15	Dist: Cust to Ware	W1	W2	W3			W1	W2	W3	Sum	Total warehouse per customer					$K$16:$K$19	=	1	Each cust. can be served by 1 warehouse at most
16	C1	3.16	4.12	7.28		C1	1	0	0	=	1					$K$21	=	2	The value of P i.e. number of open facilities
17	C2	5.00	6.32	7.07		C2	1	0	0	=	1								
18	C3	7.21	6.40	8.54		C3	0	0	1	=	1								
19	C4	10.00	10.63	9.85		C4	1	0	0	=	1								
20																			
21							1	0	1	=	2	<<< Total open warehouse = future P							
22	Total Annual Distance	6091	<< Optimal																

Appendix D. Spreadsheet Model of Loose Formulated P-Median Problem

This appendix shows a spreadsheet model which represents the p-median problem's loose formulation. It is implemented for the warehouse location problem to handle four customers and three warehouses. The optimal value is stored in cell M18. The value of P is stored in L9 and is shown to be set to 2. The constraints are shown in the cell range P1 to S17.

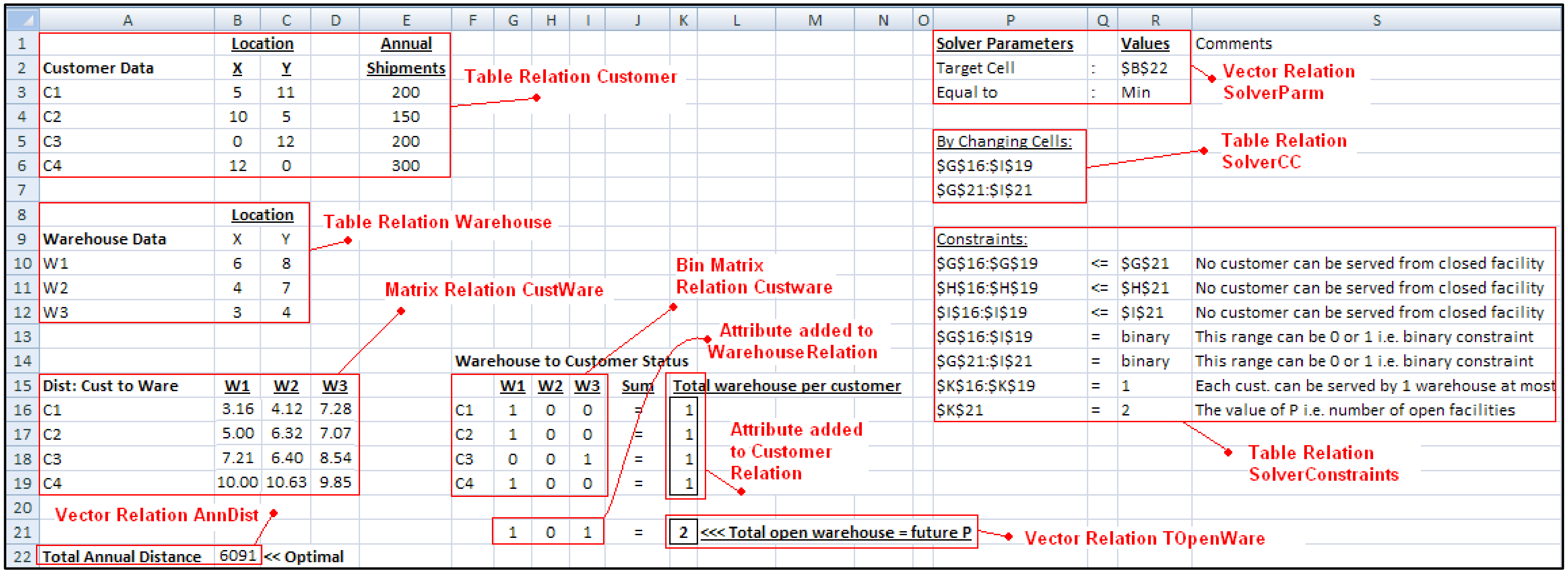


	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Location		Annual			Warehouse to Customer Status						
2	Customer Info	X	Y	Shpmnts				Whs1	Whs2	Whs3	Sum Whs/customer		
3	Cst 1	5	11	200			Cst 1	1	0	0	=	1	
4	Cst 2	10	5	150			Cst 2	1	0	0	=	1	
5	Cst 3	0	12	200			Cst 3	0	1	0	=	1	
6	Cst 4	12	0	300			Cst 4	1	0	0	=	1	
7							Sum	3	1	0	<<< Σx_{ij}		
8		Location									Total Open Whs		
9	Warehouse Info	X	Y					1	1	0	=	2	
10	Whs1	6	8				My_j	5	5	0	<<< My_j		
11	Whs2	4	7										
12	Whs3	3	4										
13													
14	Cust to Whs Distance	Whs1	Whs2	Whs3									
15	Cst 1	3.16	4.12	7.28									
16	Cst 2	5.00	6.32	7.07									
17	Cst 3	7.21	6.40	8.54							M=		5
18	Cst 4	10.00	10.63	9.85				Total Annual Distance (OPTIMAL)=					5663

	N	O	P	Q	R	S
1					SOLVER INFORMATION	
2						
3			Parameters		Values	Comments
4			Target Cell	:	$M$18	
5			Equal to	:	Min	
6						
7			Changing Cells:		$H3:$J$6	
8					$H$9:$J$9	
9						
10			Constraints:			
11			$H$7	<=	$H$10	$\sum_i x_{ij} \le My_j \ \forall j$
12			$I$7	<=	$I$10	
13			$J$7	<=	$J$10	
14			$H$3:$J$6	=	binary	This range should be 0/1 i.e. binary constraint
15			$H$9:$J$9	=	binary	This range should be 0/1 i.e. binary constraint
16			$L$3:$L$6	=	1	Each cust. can be served by 1 warehouse at most
17			$L$9	=	2	The value of P i.e. number of open facilities
18						

Appendix E. Spreadsheet to FRL for the Tight Formulation

This appendix shows the same spreadsheet model as shown in Appendix C which represents the p-median problem’s tight formulation. However, in this version, all the relevant sections are highlighted as described in Appendix A. These sections serve as the starting point for factorizing this spreadsheet model as described in Appendix A. Following the spreadsheet model, its factorized FRL schema in BNF is shown. Note that this includes extended features not part of the original schema as defined by Isakowitz et al. (1995), such as mn_table and mn_bin_table types.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		Location			Annual											Solver Parameters		Values	Comments
2	Customer Data	X	Y		Shipments											Target Cell	:	$B$22	
3	C1	5	11		200											Equal to	:	Min	
4	C2	10	5		150														
5	C3	0	12		200											By Changing Cells:			
6	C4	12	0		300											$G$16:$I$19			
7																$G$21:$I$21			
8		Location																	
9	Warehouse Data	X	Y													Constraints:			
10	W1	6	8													$G$16:$G$19	<=	$G$21	No customer can be served from closed facility
11	W2	4	7													$H$16:$H$19	<=	$H$21	No customer can be served from closed facility
12	W3	3	4													$I$16:$I$19	<=	$I$21	No customer can be served from closed facility
13																$G$16:$I$19	=	binary	This range can be 0 or 1 i.e. binary constraint
14						Warehouse to Customer Status										$G$21:$I$21	=	binary	This range can be 0 or 1 i.e. binary constraint
15	Dist: Cust to Ware	W1	W2	W3			W1	W2	W3	Sum	Total warehouse per customer					$K$16:$K$19	=	1	Each cust. can be served by 1 warehouse at most
16	C1	3.16	4.12	7.28		C1	1	0	0	=	1					$K$21	=	2	The value of P i.e. number of open facilities
17	C2	5.00	6.32	7.07		C2	1	0	0	=	1								
18	C3	7.21	6.40	8.54		C3	0	0	1	=	1								
19	C4	10.00	10.63	9.85		C4	1	0	0	=	1								
20																			
21							1	0	1	=	2	<<< Total open warehouse = future P							
22	Total Annual Distance	6091	<< Optimal																

Table Relation Customer

Table Relation Warehouse

Matrix Relation CustWare

Bin Matrix Relation Custware

Attribute added to WarehouseRelation

Attribute added to Customer Relation

Vector Relation AnnDist

Vector Relation SolverParm

Table Relation SolverCC

Table Relation SolverConstraints

Vector Relation TOpenWare

Using the process described in Isakowitz et al. (1995), the spreadsheet's components are identified as a series of relations which can be used in order to convert the model into its FRL format. This schema can then be stored or used to re-create the original spreadsheet model.

```
Relation Customer Type Table
    CustomerID: Text Key
    Xlocation: Numeric
    Ylocation: Numeric
    AnnShip: Numeric
    TotalWarehousePerCust: Sum(CustWareBin[CustomerID, *])
    /* last attribute gets the total of all warehouses for a specific CustomerID */

Relation Warehouse Type Table
    WarehouseID: Text Key
    Xlocation: Numeric
    Ylocation: Numeric
    WarehouseOpenFlag: Logical
    /* last attribute is a flag used to determine if a warehouse is open */

Relation CustWare Type mn_table
    (CustomerID, WarehouseID): Text Key Ref[Customer.CustomerID,
                                            Warehouse.WarehouseID]
    C2W: SQRT((Customer[CustWare.CustomerID].Xlocation-
               Warehouse[CustWare.WarehouseID].Xlocation)^2 +
               (Customer[CustWare.CustomerID].Ylocation-
                Warehouse[CustWare.WarehouseID].Ylocation)^2)

Relation CustWareBin Type mn_bin_table
    (CustomerID, WarehouseID): Text Key Ref [Customer.CustomerID,
                                             Warehouse.WarehouseID]
    C2W: Logical

Relation TOpenWare Type Vector
    SumOpenWarehouses: Sum(Warehouse[*].WarhouseOpenFlag)

Relation AnnDist Type Vector
    TotalAnnualDistance: Numeric

Relation SolverParm Type Vector
    TargetCell: AnnDist.TotalAnnualDistance
    EqualTo: Min

Relation SolverCC Type ChangingCells
    CustWareBin[*, *].C2W
    Warehouse[*].WarhouseOpenFlag

Relation SolverConstraint Type Constraint
    CustWareBin.[n, *].C2W <= Warehouse[n].WarhouseOpenFlag
```

```
Customer[*].TotalWarehousePerCust = 1
TOpenWare.SumOpenWarehouses = 2
```

Appendix F. Spreadsheet to FRL for the Loose Formulation

This appendix shows the same spreadsheet model as shown in Appendix D which represents the p-median problem's loose formulation. However, in this version, all the relevant sections are highlighted as described in Appendix A. These sections serve as the starting point for factorizing this spreadsheet model as described in Appendix A. Following the spreadsheet model, its factorized FRL schema in BNF is shown. Note that this includes extended features not part of the original schema as defined by Isakowitz et al. (1995), such as mn_table and mn_bin_table types.

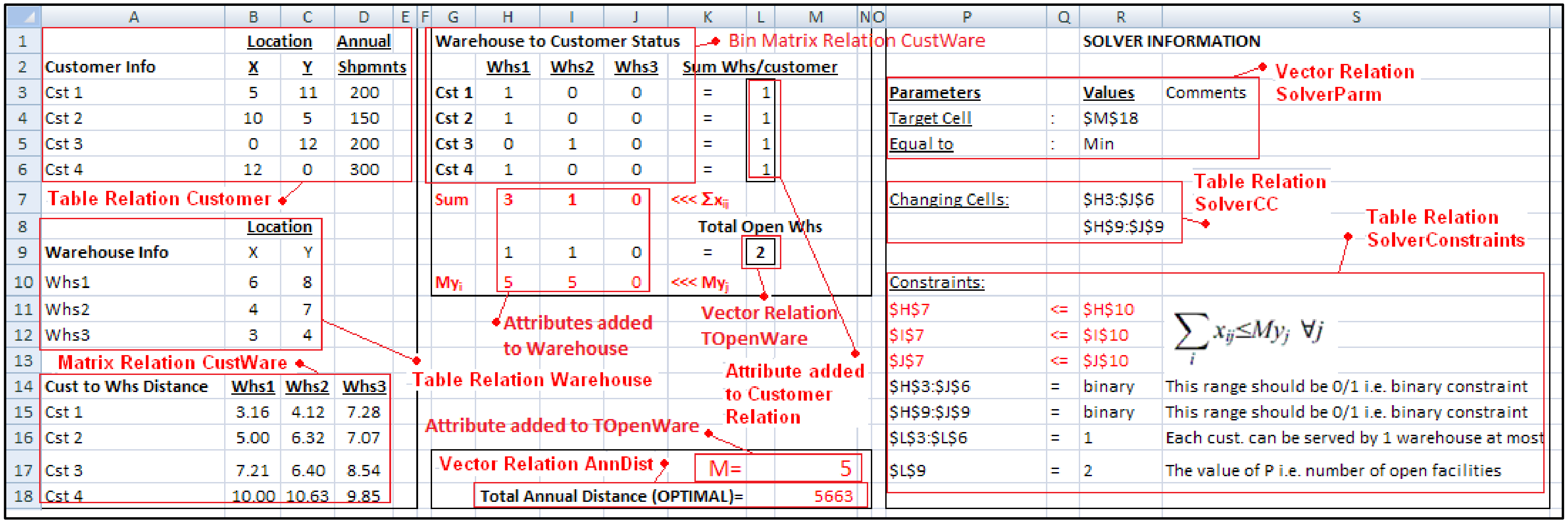


Table Relation Customer
Matrix Relation CustWare
Table Relation Warehouse
Bin Matrix Relation CustWare
Attributes added to Warehouse
Vector Relation TOpenWare
Attribute added to Customer Relation
Attribute added to TOpenWare
Vector Relation AnnDist
Vector Relation SolverParm
Table Relation SolverCC
Table Relation SolverConstraints

	A	B	C	D
1		Location		Annual
2	Customer Info	X	Y	Shpmnts
3	Cst 1	5	11	200
4	Cst 2	10	5	150
5	Cst 3	0	12	200
6	Cst 4	12	0	300
8		Location		
9	Warehouse Info	X	Y	
10	Whs1	6	8	
11	Whs2	4	7	
12	Whs3	3	4	
14	Cust to Whs Distance	Whs1	Whs2	Whs3
15	Cst 1	3.16	4.12	7.28
16	Cst 2	5.00	6.32	7.07
17	Cst 3	7.21	6.40	8.54
18	Cst 4	10.00	10.63	9.85

	G	H	I	J	K	L	M
1	Warehouse to Customer Status						
2		Whs1	Whs2	Whs3	Sum Whs/customer		
3	Cst 1	1	0	0	=	1	
4	Cst 2	1	0	0	=	1	
5	Cst 3	0	1	0	=	1	
6	Cst 4	1	0	0	=	1	
7	Sum	3	1	0	<<< Σx_{ij}		
8					Total Open Whs		
9		1	1	0	=	2	
10	My_i	5	5	0	<<< My_j		
17					M=		5
18		Total Annual Distance (OPTIMAL)=					5663

	P	Q	R	S
1			SOLVER INFORMATION	
3	Parameters		Values	Comments
4	Target Cell	:	$M$18	
5	Equal to	:	Min	
7	Changing Cells:		$H3:$J$6	
8			$H$9:$J$9	
10	Constraints:			
11	$H$7	<=	$H$10	$\sum_i x_{ij} \leq My_j \ \forall j$
12	$I$7	<=	$I$10	
13	$J$7	<=	$J$10	
14	$H$3:$J$6	=	binary	This range should be 0/1 i.e. binary constraint
15	$H$9:$J$9	=	binary	This range should be 0/1 i.e. binary constraint
16	$L$3:$L$6	=	1	Each cust. can be served by 1 warehouse at most
17	$L$9	=	2	The value of P i.e. number of open facilities

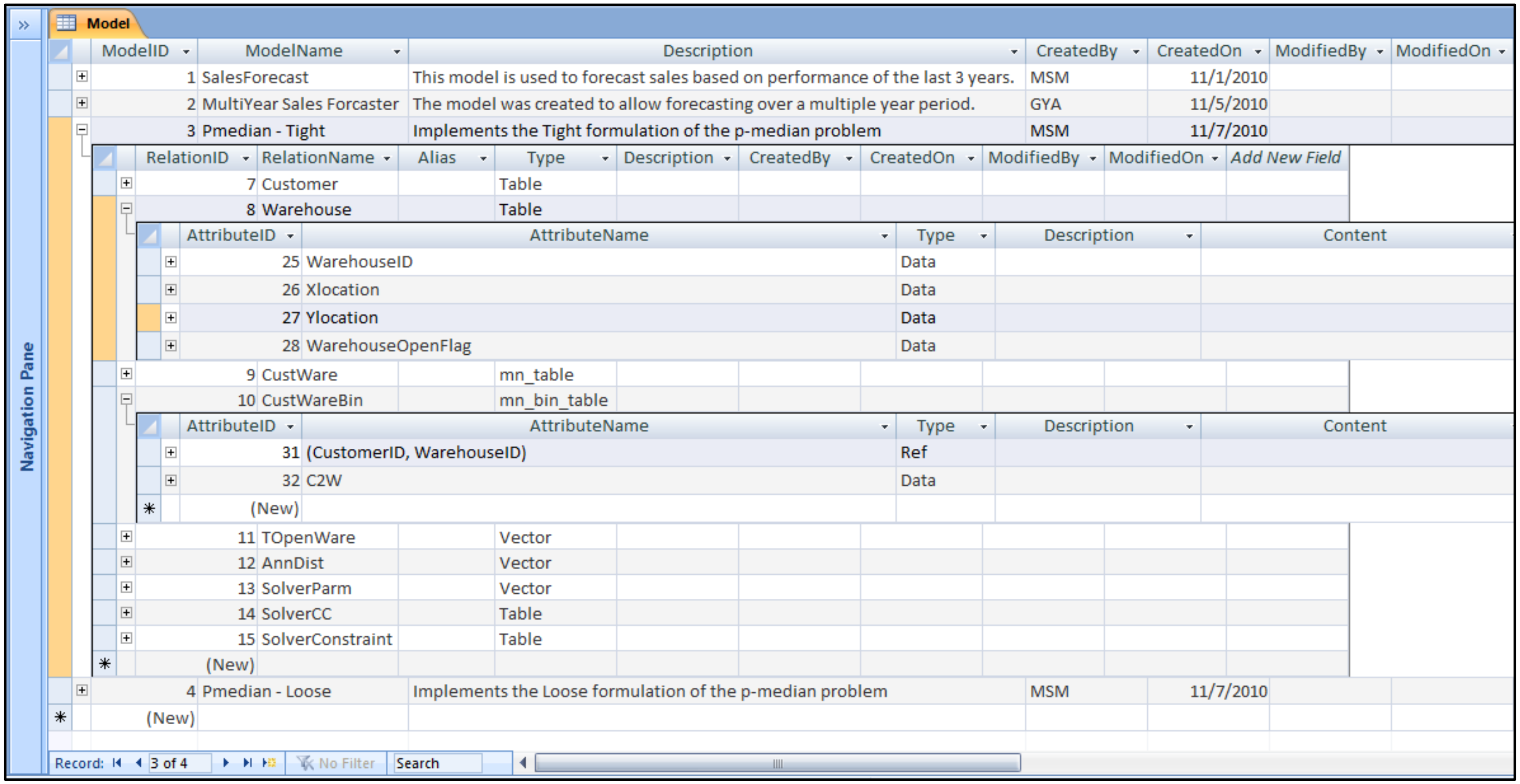
Using the process described in Isakowitz et al. (1995), the spreadsheet's components are identified as a series of relations which can be used in order convert the model into its FRL format. This schema can then be stored or used to re-create the original spreadsheet model.

```
Relation Customer Type Table
    CustomerID: Text Key
    Xlocation: Numeric
    Ylocation: Numeric
    AnnShip: Numeric
    TotalWarehousePerCust: Sum(CustWareBin[CustomerID, *])
    /* last attribute gets the total of all warehouses for a specific CustomerID */

Relation Warehouse Type Table
    WarehouseID: Text Key
    Xlocation: Numeric
    Ylocation: Numeric
    SumXij: Numeric
    WarehouseOpenFlag: Logical
    Myi: Numeric
    /* last attribute is a flag used to determine if a warehouse is open */

Relation CustWare Type mn_table
    (CustomerID, WarehouseID): Text Key Ref[Customer.CustomerID,
                                            Warehouse.WarehouseID]
    C2W: SQRT((Customer[CustWare.CustomerID].Xlocation-
               Warehouse[CustWare.WarehouseID].Xlocation)^2 +
               (Customer[CustWare.CustomerID].Ylocation-
                Warehouse[CustWare.WarehouseID].Ylocation)^2)

Relation CustWareBin Type mn_bin_table
    (CustomerID, WarehouseID): Text Key Ref [Customer.CustomerID,
                                            Warehouse.WarehouseID]
    C2W: Logical

Relation TOpenWare Type Vector
    SumOpenWarehouses: Sum(Warehouse[*].WarhouseOpenFlag)
    ConstantM: Numeric

Relation AnnDist Type Vector
    TotalAnnualDistance: Numeric

Relation SolverParm Type Vector
    TargetCell: AnnDist.TotalAnnualDistance
    EqualTo: Min
```

```
Relation SolverCC Type ChangingCells
    CustWareBin[*, *].C2W
    Warehouse[*].WarhouseOpenFlag

Relation SolverConstraint Type Constraint
    Warehouse.[n].SumXij <= Warehouse[n].Myi
    Customer[*].TotalWarehousePerCust = 1
    TOpenWare.SumOpenWarehouses = 2
```

Appendix G. Schema of Tight Formulation Stored in Sample SQL DB

This appendix shows the internal storage format of the proposed prototype system. It shows the relationship between three main tables which store various parts of a model: Model, Relation, and Attribute. Model with ModleID = 3 is expanded to show the list of Relations that compose it. And for the same model, where RelationID = 8 and 10, the attributes that make it up are shown. More information about the structure of these tables is available in Chapter 3. The example shown in the following figure is the instantiation of the *tight* formulation of the p-median problem as used in Scenario One of Chapter 4, which is also shown in Appendix C and Appendix E.



ModelID	ModelName	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
1	SalesForecast	This model is used to forecast sales based on performance of the last 3 years.	MSM	11/1/2010		
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.	GYA	11/5/2010		
3	Pmedian - Tight	Implements the Tight formulation of the p-median problem	MSM	11/7/2010		

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn	*Add New Field*
7	Customer		Table						
8	Warehouse		Table						

AttributeID	AttributeName	Type	Description	Content
25	WarehouseID	Data		
26	Xlocation	Data		
27	Ylocation	Data		
28	WarehouseOpenFlag	Data		

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
9	CustWare		mn_table					
10	CustWareBin		mn_bin_table					

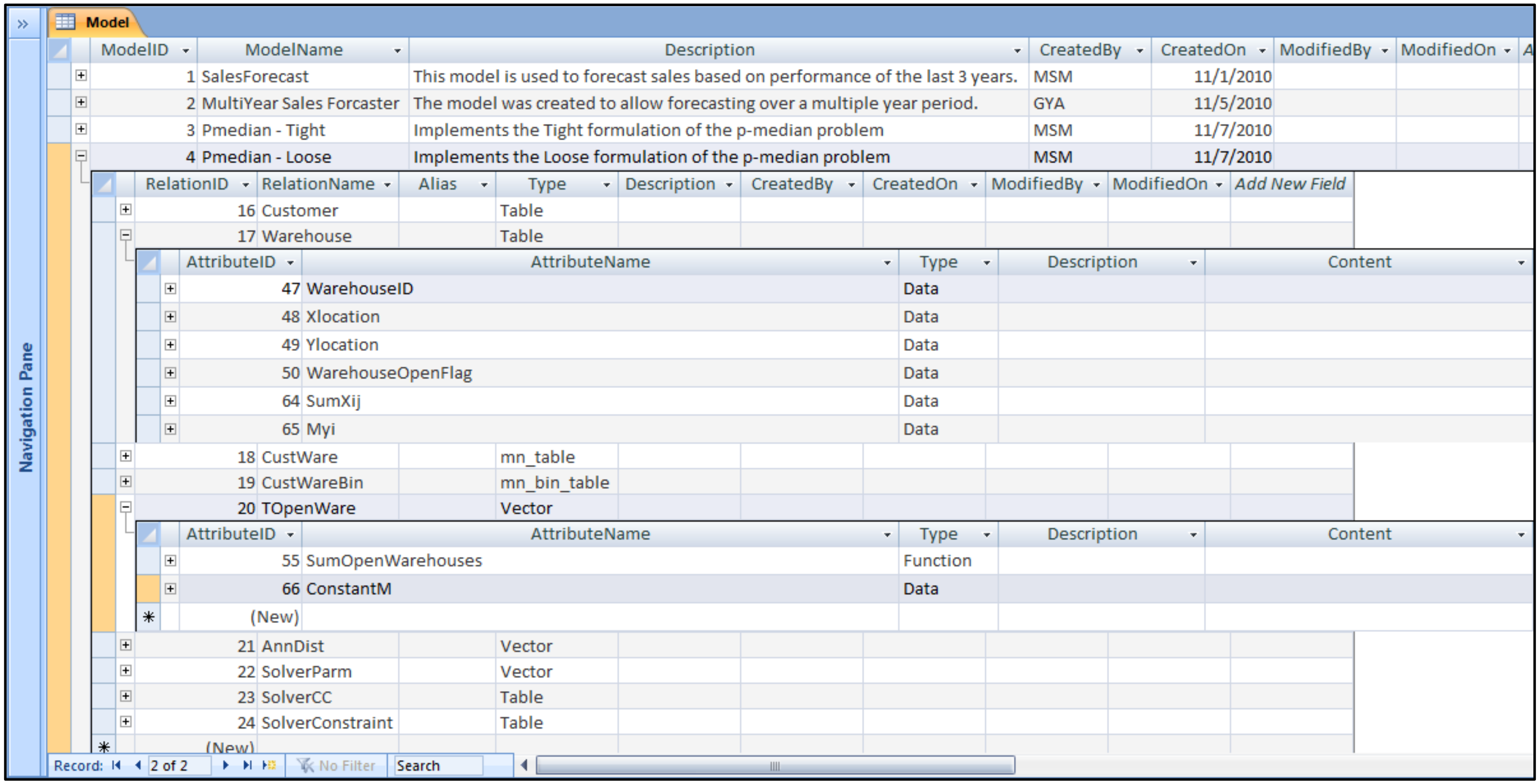
AttributeID	AttributeName	Type	Description	Content
31	(CustomerID, WarehouseID)	Ref		
32	C2W	Data		
(New)				

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
11	TOpenWare		Vector					
12	AnnDist		Vector					
13	SolverParm		Vector					
14	SolverCC		Table					
15	SolverConstraint		Table					
(New)								

ModelID	ModelName	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
4	Pmedian - Loose	Implements the Loose formulation of the p-median problem	MSM	11/7/2010		
(New)						

Appendix H. Schema of Loose Formulation Stored in Sample SQL DB

This appendix shows the internal storage format of the proposed prototype system. It shows the relationship between three main tables which store various parts of a model: Model, Relation, and Attribute. Model with ModleID = 4 is expanded to show the list of Relations that compose it. And for the same model, where RelationID = 17 and 20, the attributes that make it up are shown. More information about the structure of these tables is available in Chapter 3. The example shown in the following figure is the instantiation of the *loose* formulation of the p-median problem as used in Scenario One of Chapter 4, which is also shown in Appendix D and Appendix F.



ModelID	ModelName	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn	A
1	SalesForecast	This model is used to forecast sales based on performance of the last 3 years.	MSM	11/1/2010			
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.	GYA	11/5/2010			
3	Pmedian - Tight	Implements the Tight formulation of the p-median problem	MSM	11/7/2010			
4	Pmedian - Loose	Implements the Loose formulation of the p-median problem	MSM	11/7/2010			

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn	*Add New Field*
16	Customer		Table						
17	Warehouse		Table						

AttributeID	AttributeName	Type	Description	Content
47	WarehouseID	Data		
48	Xlocation	Data		
49	Ylocation	Data		
50	WarehouseOpenFlag	Data		
64	SumXij	Data		
65	Myi	Data		

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn	*Add New Field*
18	CustWare		mn_table						
19	CustWareBin		mn_bin_table						
20	TOpenWare		Vector						

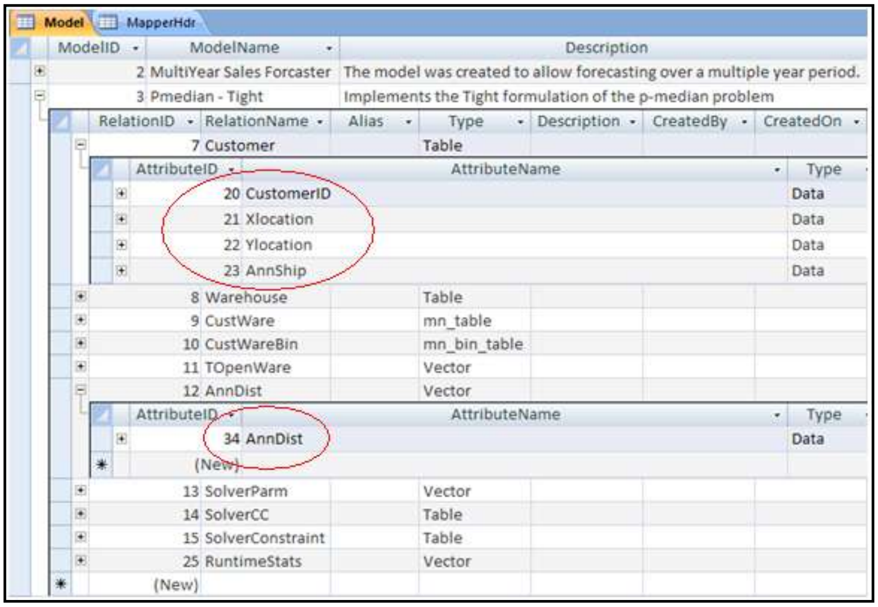
AttributeID	AttributeName	Type	Description	Content
55	SumOpenWarehouses	Function		
66	ConstantM	Data		
(New)				

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn	*Add New Field*
21	AnnDist		Vector						
22	SolverParm		Vector						
23	SolverCC		Table						
24	SolverConstraint		Table						
(New)									

Appendix I. Model Mapping

Chapter 3 describes the logic behind mapping different models and provides an example of the mechanism and data structures required. This appendix shows an instantiation of mapping for the problem shown in Scenario One of Chapter 4.

As described in Chapter 3, at the lowest level of a model's schema information hierarchy are its attributes. Mapping two or more models consists of creating a link between the attributes of the models. Such a link serves as a means to identify data attributes that are similar, but have different names. Some of the attributes of the 'Pmedian – Tight' model are highlighted in red within Figure I-1.



ModelID	ModelName	Description
2	MultiYear Sales Forcaster	The model was created to allow forecasting over a multiple year period.
3	Pmedian - Tight	Implements the Tight formulation of the p-median problem

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
7	Customer		Table			

AttributeID	AttributeName	Type
20	CustomerID	Data
21	Xlocation	Data
22	Ylocation	Data
23	AnnShip	Data

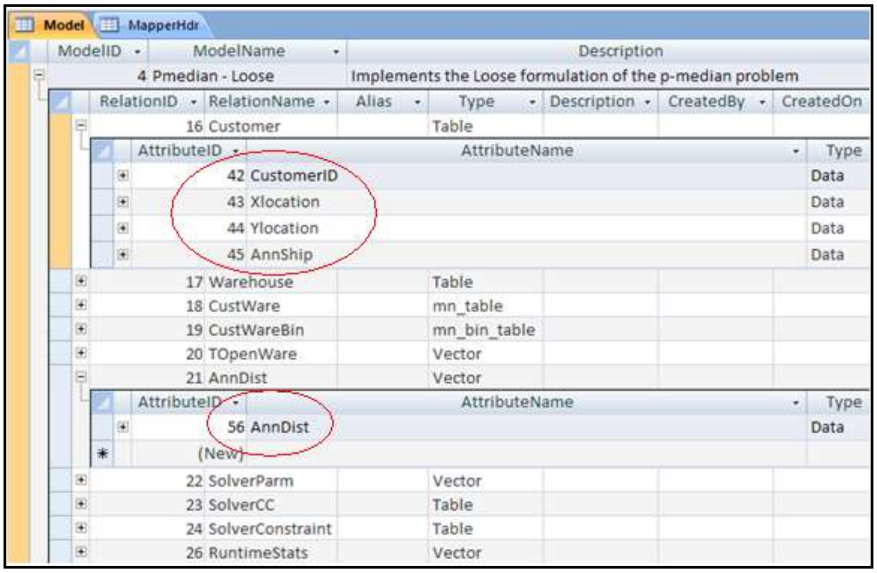
RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
8	Warehouse		Table			
9	CustWare		mn_table			
10	CustWareBin		mn_bin_table			
11	TOpenWare		Vector			
12	AnnDist		Vector			

AttributeID	AttributeName	Type
34	AnnDist	Data
(New)		

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
13	SolverParm		Vector			
14	SolverCC		Table			
15	SolverConstraint		Table			
25	RuntimeStats		Vector			
(New)						

Figure I-1. Some Attributes of the 'Pmedian – Tight' Model

Figure I-2 highlights in red the attributes of the 'Pmedian – Loose' model. Figure I-2 and Figure I-3 are showing that their attributes have the same names. Such a case would not be common. However, even if attributes names are the same, their internal unique identifiers are different. In this case, AttributeID for CustomerID in Figure I-1 is 20 while that of CustomerID in Figure I-2 is 42.



ModelID	ModelName	Description
4	Pmedian - Loose	Implements the Loose formulation of the p-median problem

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
16	Customer		Table			

AttributeID	AttributeName	Type
42	CustomerID	Data
43	Xlocation	Data
44	Ylocation	Data
45	AnnShip	Data

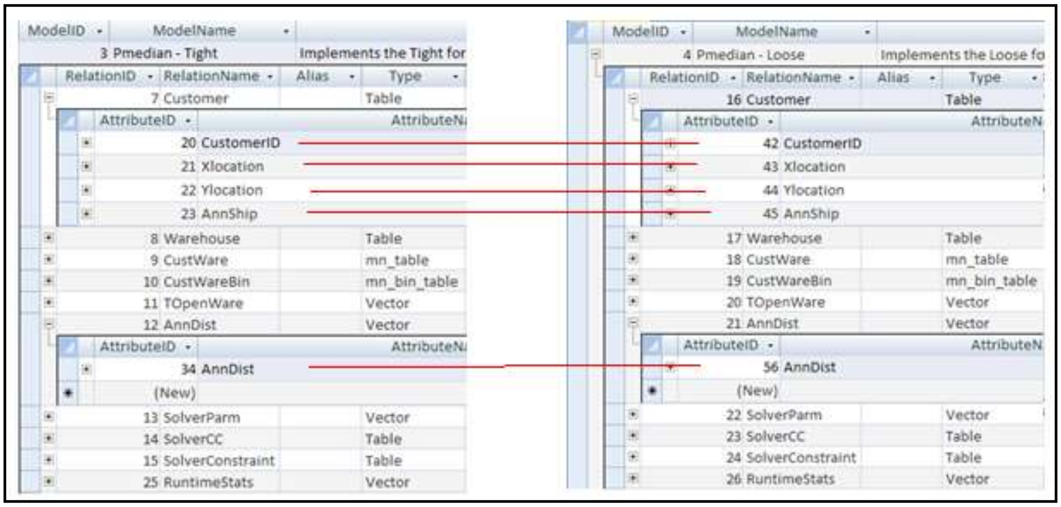
RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
17	Warehouse		Table			
18	CustWare		mn_table			
19	CustWareBin		mn_bin_table			
20	TOpenWare		Vector			
21	AnnDist		Vector			

AttributeID	AttributeName	Type
56	AnnDist	Data
(New)		

RelationID	RelationName	Alias	Type	Description	CreatedBy	CreatedOn
22	SolverParm		Vector			
23	SolverCC		Table			
24	SolverConstraint		Table			
26	RuntimeStats		Vector			

Figure I-2. Attributes of the ‘Pmedian – Loose’ Model

Mapping these two models will require a user interface which aids the user in creating a relationship between the various model attributes. Figure I-3 show one such possible interface whereby the user drags-and-drops attributes from one model to the other.



ModelID	ModelName	
3	Pmedian - Tight	Implements the Tight for

RelationID	RelationName	Alias	Type
7	Customer		Table
8	Warehouse		Table
9	CustWare		mn_table
10	CustWareBin		mn_bin_table
11	TOpenWare		Vector
12	AnnDist		Vector
13	SolverParm		Vector
14	SolverCC		Table
15	SolverConstraint		Table
25	RuntimeStats		Vector

AttributeID	AttributeN
20	CustomerID
21	Xlocation
22	Ylocation
23	AnnShip
34	AnnDist
(New)	

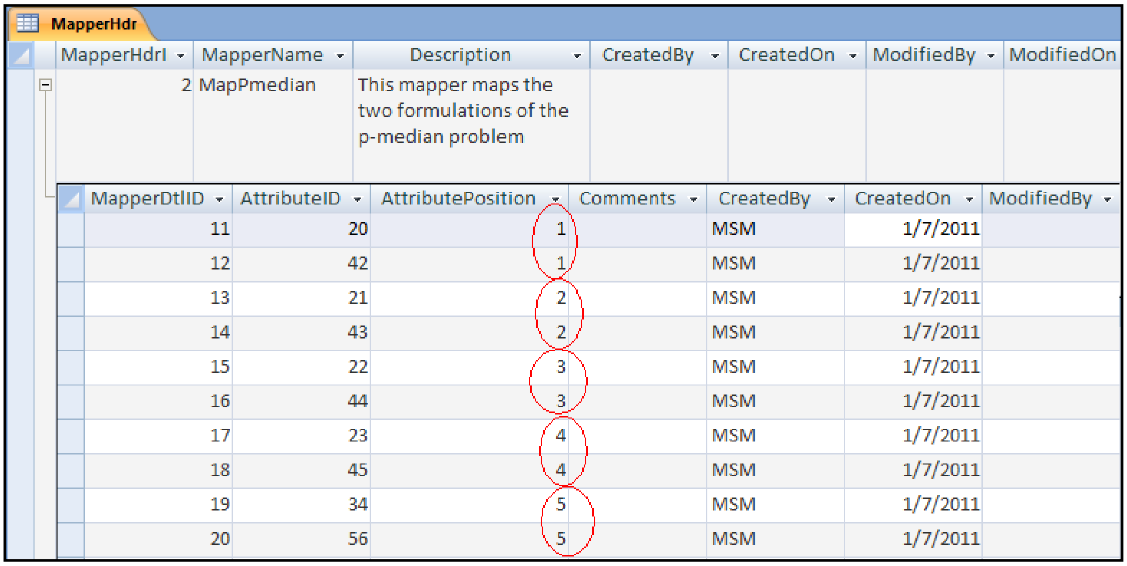
ModelID	ModelName	
4	Pmedian - Loose	Implements the Loose fo

RelationID	RelationName	Alias	Type
16	Customer		Table
17	Warehouse		Table
18	CustWare		mn_table
19	CustWareBin		mn_bin_table
20	TOpenWare		Vector
21	AnnDist		Vector
22	SolverParm		Vector
23	SolverCC		Table
24	SolverConstraint		Table
26	RuntimeStats		Vector

AttributeID	AttributeN
42	CustomerID
43	Xlocation
44	Ylocation
45	AnnShip
56	AnnDist
(New)	

Figure I-3. Mapping Model Attributes Between Two Models

Such a mechanism will internally save the model mapping in two different tables: MapperHdr and MapperDtl. The first table contains general information about the mapping, while the second contains the list of AttributeIDs and their mapping. Figure I-4 shows the internal storage of the mapping. The column AttributePosition contains a common number for the attributes that are mapped.



MapperHdrI	MapperName	Description	CreatedBy	CreatedOn	ModifiedBy	ModifiedOn
2	MapPmedian	This mapper maps the two formulations of the p-median problem				

MapperDtlID	AttributeID	AttributePosition	Comments	CreatedBy	CreatedOn	ModifiedBy
11	20	1		MSM	1/7/2011	
12	42	1		MSM	1/7/2011	
13	21	2		MSM	1/7/2011	
14	43	2		MSM	1/7/2011	
15	22	3		MSM	1/7/2011	
16	44	3		MSM	1/7/2011	
17	23	4		MSM	1/7/2011	
18	45	4		MSM	1/7/2011	
19	34	5		MSM	1/7/2011	
20	56	5		MSM	1/7/2011	

Figure I-4. Mapping of Attributes

Appendix J. Sample Output From Microsoft Excel Solver

This appendix shows a sample of the output generated by the Microsoft Excel Solver. The information in this output is specific to the instance of the tight formulated p-median as used in Scenario One of Chapter 4, which is also shown in Appendix C and Appendix E. The following output shows the cells which hold the optimal value (3353), the adjustable cells used by the solver, and cells containing the constraints of the model. This output is provided as a reference to be compared with the output of other solvers such as Palisade Evolver (Appendix K) and Lindo’s What’s Best (Appendix L).

Microsoft Excel 12.0 Answer Report

Worksheet: [p-median3 - 4C3W - Tight - 0.xlsx]Sheet1

Report Created: 12/1/2010 8:22:33 PM

Target Cell (Min)

Cell	Name	Original Value	Final Value
$B$22	Total Annual Distance W1	0	3353

Adjustable Cells

Cell	Name	Original Value	Final Value
$G$16	C1 W1	0	1
$H$16	C1 W2	0	0

$I$16	C1 W3	0	0
$G$17	C2 W1	0	0.999999999
$H$17	C2 W2	0	0
$I$17	C2 W3	0	0
$G$18	C3 W1	0	0
$H$18	C3 W2	0	0
$I$18	C3 W3	0	1
$G$19	C4 W1	0	1
$H$19	C4 W2	0	0
$I$19	C4 W3	0	0
$G$21		0	1
$H$21		0	0
$I$21		0	1

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$K$21	#NAME?	2	$K$21=2	Not Binding	0
$G$16	C1 W1	1	$G$16<=$G$21	Binding	0
$G$17	C2 W1	0.999999999	$G$17<=$G$21	Binding	0
$G$18	C3 W1	0	$G$18<=$G$21	Not Binding	1
$G$19	C4 W1	1	$G$19<=$G$21	Binding	0
$H$16	C1 W2	0	$H$16<=$H$21	Binding	0
$H$17	C2 W2	0	$H$17<=$H$21	Binding	0
$H$18	C3 W2	0	$H$18<=$H$21	Binding	0
$H$19	C4 W2	0	$H$19<=$H$21	Binding	0
$I$16	C1 W3	0	$I$16<=$I$21	Not Binding	1

$I$17	C2 W3		0	$I$17<=$I$21	Not Binding	1
$I$18	C3 W3		1	$I$18<=$I$21	Binding	0
$I$19	C4 W3		0	$I$19<=$I$21	Not Binding	1
$K$16		#NAME?	1	$K$16=1	Not Binding	0
$K$17		#NAME?	0.999999999	$K$17=1	Not Binding	0
$K$18		#NAME?	1	$K$18=1	Not Binding	0
$K$19		#NAME?	1	$K$19=1	Not Binding	0
$G$21			1	$G$21=binary	Binding	0
$H$21			0	$H$21=binary	Binding	0
$I$21			1	$I$21=binary	Binding	0
$G$16	C1 W1		1	$G$16=binary	Binding	0
$H$16	C1 W2		0	$H$16=binary	Binding	0
$I$16	C1 W3		0	$I$16=binary	Binding	0

$G$17	C2 W1	0.999999999	$G$17=binary	Binding	0
$H$17	C2 W2	0	$H$17=binary	Binding	0
$I$17	C2 W3	0	$I$17=binary	Binding	0
$G$18	C3 W1	0	$G$18=binary	Binding	0
$H$18	C3 W2	0	$H$18=binary	Binding	0
$I$18	C3 W3	1	$I$18=binary	Binding	0
$G$19	C4 W1	1	$G$19=binary	Binding	0
$H$19	C4 W2	0	$H$19=binary	Binding	0
$I$19	C4 W3	0	$I$19=binary	Binding	0

Appendix K. Sample Output From Palisade Evolver Solver

This appendix shows a sample of the output generated by the Palisade Evolver Solver. The information in this output is specific to the instance of the tight formulated p-median as used in Scenario One of Chapter 4, the model which is also shown in Appendix C and Appendix E. This output is provided as a reference to be compared with the output of other solvers such as Microsoft Excel Solver (Appendix I) and Lindo’s What’s Best (Appendix L).

Evolver: Optimization Summary (Constraint Solver)

Performed By: acer

Date: Tuesday, November 23, 2010 6:51:15 PM

Model: p-median3 - 4C3W - Tight - Evolver.xlsx

Goal	
Type of Goal	14 Constraints Met.

Results	
Total Trials	101
Original Value	12 Constraints Met.
Best Value Found	14 Constraints Met.
Best Simulation Number	101
Time to Find Best Value	0:00:14
Reason Optimization Stopped	Target value reached
Time Optimization Started	11/23/2010 18:50
Time Optimization Finished	11/23/2010 18:50
Total Optimization Time	0:00:14
Adjustable Cell Values	'Sheet1'!$G$16
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$H$16
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$I$16
Original	0
Best	1
Adjustable Cell Values	'Sheet1'!$G$17

Original	0
Best	1
Adjustable Cell Values	'Sheet1'!$H$17
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$I$17
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$G$18
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$H$18
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$I$18
Original	0
Best	1
Adjustable Cell Values	'Sheet1'!$G$19
Original	0
Best	1
Adjustable Cell Values	'Sheet1'!$H$19
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$I$19
Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$G$21
Original	0
Best	1
Adjustable Cell Values	'Sheet1'!$H$21

Original	0
Best	0
Adjustable Cell Values	'Sheet1'!$I$21
Original	0
Best	1

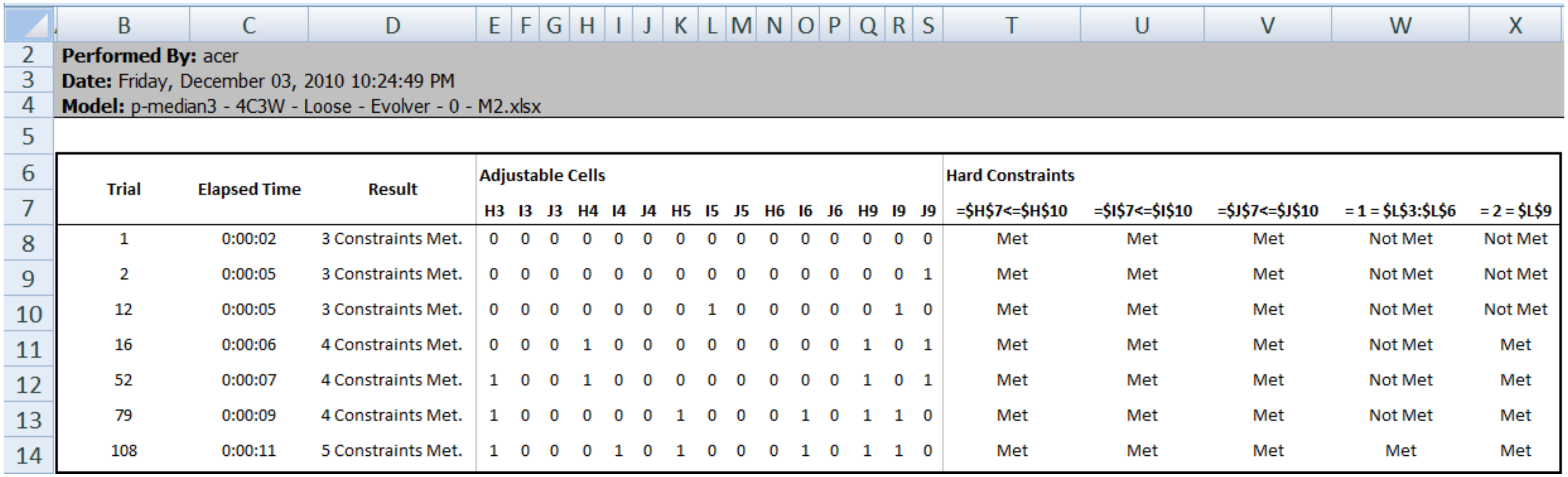
Constraints	
Description	
Definition	=Sheet1!$G$16<=Sheet1!$G$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$G$17<=Sheet1!$G$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$G$18<=Sheet1!$G$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$G$19<=Sheet1!$G$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$H$16<=Sheet1!$H$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$H$17<=Sheet1!$H$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$H$18<=Sheet1!$H$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$H$19<=Sheet1!$H$21

Constraint Type	Hard
Description	
Definition	=Sheet1!$I$16<=Sheet1!$I$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$I$17<=Sheet1!$I$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$I$18<=Sheet1!$I$21
Constraint Type	Hard
Description	
Definition	=Sheet1!$I$19<=Sheet1!$I$21
Constraint Type	Hard
Description	
Definition	= 1 = Sheet1!$K$16:$K$19
Constraint Type	Hard
Description	
Definition	= 2 = Sheet1!$K$21
Constraint Type	Hard

Adjustable Cells	
Description	
Solving Method	Recipe
Mutation Rate	0.1
Crossover Rate	0.5
Cell Range	0 <= 'Sheet1'!$G$16:$I$19 <= 1 [integers]
Cell Range	0 <= 'Sheet1'!$G$21:$I$21 <= 1 [integers]
Operators	Default parent selection Default mutation Default crossover

	Default backtrack
	Arithmetic crossover
	Heuristic crossover
	Cauchy mutation
	Boundary mutation
	Non-uniform mutation
	Linear
	Local search

Optimization Settings	
General	
Population Size	50
Optimization Random Number Seed	186963400 (Chosen Randomly)
Optimization Runtime	
Trials	FALSE
Time	FALSE
Progress	FALSE
Formula	FALSE
Stop on Error	FALSE
View	
Minimize Excel at Start	FALSE
Show Excel Recalculations	Every New Best Trial
Keep Log of All Trials	TRUE
Macros	
At Start of Optimization	N/A
Before Recalculation	N/A
After Recalculation	N/A
After Storing Output	N/A
At End of Optimization	N/A



Performed By: acer
Date: Friday, December 03, 2010 10:24:49 PM
Model: p-median3 - 4C3W - Loose - Evolver - 0 - M2.xlsx

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
6	Trial	Elapsed Time	Result	Adjustable Cells															Hard Constraints				
7				H3	I3	J3	H4	I4	J4	H5	I5	J5	H6	I6	J6	H9	I9	J9	=$H$7<=$H$10	=$I$7<=$I$10	=$J$7<=$J$10	= 1 = $L$3:$L$6	= 2 = $L$9
8	1	0:00:02	3 Constraints Met.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Met	Met	Met	Not Met	Not Met
9	2	0:00:05	3 Constraints Met.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Met	Met	Met	Not Met	Not Met
10	12	0:00:05	3 Constraints Met.	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	Met	Met	Met	Not Met	Not Met
11	16	0:00:06	4 Constraints Met.	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	Met	Met	Met	Not Met	Met
12	52	0:00:07	4 Constraints Met.	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	Met	Met	Met	Not Met	Met
13	79	0:00:09	4 Constraints Met.	1	0	0	0	0	0	1	0	0	0	1	0	1	1	0	Met	Met	Met	Not Met	Met
14	108	0:00:11	5 Constraints Met.	1	0	0	0	1	0	1	0	0	0	1	0	1	1	0	Met	Met	Met	Met	Met

Appendix L. Sample Output From Lindo's What's Best Solver

This appendix shows a sample of the output generated by Lindo's What's Best Solver. The information in this output is specific to the instance of the tight formulated p-median as used in Scenario One of Chapter 4, the model which is also shown in Appendix C and Appendix E. This output is provided as a reference to be compared with the output of other solvers such as Microsoft Excel Solver (Appendix I) and Palisade Evolver (Appendix K).

```
What'sBest!® 9.0.5.0 (Sep 24, 2009) - Library 5.0.1.431 - Status Report -

DATE GENERATED:                              Nov 10, 2010                    12:43 PM


MODEL INFORMATION:

CLASSIFICATION DATA          Current   Capacity Limits

--------------------------------------------------------

Numerics                     134

Variables                    38

Adjustables                   15          300

Constraints                  17          150

Integers/Binaries             0/15          30

Nonlinears                   15           30

Coefficients                 82


Minimum coefficient value:        1  on Sheet1!K16

Minimum coefficient in formula:   Sheet1!K16

Maximum coefficient value:        2  on <RHS>

Maximum coefficient in formula:   Sheet1!P21


MODEL TYPE:            Mixed Integer / Nonlinear


SOLUTION STATUS:       LOCALLY OPTIMAL


OPTIMALITY CONDITION:   SATISFIED
```

```
OBJECTIVE VALUE:       3358.9046030819

DIRECTION:             Minimize

SOLVER TYPE:           Branch-and-Bound

TRIES:                 16714

INFEASIBILITY:         6

BEST OBJECTIVE BOUND:   3358.9046030819

STEPS:                 161

ACTIVE:                 0

SOLUTION TIME:         0 Hours  3 Minutes  0 Seconds

ERROR / WARNING MESSAGES:

***WARNING***
Infeasibility too large for a trusted solution (Help Reference: INFLARG):
Constraint violations exceeding tolerances are found. Check the solution carefully
before proceeding. You may be able to resolve this warning by decreasing the
Feasibility Tolerance in the General Options dialog, or by unchecking the Scale
option in the Linear option dialog box.

***WARNING***
Trial/Temporary License Key.
```

```
***WARNING***

Nonlinearities Present (Help Reference: NLINCELL):

The cells below contain nonlinear expressions. If these cells are used only for

reporting, then, for efficiency, they should be included in a WBOMIT range (refer

to documentation). In some cases, nonlinear cells may be linearized automatically

by the Linearization option that is set in the General Options dialog box. This

warning can be turned off with the Nonlinearity Present checkbox in the

General Options dialog box

(cell addresses listed at bottom of tab).

LISTING:

***WARNING***

List of nonlinear cells:

Sheet1!B22

End of Report
```

Appendix M. Experiment Instance Data Retrieved From Sample SQL DB

This appendix shows the model instance data stored in the prototype system. As shown, it includes 48 different instances, 45 of which are the instances used in the experiments of Scenario One in Chapter 4. The report in this appendix contains the following fields along with descriptions:

ModelID: An internal unique ID generated by the system in order to individually track each model.

RelationID: An internal unique ID generated by the system in order to individually track each *relation* within a model.

AttributeID: An internal unique ID generated by the system in order to individually track each *attribute* within each *relation*.

InstanceHdrID: An internal unique ID generated by the system in order to individually track distinct instances (data) of a model.

InstanceDtlID: An internal unique ID generated by the system in order to individually track distinct instances (data) of a model.

AttributeName: The names given to the different attributes of a model.

AttributeDataType: The data type of the attribute just listed.

AttributeValue: The data value assigned to the attribute

.

```
InstanceHdrID        InstanceDtlID AttributeName        AttributeDataType        AttributeValue          RelationID   AttributeID


        2       Instance: ExcelSolver1   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start zeros           ModelID: 3

                         32        TotalAnnualDista     Numeric                  4495                     12             34

                         33        Runtime              Numeric                  2.1                      25             67


        3       Instance: ExcelSolver2   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start ones            ModelID: 3

                         35        TotalAnnualDista     Numeric                  5663                     12             34

                         36        Runtime              Numeric                  8.6                      25             67


        4       Instance: ExcelSolver3   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start random          ModelID: 3

                         38        TotalAnnualDista     Numeric                  5663                     12             34

                         39        Runtime              Numeric                  10.1                     25             67


        5       Instance: WB1            Desc: Pmedian Tight Formulation instance run on Whats Best - Start zeros             ModelID: 3

                         41        TotalAnnualDista     Numeric                  5663                     12             34

                         42        Runtime              Numeric                  97                       25             67


        6       Instance: WB2            Desc: Pmedian Tight Formulation instance run on Whats Best - Start ones              ModelID: 3

                         53        TotalAnnualDista     Numeric                  5663                     12             34

                         54        Runtime              Numeric                  115                      25             67
```

```
7       Instance: WB3           Desc: Pmedian Tight Formulation instance run on Whats Best - Start random            ModelID: 3

                 65       TotalAnnualDista     Numeric                    5663                      12            34

                 66       Runtime              Numeric                    417                       25            67


8       Instance: Evolver1      Desc: Pmedian Tight Formulation instance run on Evolver - Start zeros                ModelID: 3

                 77       TotalAnnualDista     Numeric                    6091                      12            34

                 80       Runtime              Numeric                    10                        25            67


9       Instance: Evolver2      Desc: Pmedian Tight Formulation instance run on Evolver - Start ones                 ModelID: 3

                 78       TotalAnnualDista     Numeric                    6121                      12            34

                 81       Runtime              Numeric                    14                        25            67


10      Instance: Evolver3      Desc: Pmedian Tight Formulation instance run on Evolver - Start random               ModelID: 3

                 79       TotalAnnualDista     Numeric                    7303                      12            34

                 82       Runtime              Numeric                    14                        25            67


11      Instance: ExcelSolver1  Desc: Pmedian Tight Formulation instance run on Excel Solver - Start zeros           ModelID: 4

                 83       TotalAnnualDista     Numeric                    6046                      21            56

                 84       ConstantM            Numeric                    2                         20            66

                 85       Runtime              Numeric                    5.3                       26            68
```

```
12      Instance: ExcelSolver2   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start ones            ModelID: 4

                 86        TotalAnnualDista     Numeric                 6121                    21             56

                 87        ConstantM            Numeric                 2                       20             66

                 88        Runtime              Numeric                 5.3                     26             68


13      Instance: ExcelSolver3   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start random          ModelID: 4

                 89        TotalAnnualDista     Numeric                 6046                    21             56

                 90        ConstantM            Numeric                 2                       20             66

                 91        Runtime              Numeric                 5.1                     26             68


14      Instance: ExcelSolver1   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start zeros           ModelID: 4

                 92        TotalAnnualDista     Numeric                 5663                    21             56

                 93        ConstantM            Numeric                 3                       20             66

                 94        Runtime              Numeric                 4.3                     26             68


15      Instance: ExcelSolver2   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start ones            ModelID: 4

                 95        TotalAnnualDista     Numeric                 5779                    21             56

                 96        ConstantM            Numeric                 3                       20             66

                 97        Runtime              Numeric                 5.5                     26             68


16      Instance: ExcelSolver3   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start random          ModelID: 4

                 98        TotalAnnualDista     Numeric                 5779                    21             56

                 99        ConstantM            Numeric                 3                       20             66
```

```
          100       Runtime             Numeric                  5.2                      26             68

 17      Instance: ExcelSolver1   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start zeros           ModelID: 4
          101       TotalAnnualDista     Numeric                  5663                     21             56
          102       ConstantM            Numeric                  4                        20             66
          103       Runtime              Numeric                  12.8                     26             68

 18      Instance: ExcelSolver2   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start ones            ModelID: 4
          104       TotalAnnualDista     Numeric                  5663                     21             56
          105       ConstantM            Numeric                  4                        20             66
          106       Runtime              Numeric                  9.3                      26             68

 19      Instance: ExcelSolver3   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start random          ModelID: 4
          107       TotalAnnualDista     Numeric                  5663                     21             56
          108       ConstantM            Numeric                  4                        20             66
          109       Runtime              Numeric                  12.4                     26             68

 23      Instance: ExcelSolver1   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start zeros           ModelID: 4
          110       TotalAnnualDista     Numeric                  5663                     21             56
          111       ConstantM            Numeric                  5                        20             66
          112       Runtime              Numeric                  6.3                      26             68
```

```
24      Instance: ExcelSolver2   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start ones           ModelID: 4

                 113       TotalAnnualDista     Numeric                  5663                     21             56

                 114       ConstantM            Numeric                  5                        20             66

                 115       Runtime              Numeric                  9.4                      26             68


25      Instance: ExcelSolver3   Desc: Pmedian Tight Formulation instance run on Excel Solver - Start random         ModelID: 4

                 116       TotalAnnualDista     Numeric                  5779                     21             56

                 117       ConstantM            Numeric                  5                        20             66

                 118       Runtime              Numeric                  5.3                      26             68


26      Instance: WB1-M2         Desc: Pmedian Tight Formulation instance run on WB - Start zeros                    ModelID: 4

                 119       TotalAnnualDista     Numeric                  5852                     21             56

                 120       ConstantM            Numeric                  2                        20             66

                 121       Runtime              Numeric                  56                       26             68


27      Instance: WB2-M2         Desc: Pmedian Tight Formulation instance run on WB - Start ones                     ModelID: 4

                 122       TotalAnnualDista     Numeric                  5852                     21             56

                 123       ConstantM            Numeric                  2                        20             66

                 124       Runtime              Numeric                  63                       26             68


28      Instance: WB3-M2         Desc: Pmedian Tight Formulation instance run on WB - Start random                   ModelID: 4

                 125       TotalAnnualDista     Numeric                  5852                     21             56

                 126       ConstantM            Numeric                  2                        20             66
```

```
             127       Runtime             Numeric                   51                        26            68

29      Instance: WB1-M3         Desc: Pmedian Tight Formulation instance run on WB - Start zeros                     ModelID: 4
             128       TotalAnnualDista    Numeric                   5663                      21            56
             129       ConstantM           Numeric                   3                         20            66
             130       Runtime             Numeric                   264                       26            68

30      Instance: WB2-M3         Desc: Pmedian Tight Formulation instance run on WB - Start ones                      ModelID: 4
             131       TotalAnnualDista    Numeric                   5663                      21            56
             132       ConstantM           Numeric                   3                         20            66
             133       Runtime             Numeric                   388                       26            68

31      Instance: WB3-M3         Desc: Pmedian Tight Formulation instance run on WB - Start random                    ModelID: 4
             134       TotalAnnualDista    Numeric                   5663                      21            56
             135       ConstantM           Numeric                   3                         20            66
             136       Runtime             Numeric                   277                       26            68

32      Instance: WB1-M4         Desc: Pmedian Tight Formulation instance run on WB - Start zeros                     ModelID: 4
             137       TotalAnnualDista    Numeric                   5663                      21            56
             138       ConstantM           Numeric                   4                         20            66
             139       Runtime             Numeric                   496                       26            68
```

```
33      Instance: WB2-M4         Desc: Pmedian Tight Formulation instance run on WB - Start ones                      ModelID: 4
                 140       TotalAnnualDista     Numeric                 5663                     21            56
                 141       ConstantM            Numeric                 4                        20            66
                 142       Runtime              Numeric                 648                      26            68

34      Instance: WB3-M4         Desc: Pmedian Tight Formulation instance run on WB - Start random                    ModelID: 4
                 143       TotalAnnualDista     Numeric                 5663                     21            56
                 144       ConstantM            Numeric                 4                        20            66
                 145       Runtime              Numeric                 504                      26            68

35      Instance: WB1-M5         Desc: Pmedian Tight Formulation instance run on WB - Start zeros                     ModelID: 4
                 146       TotalAnnualDista     Numeric                 5663                     21            56
                 147       ConstantM            Numeric                 5                        20            66
                 148       Runtime              Numeric                 455                      26            68

36      Instance: WB2-M5         Desc: Pmedian Tight Formulation instance run on WB - Start ones                      ModelID: 4
                 149       TotalAnnualDista     Numeric                 5663                     21            56
                 150       ConstantM            Numeric                 5                        20            66
                 151       Runtime              Numeric                 648                      26            68

37      Instance: WB3-M5         Desc: Pmedian Tight Formulation instance run on WB - Start random                    ModelID: 4
                 152       TotalAnnualDista     Numeric                 5663                     21            56
                 153       ConstantM            Numeric                 5                        20            66
```

```
              154       Runtime              Numeric                 477                      26            68

38     Instance: Evolver1-M2    Desc: Pmedian Tight Formulation instance run on Evolver - Start zeros                ModelID: 4
              155       TotalAnnualDista     Numeric                 6212                     21            56
              156       ConstantM            Numeric                 2                        20            66
              157       Runtime              Numeric                 11                       26            68

39     Instance: Evolver2-M2    Desc: Pmedian Tight Formulation instance run on Evolver - Start ones                 ModelID: 4
              158       TotalAnnualDista     Numeric                 6873                     21            56
              159       ConstantM            Numeric                 2                        20            66
              160       Runtime              Numeric                 19                       26            68

40     Instance: Evolver3-M2    Desc: Pmedian Tight Formulation instance run on Evolver - Start random               ModelID: 4
              161       TotalAnnualDista     Numeric                 6603                     21            56
              162       ConstantM            Numeric                 2                        20            66
              163       Runtime              Numeric                 15                       26            68

41     Instance: Evolver1-M3    Desc: Pmedian Tight Formulation instance run on Evolver - Start zeros                ModelID: 4
              164       TotalAnnualDista     Numeric                 6212                     21            56
              165       ConstantM            Numeric                 3                        20            66
              166       Runtime              Numeric                 10                       26            68
```

```
42    Instance: Evolver2-M3    Desc: Pmedian Tight Formulation instance run on Evolver - Start ones                ModelID: 4

             167      TotalAnnualDista     Numeric                 6091                    21            56

             168      ConstantM            Numeric                 3                       20            66

             169      Runtime              Numeric                 18                      26            68


43    Instance: Evolver3-M3    Desc: Pmedian Tight Formulation instance run on Evolver - Start random              ModelID: 4

             170      TotalAnnualDista     Numeric                 6915                    21            56

             171      ConstantM            Numeric                 3                       20            66

             172      Runtime              Numeric                 19                      26            68


44    Instance: Evolver1-M4    Desc: Pmedian Tight Formulation instance run on Evolver - Start zeros               ModelID: 4

             173      TotalAnnualDista     Numeric                 6023                    21            56

             174      ConstantM            Numeric                 4                       20            66

             175      Runtime              Numeric                 12                      26            68


45    Instance: Evolver2-M4    Desc: Pmedian Tight Formulation instance run on Evolver - Start ones                ModelID: 4

             176      TotalAnnualDista     Numeric                 6869                    21            56

             177      ConstantM            Numeric                 4                       20            66

             178      Runtime              Numeric                 11                      26            68


46    Instance: Evolver3-M4    Desc: Pmedian Tight Formulation instance run on Evolver - Start random              ModelID: 4

             179      TotalAnnualDista     Numeric                 7303                    21            56

             180      ConstantM            Numeric                 4                       20            66
```

```
                181       Runtime              Numeric                  9                          26             68


47      Instance: Evolver1-M5    Desc: Pmedian Tight Formulation instance run on Evolver - Start zeros                ModelID: 4

                182       TotalAnnualDista     Numeric                  5663                       21             56

                183       ConstantM            Numeric                  5                          20             66

                184       Runtime              Numeric                  12                         26             68


48      Instance: Evolver2-M5    Desc: Pmedian Tight Formulation instance run on Evolver - Start ones                 ModelID: 4

                185       TotalAnnualDista     Numeric                  5779                       21             56

                186       ConstantM            Numeric                  5                          20             66

                187       Runtime              Numeric                  18                         26             68


49      Instance: Evolver3-M5    Desc: Pmedian Tight Formulation instance run on Evolver - Start random               ModelID: 4

                188       TotalAnnualDista     Numeric                  6091                       21             56

                189       ConstantM            Numeric                  5                          20             66

                190       Runtime              Numeric                  12                         26             68
```

Appendix N. Experiment Execution Raw Data in Excel Format

This appendix shows data of the same nature as that shown in Appendix M. It shows the results of the 45 different experiments as described in Scenario One of Chapter 4. The columns of the following table include:

ID: A unique ID given to each experiment in order to easily refer to each within this study

Model Name: The spreadsheet model's filename

Version: The model formulation loose or tight.

Value of M: The value of the M parameter which is only valid for loose formulations.

Solver: The name of the solver used with the instance

Binary Starting Values: The starting values in the binary tables of the spreadsheet models which can be set to all zeros, all ones, or a combination of zeros and ones.

Total Duration in Seconds: The total duration it took the solver to find a solution

Optimal Value: The optimal value found by the solver

Comments: Additional comments/observations

Runtime								
ID	**Model Name**	**Version**	**Value of M**	**Solver**	**Binary Starting Values**	**Total Duration in Seconds**	**Optimal Value**	**Comments**
1	p-median3 - 4C3W	Loose	2	Evolver	0	11	6212	
2	p-median3 - 4C3W	Loose	2	MSSolver	0	5.3	6046	non binary
3	p-median3 - 4C3W	Loose	2	WB	0	56	5852	
4	p-median3 - 4C3W	Loose	2	Evolver	1	19	6873	
5	p-median3 - 4C3W	Loose	2	MSSolver	1	5.3	6121	
6	p-median3 - 4C3W	Loose	2	WB	1	63	5852	
7	p-median3 - 4C3W	Loose	2	Evolver	10	15	6603	
8	p-median3 - 4C3W	Loose	2	MSSolver	10	5.1	6046	non binary
9	p-median3 - 4C3W	Loose	2	WB	10	51	5852	
10	p-median3 - 4C3W	Loose	3	Evolver	0	10	6212	

Runtime								
ID	Model Name	Version	Value of M	Solver	Binary Starting Values	Total Duration in Seconds	Optimal Value	Comments
11	p-median3 - 4C3W	Loose	3	MSSolver	0	4.3	5663	
12	p-median3 - 4C3W	Loose	3	WB	0	264	5663	
13	p-median3 - 4C3W	Loose	3	Evolver	1	18	6091	
14	p-median3 - 4C3W	Loose	3	MSSolver	1	5.5	5779	
15	p-median3 - 4C3W	Loose	3	WB	1	388	5663	
16	p-median3 - 4C3W	Loose	3	Evolver	10	19	6915	
17	p-median3 - 4C3W	Loose	3	MSSolver	10	5.2	5779	non binary
18	p-median3 - 4C3W	Loose	3	WB	10	277	5663	
19	p-median3 - 4C3W	Loose	4	Evolver	0	12	6023	
20	p-median3 - 4C3W	Loose	4	MSSolver	0	12.8	5663	

Runtime								
ID	Model Name	Version	Value of M	Solver	Binary Starting Values	Total Duration in Seconds	Optimal Value	Comments
21	p-median3 - 4C3W	Loose	4	WB	0	496	5663	
22	p-median3 - 4C3W	Loose	4	Evolver	1	11	6869	
23	p-median3 - 4C3W	Loose	4	MSSolver	1	9.3	5663	
24	p-median3 - 4C3W	Loose	4	WB	1	648	5663	
25	p-median3 - 4C3W	Loose	4	Evolver	10	9	7303	
26	p-median3 - 4C3W	Loose	4	MSSolver	10	12.4	5663	
27	p-median3 - 4C3W	Loose	4	WB	10	504	5663	
28	p-median3 - 4C3W	Loose	5	Evolver	0	12	5663	
29	p-median3 - 4C3W	Loose	5	MSSolver	0	6.3	5663	
30	p-median3 - 4C3W	Loose	5	WB	0	455	5663	

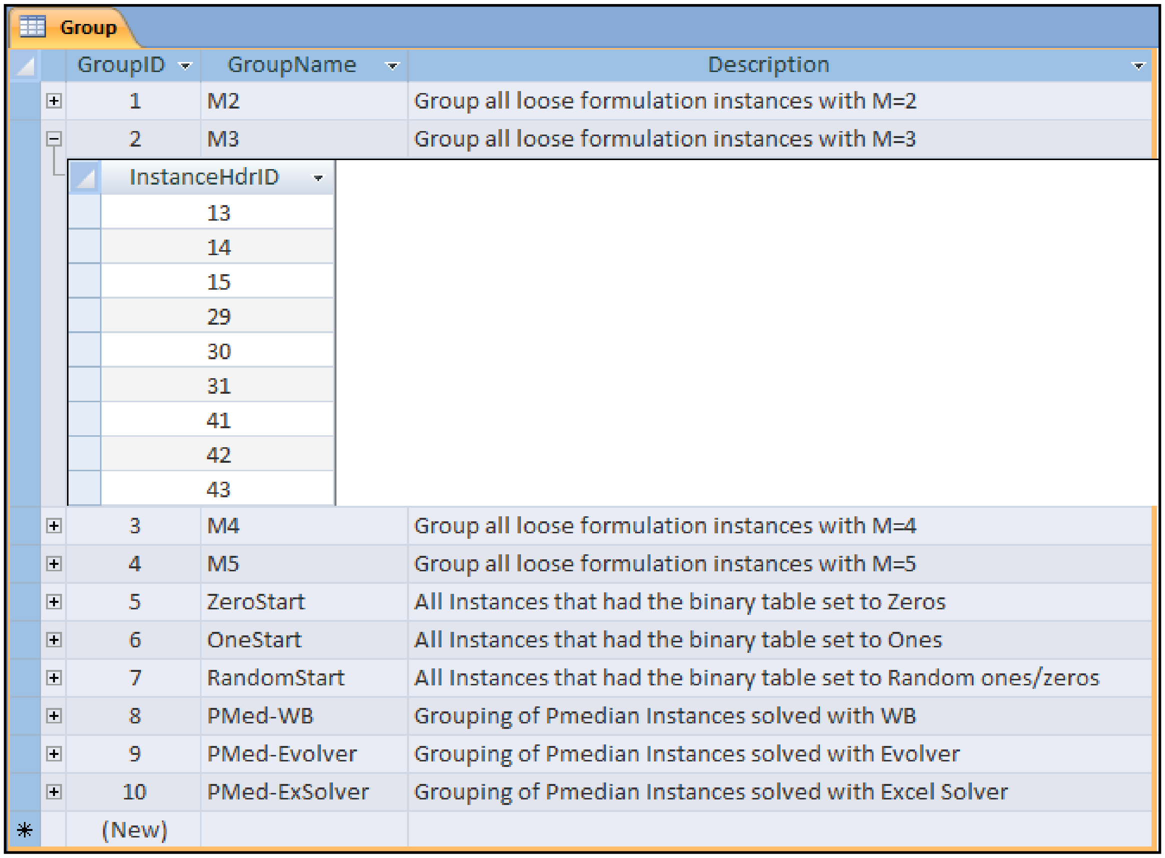
Runtime								
ID	**Model Name**	**Version**	**Value of M**	**Solver**	**Binary Starting Values**	**Total Duration in Seconds**	**Optimal Value**	**Comments**
31	p-median3 - 4C3W	Loose	5	Evolver	1	18	5779	
32	p-median3 - 4C3W	Loose	5	MSSolver	1	9.4	5663	
33	p-median3 - 4C3W	Loose	5	WB	1	648	5663	
34	p-median3 - 4C3W	Loose	5	Evolver	10	12	6091	
35	p-median3 - 4C3W	Loose	5	MSSolver	10	5.3	5779	
36	p-median3 - 4C3W	Loose	5	WB	10	477	5663	
37	p-median3 - 4C3W	Tight		Evolver	0	10	6091	
38	p-median3 - 4C3W	Tight		MSSolver	0	2.1	4495	BAD non bin .8 and .5
39	p-median3 - 4C3W	Tight		WB	0	97	5663	
40	p-median3 - 4C3W	Tight		Evolver	1	14	6121	

Runtime								
ID	Model Name	Version	Value of M	Solver	Binary Starting Values	Total Duration in Seconds	Optimal Value	Comments
41	p-median3 - 4C3W	Tight		MSSolver	1	8.6	5663	
42	p-median3 - 4C3W	Tight		WB	1	115	5663	
43	p-median3 - 4C3W	Tight		Evolver	10	14	7303	
44	p-median3 - 4C3W	Tight		MSSolver	10	10.1	5663	
45	p-median3 - 4C3W	Tight		WB	10	417	5663	

Appendix O. Grouping Model Instances

This appendix shows a method to group instances in anticipation of entering insights to each group. Model instance grouping is described generically in Chapter 3. This appendix shows an instantiation of the Scenario One shown in Chapter 4.

The VBE allows entry and storage of user feedback about models as described in Chapter 3. Comments can be attached to various levels of a model. For example, comments can be attached at the schema level: 1) *model*, 2) *relation*, and/or 3) *attribute*. Comments can also be attached at the data level: 1) *instance header*, 2) *instance detail*, and 3) *instance groupings*. All of these levels except for *instance grouping* are automatically derived by the factorization process (see Appendix A) of the original spreadsheet model. The *instance grouping* is performed by the users themselves. For example, for the experiments with the loose and tight formulations discussed in Chapter 4, the user might group the different instances into various logical groups. The following figure shows 10 different groupings of the model instances, each with a description of the underlying logic. The first four records (GroupName M2, M3, M4, and M5) group instances based on the value of the constant M. The figure is also showing that grouping M2 contains nine different instances (Instance IDs are displayed). Note that it is also possible to derive this same grouping based on the value of M which is stored as an attribute within the model schema. However, with such a scheme, the user will have to create special reports to filter out the needed instances, whereas with the grouping feature the instances remain statically linked.



Group

GroupID	GroupName	Description
1	M2	Group all loose formulation instances with M=2
2	M3	Group all loose formulation instances with M=3

InstanceHdrID
13
14
15
29
30
31
41
42
43

GroupID	GroupName	Description
3	M4	Group all loose formulation instances with M=4
4	M5	Group all loose formulation instances with M=5
5	ZeroStart	All Instances that had the binary table set to Zeros
6	OneStart	All Instances that had the binary table set to Ones
7	RandomStart	All Instances that had the binary table set to Random ones/zeros
8	PMed-WB	Grouping of Pmedian Instances solved with WB
9	PMed-Evolver	Grouping of Pmedian Instances solved with Evolver
10	PMed-ExSolver	Grouping of Pmedian Instances solved with Excel Solver
(New)		

Groupings five through seven group instances based on the starting values of the binary tables. For example, grouping six sets all binary table values to one. Groupings eight to ten group the instances based on the solver used. Once such groupings are formed, users will be able to attach comments to them for later review by other users.

Appendix P. Summary of Insights Created From Scenario One

As described in Chapter 3, at any point while using the VBE a user can record insights based on the categories shown in Figure 27. The last item of this category list titled ‘Other’ allows users to create new categories not covered in the initial list. This appendix shows the summary of insights entered after using Scenario One. The following table shows the following list of items:

FeedbackID: An internal unique ID generated by the system in order to individually track each feedback recorded.

Scope: This field determines the scope to which the current insight refers. The domain of the scope is (Id, Ih, Ig, A, R, M) where these codes respectively refer to ‘InstanceDtl’, ‘InstanceHdr’, ‘Group (grouped instances)’, ‘Attribute’, ‘Relation’, ‘Model’.

RelationID: The internal ID of the record as determined by the Scope. For example, if Scope = ‘A’ i.e. the comment is at the Attribute level, then this RelationID refers to the ID of the record in the Attribute table. For details about the internal representation and storage of user feedback, see Chapter 3 (Figure 28).

Level: This field shows the name of the model and it is shown for reference only to facilitate model identification.

FeedbackCatID: The internal unique ID of the criteria as shown in Figure 27.

Feedback: The title of the criteria as shown in Figure 27.

Comments: The actual user insights in textual format.

FeedbackID	Scope	RelationID	Level	FeedbackCatID	Feedback	Comments
7	Ig	9	Evolver	1	Accuracy of Model	Although this solver (Evolver) runs pretty fast, it most often does not find the optimal value. During the experiments, only 7% of time an optimal value was found.
9	Ig	10	Excel Solver	1	Accuracy of Model	The Excel solver lies somewhere in the middle between Evolver and WB: It found the correct optimal solution 53% of the time during experimentation.
8	Ig	8	WB	1	Accuracy of Model	Although this solver (WB) runs the slowest, it was found to be the most reliable for these p-median problems: the correct optimal value was found 80% of the times during these experiments.
31	Ig	9	Evolver	3	Trust in Model	Least trusted solver… it needs a professional who knows how to tweak in order to get acceptable numbers
32	Ig	10	Excel Solver	3	Trust in Model	Besides the binary table robustness issue, this solver is somewhere in the middle between WB and Evolver. For its price (free), it's not bad granted that the user must be careful checking the values of the constraints to make sure they are not violated
30	Ig	8	WB	3	Trust in Model	This seems to be the most solid package… this reviewer trusts this solver the most.
23	M	4	Loose Model	5	Input Needs	As opposed to tight formulation, the loose formulation requires setting of the constant value, M. It's important to properly set this variable; otherwise an optimal value may not be obtained. From experimentation, the value of this constant should be set

24	M	3	Tight Model	5	Input Needs	No need to set and adjust any parameters for the tight formulation
18	Ig	9	Evolver	7	Performance	Evolver performed a bit slower than Excel Solver (pretty close in fact), but was much faster than WB. Average duration for the loose formulation was 13.8 seconds
19	Ig	9	Evolver	7	Performance	Evolver performed a bit slower than Excel Solver (pretty close in fact), but was much faster than WB. Average duration for the tight formulation was 12.7 seconds
14	Ig	10	Excel Solver	7	Performance	Excel Solver shows the fastest execution times with means of 7.2 seconds for the loose formulation
15	Ig	10	Excel Solver	7	Performance	Excel Solver shows the fastest execution times with means of 6.9 seconds for the tight formulations
16	Ig	8	WB	7	Performance	WB has the slowest execution time: a minimum of 97 seconds for the Tight formulation. These numbers represent roughly a 10 times in orders of magnitude compared to the lowest running times. As the models inputs increase, this could represent a serious performance issue.
17	Ig	8	WB	7	Performance	WB has the slowest execution time ranging up to 648 seconds for the maximum runtime for the Loose formulation. These numbers represent roughly a 65 times in orders of magnitude compared to the lowest running times. As the models inputs values increase, this could represent a serious performance issue
28	Ig	9	Evolver	9	Cost	Evolver costs somewhere between 750 British pounds (for the professional version) and 1000 pounds for the industrial version.

27	Ig	10	Excel Solver	9	Cost	The excel solver comes free with the Microsoft Excel tool.
29	Ig	8	WB	9	Cost	WB comes in different packages ranging from $500 to $5000 depending on version and options
22	Ig	9	Evolver	10	Robustness of Model	Evolver seems pretty robust: all binary constraints remained binary, as opposed to Excel's Solver… however, there are concerns about finding the optimal value: rarely found
20	Ig	10	Excel Solver	10	Robustness of Model	Excel solver exhibits some problems with setting the binary table values. Although these values should be zeros or ones, sometimes there are values other than binary. For example, this instance shows that the optimal value is 4495… this is actually not a correct optimal value since the optimal is 5663. Therefore, the binary constraints were violated in order to get lower optimal number.
21	Ig	8	WB	10	Robustness of Model	WB seems pretty robust: all binary constraints remained binary, as opposed to Excel's Solver.
25	M	4	Loose Model	14	Designer Comments	This model does not contain any designer comments. It includes basic field labels, mostly abbreviated: Cst for Customer and Whse for Warehouse.
26	M	3	Tight Model	14	Designer Comments	This model does not contain any designer comments.
12	Ig	9	Evolver	15	Other: Control and feedback	Evolver provides a detailed progress status as it executes. It provides a detailed final report, however its content is very cryptic: specialized knowledge is required to decipher its content. While executing, it allows the user to pause execution, and later continue execution.

11	Ig	10	Excel Solver	15	Other: Control and feedback	Excel solver shows minimal progress status as it solver executes. It provides minimal final report when the solver completes. While executing, it does not allow the user to pause execution.
13	Ig	8	WB	15	Other: Control and feedback	WB provides an intermediate level (much better than Excel Solver and much less than Evolver) of progress status as it executes. It provides a detailed final report, and its content is in plain English: Most feedback is actionable without the need for specialized knowledge.

Appendix Q. Assigning Weights to Criteria by Performing Pairwise Comparisons

This appendix shows an instance of finding preference weights for the criteria of Scenario Two of Chapter 4. It uses the pairwise comparison method as seen by the user, and shows the internal mechanisms involved in converting user comparative preference into preference weights.

One of the main required tasks of AHP is the assignment of preference weights to the available set of criteria and alternatives. One possible method is to ask the user to assign an importance weight (a percentage figure) for each of these criteria. However, such arbitrary assignment may be cognitively demanding on the user. Therefore, the AHP method recommends a pairwise comparison method which eases the preference weight assignment process. This method solicits user feedback for every two criteria at a time, and the importance of one criterion in reference to the other is recorded. This appendix shows the mechanics of this pairwise comparison process for the problem described in Scenario Two of Chapter 4.

Figure 27 provides a list of possible criteria that may help in the model evaluation and selection process. The following is a list of the reduced set of criteria for which insights were entered in Appendix P: '*Accuracy of Model', 'Trust in Model', 'Input Needs', 'Performance', 'Robustness of Model', 'Designer Comments', 'Cost', and 'Other: Control and feedback'*. Let us assume that the user is only interested in the first five criteria, therefore disregarding the rest.

Figure Q-1 shows a sample criteria comparison tool where a scale is presented in the between two criteria. At the center of the scale is the equality score (1) designating that both criteria are equally important to the user. On either side of the equality mark are the preference scores for the appropriate criterion. For example, by selecting 'strongly favors' located to the right of the equality mark on the comparison scale designates that the user *strongly* favors Criterion2 over Criterion1. Similarly, selecting a score to the left of the equality mark designates that the left-side criterion (Criterion1) is preferred over the right one (Criterion1).

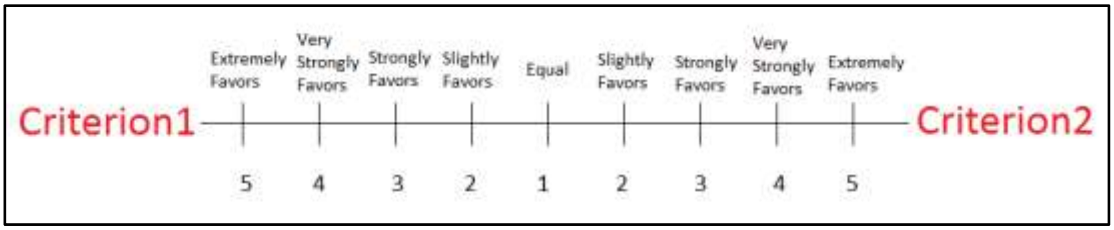
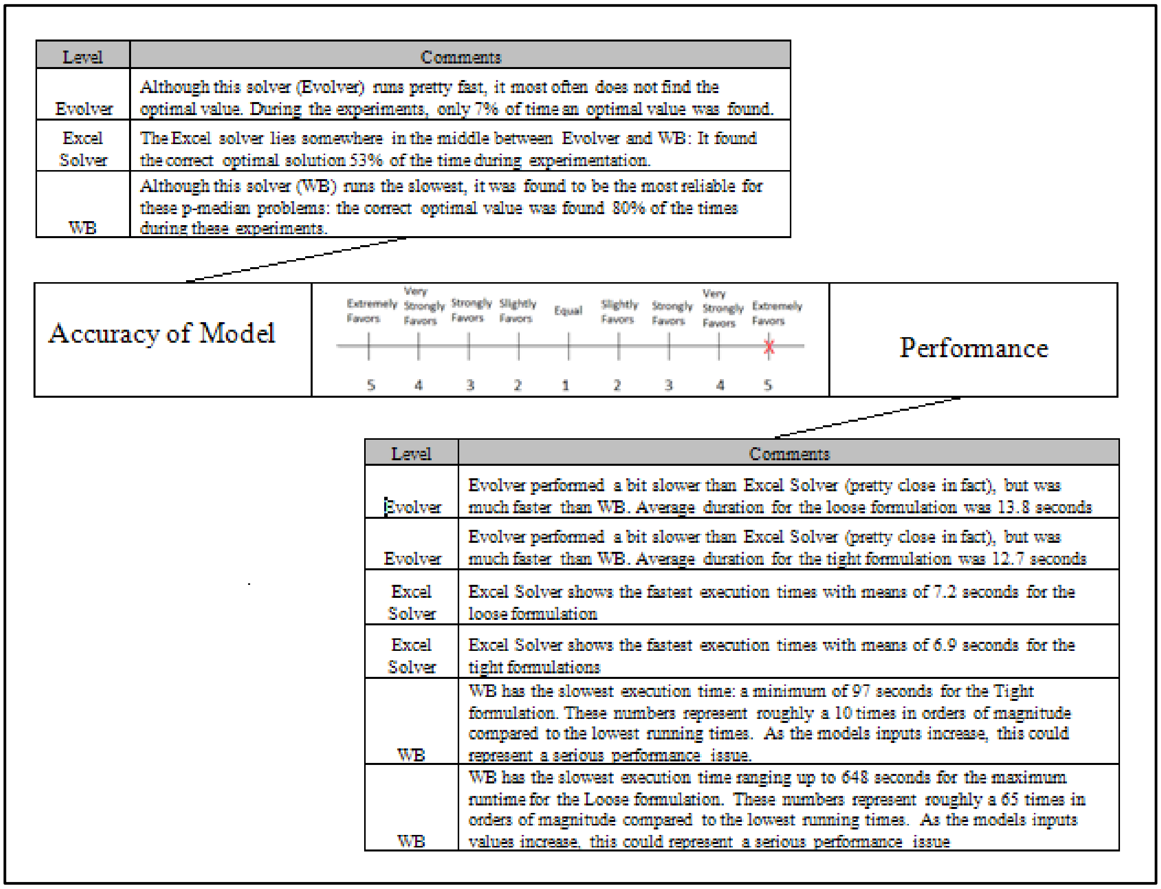


Figure Q-1. Pairwise Comparison Scale

The VBE will present each pair of criteria, and the user will click one of the nine preference positions going from extremely favoring one criterion over another to equally preferring both, and all the ranges in between (equal, slightly favors, strongly favors, very strongly favors, and extremely favors).

Figure Q-2 shows the solicitation of user preferences for the ‘Accuracy of Model’ vs. ‘Performance’ criteria. It shows how the VBE can be equipped with a context sensitive menu whereby clicking on a criterion presents insights recorded in prior uses. Filtering mechanisms can be devised to eliminate displaying insights that may be irrelevant for a user. The display of prior feedback/insights will help the user in deciding on scores for each pairwise criteria comparison.



Level	Comments
Evolver	Although this solver (Evolver) runs pretty fast, it most often does not find the optimal value. During the experiments, only 7% of time an optimal value was found.
Excel Solver	The Excel solver lies somewhere in the middle between Evolver and WB: It found the correct optimal solution 53% of the time during experimentation.
WB	Although this solver (WB) runs the slowest, it was found to be the most reliable for these p-median problems: the correct optimal value was found 80% of the times during these experiments.

Accuracy of Model	Extremely Favors	Very Strongly Favors	Strongly Favors	Slightly Favors	Equal	Slightly Favors	Strongly Favors	Very Strongly Favors	Extremely Favors	Performance
	5	4	3	2	1	2	3	4	5 ✗	

Level	Comments
Evolver	Evolver performed a bit slower than Excel Solver (pretty close in fact), but was much faster than WB. Average duration for the loose formulation was 13.8 seconds
Evolver	Evolver performed a bit slower than Excel Solver (pretty close in fact), but was much faster than WB. Average duration for the tight formulation was 12.7 seconds
Excel Solver	Excel Solver shows the fastest execution times with means of 7.2 seconds for the loose formulation
Excel Solver	Excel Solver shows the fastest execution times with means of 6.9 seconds for the tight formulations
WB	WB has the slowest execution time: a minimum of 97 seconds for the Tight formulation. These numbers represent roughly a 10 times in orders of magnitude compared to the lowest running times. As the models inputs increase, this could represent a serious performance issue.
WB	WB has the slowest execution time ranging up to 648 seconds for the maximum runtime for the Loose formulation. These numbers represent roughly a 65 times in orders of magnitude compared to the lowest running times. As the models inputs values increase, this could represent a serious performance issue

Figure Q-2. Pairwise Comparison With Prior Insights Displayed

Figure Q-3 shows the collected preferences for Scenario Two. The formula to calculate the number of comparisons is as follows:

Number of Comparisons = n (n-1)/2 where n is the number of items to be compared. Therefore, since n = 5, the number of comparisons for Scenario Two is 10.

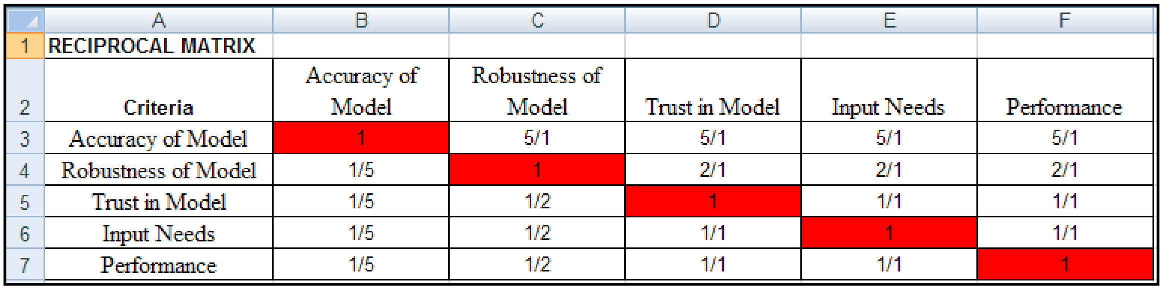
Accuracy of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · [✓ Extremely Favors 5]	Robustness of Model
Accuracy of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · [✓ Extremely Favors 5]	Trust in Model
Accuracy of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · [✓ Extremely Favors 5]	Input Needs
Accuracy of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · [✓ Extremely Favors 5]	Performance
Robustness of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · [✓ Slightly Favors 2] · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Trust in Model
Robustness of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · [✓ Slightly Favors 2] · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Input Needs
Robustness of Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · Equal 1 · [✓ Slightly Favors 2] · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Performance
Trust in Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · [✓ Equal 1] · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Input Needs
Trust in Model	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · [✓ Equal 1] · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Performance
Performance	Extremely Favors 5 · Very Strongly Favors 4 · Strongly Favors 3 · Slightly Favors 2 · [✓ Equal 1] · Slightly Favors 2 · Strongly Favors 3 · Very Strongly Favors 4 · Extremely Favors 5	Input Needs

Figure Q-3. Pairwise Criteria Comparison Feedback Collected From User

The preferences shown in Figure Q-3 must be converted to preference weights. Saaty (1990) demonstrated that the Eigen Vector solution is the best approach. Teknomo (2006) provides an approximation method to find the Eigen Vector followed by a validation method to check the consistency of user preferences. This approximation procedure can be summarized as follows: 1) create an n by n matrix and transfer all user preference values; 2) generate the priority vector by approximation; and 3) verify the consistency of the user preferences. This method is demonstrated next.

1.0 - Create an n by n Matrix and Transfer User Preference Values

The first step after collecting user preferences in a pairwise fashion as shown in Figure Q-3 is to store and process these preferences in a matrix as shown in Figure Q-4. The labels at the left and the top of the matrix are the criteria to be compared. The list of diagonal cells which span from the upper left to the lower right of the matrix hold the score of one, which designates that each criterion is equally important to itself.



	A	B	C	D	E	F
1	RECIPROCAL MATRIX					
2	Criteria	Accuracy of Model	Robustness of Model	Trust in Model	Input Needs	Performance
3	Accuracy of Model	1	5/1	5/1	5/1	5/1
4	Robustness of Model	1/5	1	2/1	2/1	2/1
5	Trust in Model	1/5	1/2	1	1/1	1/1
6	Input Needs	1/5	1/2	1/1	1	1/1
7	Performance	1/5	1/2	1/1	1/1	1

Figure Q-4. Reciprocal Matrix With Values as Entered by User

The scores located above and to the right of the diagonal show the importance of the horizontally listed criteria in reference to those listed vertically. And similarly, the scores listed below and to the left of the diagonal show the importance of the vertically listed criteria in reference to those listed horizontally, and therefore are the inverse of the scores on the other side of the diagonal.

For example, the score of ‘2/1’ in cell E4 of Figure Q-4 shows the preference as shown in Figure Q-5. The value of ‘1/2’ in cell C6 is the inverse of the value of cell E4.

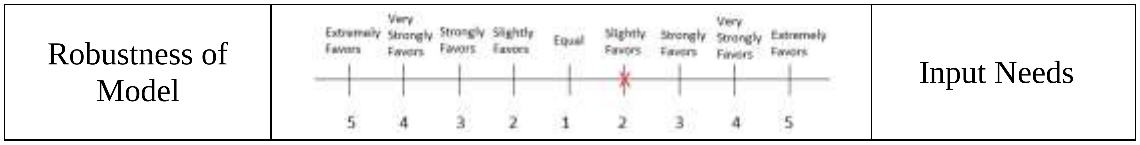
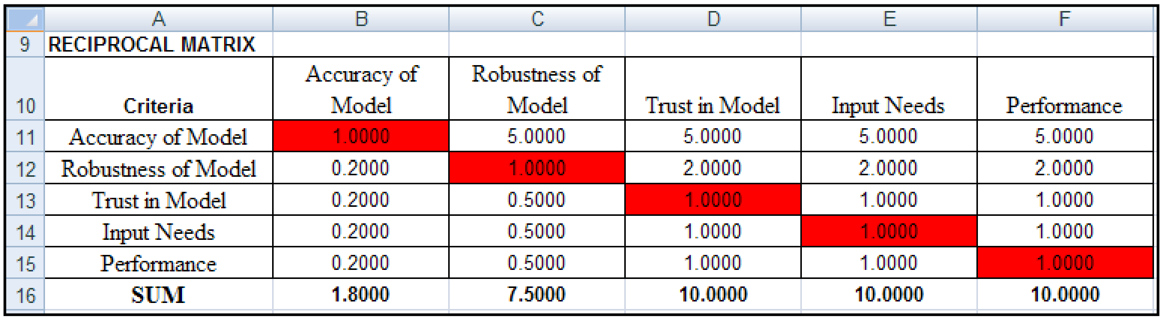


Figure Q-5. Sample Pairwise Comparison Scale

The next step is to normalize the comparison matrix by first computing the sum of each criteria column as shown in Figure Q-6.



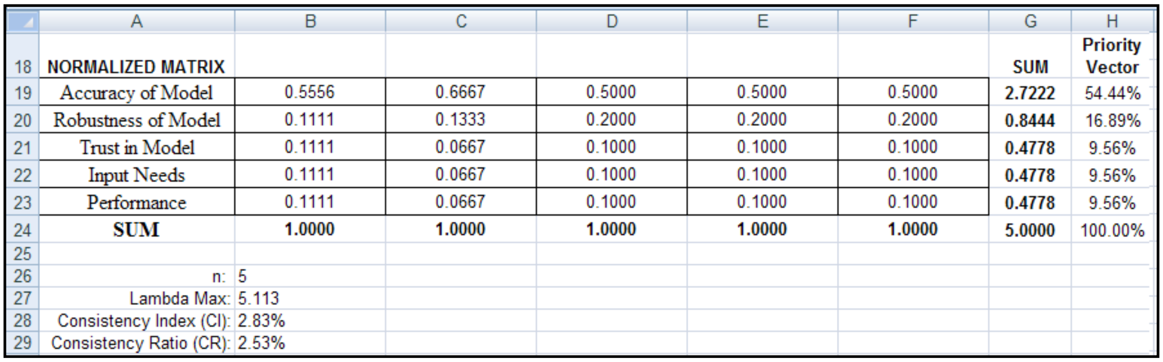
	A	B	C	D	E	F
9	RECIPROCAL MATRIX					
10	Criteria	Accuracy of Model	Robustness of Model	Trust in Model	Input Needs	Performance
11	Accuracy of Model	1.0000	5.0000	5.0000	5.0000	5.0000
12	Robustness of Model	0.2000	1.0000	2.0000	2.0000	2.0000
13	Trust in Model	0.2000	0.5000	1.0000	1.0000	1.0000
14	Input Needs	0.2000	0.5000	1.0000	1.0000	1.0000
15	Performance	0.2000	0.5000	1.0000	1.0000	1.0000
16	**SUM**	**1.8000**	**7.5000**	**10.0000**	**10.0000**	**10.0000**

Figure Q-6. Reciprocal Matrix Internal Representation

Then, the score in each cell is divided by the sum of the column in which it is located, and is placed in a new matrix as shown in Figure Q-7.

2.0 - Generate the Priority Vector by Approximation

The next step is to sum up the scores in each row as shown in cells G19 to G24 of Figure Q-7. The final step is to divide each of these sums by the total of the sums located in cell G24 and placing them accordingly in cells H19 to H24. This final step creates the Priority Vector, which is the weight assigned to each criterion.



	A	B	C	D	E	F	G	H
18	**NORMALIZED MATRIX**						**SUM**	**Priority Vector**
19	Accuracy of Model	0.5556	0.6667	0.5000	0.5000	0.5000	**2.7222**	54.44%
20	Robustness of Model	0.1111	0.1333	0.2000	0.2000	0.2000	**0.8444**	16.89%
21	Trust in Model	0.1111	0.0667	0.1000	0.1000	0.1000	**0.4778**	9.56%
22	Input Needs	0.1111	0.0667	0.1000	0.1000	0.1000	**0.4778**	9.56%
23	Performance	0.1111	0.0667	0.1000	0.1000	0.1000	**0.4778**	9.56%
24	**SUM**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**5.0000**	100.00%
25								
26	n:	5						
27	Lambda Max:	5.113						
28	Consistency Index (CI):	2.83%						
29	Consistency Ratio (CR):	2.53%						

Figure Q-7. Normalized Matrix and Priority Vector

3.0 - Verify the Consistency of User Preferences

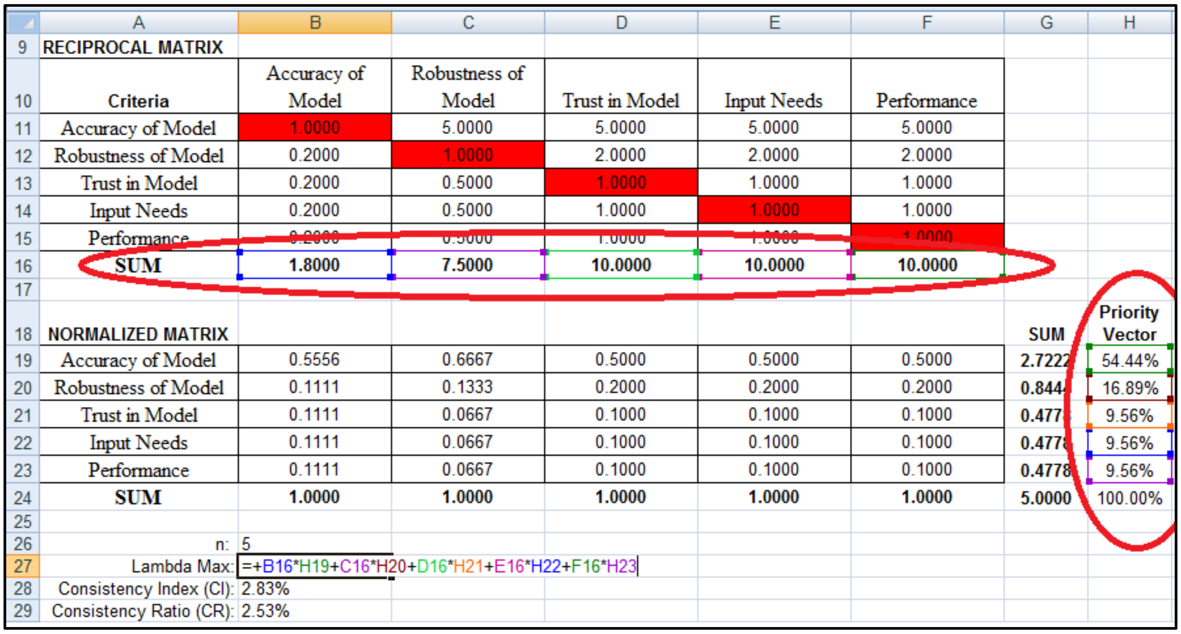
Since the user provides feedback by comparing every two items (criteria) at a time, it is possible that sometimes the preference values between three items may be inconsistent. For example, let us suppose that we have three criteria to compare: C1, C2, and C3. Let us assume that in a first pairwise comparison, the user indicates that C2 is extremely more favored than C1. In a second pairwise comparison, the user indicates that C3 is extremely more favored than C2. And in a final comparison, the user indicates that C1 is extremely more favored than C3. This last comparison contradicts the earlier two since by transitivity, C3 should be extremely more favored than C1.

The final step is to check for consistency concerns in the feedback provided by the user.

First, the Principal Eigen value known as λ_{max} needs to be calculated. This value is computed by summing up ‘the products of the sum of the columns in the reciprocal matrix’ (shown by a horizontally stretched red circle in Figure Q-8) multiplied by ‘the values of the priority vector’ (shown by a vertically stretched red circle in Figure Q-8).

The following cell formula shows how the Principal Eigen value is computed for Figure Q-8:

```
Principal Eigen    = Lambda Max =
                   = +B16*H19+C16*H20+D16*H21+E16*H22+F16*H23
                   = 5.113
```



	A	B	C	D	E	F	G	H
9	RECIPROCAL MATRIX							
10	Criteria	Accuracy of Model	Robustness of Model	Trust in Model	Input Needs	Performance		
11	Accuracy of Model	1.0000	5.0000	5.0000	5.0000	5.0000		
12	Robustness of Model	0.2000	1.0000	2.0000	2.0000	2.0000		
13	Trust in Model	0.2000	0.5000	1.0000	1.0000	1.0000		
14	Input Needs	0.2000	0.5000	1.0000	1.0000	1.0000		
15	Performance	0.2000	0.5000	1.0000	1.0000	1.0000		
16	SUM	1.8000	7.5000	10.0000	10.0000	10.0000		
17								
18	NORMALIZED MATRIX						SUM	Priority Vector
19	Accuracy of Model	0.5556	0.6667	0.5000	0.5000	0.5000	2.7222	54.44%
20	Robustness of Model	0.1111	0.1333	0.2000	0.2000	0.2000	0.8444	16.89%
21	Trust in Model	0.1111	0.0667	0.1000	0.1000	0.1000	0.4778	9.56%
22	Input Needs	0.1111	0.0667	0.1000	0.1000	0.1000	0.4778	9.56%
23	Performance	0.1111	0.0667	0.1000	0.1000	0.1000	0.4778	9.56%
24	SUM	1.0000	1.0000	1.0000	1.0000	1.0000	5.0000	100.00%
25								
26	n:	5						
27	Lambda Max:	=+B16*H19+C16*H20+D16*H21+E16*H22+F16*H23						
28	Consistency Index (CI):	2.83%						
29	Consistency Ratio (CR):	2.53%						

Figure Q-8. Calculation of the Principal Eigen Value i.e. Lambda Max

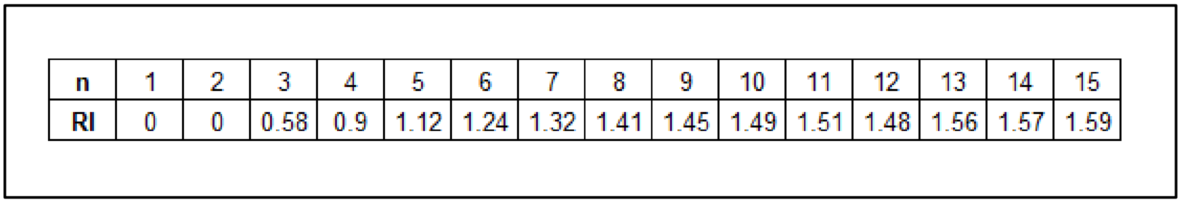
Next, the Consistency Index (CI) should be calculated as follows:

$= (\lambda_{max} - n)/(n-1)$
$= (5.113 – 5) / (5-1)$
$= 2.8$

And finally, the Consistency Ratio (CR) must be calculated.

CR defined as: Consistency Index (CI) / Random Index (RI)

The RI as shown in Figure Q-9 represents averaging the consistency indexes of 500 matrices, based on randomly generated reciprocal matrices (Teknomo, 2006). These figures are used as a benchmark against which to check, based on the value of n.



n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Figure Q-9. Random Consistency Index - Teknomo (2006)

The Consistency Ratio (CR) for Scenario Two is
= Consistency Index (CI) / Random Index (RI)
= 2.83 / 1.12
= 2.53

As long as the Consistency Ratio (CR) is below 10, then there is no inconsistency in the user feedback. Otherwise, the user must recheck the preference scores provided.

Reference List

Barbarosoglu, G., & Yazgac, T. (2000). A decision support model for customer value assessment and supply quota allocation. *Production Planning & Control*, *11*(6), 608-616.

Barkhi, R., Rolland, E., Butler, J., & Fan, W. (2005). Decision Support System induced guidance for model formulation and solution. *Decision Support Systems, 40*(2), 269– 281.

Basu, A., & Blanning, R. W. (1995). Metagraphs. *Omega International Journal of Management Science, 23*(1), 13-25.

Basu, A., & Blanning, R. W. (1997). A graph-theoretic approach to analyzing knowledge bases containing rules, models and data. *Annals of Operations Research, 75*(1-4), 3-23.

Basu, A., & Blanning, R. W. (1998). The analysis of assumptions in model bases using metagraphs. *Management Science, 44*(7), 982.

Bharadwaj, A., Choobineh, J., Lo, A., & Shetty, B. (1992). Model management systems: a survey. *Annals of Operations Research, 38*(1-4), 17-67.

Bhargava, H. K., Krishnan, R., & Mukherjee, S. (1992). On the integration of data and mathematical modeling languages. *Annals of Operations Research, 38*(1-4), 69-95.

Chari, K. (2002). Model Composition Using Filter Spaces. *Information Systems Research*, *13*(1), 15-35.

Chari, K. (2003). Model composition in a distributed environment. *Decision Support Systems*, *35*(3), 399-413.

Chari, K., & Krishnan, R. (2000). Model management: Survey, futures directions and a bibliography. Retrieved May, 2009, from http://catt.okstate.edu/sharda/docs/itorms/kchari/kchari/doc.html

Chari, K., & Sen, T. K. (1998). An implementation of a graph-based modeling system for structured modeling (GBMS/SM). *Decision Support Systems, 22*(2), 103-120.

Colombo, E., & Francalanci, C. (2004). Selecting CRM packages based on architectural, functional, and cost requirements: Empirical validation of a hierarchical ranking model. *Requirements Engineering, 9*(3), 186-203.

Davies, M. (2000). Using an Analytic Hierarchy Process in advertising creativity. *Creativity & Innovation Management, 9*(2), 100.

Dolk, D. R. (1986). Data as models: An approach to implementing model management. *Decision Support Systems, 2*(1), 73-80.

Dolk, D. R. (2000). Integrated model management in the data warehouse era. *European Journal of Information Systems, 122*, 199-218.

Dyer, J. S., Fishburn, P. C., Steuer, R. E., Wallenius, J., & Zionts, S. (1992). Multiple Criteria Decision Making. Multiattribute Utility Theory: The next ten years. *Management Science, 38*(5), 645-654.

Fazlollahi, B., & Vahidov, R. (2001). A method for generation of alternatives by decision support systems. *Journal of Management Information Systems, 18*(2), 229-250.

Forgionne, G. A. (1999). An AHP model of DSS effectiveness. *European Journal of Information Systems*, 95–106.

Forman, E. H., & Gass, S. I. (2001). The analytic hierarchy process - An exposition. *Operations Research, 49*(4), 469-486.

Foulds, L. R., & Partovi, F. Y. (1998). Integrating the analytic hierarchy process and graph theory to model facilities layout. *Annals of Operations Research, 82*(1-4), 435-451.

Gagliardi, M., & Spera, C. (1995). Toward a formal theory of model integration. *Annals of Operations Research, 58*(1-4), 405-440.

Geoffrion, A. M. (1987). An Introduction to Structured Modeling. *Management Science, 33*(5), 547.

Geoffrion, A. M. (1989). The formal aspects of Structured Modeling. *Operations Research, 37*(1), 30.

Geoffrion, A. M. (1991). FW/SM: A prototype Structured Modeling environment. *Management Science, 37*(12), 1513.

Geoffrion, A. M. (1992). The SML language for Structured Modeling: Levels 1 and 2; Levels 3 and 4. *Operations Research, 40*(1), 38.

Hevner, A. R., March, S. T., Jinsoo, P., & Ram, S. (2004). Design Science In Information Systems Research. *MIS Quarterly, 28*(1), 75-105.

Isakowitz, T., Schocken, S., & Lucas, H. C., Jr. (1995). Toward a logical/physical theory of spreadsheet modeling (Vol. 13, pp. 1-37): ACM Press.

Iyer, B., Shankaranarayanan, G., & Lenard, M. L. (2005). Model management decision environment: a Web service prototype for spreadsheet models. *Decision Support Systems, 40*(2), 283-304.

Jones, C. V. (1990). An introduction to Graph-Based Modeling Systems, part I: An overview. *ORSA Journal on Computing, 2*(2), 136.

Jones, C. V. (1991). An introduction to Graph-Based Modeling Systems, part II: Graph-Grammars and the implementation. *ORSA Journal on Computing, 3*(3), 180.

Jones, C. V. (1992). Attributed Graphs, Graph-Grammars, and Structured Modeling. *Annals of Operations Research, 38*(1-4), 281-324.

Jones, C. V., & Schocken, S. (1993). Reframing decision problems: A Graph-Grammar approach. *Information Systems Research, 4*(1), 55-87.

Lee, C. E., & Hsu, S. C. (2004). Outsourcing capacity planning for an IC design house. *International Journal of Advanced Manufacturing Technology, 24*(3/4), 306-320.

Lenard, M. L. (1986). Representing models as data. *Journal of Management Information Systems, 2*(4), 36-48.

Liang, T.-P. (1988a). Development of a Knowledge-Based Model Management System. *Operations Research, 36*(6), 849.

Liang, T.-P. (1988b). Model management for group decision support. *MIS Quarterly, 12*(4), 667.

Liang, T.-P., & Jones, C. V. (1988). Meta-Design considerations in developing model management systems. *Decision Sciences, 19*(1), 72.

Muhanna, W. A. (1987). *A systems framework for model management in organizations.* University of Wisconsin-Madison.

Muhanna, W. A. (1990). Issues in distributed model management systems. *Proceedings of the 11th Annual International Conference on Information Systems, ICIS '90* (pp. 231-242).

Muhanna, W. A. (1992). On the organization of large shared model bases. *Annals of Operations Research, 38*(1-4), 359-396.

Muhanna, W. A., & Pick, R. A. (1988). Composite models in SYMMS. *21st Annual Hawaii International Conference on System Sciences* (pp. 418-427).

Muhanna, W. A., & Pick, R. A. (1994). Meta-Modeling concepts and tools for model management: A systems approach. *Management Science, 40*(9), 1093-1123.

Panko, R. R. (1998). What we know about spreadsheet errors. *Journal of End User Computing's Special issue on Scaling Up End User Development 10*(2), 15-21.

Panko, R. R. (1999). Applying code inspection to spreadsheet testing. *Journal of Management Information Systems, 16*(2), 159-176.

Panko, R. R. (2006). Spreadsheets and Sarbanes-Oxley: regulations, risks, and control frameworks. *Communications of AIS, 2006*(17), 2-50.

Partovi, F. Y., Burton, J., & Banerjee, A. (1990). Application of Analytical Hierarchy Process in Operations Management. *International Journal of Operations & Production Management, 10*(3), 5-19.

Phillips-Wren, G., Mora, M., Forgionne, G. A., & Gupta, J. N. D. (2009). An integrative evaluation framework for intelligent decision support systems. *European Journal of Operational Research, 195*, 642-652.

Ronen, B., Palley, M. A., & Henry, C. L., Jr. (1989). Spreadsheet analysis and design (Vol. 32, pp. 84-93): ACM Press.

Ruhe, G., Eberlein, A., & Pfahl, D. (2003). Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering, 13*(4), 345–366.

Saaty, T. L. (1990). How to make a decision: The Analytic Hierarchy Process. *European Journal of Operational Research, 48,* 9-26.

Sarkis, J., & Sundarraj, R. P. (2001). A decision model for strategic evaluation of enterprise information technologies. *Information Systems Management, 18*(3), 62.

Singh, R. K., Choudhury, A. K., Tiwari, M. K., & Maull, R. S. (2006). An integrated fuzzy-based decision support system for the selection of lean tools: a case study from the steel industry. *Proceedings of the Institution of Mechanical Engineers -- Part B* -- *Engineering Manufacture, 220*(10), 1735-1749.

Stannard, B., Zahir, S., & Rosenbloom, E. S. (2006). Application of Analytic Hierarchy Process in multi-objective mixed integer programming for airlift capacity planning. *Asia-Pacific Journal of Operational Research, 23*(1), 61-76.

Steiger, D. M. (1998). Enhancing user understanding in a Decision Support System: A theoretical basis and framework. *Journal of Management Information Systems, 15*(2), 199-220.

Teknomo, K. (2006). Analytic Hierarchy Process (AHP) Tutorial.   Retrieved January 2010, from http://people.revoledu.com/kardi/tutorial/ahp/

Uzoka, F. M. E. (2005). AHP-based system for strategic evaluation of financial information. *Information Knowledge Systems Management, 5*(1), 49-61.

van Aken, J. E. (2004). Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules. *Journal of Management Studies, 41*(2), 219-246.

van Aken, J. E. (2005). Valid knowledge for the professional design of large and complex design processes. *Design Studies, 26*(4).

Wallenius, J., Dyer, J. S., Fishburn, P. C., Steuer, R. E., & Deb, K. (2008). Multicriteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management Science, 54*(7), 1336-1349.

Yuan-Jye, T., & Yu-Hua, L. (2005). A Model for Supplier Selection and Tasks Assignment. *Journal of American Academy of Business, Cambridge, 6*(2), 197-207.

Zigurs, I., & Buckland, B. K. (1998). A theory of task/technology fit and group support systems effectiveness. *MIS Quarterly, 22*(3), 313-334.

NSU NOVA
SOUTHEASTERN
UNIVERSITY
The Graduate School of Computer and Information Sciences

Certification of Authorship of Dissertation Work

Submitted to (Advisor's Name): Dr. S. Mukherjee

Student's Name: Mario Sarkis Missakian

Date of Submission: 2/8/2011

Title of Submission: Automated Support for Model Selection Using Analytic Hierarchy Process

Certification of Authorship: I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for this purpose.

Student's Signature: Mario Sarkis Missakian
____