What's Wrong with Spreadsheets – and How To Fix them with Analytica

Draft 14 July, 2004

Max Henrion, PhD

Lumina Decision Systems, Inc
26010 Highland Way
Los Gatos, CA 95033, USA

www.lumina.com

Comments are welcome to Henrion@lumina.com
Please check www.lumina.com for a final version before citing or quoting.

The original “killer app”

The twenty-fifth anniversary of the first spreadsheet is almost upon us: The first spreadsheet, VisiCalc, was developed by Dan Bricklin and Bob Frankston, and released in May 1979 for the Apple II. Now that there are many tens of millions of spreadsheet users, spreadsheets as a category rarely make headlines, it is easy to forget that the spreadsheet was the original “killer app”: In the early 1980s, the spreadsheet was the vehicle on which the personal computer first rode to fame and fortune. Before spreadsheets, the personal computer was mostly seen as a complicated toy for hobbyists. Spreadsheets, by offering immediate practical value to businesses, convinced millions to buy their first personal computer.

Following VisiCalc, Mitch Kapor developed Lotus 123 and released it for the IBM PC in 1983. Lotus 123 rapidly became the dominant spreadsheet for the rest of the 1980s. Microsoft first released Excel as one of the first applications on the new Macintosh in 1984-5. Excel became Microsoft’s flagship product on Windows with the release of Windows 3.0 in 1989. Since then it has achieved near complete domination of the category.

The application takes its name from the accountants’ *spreadsheet* — a sheet of paper with ruled rows and columns, in use for hundreds of years. Part of its immediate success lay in its user-interface metaphor — the *electronic grid* that looked seductively familiar to those accustomed to paper spreadsheets — the layout with row labels down the left-hand column and columns of numbers with totals at the bottom. The ability of the computer spreadsheet to recalculate results as you change input numbers seemed almost miraculous to those accustomed to using a hand – or head – calculator. Even for those few who had been programming computers to perform such calculations, the way in which the spreadsheet maintains the dependencies and automatically recalculate formulas was a breakthrough in freeing them from writing programs to sequence the calculations.

Spreadsheet applications, notably Microsoft Excel, have continued to add a wide variety of new features over the last 25 years, but the essential user interface – the grid metaphor – has barely changed since VisiCalc [Dan Bricklin, 2001]. It still displays rows and columns labeled by numbers and letters, with formulas referring to cells by their grid addresses, and a field at the top showing the contents of the selected cell.

Using spreadsheets for quantitative modeling for decision making

People use spreadsheets nowadays for a great deal more than accounting. They use them to help manage lists of items, such as product inventories, assets, contacts, and employees – essentially small databases. They use them for budgets. They also use them for analyzing and forecasting the performance of investments, businesses, and other organizations. These latter applications involve significant *quantitative modeling* – the spreadsheet models the past, current, and future revenues, costs, profits, and often staff, customers, products, and material inputs and outputs of an industrial process. These kinds of applications are sometimes called *business analytics*.

The grid metaphor is ideal for simple accounting, for which it was created, and works well for managing lists, provided the lists are not too large. It is less suited for quantitative modeling and business analytics. Indeed, spreadsheets offer major obstacle to the effective building and use of quantitative models. Why do I think that? Below, I will identify ten key deficiencies in the design of spreadsheet applications that get in the way of effective modeling. But, first, let us examine the activity of building and using quantitative models, so that we can understand what makes modeling different from basic accounting, list management, and other simple applications of spreadsheets.

Insight not numbers: Quantitative modeling involves, of course, working with numbers – but the ultimate goal of the exercise is insight, not numbers. To be effective the model results in improved decisions, because the decision makers can base their choices on a deeper understanding of the situation. Effective modeling does not replace the decision maker’s intuition, but rather enriches and extends it. Few executives or policy makers are willing to base a decision purely on the numerical outputs of a model -- which option offers the highest return on investment or net present value – most want to know why – what factors and which assumptions lead to these results, and how important are their contributions to the conclusions.

A collaborative process: Making important decisions in organizations, whether business or public sector, is a collaboration involving multiple players. Even when there is a single "boss" with exclusive responsibility for the decision, he or she almost always relies on inputs from other people. More often, organizations have some kind of consultative decision process that involves multiple people in discussing and evaluating the options. In either case, the participants give little credence to a black-box model that provides only "the numbers". An effective model is one that enables exploration and explanation of its results and helps the group reach a deeper collective understanding of the assumptions and implications.

An evolving process: Occasionally, decisions are "one-off", but usually a series of similar decisions must be made daily, monthly, or yearly – such as who to hire or fire, which R&D projects to fund, which products to manufacture, whether to acquire or sell capital equipment or a manufacturing plant. Ideally, the decision team learns from experience and improves its decision process over time.

When we understand decision making to be an evolving team process and that the goal of quantitative modeling is insight not numbers, we can see the importance of two aspects of the model:

Transparency: If a model is to help provide insights into the problem and serve as basis for a team to arrive at a collective understanding, the model must be transparent – that is, it must be easy to see what assumptions, relationships, and numbers go into it, and how those inputs influence the results.

Flexibility: As the decision process evolves, participants will suggest new options, more factors to consider, and criteria to evaluate them. To support this process, the model should be flexible enough so that it is easy to explore, adapt, and extend to include these new issues.

In practice, spreadsheet models are highly opaque and inflexible relative to what is possible. Their ubiquity and value for simple applications has blinded many of us to the shackles they impose on the creation and analysis of real decision models. There is increasing evidence that they are commonly a source of serious errors, as I shall explain below. Spreadsheets are catastrophically unsuitable as a tool to facilitate clear thinking and deep insights – the real reason that we create most quantitative models. What was once an instrument of liberation to the savvy businessperson has become, all too often, a source of confusion and frustration.

How often are spreadsheets wrong?

Anyone who has built a spreadsheet knows that it is easy to make mistakes. How often do mistakes remain uncaught and lead to seriously bad decisions? There exist a few worrying anecdotes, but it is hard to know if they are isolated incidents. Naturally those involved in major spreadsheet disasters are rarely willing to publicize them. There have, however, been several substantial empirical studies that audit operational spreadsheets, providing disturbing results.

Study	Number of spread sheets	Criterion	% Models with Errors
Davies & Ikin [1987]	19	"Serious" errors.	21%
Butler [1992]	273	Errors large enough to need additional tax payments. Audited by UK tax inspectors	11%
Cragg & King [1993]	20	150 to 10,000 cells, serious errors.	25%
Coopers & Lybrand [1997]	23	Results off by at least 5%	91%
KPMG [1997]	22	Only major errors.	91%
Butler [2000]	7	Tax spreadsheets audited with enhanced methodology and software.	86%
Total	**367**	Weighted average	**24%**
Total since 1997	**54**	Weighted average	**91%**

Table 1: Adapted from Raymond R. Panko, 2000. "What we know about spreadsheet errors"

According to Professor Raymond Panko's excellent summary of this work (Panko, 2000), field audits of real-world spreadsheets have found 0.5% to 5% of formula cells contain bugs, and between 11% and 91% of the spreadsheets audited were found to contain bugs. Table 1 summarizes some of these studies. Note that the spreadsheets studied are ones in operational use within organizations, in many cases over many years. It is interesting that the more recent studies, since 1997, found higher error rates, averaging 91% than the earlier ones. Panko suggests that this is not because the actual error rates are increasing. Rather the more recent studies audited the spreadsheets more carefully, and identified a higher percentage of the bugs. It turns out to be easy to miss bugs: Experiments introducing deliberate errors find that auditors miss 30% to 50% of them.

How can errors be so common? Panko points out that error rates per cell are actually similar to those in studies of comparably complex tasks, such initial error rates per line in writing software. But, commercial software is usually written by professionals and subject to systematic testing by separate quality assurance teams, unlike spreadsheet builders who are mostly untrained end users. And, most software bugs create obviously wrong behavior or crashes, where most spreadsheet errors, such as the common omissions in summation, give results that are off by modest percentages, and are so hard to detect. For these reasons, initial mistakes in spreadsheets are often uncaught, and error rates are likely to be much higher than in commercial software.

Several studies have shown that most spreadsheet users are unaware of the frequency of errors, and are overconfident about their reliability (Brown & Gould, 1987; Davies & Klein, 1987). They are even more confident about larger spreadsheets (Reithel, Nichols, & Robinson, 1996), which because of their size are actually more likely to contain errors. This may explain why spreadsheet builders often neglect the importance of careful testing and auditing.

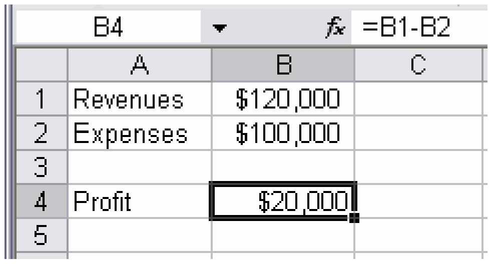
Ten problems of spreadsheets – and how to fix them

The focus of the rest of this paper are ten key deficiencies of spreadsheet design which, I shall argue, are, to a major extent, responsible for these problems — summarized in Table 2. The spreadsheet examples use Microsoft Excel, because, as the dominant product, it will be familiar to most readers – certainly not because it deserves to be singled out as worse than its competitors.

The good news is that we can avoid, or at least significantly alleviate, many of these problems by the use of software tools that are designed specifically for modeling rather than accounting. For concreteness, I will illustrate how each of these ten problems can be addressed by the use of Analytica, software application that I helped create, along with several associates, with precisely this purpose in mind.

1. Meaningless cell references

As any spreadsheet user knows all too well, the formulas refer to other variables using cell addresses — such as B2 for column B of row 2 — rather than meaningful names — such as Revenues or Expenses. Anyone who has tried to read such formulas, whether written by someone else or oneself, knows how hard they can be to understand or verify.



B4 =B1-B2

	A	B	C
1	Revenues	$120,000	
2	Expenses	$100,000	
3			
4	Profit	$20,000	
5			

Figure 1: The formula for the selected cell, B4, uses cell references, B1 and B2, to identify those variables

Cell addresses reflect how the model happens to be laid out on the grid, and usually have little functional relevance to the model. The earliest computer programs in the 1940s used machine languages, which refer to each variable using a number that is its physical address in memory. In the 1950s, computer scientists developed assembly languages and higher-level computer languages, such as Fortran, which let programmers use meaningful names for variables. It is amazing that still, fifty years later, the vast majority of spreadsheets still use cell addresses as their primary way to refer to variables.

Table 2: Summary of the ten key deficiencies of spreadsheets

What's wrong with spreadsheets?	**How does Analytica handles the issue?**	**What's the benefit?**
1. Meaningless cell references	It uses a meaningful name to identify each variable	Formulas are much easier to write, read, and debug
2. Unstructured documentation	Each variable is an object: fields include name, description, units, as well as definition (formula) and value.	Modelers include clear, complete model documentation as they go
3. No variables with defined types or roles	Each object has a *class*, such as *decision, constant, chance variable, objective*, or *index*, defining what role it has in the model.	Modeler and application both understand the role of each variable, which prevents common conceptual errors.
4. Invisible dependencies	Influence diagrams provide an intuitive graphical view, depicting variables as nodes and dependencies as arrows.	Modelers and decision makers communicate clearly with each other about key assumptions and model structure. It is easy to navigate large models.
5. Little support for modularity	You organize a large model as a hierarchy of simple comprehensible modules.	Complex models become manageable. Each diagram shows key variables and relations, and hides irrelevant details in submodules.
6. Formulas define and refer to cells not tables	With Analytica's Intelligent Arrays™, a single formula expresses an operation on named tables. It automatically iterates over all dimensions.	The number of formulas to write, verify, and debug is often 100 to 1,000 times less, hugely reducing chances for error.
7. Changing a dimension demands major surgery	Analytica understands the indexes that identify dimensions of a table or array. Editing or adding a dimension automatically updates all affected arrays and keeps formulas correct.	Modifying and adding dimensions to tables is much faster and less error-prone. It is easy to manage three or more dimensions.
8. No built-in treatment of uncertainty	The value of any variable can be a probability distribution. Efficient Latin hypercube sampling (a type of Monte Carlo simulation) generates corresponding distributions on results.	Modelers without special training can treat uncertainties explicitly and analyze risks.
9. Minimal support for sensitivity analysis	Automated importance analysis shows the relative effect of each uncertain variable on affected outcomes and decisions.	Modelers without special training can easily do importance analyses and generate valuable insights.
10. No separation of end-user interface	It's easy to create a "dashboard" module, to access the key inputs and outputs.	The dashboard offers a simple user interface for model users, protecting them from seeing or messing with irrelevant details.

Microsoft Excel does provide the option of assigning a text name to a cell or range. But, if you try to use names systematically for all variables in model (cells and tables), you soon find out why few spreadsheet users do that. Defining a new name for a cell takes 5 steps (mouse clicks and menu selections). Naming does not scale well for handling models with hundreds or thousands of named elements: Excel offers pull-down menus and list boxes to select names from all the defined names in a spreadsheet — so it is difficult to use more than about 40 names in a model, a tiny fraction of the number of variables in most interesting models. If you expand a table named by a cell range by appending columns or rows, you have to reattach the name the expanded range. Failing to do that properly is a common source of errors. If you modify a name, it invalidates all the formulas using the name — rather than automatically updating them to retain consistency, as Analytica does.

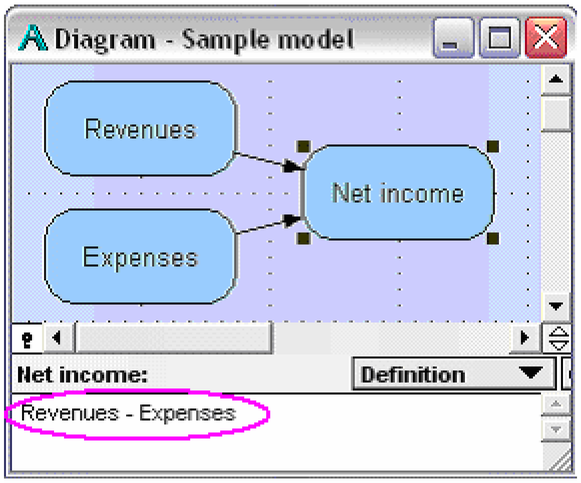
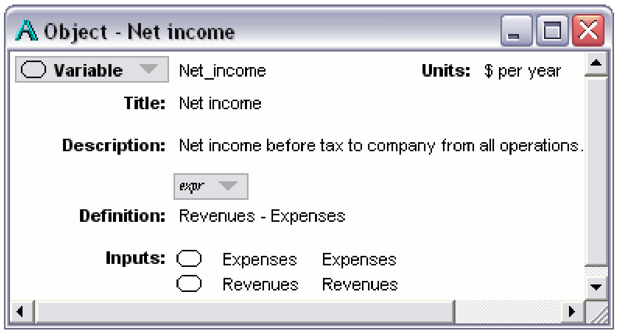


Figure 2: Analytica shows the definition of the selected variable, Net income, referring to the other variables by name. The arrows in the influence diagram reflects the dependencies in the expression

Almost every other kind of modeling tool or computer language refers to variables by name, and Analytica is no exception. However, unlike most other systems, if you decide to change the name of a variable, Analytica will keep the model self-consistent by automatically updating all formulas (expressions) to reflect the new name.

2. No structured documentation

As a spreadsheet builder, you can type text as documentation into any cell that doesn't contain a number or formula. The text may include the name, title, units of measurement, or even an explanation. It is conventional, but far from universal, to place this text in the cell to the left of each cell containing a number or formula. There is no consistent relationship between documentation and the formal model (numbers and formulas). Without any inherent link between documentation and model, the spreadsheet cannot reliably assist the user by prompting for documentation, or maintaining consistency between documentation and model.



Net_income **Units:** $ per year
Title: Net income
Description: Net income before tax to company from all operations.
expr
Definition: Revenues - Expenses
Inputs: Expenses Expenses
Revenues Revenues

Fig 3: The Object view shows functional and documentation attributes, including its Class (Variable in this case), Units (of measurement), Description, and Definition. The Inputs list automatically reflects the variables in the Definition.

In Analytica (see Figure 3), each variable is an object with a standard set of documentation attributes, including a Class, Identifier, Title, Units of measurement, and Description, as well as the Definition containing a number or formula. Since Analytica represents each variable as a coherent object, it can be smart in keeping the elements consistent in each view. It automatically displays the correct Title and Units on any table or graph. If you change the identifier of a variable in the Object window view, it automatically updates the identifier in the diagram node and in all formulas that refer to it. The standard structure encourages you to include complete, well-organized documentation as you define each variable. It also makes it easier for reviewer to understand a model.

3. No variables with defined types or roles

A spreadsheet cell can contain a number, formula, text value, documentary text, or empty space. It can be an input, output, or intermediate calculation — a decision variable, a constant, the index of a table, or an objective to be optimized — among many other things. The problem is that a cell is just a cell as far as the spreadsheet is concerned — there is no explicit representation of what role it is intended to play in the model. As a reader of a spreadsheet, you do not know what kind of thing to expect in each cell. There is no easy way to tell text values from documentation, inputs from outputs, or decisions from objectives. Without representing this information explicitly, the application cannot assist the user by providing appropriate options, checking that the contents of the cell is consistent with its role, and so forestalling common conceptual errors.

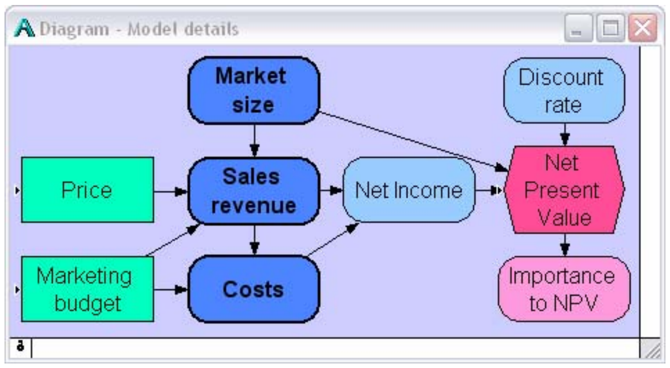
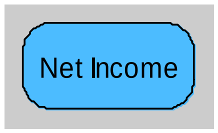
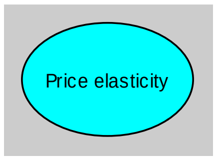


Figure 3: An Analytica influence diagram identifies the type of objects by node shape and type, and the influences (arrows) between them.

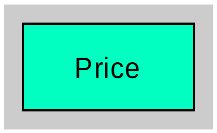
A user of Analytica may designate any variable as an input or output, making it easily available for changing or viewing to end users. In Browse mode, the user may change only variables designated as inputs. Analytica adapts and extends the widely used conventions of influence diagrams [Howard & Matheson, 1984] to denote classes or roles of variable by node shapes:



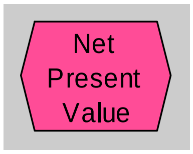
A standard **variable** is a deterministic function of its inputs.



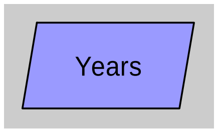
A **chance** variable is uncertain, and not under direct control of the decision makers. It is usually specified as a probability distribution.



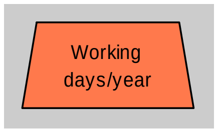
Decision or policy variable, under the direct control of decision makers or policy makers.



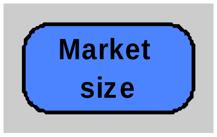
An **objective** quantifies the degree to which decisions satisfy the stakeholders. It might be the expected profit, the discounted net present value, or a measure of utility with multiple goals.



An **index** variable identifies the dimensions of tables or arrays — such as time periods of interest (years), geographic sub regions (e.g. nations), or crop types. Indexes are important in defining the scope and level of detail of the analysis.



A constant identifies a simple constant such as the number of working days in a year, or conversion factor from kilowatt-hours to Joules.



Modules are nodes that contain a set of variables that comprise a sub diagram of the diagram containing a module. By organizing a large model into a hierarchy of modules, and hence a hierarchy of subdiagrams, you can make a complex model more comprehensible.

These graphic conventions make the role of each variable in the model clear. Since Analytica represents the class of each object, it can perform automated checks – for example, ensuring that chance variables are probabilistic and others are not. If a standard variable is assigned a probability distribution, it will automatically change it to a chance variable, updating the node to an oval. Perhaps most importantly, by requiring the modeler to choose a class for each variable when first creating a model, this representation encourages the modeler to think clearly about the structure and goals of the model.

The spreadsheet builder can choose graphic conventions, such as background colors, shading, or borders, to distinguish inputs and outputs, or other variable roles – but it offers no consistent notation or means of checking its use. Excel also provides the ability (from the data menu) to specify validity checks on inputs and to lock other cells. However, there is no standard convention for distinguishing classes of variables, and the spreadsheet itself contains no explicit representation of the intended role of each cell in the model. Hence, it cannot automatically maintain consistent displays and check the validity of variables. For these reasons, modelers rarely use these featurs consistently.

4. Obscure dependencies

Spreadsheets offer no way to visualize the overall structure of a model. One effect of using cell references in formulas is that it is laborious to find out which variables – or rather cells – depend on each other by tracing formulas from one cell to another. Recent releases of Excel offer two ways to trace the inputs to a selected cell: A "formula view" shows formulas within each cell, and color codes cell references to identify its inputs. An "audit tool" will display arrows from the inputs (or to the outputs) of a selected cell. These tools, however, can show the dependencies of only one, or a few, cells at a time. If they could show more, they would usually appear as incomprehensible spaghetti, because spreadsheets are not normally laid out to provide a clear dependency structure.

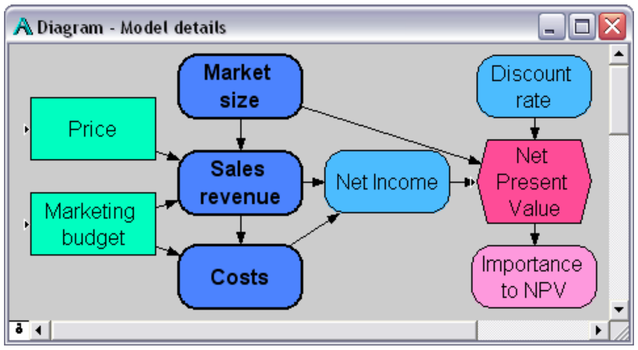


Figure 4: An Analytica influence diagram depicting a model with decisions (green rectangles), intermediate variables and modules (blue rounded rectangles), and an objective (red hexagon). The arrows ("influences") depict dependencies between variables.

In Analytica, the influence diagrams show dependencies as arrows ("influences") between variables. These diagrams are the primary tool for creating models, so the diagrams are, by construction, meaningful views of the model structure, not post-hoc ways to visualize a model as in spreadsheet. As an Analytica modeler, you start building a model by selecting node types, arranging them on a diagram, and drawing arrows between them to depict the dependencies. Having created such a purely qualitative depiction of the model, you can subsequently quantify the model by adding numbers and formulas. As you add a definition, you can paste in variable and function names by selecting them from a pulldown list of the inputs identified by the arrows you drew into that variable. If you paste or type other variables into a definition, Analytica will automatically redraw the arrows to maintain consistency between diagram and definition.

Influence diagrams provide an intuitive, qualitative representation of the model that is valuable initially when building a model, and later when reviewing and explaining it. The diagrams are intuitive enough to be used by top decision makers and other stakeholders in collaboration with modelers to help identify key issues, decisions, and objectives. Together, they can arrive at a shared understanding of the problem by drawing influence diagrams, often using a digital projector. This kind of collaboration is much more inviting and practical for those who are not "quant jocks" than sitting together around a spreadsheet.

.

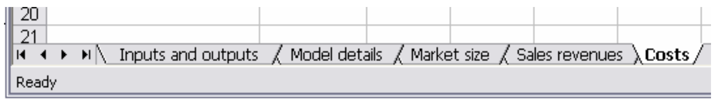


Figure 5: The tabs along the bottom of an Excel window offer access to multiple worksheets

5. Little support for modularity

The best way to manage large models is to organize them into a hierarchy of modules, where each module is small enough to be comprehensible. Excel supports one level of modularity: organizing a Workbook into a number of Worksheets, accessible by tabs along the bottom (see Figure 5). If you use multiple worksheets to represent a third dimension of a table, it will interfere with even this kind of modularity. Experienced spreadsheet users may organize complex models within each worksheet to reflect an underlying modular structure, but the application itself is unaware of any such structure and so can offer no assistance in laying out or organizing the model.

Analytica lets you group a set of related variables together within a module: Each module appears its parent diagram simply as a single node with bold outline. It automatically displays any dependencies between modules due to dependencies between the variables they contain as arrows between the module nodes. In this way, you can organize a complex model into a hierarchy of modules with an unlimited number of levels (see Figure 6). Analytica also offers an expandable outline view of the module hierarchy to help you understand and manage larger models.

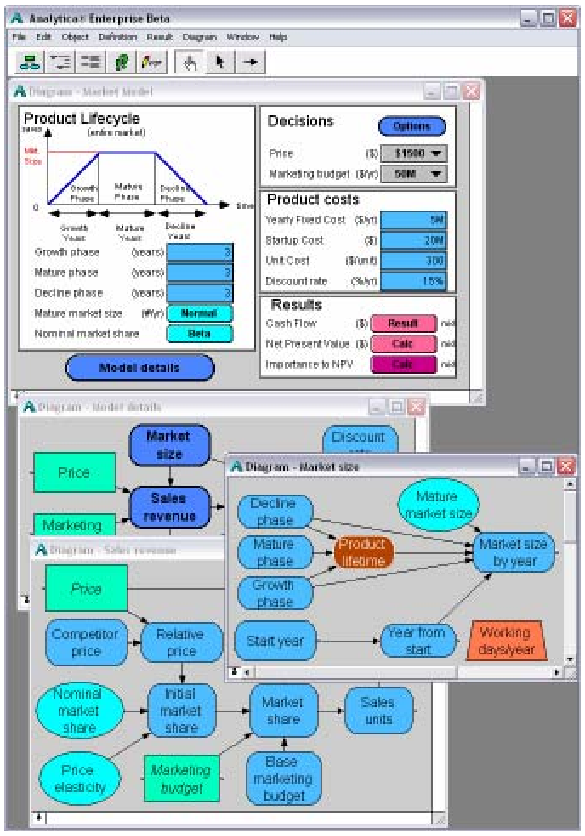


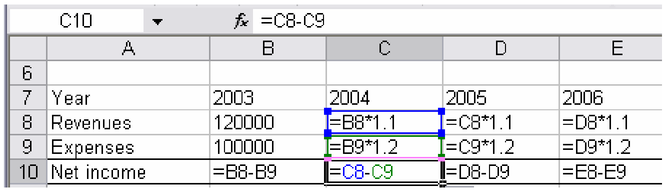
Figure 6: An example three-level module hierarchy: In the top diagram, the blue node "Model details" opens to show the central diagram. In "Model details", nodes "Market size" and "Sales revenue" are opened to show third level diagrams.

Figure to be inserted.

Figure 7: An outline view of the module hierarchy.

6. Formulas work with cells not tables

Most spreadsheet models consist largely of tables with one or two dimensions. The builder is forced to work at the level of cells rather than tables. Spreadsheet formulas usually refer to individual cells rather than tables. Each cell must contain a separate formula, even when all the cells in the table have identical relationships with the cells of other tables. Excel offers a relatively convenient way to create such tables, by "stretching" one cell across the table – provided you use correct absolute and relative cell referencing in the initial formula. But, after you have created the table, the fact that all the formulas are essentially the same gets lost. Indeed, if someone changes a single formula in a table, by design or by accident, it is very difficult to detect – which is a common source of mistakes.



C10 fx =C8-C9

	A	B	C	D	E
6					
7	Year	2003	2004	2005	2006
8	Revenues	120000	=B8*1.1	=C8*1.1	=D8*1.1
9	Expenses	100000	=B9*1.2	=C9*1.2	=D9*1.2
10	Net income	=B8-B9	=C8-C9	=D8-D9	=E8-E9

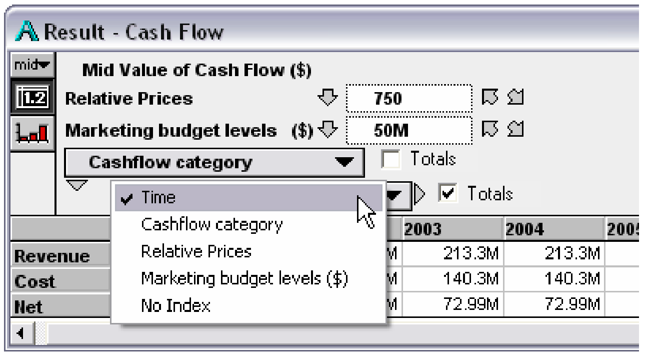
Figure 8: A spreadsheet table showing the formulas in every cell.

With Analytica, the expression

Profit := Revenues – Costs

is just the same whether the variables are simple scalars (single cells), a one-dimensional table for each of 5 years, or a two-dimensional table over 10 divisions of the company. The relationship is easy to write and easy to understand. More important, there is only one relationship to write and review (and store), instead of 50 in the 2-dimensional case. Many models have large tables with hundreds or thousands of elements, making them hundreds or thousands of times simpler to audit – even before taking into account the advantage of using meaningful names rather than cell references. This is part of the reason that model files usually reduce in size by one or two orders of magnitude when translated from Excel into Analytica, despite the addition of improved documentation and hierarchical influence diagrams.

In Excel, it is possible to create tables with three dimensions, using a set of worksheets to represent the third dimension. However, it is challenging to manipulate a three-dimensional table – for example, to display a two-dimensional slice or subtotal over one of the first two dimensions. Managing tables with *more* than three dimensions becomes increasingly more challenging with each additional dimension.



Result - Cash Flow

mid

Mid Value of Cash Flow ($)

Relative Prices: 750

Marketing budget levels ($): 50M

Cashflow category ☐ Totals

☑ Totals

		2003	2004	200
Revenue	M	213.3M	213.3M	
Cost	M	140.3M	140.3M	
Net	M	72.99M	72.99M	

Figure 9: Show a slice or summary of a multi-dimensional table by selecting the dimensions to show down the rows and across the columns.

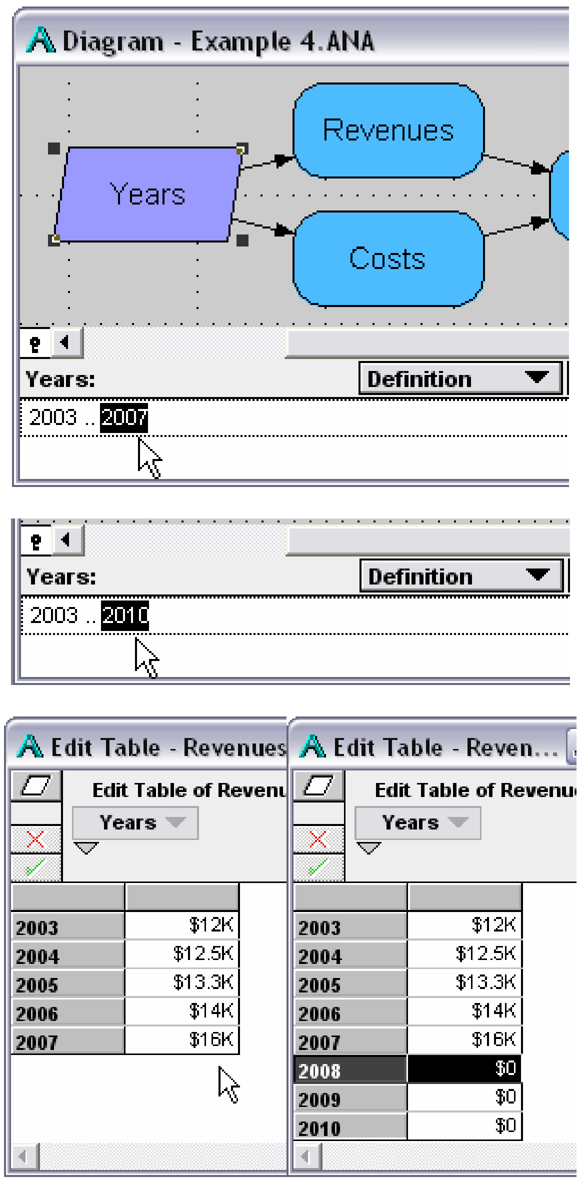
In Analytica, it is quite easy to create and manage arrays of many dimensions. The relationship between the dimensions and the display layout is quite flexible. You can choose any dimension to display down the rows, another along the columns, simply by picking from a pull-down menu. Like other OLAP (Online Analytical Processing), or multidimensional tools, Analytica lets you flexibly slice, dice, and total over the dimensions of your choosing.

7. Editing or adding a dimension demands major surgery

When building a model, the most important and challenging decisions are typically about how much detail to include in table dimensions. How far ahead should your time horizon be? Is it okay to do your analysis by year, or should it be by quarter or by month? What about dividing up by sales regions, by product type, or alternative economic scenarios? Each new dimension, and each increase in the level of detail, expands your model and your work substantially. Will this extra work be worth the improved accuracy and insight? Ideally, you should start out simply and experiment with adding detail to see how it affects the results. Unfortunately, this is usually too much work when building a spreadsheet. Changing the size of a dimension, or adding a dimension, to spreadsheet tables requires much more effort that ought to be necessary.

Suppose your spreadsheet evaluates a new business over 5 years, from 2003 to 2007. Your boss then tells you, she needs a 7-year horizon. You will need to extend every table over time by adding three columns, after the year 2007, and before the total column if there is one. If the tables are aligned — and there are no other tables more than 7 columns wide — you can insert three more columns into the entire worksheet. You can then stretch the formulas 2007 column for each table across the three new columns – assuming there is nothing special about the 2007 column, which you can't easily tell until you look at all the formulas in several columns. You must also enter new formulas into the Total column, which does not include the new columns – a common source of error. If your model contains multiple worksheets, tables with more than two dimensions, or tables in which Year goes down rows rather than across columns, it will take additional work. If there are computations relying on the number of years, for example, computing the annual average from the total, you must remember to fix those two. There is plenty of scope for making errors, and the errors may be hard to spot.

Compare this process with the equivalent change in Analytica: You select the Years index and click <Expr> to view its definition. Select the final year 2007 and type 2010 to replace it. This Years index applies to all tables indexed by Years. So, all definitions and tables indexed by Year are immediately updated to encompass the expanded range, and guaranteed correct. The definition of Net Revenue (Revenues – Costs) remains the same. Any Sum or other array function over Years automatically extends to the extended Years. All tables indexed by Years show the new Years, and correct Totals. Analytica automatically extends any input tables (Edit tables) indexed by Years, like Revenues in Fig 10, initializing the added cells or slices to zero. The only task remaining for you is to fill in the correct values for those new input cells.



Years: 2003 .. 2007

Years: 2003 .. 2010

2003	$12K
2004	$12.5K
2005	$13.3K
2006	$14K
2007	$16K

2003	$12K
2004	$12.5K
2005	$13.3K
2006	$14K
2007	$16K
2008	$0
2009	$0
2010	$0

Figure 10: Expanding the Years index by changing the time horizon from 2007 to 2010. This action automatically inserts cells containing $0 for the three new Years in the Edit table for Revenues.

What if you want to add an entire new dimension, say over five sales regions? You define a new Index as a list of the sales regions, and add it to the input table(s) you want to vary by sales region. Any variables that depend on those tables will automatically extend to include the new dimension. For example, if you add the dimension to revenues, then net income will automatically expand to compute by region. Of course, you can display net income total or by region, as you prefer.

Analytica's ability to generalize expressions and operations automatically over multiple dimensions is known as *Intelligent Arrays*™. This feature makes it vastly easier to create and modify multidimensional models, and to avoid making mistakes. It allows the modeler and the tool both to work at the level of tables rather than cells, a more natural and appropriate a level of abstraction.

8. No treatment of uncertainty

When working with numbers, we know when we think about it that almost all numbers are uncertain. Some, like the speed of light, are accurate to one part in 10^{12}. But, most quantities in practical business modeling or government policy are far less accurate. It is often helpful to represent those uncertainties explicitly as probability distributions, so that you can evaluate and manage the risks, and discover which sources of uncertainty are the most important.

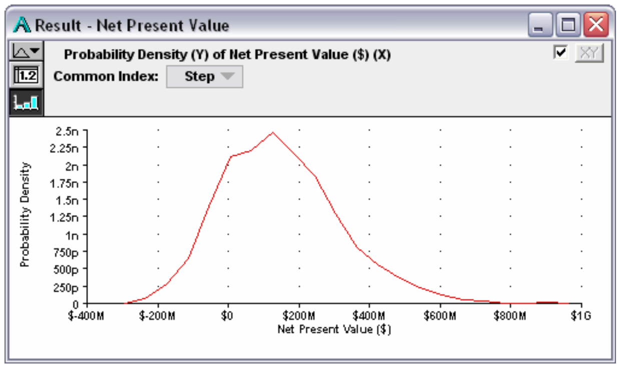
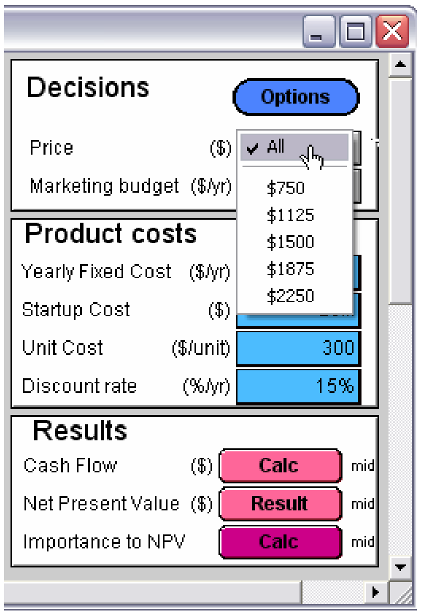

Result - Net Present Value
Probability Density (Y) of Net Present Value ($) (X)
Common Index: Step

Figure 11: Example probability distributions on corresponding results.

Common spreadsheet applications do not let you express probabilistic uncertainties directly, although there are spreadsheet add-ins that make this possible. Analytica offers probabilistic modeling of uncertainty as a built-in facility. Being fully integrated into the tool, the probabilistic analysis is much faster to compute and usable with minimal training. The uncertainty about any quantity may be expressed by a probability distribution – a standard distribution, such as a uniform, normal, or triangular – or as a custom distribution. To calculate a model probabilistically, it uses Monte Carlo simulation, or the more efficient Latin Hypercube simulation to generate a random sample from each distribution and propagate them through the model. The sample is treated simply as an additional dimension, and propagated with the same Intelligent Array features just discussed. Analytica automatically estimates the probability distributions on results and can display them as probability ranges, density functions, or cumulative probability functions, as you prefer.


Decisions

Price ($) ✓ All
$750
$1125
$1500
$1875
$2250

Marketing budget ($/yr)

Product costs

Yearly Fixed Cost ($/yr)
Startup Cost ($)
Unit Cost ($/unit) 300
Discount rate (%/yr) 15%

Results

Cash Flow ($) Calc mid
Net Present Value ($) Result mid
Importance to NPV Calc mid

Figure 12: If you select "All" as an option for an input variable, in this case **Price**, the model performs a parametric analysis to display the effects of changing the values on the results of interest, as shown in Figure 13.

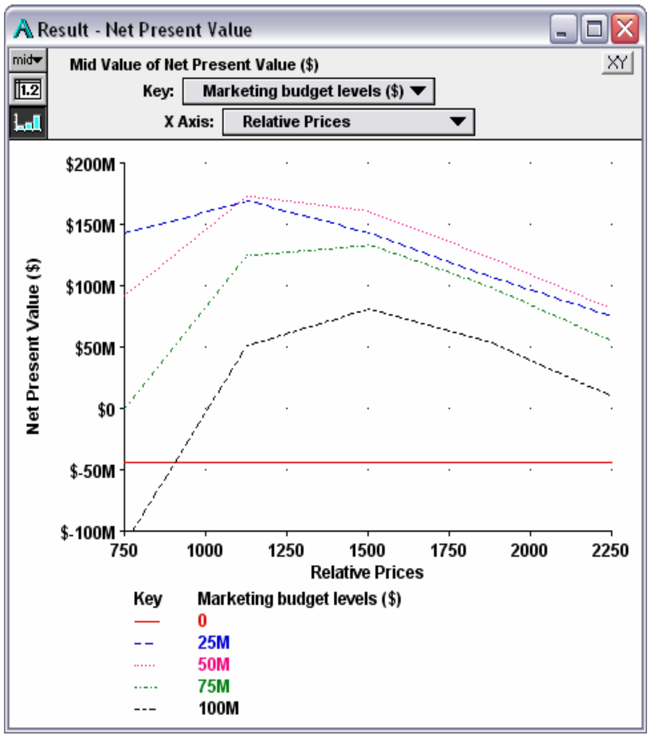

Result - Net Present Value
Mid Value of Net Present Value ($)
Key: Marketing budget levels ($)
X Axis: Relative Prices

Figure 13: Parametric analysis showing how the net present value of the new product varies with pricing and marketing budget.

9. Minimal support for sensitivity analysis

Often the most potent source of insights from modeling arises from sensitivity analysis, to help figure which inputs and assumptions have the most effect on the results, and why. Sensitivity analysis includes simple *what-if analysis* to see how a change to one input affects the results, *scenario analysis* to examine the effects of combinations of input values, *parametric analysis* to graph how changing one or more inputs across several values affects the results (see), and *importance analysis* to identify the relative effect of the uncertainties about the various input assumptions (see Figure 14).

It is, of course, easy to do single cell what-if analysis with a spreadsheet. You just change an input and look at the change in the result — provided you can remember or record its previous value. Excel also provides powerful tools for scenario analysis, allowing you to define and compare the results of scenarios. Parametric analysis is a little harder but can be achieved by judicious definition of scenarios.

Importance analysis requires explicit representation of uncertainty. You will need a proprietary add-in for the spreadsheet. Analytica provides these kinds of sensitivity analysis as basic functions. You can easily provide any input with a list of alternative values. By default, the input node will show a pulldown list which includes "All". (See figures 12 and 13.) Selecting "All" for one or more inputs automatically generating a one or more-way parametric analysis over those variables.

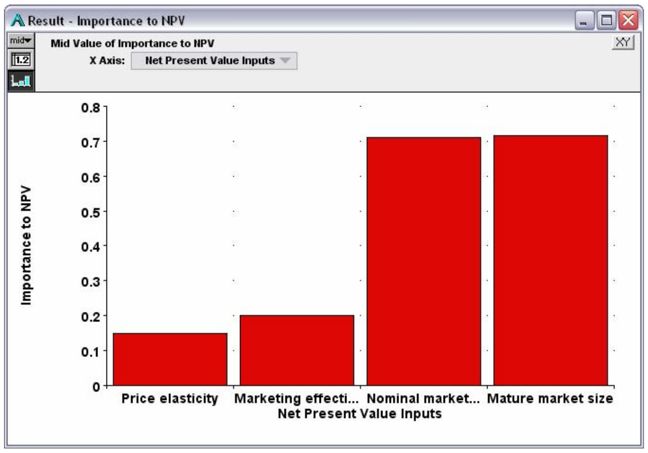
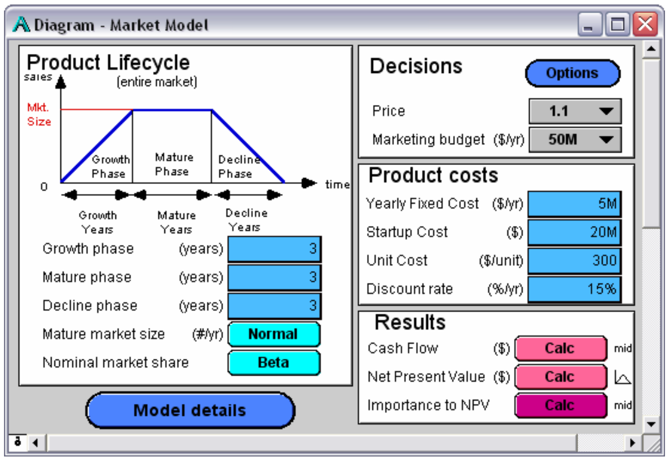


Figure 14: Importance analysis, showing the relative importance (rank order correlation of Monte Carlo samples) of the uncertain inputs on the uncertainty in the resulting NPV. It suggests that reducing uncertainty about market size is much more important than price elasticity.

10. No separation between end-user interface and model details

Frequently-used spreadsheets in an organization are usually built by one person and used by many others. A good spreadsheet builder knows to separate the inputs and outputs of interest to the users in one worksheet, as the user interface or "dashboard" – from the other internal components segregated into other worksheets. Ideally, the builder identifies the input cells clearly by color or shading, and locks all the other cells to prevent accidental or deliberate tampering with other parts of the model not designed for changing by end users. However, audits of operational spreadsheets find that builders often fail to maintain this clear separation and locking. Unfortunately, spreadsheet applications provide them little assistance in doing so.



Growth phase	(years)	3
Mature phase	(years)	3
Decline phase	(years)	3
Mature market size	(#/yr)	Normal
Nominal market share		Beta
Price		1.1
Marketing budget	($/yr)	50M
Yearly Fixed Cost	($/yr)	5M
Startup Cost	($)	20M
Unit Cost	($/unit)	300
Discount rate	(%/yr)	15%
Cash Flow	($)	Calc
Net Present Value	($)	Calc
Importance to NPV		Calc

Figure 15: An example end user interface for the Market model. It arranges three groups of inputs and one group of outputs (Results). "Model details" in the module at the bottom are available for review but not to change.

Analytica provides a special class of module, a *form*, designed to provide the human interface for end users. Drawing an influence arrow from a form module to a variable automatically creates a corresponding input node in the form, displayed as an input field, pull-down menu, or edit table, according to the type and dimensions of the variable. Drawing an arrow from a variable to a form creates a corresponding output, displayed as a number or button opening up to display a table or graph. The input and output nodes are aliases to the original variables, allowing the original variables to remain in their functional positions in their original modules, but with links set in the form.See Figure 15 for an example.

In a spreadsheet, the modeler who wants to preserve the logical structure of the internal worksheets would need to make cells or tables that copy from inputs or to outputs. In Browse mode, Analytica lets the user change only designated inputs, and prevents direct changes to any other part of the model. The Analytica Player offers only Browse Mode. The Enterprise version allows any model to be locked in Browse mode, no matter which version of Analytica the end user has.

Analytica's Intelligent Arrays offers an extra flexibility for end users. If an Index variable, such as Years, is made available as an Input, the user may modify the corresponding dimension. Any input table indexed by Years will then automatically expand or contract accordingly. In Spreadsheets, the only practical approach is to create all tables with enough rows and columns to handle the largest possible application – most inefficient when most end users need many fewer. Allowing end users to add dimensions to tables is quite impractical – something which is quite easy in Analytica's methods for "parametric analysis"

User-interface metaphors and levels of representation

Like any user-oriented software, the usability of spreadsheets lives or dies by its user-interface metaphors. The spreadsheet grid as a user-interface metaphor — so well-adapted for accounting — is poorly suited for more sophisticated modeling. The limitations of the grid metaphor are, I believe, a major cause of the high frequency of errors in spreadsheets. It makes errors easy to make and hard to detect. It requires massive redundancy in the representation of relationships between tables. It does little to encourage structured documentation and modular organization. Worst of all, it does not represent high-level knowledge about a model: It does not match the way an expert analyst thinks about the problem. Spreadsheets are therefore unable to assist the user in creating and testing a model at the appropriate level of representation.

We can summarize the deficiencies of spreadsheets by saying that they work at too low a level of representation. The grid deals with cells. The modeler is – or should be – thinking about higher level entities, such as variables, influences, modules, hierarchies, dimensions and arrays, uncertainties, and sensitivities. A modeling tool that displays these entities and lets the user interact with them directly feels much more intuitive and empowering to a modeler. It reduces the need for mental translation between the representations used by the modeler and by the software. It makes it much to write, review, verify, explain, and extend models. It reduces the number of errors by preventing many kinds of errors from being made in the first place, and by making remaining errors easier to detect and fix.

Building and analyzing business models and other kinds of quantitative models is primarily about improving your qualitative understanding of the problem leading to better decisions. This understanding involves identifying what decision options or strategies are worth looking at, what the objectives are, which issues and uncertainties might have the largest effects on the results, and the reasons or conditions to prefer one decision over another. It is usually a social process that involves several people developing an improved shared understanding. For all these reasons, an effective model is one that improves communication. Representations like influence diagrams provide an excellent way for people to discuss and explain the qualitative structure of a model. High quality documentation is essential if people are to understand the assumptions of a model. Sensitivity and uncertainty analysis helps answer questions about what factors are important to the recommendations and why.

For coherence and consistency, the diagrams and documentation should be completely integrated into the quantitative aspects of the model, rather than, as they often are, entirely separate documents, in different applications — say Powerpoint, Word, and Excel. In principle, such documents can be linked together using OLE, but these links cannot easily maintain consistency of structure between representations.

Why do we need a different application to fix spreadsheets?

The deficiencies of spreadsheets are scarcely a negative reflection on their developers. Spreadsheet developers, notably of Microsoft Excel, seem to be aware of many of these problems, and they have made heroic efforts to address some of them — including named cells and ranges, the audit tool, the outliner, cell locking, data validation, pivot tables, among others. The problem is that it is very difficult to retrofit the original spreadsheet paradigm with the needed higher levels of representation. It is hard to make these features integrate well with each other while being completely backward compatible. For example, Excel lets you choose the text to the right of a cell, or row, as its name, but it does not remember the connection when you update the text cell. Even if you change the name of a cell, it does not update the formulas that use that name. If you add columns or rows adjacent to a named range, it does not extend the range. It would be very hard to provide automatic propagation of changes to table dimensions in a spreadsheet. Because the tables are embedded in the grid, it would require automatic insertion or deletion of rows or columns in the grid, which might break other parts of the model using the same rows or columns but not indexed by those dimensions.

It was much easier for the author and his team to design Analytica as a modeling tool without having to worry about backward compatibility (although, the first version, known as Demos, was in fact designed around the same time as the first spreadsheet in 1979).

By designing the features as a coherent whole, we were able to obtain great synergies among them. For example, since variables can represent arrays, an influence diagram can represent a large multidimensional model with relatively few variables. If you needed a separate variable for each cell, the diagrams and influence arrows, would be overwhelming and incomprehensible – imagine using Excel's audit tool to display arrows for the dependencies among all the cells in a spreadsheet. Hierarchical modules allow organization of much more complex models into a set of comprehensible diagrams. Automatic propagation of changes to indexes and addition of dimensions works well in Analytica, but would be problematic if the tables of varying sizes were embedded in a fixed spreadsheet grid.

Analytica is not the only modeling tool to offer any of the features that fix the listed deficiencies in spreadsheets. There are decision analysis tools that provide influence diagrams (if not hierarchical modules). There are spreadsheet add-ins that support treatment of uncertainty with Monte Carlo simulation. And there are OLAP tools that provide versions of array abstraction. However, Analytica is the only tool we know of that provides all these three key elements as an integrated whole, and can therefore benefit from the considerable synergies among them.

References

Bricklin, Dan (2001) "The first spreadsheet" http://www.bricklin.com/firstspreadsheetquestion.html

Panko, Raymond. "Errors in spreadsheets" *The Journal of End User Computing*, Vol 10, No 2, Spring 1998, pp 15-21 1998. http://panko.cba.hawaii.edu/ssr/Mypapers/whatknow.htm

Howard. RM & Matheson, J. 1984. "Influence diagrams". Principles and Practice of Decision Analysis, Strategic Decisions Group.