PART

One

Introduction

CHAPTER 1

Why? What? Who? Where? and How?

THE IMMORTAL QUESTION(S)

If you use the Search feature for Amazon.com and search for the combination of subjects "Excel," "VBA," and "Modeling," it currently returns a total of almost 500 results. Some of these are essentially duplicate entries. A book with three editions will show up once for each of the editions.

What makes this book different?

Why should you spend the time reading it, and even much more time learning the concepts and practices laid out within?

What do you need to know even before you start?

If you are already proficient or even expert with Excel, what possible advantage can you gain by learning the Visual Basic Application (VBA) language?

What can you do with this knowledge and how will that help you?

Is learning a programming language as difficult as everyone says it is?

These questions are all reasonable and valid. Except for the specific reference to Excel, they are the broad questions that any author should be prepared to answer whatever the subject of their work.

I am prepared to do just that.

Why Is This Book Different?

This book is different because it is the only book that equally balances the mindset of modeling with a code intensive approach, and at the same time is aimed at the complete novice.

What do I mean by that?

Although this book has as its subject a structured finance model application, its intellectual focus is to get you to start thinking about dynamic problem solving. We will take a moderately complex set of rules and relationships and help you to break them down into some very small components. The process of the book will work with you to implement these discrete elements without losing sight of the bigger picture.

To accomplish this you will be exposed to VBA code, more VBA code, and even more VBA code! You will see a few, then tens, then hundreds, and finally thousands of lines of VBA code. At the end of each chapter you will see the model

in a progressive and additive state of development. First there will be just the Excel workbook of a single worksheet containing the deal waterfall and nothing else. Then over the course of the following chapters we will add one piece of code and then another and another until at last we have a finished model. I will tell you now that you cannot learn VBA by just reading it. At the end of each chapter you need to take the code out and play with it. Step through its operations using the VBA debugger tool. Look at the structure and the types of VBA statements used to achieve the results of the incremental addition of the chapter. Then take the code copy it, muck around with it and see if you can replicate its functionality on your own. Start simply and start small. Don't attempt to build an elaborate menu error-checking subroutine the first time out. Make a scaled down version of what you are seeing and then try to build up once you have it working. If you do this you will make much more progress and at a faster rate than you would if you don't. Don't worry, I will provide you with lots of code to look at! You will see practically every single line of VBA code used to create the model. It is up to you to dive in and start swimming. In fact it is the only way to learn.

There will be little or no theoretical discussions about anything. There will be no discussions about the Zen of design, or the Tao of programming. That can wait! It can wait until you have mastered a critical minimum subset of the VBA language. It can wait until you can fashion problem-solving code in a manner that will not be a danger to yourself or others. It can wait until you know enough to have your own informed opinions. That time is *not* now.

I have strong opinions about what works and what does not work, all based on 25 years of experience. You will hear them expressed from time to time in the material to come. The focus of this book is on concrete, immediate, measurable progress toward problem solving through the use of VBA.

The two effects that this will have are:

1. You will become much more facile in jumping between the specific and the general scope of decomposing and reconstructing complex problems. In a phrase, you will increase your mental agility.
2. Upon recognizing specific problem elements, you will be able to reflexively choose the correct and concise VBA code to express its logical or computational solution.

By the end of this book I would hope that you have substantially improved your mental acumen in problem decomposition and synthesis, and that you have begun to think in VBA. Here I will make one final point. When learning a foreign language, one starts with basic vocabulary and the rudimentary syntactical rules and works from there. As one builds upon this early knowledge, the vocabulary is expanded, and certain short expressions are integrated into one's repertoire. There, then, is a stage where entire sets of sentences spring to mind as complete entities. Final mastery of the language is accomplished when one dreams in the language.

I do not expect you to dream in VBA by the end of this book, but I do expect you to at least catnap in it.

In a word, this is not a book about a computer language.

It is not a book about structured finance per se although the applications are particularly relevant to the conditions in the financial markets today.

It is a book aimed specifically at improving your complex problem solving abilities.

It is a book about giving you the ability to translate your thoughts into a medium that is relatively easy to learn and that can be powerfully applied across a wide Range of practical problem solving processes.

It is a book whose goal is to add to the inventory of your personal tool kits an increasingly valuable, and monetarily rewarded, dimension. By the way, in case you missed it, the crucial word in the preceding sentence was the word **monetarily**.

I want everyone who reads this book, who does the work, and who perseveres in the pursuit of this knowledge and these skills to understand that this book will make you more valuable. It will do so by giving you an added dimension that many of your peers do not have. These skills will improve your personal market value and your career potential.

Who Is This Book Aimed At?

In order of immediacy, this book is aimed at the following people:

- Men and women working in the financial industry, especially at the levels of vice president and below, who want or need to develop their modeling skills in a self-paced environment. This especially relates to the current environment. The volumes of structured finance deals being currently created and issued is significantly less than it was last year or two years ago. Much of the analytical effort is now directed to helping risk managers understand what they have and the prospects for their positons in the current market environment. For people involved in these activities the book packs a double punch! You will see the development of a structuring model that will teach you the fundamental elements common to all structured finance deals. You will then see the transformation of the structuring model we will create into a monitoring and valuation model. This is an especially valuable twofold experience for people working in today's marketplace.
- Intermediate level managers who supervise these people but have little or no knowledge of, or experience with modeling. They want to know what is possible, and how it is to be accomplished. More importantly, they need to be able to form realistic ideas of what is possible in a given time frame. There is a saying about tradeoff: "I can be fast, accurate, and cheap. The problem is that you can have only two of the three." As mentioned above, in today's environment the aspect of risk management and portfolio monitoring is critical. Intermediate level managers will need to be up-to-date as to the current health of their businesses. Modeling in VBA/Excel providers them with an excellent platform to improve their information flows and knowledge of the sensitivity of their holdings.
- Students in graduate or undergraduate programs wishing to develop modeling skills.
- Anyone that wants to have the ability to intensively explore problems that require a quick, fluid medium of analysis. This includes anyone in the financial world dealing in risk and especially those that seek to measure it and evaluate risk arbitrage through modeling.

- Anyone in any commercial, nonprofit, government, or military function that needs a tool for dynamic problem solving.
- Anyone that is a regular Excel user and wants to significantly expand the scope of their ability to address issues that the use of Excel alone cannot.
- Anyone that wants to ask the question "What If?" and needs a powerful, easy-to-learn tool to translate their thoughts into concrete quantitative reality.
- Lastly, anyone that thinks they know and are good at VBA. I am certain that you will find many useful techniques here that you may have overlooked.

What Assumptions Do I Make About You?

- Basic computer literacy.
- A working, although not expert, knowledge of Excel.
- A working knowledge of algebra is required to understand some of the bond math and mortgage amortization processes.
- A positive attitude and a willingness to mentally engage in the process. It is assumed that the reader is willing to put in the work and to think about the work that they are doing. Learning a programming language, especially VBA, is not easy. It is however, not that difficult either. This of course posits two assumptions:
  1. You really want to learn.
  2. You will continue to want to learn and will persevere beyond the first, second, or third setback.
- That you are a professional in the broadest sense of the word.
  - That you want to produce a quality product and that you will take the time and make the effort to do so.
  - That you will feel a sense of responsibility to yourself and others not to produce a model that is hard to work with, easy to make mistakes with, or inaccurate in its function.
  - That you will take responsibility for your work by completely testing and validating the model results.
  - Lastly, that you will take the time to protect yourself and others by providing a minimum of documentation necessary to prevent the model from being inappropriately applied.

WHAT ARE THE ADVANTAGES OF LEARNING VBA?

When I was ten years old I announced to my grandfather, who was a plumber, that I felt that I had mastered enough mathematics to last me the rest of my life. The scene of this revelation was a modest summer cottage by a lake. I was feeling rather reflective. It was a hot day in July, and the reality of returning to school was a distant idea. My conclusions seemed quite reasonable.

He smiled at this and made little in the way of return comment.

The next day he asked me to dig a hole off to the side of the front yard. He gave me a small plastic coffee scoop and an old table knife. He led me over to a stake he had placed in the ground about three feet from a hedge. He told me the hole must be exactly one foot on each side with sharp clean corners. I was to do nothing else

until I had finished the task. Nothing else specifically included playing with friends, swimming, and fishing with him.

At first everything went smoothly. I cut through the sod and the first six inches of topsoil. I then hit the roots of the hedge, a rock the size of dinner plate, and lastly a mixture of clay and pebbles. Two-and-a-half hot days later, complaining, bored, frustrated, and angry I finished.

The next day he had me repeat the process using any tool I wanted from the shed. I was finished in 15 minutes. That, he said, was the difference between life with and without the knowledge of mathematics.

I did not think about this incident for years until I was called to travel to the London office of the investment bank with which I worked to debrief a leasing specialist who was leaving the firm. He had created an Excel model to run the cash flows of a structured finance asset-backed securitization (ABS) deal. This model was viewed by everyone in that department with a mixture of awe and reverence. The fastest PC in the office was dedicated solely to the running of this model.

We spent the first two days reviewing it. It consisted of over 65 spreadsheets that amortized 1,900 individual leases. He indicated to me that he had made some changes to it recently. These changes had decreased the runtime from its original eight to nine hours to the present six and a half hours. At the moment, however, we needed to produce 15 scenarios for one rating agency and nine for another. Eight more were required for internal credit approval and four by the credit wrap provider of the deal. In total, that was 216 hours of runtime, not factoring the setup times for each scenario, and assuming that everything went well thereafter. The vast majority of the computational time was taken up in determining the monthly cash flows from the existing lease agreements and then calculating additional cash flows from a re-leasing agreement that served to extend the terms of the original contract. The uses of the cash from the collateral were straightforward and represented a small portion of the computational burden of the program.

I began by replacing the thousands of columns of Excel that calculated the individual leases. Using amortization subroutines that I had previously written in VBA (very similar to those in this book), we were able to eliminate all but seven of the spreadsheets. Five of these spreadsheets were menus; one displayed the monthly collateral cash flow summary and one the performance of the debt supported by those cash flows. This moved the vast majority of the computational burden of the model from the Excel spreadsheets and into the VBA code.

The runtime of the new program, (even before we tried to optimize the VBA calculation sequences), was now reduced to eleven minutes per scenario. This was a 97% reduction in the time it took to produce a single scenario. We next implemented a simple looping structure that allowed the model to run groups of related scenarios without human intervention. Finally we separated the output reports from the model itself into standalone Excel files that used a common report format. This allowed us to produce up to 50 variants of a base scenario without human intervention. Now we could put in the scenario specifications, turn the model on, and go to a series of well-deserved extended lunches. Upon our return we would find various sets of files, each containing a unique scenario. The model had finished, printed the files including graphics, sorted the scenario files based on their characteristics into particular directories, and produced a summary report resident in the model.

Some holes can be dug far more efficiently with knowledge of VBA than without such knowledge. There are also entire classes of problems that cannot be solved by Excel alone!

The combination of Excel and VBA is far more powerful than either product alone. Excel is a wonderful medium for quickly changing report configurations. It also offers a wide Range of options for the graphic display of information.

VBA can be used to control the Excel environment in a surprisingly wide Range of activities. It can also replicate complex sequences of tasks that an Excel user would have to perform manually. This frees the analyst for other, more valuable work (interpreting all that newly available data), and increases individual and team productivity.

VBA allows the user to explore a wider Range of situations and to pursue the analysis to a deeper level than would be possible by using Excel alone.

There is one further lesson to be emphasized from the above. The key to the rapid rewrite of the model was the fact that I had previously written a collection of VBA subroutines that performed cash flow amortization for leases. The VBA code was written for leases on railroad cars, not aircraft, but was easily changed to meet the somewhat different aspects and conditions of the second deal. I also had a set of preconfigured report files and VBA code to transfer the information from the model to these files. All of this work we were able to employ immediately in the aircraft model. It was intact, tested, and available to the new model. The ability to transfer previously developed work to new projects is an enormous productivity boost. We will engage in just this kind of activity late in the book. We will use a structuring model as the basis for the development of a risk assessment and valuation model for the securities we created with the first model.

As you learn and practice VBA you will find that two things will happen. First, you will be able to develop new VBA modules faster. (Practice may not make you perfect, but it will make you formidable.) Second, if properly written, you will be able to repeatedly reuse VBA code you have previously developed in new projects. These two factors will synergistically interact, allowing you to dramatically reduce your model development times.

WHAT ARE THE DISADVANTAGES OF LEARNING VBA?

As enticing as I hope you have found the preceding section to be, we must balance it by presenting the disadvantages and barriers to learning VBA.

The first and most obvious is the natural fear of change. This seems paradoxical if one believes that the change will ultimately be beneficial. There is still a hesitation to abandon the tried and true, the familiar ways of doing something. There is always comfort in the ways of the devil that you know, as opposed to the one that you do not. Different ways of doing things can change relationships, workflow, and the perception of relative value between peers and managers. New ways can leave some people who are not a party to them feel excluded and disenfranchised. They can also create new classes of problems for others who are not directly involved in the change but who are affected nonetheless.

My answer to the above dilemma is simply that LIFE = CHANGE. Change occurs whether we want it to or not. The one thing that is known is that those

who have more capacity to deal with change deal with it better. VBA can improve your productivity, which will, in the long run, give you more time, not less. It also gives you another response option to business problems. I am not representing that learning VBA is a panacea that can be applied against all possible future career situations. Learning VBA will ***not*** solve all your troubles. It is self-evident, however, that having more resources rather than fewer are better when dealing with change.

The second is the issue of time. I can barely keep up with my work now. I need to balance time spent on my career and time spent on my family. I feel that I am overburdened at present and now you want this and me to do everything I'm doing now, too? Taken as a whole, VBA has tens of thousands of features, methods, objects, commands, and rules. You do not need to know all of them to be quickly productive in a meaningful sense. You need to know fewer than four dozen statement types, variable types, objects, and methods to produce the model we will develop in this book. I have selected the most critical and the most widely useful elements of the language as the focus for creating the model code. This is a fast track to real, working VBA code.

The third issue is the entire concept of learning to program at all. You will become a programmer. Does this mean you will immediately begin wearing plastic pocket protectors and sport inch-thick glasses? No. You will be the same person you now are, except you will have an additional and valuable skill at the end of the learning process.

Is it easy to learn how to program? Yes and no.

Yes, if the material is properly selected. Yes, if there is a manageable subset of basic concepts that is progressively developed. The pace must be manageable so that the material can be understood and internalized by the student. Most important is your commitment to learn. The first program I ever wrote had 16 statements. As it turned out it also had 25 errors! Learning how to program in VBA is not simple, *but it is not impossible*, or even terribly difficult for that matter! It is a precise process. You need to be careful. It can be frustrating to have the model produce results that are clearly wrong when you are sure you have programmed it correctly. As you practice you will become better, faster, and much more confident. We will build that confidence through the examples at the end of each chapter. If you have the patience and the commitment you will find the process rewarding.

No, if you are bombarded by too much too soon. VBA has thousands of features that you will never need to learn, or if you do, you would be better off not knowing. Why? There is a tendency in some beginner programmers to try to find a use for every feature they read about. This practice is not only counterproductive, it is also dangerous. It leads to overly elaborate programs that are nearly impossible to understand by anyone other than the original author. In a fast-paced and demanding business environment, it means that it will be difficult and time-consuming to understand and to modify. That type of code is mostly characterized by the use of the object-oriented features of the VBA language, which are inappropriate for beginning modelers to tackle their first time out. It also makes the VBA models they write less accessible to other modeling generalists, who lack a formal computer science education and who may inherit the code after them.

Lastly, there is the issue of too much of a good thing. If the model can produce five scenarios, why not run 10, 20, 50, 100, 500, 1,000, 10,000? Why not run all possible value combinations of each of the 30 dependent variables and the 10 independent

variables that are the model drivers? There is a tendency for some managers to ask for too much information simply because they can get it. It is possible to miss the relevant trees through the forest of too many scenarios. The situation can also arise where the analyst wishing to demonstrate their technical skills produces so much information that the decision maker is swamped by sheer volume of the model results. This, at best, leads to confusion and delay, at worst to errors, missed opportunities, and losses. This is not as uncommon as you may think it is.

The answer to whether VBA is easy to learn is common sense. Some work will be required, some will be optional, but common sense and communication should serve to ameliorate the worst of these situations.

So having weighted the alternate advantages/disadvantages, we now must decide. To learn VBA or not to learn! I vote for learn! I believe it is always the best course to try to improve and grow. We need to stretch ourselves and push out our boundaries. Learning VBA is a way to do that. Your problems may change, but you will have one more tool to deal with them. It has worked for me and thousands of other people.

If you have continued to read to this point you have de facto accepted that the advantages of VBA just might outweigh the disadvantages. With that premise in hand let us address the next question.

WHAT IS A MODEL?

The Basic Concept

A model is an abstraction of a real world situation. That is what everyone can agree upon. Unfortunately past that point of agreement you can find yourself in the position of the person who asked the nine blind men what an elephant was and received nine very different answers, each based on personal perspective.

My view is that a model is a computer program that is configured for investigative purposes. This is the classic "WHAT IF (conditions 1, 2, 3) = RESULTS." Its goal is to allow the model user to move from the known into the unknown. It is different in this way from a program that balances your checkbook or keeps track of the medical histories. The function of these programs is rather simple and one-dimensional. They tend to exist merely for the manipulation and reconfiguration of static information.

Models are dynamic. They tend more towards the consideration of complex problems rather than simple ones. They produce a Range of results as opposed to a single answer. They focus on the effect of multivariate degrees of change. They also seek to illuminate the interdependencies of the components of the phenomena they are built to describe. They seek to produce an informationally "dense" or "rich" result, on several different layers of complexity or of several dimensions (especially time).

More often than not they will raise just as many questions as they answer.

Difference between a Spreadsheet and a Model

A common question is "When does an application stop being a spreadsheet and start being a model?"

Spreadsheets develop a set of answers from a set of equations. This set of answers represents a single descriptive entity, or scenario. That is, however, as far as it goes.

If you have purchased a portfolio of assets and know their empirical performance as to payments and losses, a spreadsheet can produce the expected case. You can change one assumption, run the spreadsheet, and get another case.

A model, however, will be designed to produce sets of scenarios. These scenarios will have value in and of themselves if viewed in isolation. Their value will significantly increase if viewed as elements of a unified set. They may be taken as a series of risk points along a continuum from what is desirable, through acceptable, to catastrophic. The value of the individual scenario results is greater because they form the descriptive elements of a much greater whole. It is this type of collective result that is most characteristic of models versus spreadsheets. The other defining characteristic of the model is that it provides for a collective interpretation of this result set—the ability to look at one result, several results, or all results, and conceive a framework for thought or a unified perspective. That perspective might be as simple as a grid exhibiting a single result parameter for each of the component analyses, or as complex as a three-dimensional graphic.

Complexity, Synergy, Dynamism, Sensitivity

Models are particularly important for capturing and describing complex systems. Complex systems tend to exhibit behavior that is not always incrementally uniform or predictable.

They are also excellent at capturing synergistic relationships between various independent and dependent variables. Depending on the Range of scenarios produced and the degree to which the model's report package is designed, all kinds of interesting interrelationships can be described and studied. This is especially true of VBA models used in Monte Carlo simulation analysis, where the result set can contain thousands, tens of thousands, or even hundreds of thousands of scenarios, which combined together, comprise the whole of a single model run.

Dynamism is very difficult to capture with the use of a spreadsheet alone. When a model produces a collection of scenario results, it is possible to measure the degree and type of change triggered by variations of the value of a single variable. The time it takes an analyst to run a properly designed model and capture the results of as few as 25 scenarios is tiny compared to the effort required to replicate those same results from a spreadsheet model. Why is this so? Models, because they are designed to produce a set of scenarios, will have the results at hand. A spreadsheet produces one scenario at a time. The results must then be correlated and displayed in a manual process. This tends to bog the analyst down in the mechanics of data collection and presentation, instead of freeing him to focus on what the results are saying.

Sensitivity analysis is the last element that separates models from spreadsheets. Not only can we see broad trends from a collection of scenarios, we can look at incremental change between any pair-wise comparisons of available results. We can see if a given set of linear changes in an independent variable produces a similar set of changes in the value of any of the dependant variables. This is particularly important when a high degree of analytical granularity is required. How thinly can a change process be sliced before it becomes significant to the decision? It is these observations that are sometimes the most important output of the modeling effort.

WHY IS MODELING A VALUABLE SKILL?

Modeling is valuable because it is the most powerful tool we have to describe systems that are too complex for manual computational solutions.

In the business world, modeling is preponderantly focused on the issue of financial risk. Describing a continuum of risk is virtually impossible without the use of computer models. There are entire fields of finance that were created only after computers became fast enough and financial analytics became robust enough to describe the risk inherent in the products. This risk not only had to be measured, it also had to be described, and then packaged in a way that was comprehensible to the non-specialist. This often involved extensive "What If?" analysis from a wide Range of constituencies both internal and external to the business unit originating the financial product.

Learning to model will improve your analytical and product skills. You have to know what you are to model before you can model it! Modeling forces you to decompose complex systems and then rebuild them from the ground up in the world of VBA. The combination of the development of these general intellectual skills coupled with the specific skill of modeling is a useful package to have.

Having said all of the above, let us finally get to it. **How does one model?**

WHAT ARE THE STAGES OF MODEL DESIGN AND CONSTRUCTION?

Stage 1: Defining the Deliverables

The emphasis of this book is on concrete deliverables. What am I going to learn, how do I put it to use, what will be the end product?

Begin the modeling process by describing the successful end product. How will you know when you are done? This sounds sophomoric, but many modeling efforts bog down and then self-destruct because people do not set clear concise goals. As the project evolves, goals are changed, and changed, and changed again. You find yourself chasing a moving target. You can never succeed because you can never finish.

You need a set of goals to be able to point to as the deliverables! You can then claim success when they are achieved.

What Do the Decision Makers Need to Know? What are the concise, finite, and, above all, specific results you need the model to produce in order to enable the decision-makers to decide? Get a list; check it twice (even if you are not Santa Claus). Be very careful to specifically layout the calculations and form of the results the decision-makers want you to produce. Make sure that you have specific agreement as to form, Range, and granularity of the desired results.

Then get explicit signoff, by e-mail or some other document if possible.

What Is the Scope of the Modeling Effort? Is the model to address a specific case or a set of more general cases? How much effort is willing to be spent in terms of personnel, money, and time? Is this going to be a prototype for a series of models or is it for a unique situation that is unlikely to occur again? Will you need to cooperate

with other personnel? What external constituencies will be involved? Do any of them have special requirements that will have to be addressed by the model in addition to your own requirements?

Who Is the Target Audience? Who, other than yourself, will see the output? Will other parties require special outputs or, more importantly, special methodologies that are not in the original internal plan for the model? What is the level of knowledge, sophistication, and need for detail of all the parties who will see the output? Will the results have to be packaged differently for other parties in your firm, including members of the management team up the line? This last point is particularly important in that senior managers often require that an entire briefing fit on one side of a single sheet of paper.

What Will Be the Frequency, Intensity, and Longevity of Use? Who will run the model? Will there be error checking, reasonableness checking, checking for conflicting computational options, checking for infrastructure issues (for instance, is the data file where you said it was?)? How frequently is the model to be run? Will the model produce several scenarios, tens of thousands of scenarios, hundreds of thousands of scenarios? This will be particularly important when it comes time to determine how much optimization effort you will put into designing and writing the VBA code.

Is the model a one-trick-pony, or is it anticipated that it will be continually developed over many years as part of the ongoing effort for a product line? Will the model be needed to perform ancillary functions such as monitoring the deals after they are created for either risk or regulatory purposes? Remember that if the business is subject to rapid change, the model will have to continually reinvent itself to keep pace with the evolution of product development in the business environment. It will be linked to the successive challenges the business faces and must be robust enough to meet them in a timely and accurate manner.

Is the expected profit (or avoidance of losses) from the use of the model substantial enough to warrant extraordinary measures in its development? This has significant impact on the degree, extent, and complexity of the validation effort. It may also necessitate the involvement of third parties to replicate and verify the model results.

Stage 2: Research and Abstraction of the Real-World Problem

What to Put In, What to Leave Out One of the most critical early-stage decisions in model development is what aspects of the real-world problem to include in the scope of the models and which ones to ignore. Some items are obligatory, some discretionary, and others a matter of judgment. Obviously this question is intimately related to some of the issues we laid out above. What are the resources and time frame of the development effort? Will it accommodate extensive study of all the real-world elements we seek to model? Can we learn more about the impacts of the various inputs? Can outside parties give us guidance or education on these items?

A specific case in point is regulatory issues. Any insurance product in the United States or the United Kingdom is a highly regulated item. To correctly understand the behavior of various reserve accounts requirements, we hired a consulting firm.

In the end, we had a team of three insurance specialists spend over 80 hours each educating the deal group and especially myself in how to calculate certain of these requirements.

One is also faced with the absence of data or the absence of data in a directly useable form to the modeler. It then becomes necessary to make a judgment as to at what level of abstraction or approximation is sufficient or acceptable to include in the model. In another instance, we had to analyze a portfolio of home equity lines of credit. When we received the data, a key item was missing. Inquiring of the client, we found that they did not capture this information anywhere in their systems. They would be happy to do so going forward, but not retroactively, on the 10,000-loan portfolio. Unfortunately this information was critical to both an internal risk analysis and one of the rating agencies. We therefore assembled an audit team, and with the deal team, flew to the client site. We had called them beforehand and had them assemble sets of 25 randomly selected loans. We read the missing information from each loan file and then constructed a statistical estimate of the values for the portfolio as a whole. These were then used in the analysis. VBA was critical in this instance in helping to develop a statistical distribution of this data. This population was sampled repeatedly, and its new values inserted into the collateral files, allowing an estimate of the risk that a specific parameter introduced into the deal.

Description of Key Interactions There needs to be a list and description of all-important activities that make up the real-world event. The sequence of these should be determined and the computational or data manipulation requirements described in as much detail as possible.

Mathematical Formulae Any phenomena that can be reduced to mathematical formulae should be described in as precise a manner as possible. If there are existent standards for the calculation or measurement of the process, these should be strictly employed and earmarked for special validation efforts. If there are industry-accepted methods for the implementation of particular algorithms, these should be determined and adopted. If the client or any of the internal or external constituencies require an idiosyncratic calculation or approach, it must be researched and validated. It cannot be stressed enough that you ***must know these things*** before you start writing the VBA code. You can have 80%, 90%, or even 95% of the model building process complete and yet hit a snag that may force you to undo or discard hours and hours of work.

Human Element If there are judgmental issues in the modeled process, one must make allowances for them. This is to say that decision branching caused by human decisions must be anticipated and allowed for. If collective or cascading human behavior (the madness of the crowds) is a significant factor in the modeled issue, you must make allowances to include it. If it cannot be explicitly included, it may be possible to implicitly include it by allowing manipulation of the behavior of other variables.

Chance Some tasks and behaviors are deterministic and some are probabilistic. You may need to determine how you will model items that are subject to chance. Mortgage defaults are a perfect example.

Most mortgage default behavior is probabilistic. Unless there is a specific economic factor causing systemic default stresses, borrowers default in a random manner. A portfolio may consist of 20,000 mortgages that share common demographics, similar balances, and payment terms. Individual defaults that occur are generally modeled in this portfolio by deeming a certain fractional percentage of each loan to have defaulted. This is in fact standard industry practice and an accepted risk measurement approach. What happens if, instead, 20% of the portfolio is contained in 10 of the loans? If even one of these loans defaults we will see a huge increase in default rate for the period. If measuring such volatility is important (and it is), we will have to consider other methods. In the preceding case we may wish to separate these 800-pound gorillas from the rest of the zoo and treat them as a separate portfolio unto themselves. We may wish to model the defaults of these large balance loans by using a Monte Carlo technique that would randomly default their entire balance, rather than subjecting them to the proportionate attrition techniques generally applied.

Environmental or Regulatory Risk If the modeled process is subject to changes in law or regulatory modification, elements reflecting those potential changes may need to be provided for.

Organizational Constraints The last items that may need to be considered are constraints affecting the modeled phenomena by self-imposed organizational practices. One such issue is sector risk. How many companies in a particular industry are we willing (or allowed) to do business with? How much business with any one of them and how much with all of them in aggregate? What will be the maximum duration of our involvement?

Another factor to consider is the organizational policies that govern the development of models in general. Are there specific testing and development milestones that will require you to seek signoffs from other parties within the organization? Are there independent model validation groups that will need to certify the accuracy and scope of the model prior to its deployment and use?

The above are not meant to be an exhaustive list of considerations, but ones to start you thinking broadly about this part of the process.

Stage 3: Develop the Fundamental Framework

Sequence Functional Procedures of the Model Now the process begins to move from the abstract to the concrete. Having completed the initial fact-finding, you next will sequence all the activities the model will have to perform from the beginning of its run to the end.

Broadly, these will fall into three macro phases common to all computer programs of whatever type: input activity, computational activities, and output of the results.

In regards to input, you will need to determine the sources, volume, character, and method of input that the model will require. Will the inputs be in the form of menu-specified data and choices from the user, files, Internet feeds, or randomly generated information? Will the data be sparse or voluminous? Will it be simple in form, or will it require extensive preprocessing inside the model to become useful?

Will it be numerical, alphabetical, or mixed? Will it all be available at the beginning of the model run or will some of it be developed along the way?

For the computational aspects, you will have to determine the sequence and timing of all calculation activities. You will need to identify, specify, and pair each calculation with each portion of the computational activity associated with it. You will need to think through sequential dependencies. One aspect related to this will become vitally important as you become more familiar with the use of looping structures. These structures allow the repetitive execution of specified VBA statements contained within the boundaries of the loop. Model performance can be severely degraded by placing computational activities within a loop when they can be done more efficiently outside of it.

Output of results also needs to be carefully considered. In the initial stages, develop a conservative plan for the minimum amount of output that will satisfy the decision criteria. Do not be too aggressive. Carefully survey the exact scope and detail needed in the output. Confirm with the intended audiences that these results are both sufficient and robust enough to satisfy the requirements of the project. This is an excellent time to begin design of the actual form and content of the reports that will present this information.

Define the Major Activity Blocks The next step in the planning process is to move from the three phases described above to a specific breakdown of activity units. For example, at the first level, the input phase now becomes the following:

1. Read the menu worksheets of the model.
2. Verify and conduct error checking of the contents of the menus, returning the user to the input phase if errors or inconsistencies are found.
3. Read the names of any data files and determine if they are in the locations specified.
4. Open and read the data.
5. Read the names of any links to external systems; verify the pathways or site names.
6. Establish connection and download the required information.
7. Move this information to the variables of the model so that they are available for the computational phase to follow.

This degree of specification will then allow us to decompose the problem further. In the case of item 1 we can now design VBA routines to read each of the individual menus. This process of detailing will then give us a list of specific tasks that need to be accomplished at the micro level.

Writing Pseudo-Code (at least once) The final activity will be to write pseudo-code or lines that describe the activity that you will later replace with actual VBA language. Pseudo-code is an excellent way to organize and structure your model. It is, however, time consuming. Do not worry! Writing pseudo-code is a skill that you may well find valuable when faced with planning and writing horrifically complicated models.

I recommend it to beginners as a way to clarify your activities at a very macro level. For the first time modeler it will probably save more time than it will cost. It can also be a useful practice for the experienced modeler when you have to venture into new practices or a different application area. This is especially true if you are

seeking complicated issues for the first time. It can be particularly useful when used to walk others through what the model will do and how it will do it. This allows them to be part of the organization of the model with no need for programming or technical involvement, and also assures you that the correct steps are being taken in the correct order to solve the problem.

Stage 4: Implementation into Software

Translating Functionality into Code Now you will have to translate the overall macro design into a first cut at a model. Determination of what to construct in Excel and what to construct in VBA is the big decision here.

What to Implement in Excel Implement in Excel those portions of the model that have high visibility. There is a significant efficiency in being able to look right at a portion of the model and see what is going on. In the VBA sections you will need to use a facility of VBA itself to do this. It is called the debugger. In Excel you can literally see all the ancillary functionality already built into the product to assist you. Such features as the "Trace Dependents" option in the Tools menu can be invaluable as a fast, accessible, and easy-to-use diagnostic tool.

Extend the visual concept to all menus and user specifications. Use Excel to express those elements of the model most subject to change. If you are going to provide the user with any type of intermediate results, provide Excel spreadsheets to display this information.

Lastly, all of the output should be created and displayed in Excel. This will allow you to make rapid changes in formatting. It will also allow you to fully exploit the tremendous Range of visual effects Excel has to offer.

What to Implement in VBA VBA should be reserved first and foremost for the most computationally intense portions of the model calculations. This is especially true of complex calculations that have a high frequency of repetition. If, for instance, you have to sort through 30,000 items and make multiple determinations on each item, do it in VBA. Likewise, if you have to perform a complex, repetitive task, such as the lease amortizations I alluded to earlier, use VBA.

As you gain experience you will develop an instinctive nature of which parts of the model to place in what medium. What I intend to do is to provide you with enough examples so that you are well on your way to this goal by the end of the book.

Compilation and Debugging After you have finished writing the Excel and VBA portions of the model, you will need to compile the program and debug it. Debugging is the skill (and some would say art) of identifying errors in your program. I have devoted an entire chapter, Chapter 16, "Debugging the Model," to this subject. VBA has an excellent, powerful, and informative debugger capability. You may or may not have heard horror stories about debugging models. They are all true. (Just kidding.) With patience and common sense, most VBA programs can be debugged in relatively short order. With practice you will learn to avoid common mistakes in writing VBA code. This will substantially decrease the number of errors the debugger will find. You will be able to concentrate on those types of errors that are more difficult to identify and fix. Once again I would emphasis that **patience and practice** are 90% of the key to mastering debugging techniques.

Stage 5: Model Validation

The task of model validation is as unique as each model you will create.

Some elements of validation are actually accomplished in the debugging phase immediately above. That is the phase in which most simple errors are caught and fixed. These as a class are generally basic arithmetic mistakes. Often tables of results from independent sources are available to directly compare your results to. This is the case with handbooks and Web sites that deal with bond math. Many of these sites offer numerous specific examples that can be easily checked in considerable detail against what your model is doing.

For more complex issues you can sometimes use other previously validated VBA models, or you can replicate portions of the analysis in Excel. You can also enlist others for second perspectives, especially if they have more product knowledge or experience. You will find that these people (and you will soon be one of them) have an instinctual feel for what is a reasonable set of results and what is not.

If others will use this model, you should involve them from the design process onwards. It is imperative that you reconfirm that the design, scope, and computational content of the model meet their requirements before proceeding. Walk the potential users through the model and explain how it is to be run. Get their feedback. Make suggested changes now to marshal support for the model. Like the model debugging process, the model validation process has a separate chapter devoted to it, Chapter 17, "Validating the Model."

Finally, when you are satisfied with the results, you can release the model.

Stage 6: Initial Deployment of the Model

First Use Start simple. Do not overreach or over-promise in the early stages. Establish credibility for yourself and the model. If other people are going to use the model, make sure you take the time to lay out the series of steps needed to run the model correctly and then sit with them as they run the model. This will tend to defuse any negative feelings should problems arise because you are there to help and guide them. It demonstrates that you are equally invested in the process of using the model, not just in the process of developing it.

Interpreting the Results Get buy-in as soon as possible from the decision makers that the model addresses the core requirements in a sufficiently detailed and accurate manner. Listen to what they say and do not let pride of ownership interfere with your ability to hear their comments. Take criticism constructively; remember if they do not succeed, you do not succeed! Use the initial results as a framework that leads to thinking about the next steps in the model development.

Documentation This is the time to complete whatever documentation efforts you have decided are appropriate. The term "complete" is used because if you follow the practices of this book, you already will have performed a substantial amount of documentation in writing the model. This documentation will be completely contained inside the model itself. Most of it will take the form of comments. Comments are notations that are made in the VBA code, but which are not VBA statements that the computer can act upon. Comments are for the benefit of the person who has

responsibility for maintaining and expanding the code. In all likelihood that person will be you. You will be the "expert" on the model for some time to come. By leaving yourself these explanatory notes and identifying remarks you protect both your firm and serve your own interests. We will discuss commenting in much more detail later.

If there is to be additional documentation produced that is resident outside of the model, now is the time to do it. Do not procrastinate! This is especially true of training and user documentation, the dos and don'ts of running the model. The documentation you create will hopefully put bounds around what is acceptable user behavior. Despite this, you may find all manner of aberrant behavior being practiced on your helpless model! Curb these practices with guidance, training, and documentation.

There is a tendency to feel that the initial deployment of the model is a time to take a break after the development stage. Documentation can be a pain and is generally underappreciated. Despite this, get it done now, while the ideas are fresh in your mind and not dimmed by the passage of time. Documentation has much in common with a serious dental problem. It doesn't get any better by ignoring it or deferring it. It only gets harder and more painful to correct the more time goes by.

In addition you must realize that it is an ongoing process! As you make each change to the model you must update a Change Log to tell everyone (and yourself later) when and where additional changes to the base model were made.

Stage 7: Mature Deployment of the Model

Segregation and Protection Once the model has weathered its first set of deployment issues and has become temporarily stable, it is time to lock it up like a prisoner in a maximum-security jail! (You can console yourself that this is for its own good . . .)

Create a copy of the model along with any validation runs and supporting material. Place it in a secure directory inaccessible to the user community. Even you should not have the ability to change this first copy. Many companies refer to this copy as the "gold copy." It can be copied but not modified. This is for several obvious reasons:

- It protects the original work from corruption by you or others (more emphasis on the others!).
- It provides a measurable base line to compare any legitimate changes you may make in the future. If you develop other expanded versions of the model, it can be used to immediately validate the parts of the model that did not change. This is important because you do not want the incremental work to affect the existing work unless it was meant to do so. In either case, you have an immediate benchmark to measure the change, or lack thereof. This process is called "back testing."
- It provides a history of the model building activity and links the model to a specific time period and analytical approach.

Training of a General User Group If you are going to roll the model out to a wider user group, this is the time to conduct more formalized training.

Stage 8: Continued Evolution of Model Scope and Complexity

Adding Methodologies Well-constructed models often create the need for change. They excite thought and facilitate creativity. This is especially true if they are economically valuable, generating revenue or preserving or expanding competitive advantage.

At some point, and probably sooner than you think, you will be called on to expand the model. You will add new methodologies and the process of model development will begin anew. If you follow the general precepts we have talked about, this will not be traumatic. Challenging work, perhaps, but not traumatic. A set of improvements to the basic model that we will build are presented in one of the final chapters of the book. These are broadly representative of the typical kinds of model capability extension that you may be called on to implement. Chapter 19, "Building Additional Capabilities," can be a guide to what you might face.

Improving the User Interface One of the most common immediate demands is to change the way the user deals with the model through interfaces such as menus. Repeated use will suggest other and better ways to do things that were unforeseen in the original design conception. This is a positive development and far superior to one in which people use the model for a short while, become dismayed or disgusted, and abandon it.

The most common immediate improvements to the interface involve providing the user with the ability to specify more independent variables. You may also want to allow the user to combine the independent variables into combinations that were not possible in the previous version of the model.

Another issue is increased granularity of the model's inputs. For instance, inputs that were expressed as annual rates may next need to be specified on a weekly or daily basis as well.

Expanding the Model to Other Analytical Tasks

Over time the model may be applied to other tasks that are related to, but different from, the model's original role. A model that originally began life designed to structure securities may be transformed into a model that accepts historical data to monitor the ongoing, and project the future, performance of the original deals over time. The same may be said of an issuance model that is transformed into a pricing model to support trading activities in the secondary markets of the security.

OTHER ASPECTS OF MODELING

Productivity Enhancement for Follow-on Modeling Efforts

There is a very interesting, synergistic, aspect to modeling. If you build reliable, efficient VBA code for the first model, you will find that you will be able to reuse significant portions of it for the next model. Not only that, but as you build up a library of models, you may eventually get to the point where a significant portion, (30 to 50%) of the model is comprised of reusable code. This is equally true whether

you are writing models for a business that has a series of similar deals or one with only a single specialty deal. How can this be?

Let us take the case of an individual input menu. The role of this menu is to perform the standard menu tasks of receiving user inputs and choices. It then transfers them into VBA variables that make the information available to the model. Many of the fields of this menu will be common across all applications because every application needs to execute a certain irreducible set of tasks to function. Filenames and pathways must be read from the menu. Runtime options must be specified: do this, do that, do not do this. Data arranged in time periods must be ordered and made available to the VBA for calculation purposes, and so on and so forth. The VBA code that performs these functions will be virtually identical across models. All you will have to do is change the names of the variables to be better able to identify their contents in respect to the specific application. If the original names of the variables are generic enough you might not even have to rename that many of them!

It is therefore possible to design generic menus, report generators, sorting routines, and selection routines.

Similarly, it is possible to pre-configure files containing templates for frequency distributions or cumulative loss curves. Day-count methodology menus also fall into this category of reusable technology.

Report files can incorporate previously configured graphics that can be used across many different types of models. Your VBA code will simply supply the idiosyncratic results of the specific model and the titles, spacing, scaling, and color of the various components that you wish to be automatically displayed. This is especially true of graphics that display the timing and magnitude of the cash flows of the model. All amortizing assets such as loans and leases have scheduled principal repayments, coupon income, defaulted principal, prepaid principal, and recoveries of defaulted principal. Standardized graphics can be prepared and set aside to be integrated into any model that requires these performance criteria be visually displayed.

The last subject of the book is the process of using an application, the initial model, that you have already produced, to quickly build another, related, but different application. Here we will take the model that we spent the entire book building and in one chapter transform it from a structuring model for deal issuance into a risk assessment and valuation model. This second model will monitor the health of the deal. It will provide a means of placing an ongoing, (and changing), value to the securities as they respond changes in the performance of the collateral pool and market conditions.

These are only a few of the many examples of the use of replicable VBA code. If you design the code well, you will eventually construct a box of generic spare parts and components, from which you can quickly develop new and different VBA engines.

Risk Control

"Key Person" Risk One of the nightmares of everyone working on a team with tight deadlines and high financial risk reward issues is the loss of a key team member. While there is nothing that VBA can do to defeat the loss of human capital, VBA's very design can mitigate the departure of a valued team member.

A well-planned, well-commented model, with supporting documentation, will at least preserve the essence of the analytical approach to the problem. From a

managerial perspective this tends to provide a solid starting point for recovery. VBA is a straightforward language. It lacks some of the sophistication of its rivals such as C++ and Java, but is easier to maintain and learn than either of them.

PERSPECTIVE OF THIS BOOK

The preface of this book outlines the perspective the author intends to follow.

This book will assume that you are a new analyst or associate who has completed the corporate training cycle in an investment or commercial bank. In your training you will have been exposed to the various business activities of the organization and acquired a fundamental knowledge of financial theory and calculations.

You are replacing a person who is leaving in two weeks. You need to sit with them and come up to speed on the current set of projects you will be assuming. One of these projects is the modeling support for an asset-backed securitization deal that the bank feels highly confident of winning. The modeling for the deal must be complete in about two months.

The person you are replacing has gotten the project off to a good start. He has built an Excel spreadsheet that correctly models the debt structure of the deal. One part of the model that is NOT complete is the calculation of the amortization, payments, defaults, and recoveries of the several thousand loans that constitute the client's loan portfolio. Expert in Excel, the outgoing analyst has tried a number of different spreadsheet solutions. None proved feasible—the analyst is stumped as to how to develop a spreadsheet application that can amortize thousands of loans across a wide variety of assumptions.

"Well, you seem pretty bright!" he says as he is leaving. "Good luck. It's your problem now!"

Indeed it is.

You have the Excel spreadsheet but the columns that will receive the cash flows from the loan portfolio are blank. You have a data file from the client with the proposed initial loan portfolio information. You know from other deals done by the department what output reports are needed for this deal.

You also have this book. This book will walk you through, on a step-by-step basis, the construction of a robust, flexible, and easy-to-run model. It will teach you how to wrap the Excel model in a series of menus to bring in information. It will teach you how to read data from other Excel files and how to write reports from the results produced by the model. It will help you build a valid, working model that will accomplish more in a single afternoon than your predecessor was able to accomplish in a month using Excel alone. The model you will build will allow you to answer a wide Range of questions and to adroitly manage the risk assessment and structuring requirements of the deal. You will look like a star to the other members of the deal team and the management of your department. You will create an analytics engine that will describe the financial risk of the deal, serve the client, and answer the questions of the rating agencies.

After the deal has been created you will now embark on an equally important activity, measuring its performance! We will walk through the process of converting your newly created structuring model into a monitoring, surveillance, and valuation model. Once we have this monitoring model in hand we will examine the

performance of the deal at quarterly intervals for the first two years of its life. We will look at the performance of the collateral performance and the debt structure and see how they weather the current credit crisis.

When you have completed this process you will have made a significant step forward in your professional development.

This is your chance to make your predecessor's problems your opportunities!

STRUCTURE OF THE BOOK

The book is comprised of seven parts and a number of technical appendices.

Part One is the Introduction and Chapter 2 deals with concepts of modeling in general.

Part Two consists of Chapter 3 and describes the asset securitization process and the business case study we will model.

Part Three focuses on model design. Chapter 4 deals with understanding the structure of the Excel Waterfall spreadsheet of our predecessor, and Chapter 5 the design of the VBA code we will place around it.

Part Four consists of six chapters that begin to teach you the VBA language. We start, in Chapter 6, by creating the internal structure of the model in the form of VBA modules and its external structure in the form of specially named directories to hold the model and its files. Chapter 7 introduces you to VBA code that is generated from simple recorded macros. Here we will use the VBA editor for the first time and look at recompiling our edited code. You will learn how to compile code and run a short macro. In a rudimentary way, this will be your first experience programming. This chapter will lead directly to Chapter 8, where we discuss VBA data, arrays, Ranges, and objects. You will learn what objects and methods are and will experiment with the Range function. In this chapter you will begin to implement simple menus. In Chapter 9, we will address the VBA statements that allow you to control the flow of the model. Here you will learn how to have the code make decisions, assign values to variables, and perform calculations. The next chapter, Chapter 10, introduces you to the concept of VBA code output and shows you how to produce error and progress messages for the user. Chapter 11 finishes the section by describing the reporting function.

Part Five focuses on writing VBA code—lots of it. Chapter 12 tackles the main program and the declaration of a host of global variables. These variables will hold all of the input information about the portfolio and intermediate calculation results. They will also hold results from the model run, the portfolio level cash flows. Chapter 13 will take you through the process of writing a collateral selection methodology. Chapter 14 calculates the cash flows on a loan-by-loan, period-by-period basis. Here you may wish to detour to the two Appendices at the back of the book on Mortgage and Bond mathematics and concepts. We will also introduce the concept of calculation optimization. Chapter 15 will teach you how to transfer the VBA results back out to the Excel waterfall, load it, and calculate it. We will also write the code to capture the results of the Excel waterfall and transfer them back to VBA variables for later output. The last chapter of the section, Chapter 16, will teach you how to use the debugger program to detect and correct different types of errors in the code you have written.

Part Six deals with the initial deployment of the model. Chapter 17 will discuss validation techniques and testing. We will want to be sure the model is correct before we use it on a real deal. In Chapter 18 we get to run the complete model for the first time and come up with the securities structure that will meet the requirements of all of the interested parties. These Range from our own internal constituencies to investors, rating agencies, and regulatory groups. Possible solutions to different problems we encounter will mirror what can happen in a live deal. After we have run a number of scenario sets, we will move on to improving the model. Chapter 19 describes the process of expanding the model with new inputs, calculation methodologies, and reports.

Part Seven addresses what happens after the model is deployed. In Chapter 20, we will discuss the subject of documentation: how much, what, and for whom. Chapter 21 describes how you protect and grow your model so that it can continue to create value for the business unit.

Part Eight concerns the fate of the deal once it has been launched. In Chapter 22, "Building a Portfolio Monitoring Model," we will do just that, drawing heavily off of the code and reports we created for the structuring model. This will allow us to put this model in place in a fraction of the time it took us to develop our first model. In Chapter 23, "Valuation Techniques: How Do We Determine Price?" we will look at the fundamentals of valuing the deal after it has been issued. In Chapter 24, "Challenging Times for the Deal," we will analyze the progress of the deal over the first two years of its life. We will employ our newly developed monitoring model to project the cash flows based on the empirical performance of the collateral to date. We will apply these cash flows to the deal's structure along with default and prepayment projections and other assumptions about the market to arrive at a series of pricing benchmarks for the notes.

Part Nine contains a single chapter, Chapter 25, "Parting Admonitions," which gives you some last words of advice, consolation, and appreciation, and caps the entire process you have undergone.

The book closes with a pair of technical appendices. These sections include a mathematical discussion of mortgage and bond math used in the model.

PUTTING THE DELIVERABLES "ON THE TARGET"

Emphasis on Deliverables in a Business Context

Every chapter starts with an **"Overview."**

The next section is named **"Deliverables."** This is what you should have achieved by the end of the chapter.

These deliverables are broken down into knowledge goals and production goals.

The **"Modeling Knowledge"** goals will relate to concepts and terms in finance, modeling, and programming. The **"VBA Programming"** goals will be the elements of the VBA language, VBA commands, logic structures, and code deliverables in the form of subroutines and functions you have written.

At the end of each chapter you will have a **"Deliverables Checklist"** section that will test you on the contents of the chapter you have just completed tied directly to the items in the **"Deliverables"** section.

In this way the completion of every chapter will add to a list of your intellectual and product development accomplishments. You will have a detailed list of specific goals to meet, and a checklist at the end of the chapter that tells you if you have met them.

Another section that will regularly, but not always, appear in the chapters is the **"Under Construction"** section. In this section anything that is being build, new template files, additions to the model, menus, new calculation capabilities, actions buttons, or even standalone programs will be described.

The next to last section will be entitled **"Next Steps."** This section contains a brief review of the chapter you have just completed and how it connects with the next two or three chapters. This section is designed to provide you with the perspective of how the material is integrated over a three-to-four-chapter span of ideas and techniques.

The last section of most chapters will be **"On the Web Site"** and will hold a description of the material related to the chapter that is available on the Web site of the book.

There is an ancient Roman saying, "Familiarity breeds contempt." From an intellectual perspective, this concept can be dangerous. As you progress through this book you will be learning something on almost every page. You may not be aware of the amount of knowledge and skill that you are accumulating. This is particularly true in some of the introductory chapters, for example, Chapter 7, "Recorded Macros: A First Look at the VBA Language." This chapter seems a little passive in nature. We record a macro and then look at it. This is a simple macro that prints a column of numbers, sums them, and changes their font color and bold settings. Amazingly, this example introduces you to almost everything you need to build a simple model. It introduces objects and methods, variable assignments, looping structure, and output manipulation. It is the first concrete link you will see of the direct interaction between VBA and Excel. It actually contains about 80% of all the concepts we need to design models in VBA. In addition to all the VBA features of this example, we also are exposed to the VBA Editor and the VBA Debugger for the first time.

The ironic thing is that if you do not realize how much you have learned, you may feel as if you have hardly learned anything at all! The Deliverables section therefore is designed to give you a line-by-line list of your accomplishments. This is important, because the ultimate deliverable, a working model, can appear to be a large and distant goal for the beginner.

I want to re-emphasize that the way to succeed in this endeavor is to reduce this large task to bite-size entities and master them one-by-one in easy, incremental steps.

Development of Your Human Capital

There are seven major reasons why a person should learn to model using VBA/Excel:

1. Career development and enrichment
2. Managerial and peer recognition
3. Significantly increased responsibility, recognition, and reward
4. Broadening your business knowledge and perspectives

5. Potential for substantially more responsibility, exposure, experience, recognition, promotion, and reward than indicated in item 3
6. Intellectual stimulation, feelings of accomplishment, and the satisfaction of having added to your personal growth
7. Very much more responsibility (Oh! The responsibility is killing me!), promotion (Call me SIR!), and rewards (we hope!)

So take your pick of odds or evens, or even better, a mixture of both. Whatever your reasons, I hope you enjoy what is to come.