CHAPTER ONE
INTRODUCTION

This is a book of case studies in which each study is an example of how some form of Model Driven Architecture® (MDA®) has been introduced into an organization to help solve a real-life business problem. These organizations include three large corporations (two in Europe and one in the United States) whose yearly revenues range from $450 million to $182 billion (U.S. dollars), as well as three governmental agencies (two in the United States and one in Europe) whose sizes span a similar range.

The respective projects involved are commercially important–in some cases vital—to each of these organizations. Moreover, each organization made a conscious and significant commitment to a new approach to managing their software life cycle both when they first decided to use MDA and as they continued to absorb the many implications of that decision. All of them have plans to expand their use of MDA in the future, sometimes in ways they had not previously considered. MDA, it seems, can be somewhat addictive.

So, what is MDA? At the conceptual level, MDA is a holistic approach to improving the entire information technology (IT) life cycle–specification, architecture, design, development, deployment, maintenance, and integration–based on formal modeling. More specifically, MDA is a framework of technical standards progressively being developed by the members of the Object Management Group (OMG)–an open industry consortium supporting this approach–along with a set of usage guidelines for enabling the application of those standards with appropriate tools and processes.

There are a lot of MDA products, even though MDA "compliance" is as yet undefined

Exactly as intended, the release of MDA by the OMG has spawned a wide variety of commercial products purporting to be "MDA compliant." Exactly what that label means is not clear, in that at the moment neither the OMG nor any other organization has any formal mechanism for testing product compliance with any specific MDA standard. There are also a number of vendors selling products that purport to support a "model-driven approach" without specifically claiming to implement or support MDA per se–that is, as defined by the OMG.

Just as predictably, the increasing availability of MDA-based tools is spawning a host of new books and articles about MDA and related approaches (including by the authors of this book). Most of these focus on explaining the theory of MDA, surveying the various standards, or describing in detail the features of specific MDA-based products or techniques. We welcome these books, even if we don't always agree with all of them, and hope for many more to come.

For this book, however, we decided on a different approach–to produce a book about MDA based solely on real-life case studies. In our experience, a good case study is the best (and easiest) way for most people to begin to determine whether or not any particular approach is likely to work for themselves or their organization. If they come to the conclusion that the approach will work for them, they will more likely be willing to wade into the more formidable technical details. Furthermore, by including a healthy half-dozen of such case studies in a single book (drawn from a wide range of organizations and applications) we hope that nearly every reader will be able to find examples that speak directly to their own needs and experience.

So, rather than plunge into MDA per se we simply present the stories of six very different organizations, each of which used MDA to address its business needs in very different ways. For the most part, we believe these case studies stand on their own and speak for themselves (both individually and collectively) and therefore have not tried to embellish them with additional details. That is, what you will read is pretty much just what we were told by the participants.

The really good news is that you don't need to know much about MDA itself to understand these case studies. Anyone with a general familiarity with typical information technology (IT) issues and jargon will have no trouble identifying with most, if not all, of the individuals and organizations involved. That said, for those who might need it we have included a short MDA primer as an appendix, a bibliography, and a glossary that should fill in any remaining conceptual or buzzword gaps.

Otherwise, our primary purpose in writing this book is to "explain" MDA through real-life examples in order to help you, the reader, consider the business case for adopting MDA in your own organization. In each of our case studies, there is an MDA "champion," someone who sensed early on that MDA could really make a difference and who pushed through its acceptance by the organization (at least for the project in question). Perhaps you already are, or will soon be, that person in your own organization.

Champions tout MDA's business benefits rather than its shiny new technology

As you will see in the case studies, that champion may come from IT or from the business itself. He or she may be a C-level corporate officer, an enterprise architect, or a program/project manager. But in every case that champion found a way to demonstrate effectively how the introduction of MDA could substantially benefit the business. Nobody described in the book was just trying to add another

acronym to his or her resume, or merely playing around with some sexy new tools.

That is important to point out, because some people still believe that MDA is just another technical approach for generating code–a new form of CASE (computer-assisted software engineering) based on the popular Unified Modeling Language (UML)–and therefore something far from the concerns of the business. It is true that MDA was originally based on UML, and also true that it is often used to automatically generate code and other software life cycle artifacts. To that extent, MDA carries forward some of the ideas of CASE. But that's really yesterday's news.

MDA is about gaining control over the life cycle of business solutions

Today, as our case studies clearly show, MDA is being applied as an overall approach to gaining control over and systematically improving the entire life cycle of IT solutions–from modeling the overall business and capturing specific solutions requirements to developing, deploying, integrating, and managing many kinds of software components. Today's MDA is less about generating code per se and much more about precisely capturing requirements, enforcing architectural standards, maintaining traceability, and facilitating effective communication between the business and IT (and between different parts of IT).

On the surface, MDA often seems to be different things to different people. To some, MDA is still mainly about generating code from models. To others, MDA is about capturing business requirements more precisely and completely. To yet others, MDA is a way of managing the evolution and integration of existing systems. And as in the story of the blind men and the elephant, all of these perceptions–and more–are equally valid and equally incomplete. This suggests that like the proverbial elephant MDA is already emerging to be much more than the sum of its currently perceived parts.

There, we said it: MDA is a paradigm shift

We really believe that the emergence of MDA represents one of those so-called paradigm shifts, and that it will eventually change everything about the way software systems are specified and built. This is always a controversial position, particularly in IT, which has seen so many supposedly revolutionary "paradigm shifts" come and go. If the readers of this book ultimately reach the same conclusion, it will be because of what they learn from our case studies rather than our attempts to convince them with theory or technical details.

That said, before jumping into the case studies we would like to put our own view about MDA in some type of perspective by way of an analogy. Reasoning by analogy is always risky, but it can sometimes be useful where the very concepts under discussion may be new or unfamiliar to the reader.

By now, most of you have probably already heard the term *software factory*. It may even be that this beguiling label has led some of you to MDA, and even to this very book. The term itself evokes the powerful image of developing software as a form of "manufacturing."

In this context, the "big idea" is that like a modern physical factory the workings of a typical IT department may ultimately be reduced to a set of precisely defined

processes supported by appropriate tooling, which can reliably produce high-quality software at ever-decreasing costs and time-to-market. Now, who could argue with that?

Let's remember that the roots of the word *manufacture* are the Latin words for "hand" (*manus*) and "make" (*facere*). Most modern manufacturing industries began as "crafts," in which the production of each unit of a given product was largely a unique one-at-a-time process. The "factory" was just a convenient locale where this took place, not a carefully constructed enabler of the manufacturing process itself.

In the twenty-first century, we rightfully tend to think of the craft-based model of physical production as slow and inefficient, suitable only when some form of artistic output is desired. Moreover, even more serious drawbacks manifest themselves later in a product's life cycle, when it is time to repair or replace a handcrafted unit or to integrate it with other units.

For this reason, the concept of a factory based on an "assembly line" (assembling standardized parts) was invented, ultimately replacing the craft model in nearly all forms of "manufacturing." The current popular definition of manufacturing now almost completely belies its Latin etymology. Few today would expect, or even want, most "manufactured" goods to be literally "handmade."

Eli Whitney had a lot of help–and still did not succeed at his stated goal

The notion of assembling finished products from standardized parts is often attributed to the American Eli Whitney (also of cotton gin fame). Whitney famously demonstrated the idea to the U.S. Congress in the late 1790s as a more efficient way of both manufacturing guns and maintaining them in the field. Of course, the basic idea of standardization is much older than that, dating to classical times and attributable to others (such as John Hall and Marc Isambard Brunel, who deserve some of the credit for refining the idea in more modern times).

However, neither Whitney nor anyone else actually succeeded in making the idea of true "manufacturing" from standardized *interchangeable* parts practical until the 1850s. The long delay between the concept and its practical realization had several causes. Although Whitney et al. could design interchangeable parts, could make prototypes of those parts, and could show how useful that would be, what they could not do was reliably mass produce those parts or get anyone else to. The measurement standards of the time were not precise enough, nor was sufficient machining capability available, to make this possible.

In a nutshell, Whitney just didn't have the tools needed to make the tools needed to make the interchangeable parts. An entirely new industry–machine tools–had to be invented before Whitney's main idea of interchangeable parts could become a reality. But once those machine tools–and all of the standardized processes they required–became widely available every other form of manufacturing was soon revolutionized.

What emerged is now commonly called the "factory model." This factory model is not just about putting raw materials in at one end of a building and

getting out finished product at the other end. After all, even the most hidebound traditional craft-based "factory" could do that. No, it's about an entire science of breaking the products down into standardized parts, while breaking the production process down into standardized activities, all based on formal specifications. Moreover, this componentization of both parts and activities is applicable not just to assembly but to design, procurement, maintenance, and other areas.

And it's not just the end product that is ruthlessly componentized–it's the factory itself! Modern factories aren't just designed to produce one product, or even a predetermined set of products. Companies with those types of factories go out of business quickly, the first time the market for their particular product changes. Ergo, modern factories themselves must be able to change what they manufacture at will, albeit within certain boundaries. This means the factories themselves must be built from standardized parts, which can be changed or upgraded using standardized processes.

Current software development practices resemble those used by seventeenth-century craftsmen

So, what does all this have to do with today's notion of "software manufacturing"? Unfortunately, by any reasonable standard the typical IT shop is still stuck back in the craft-based world of manufacturing. That is, the specification, development, and deployment of each new solution is still a one-off project, whether we are starting from scratch or trying to integrate disparate systems. This is true even though we may use some fancy tools, sit in a modern office (the "factory"), and share some sophisticated technical infrastructure. From a modern "manufacturing" point of view, though, we still develop software not much differently than a group of craftsmen building custom firearms–one at a time–in a seventeenth-century munitions workshop.

How does this relate to MDA and the more modern vision of a "software factory"? In spite of the (at least) decade-old hype surrounding "reusable software components" and "assembling software," the truth is that nobody has yet been able to fully deliver on that vision. In practice, you either find vendor-proprietary sets of "components" or you don't get real components at all.

In any case, even the most advanced software developers today still spend an enormous amount of time and money patching together various tools, platforms, and existing applications to build the "components" they need and then to integrate them into a new solution. Most of them probably do not realize they are following in the footsteps of gunsmiths who wore out numerous files in making "standard" parts actually fit. The resulting modified "parts," of course, are even less "standard" than the originals.

Does that mean that the idea of a modern "software factory" can't be achieved? Not at all. It simply means is that the IT industry hasn't yet fully defined the notion of "machine tools" for software. Sure, we have some fancy development tools, but as yet they are not standardized to the point that they can be easily and reliably

configured to produce "interchangeable parts" or "components" that can then reliably be combined into real solutions based on a set of formal specifications.

MDA focuses on the environment–the standards and processes–that enable the creation of software components

That's where MDA comes in. MDA is an approach that focuses on the standards and processes necessary to create true components and a reliable product life-cycle process for software through formal specifications. In terms of achieving industry-wide interoperability, MDA is still a work in progress. However, as these case studies amply illustrate MDA can already be used to figure out the best way for even a single company to architect a component-based strategy and to start transitioning toward a factory model for its entire software life cycle. Today, that MDA-based software factory might still have a few bottlenecks and shortcomings but at least we can identify and focus our efforts on removing them.

To extend the analogy a bit, we can loosely compare Whitney's achievements in the 1790s to the invention, some 200 years later, of interface definition languages (IDLs) in the software industry. That is, Whitney could show standard interfaces (interchangeable parts or components) and examples of those interfaces (prototypes). He could show how you could in theory make a system (a rifle) if you assembled those parts.

What he could not do with this equivalent of an IDL was reliably manufacture the components/parts that could support those interfaces. The inability to make those parts to the required tolerances in turn meant that there was no way to manage the "rifle life cycle" other than by taking a file to a lot of not-quite-interchangeable parts.

So, an IDL-style approach is necessary–but not sufficient–to support the factory model for software. The MDA standards, in contrast, provide a much more robust framework that will ultimately let us specify both the structure of a component and all of the associated semantics related to the way a component must function throughout its life cycle–a complete functional specification, if you will. The precise semantics of an MDA model correspond to the precise tolerances in a physical model because each determines whether the parts actually do what you want them to do–as opposed to simply appearing to fit together.

So, like the proprietary software vendors of today, Eli Whitney was able to show what could be done if you could make parts to certain specifications. He successfully demonstrated being able to shoot the gun assembled from (prototype) parts, replace a part, and then shoot again. At the time, it was an amazing feat, but there were too many missing standards and tooling technologies for the vision to be realized completely.

No matter how hard he tried, Whitney just could not build a factory–and he could not describe how anyone else could build a factory either. He could not describe a reliable process by which close-tolerance parts could be created, and he could not tell you how to build a machine that could reliably support the execution of such a process. More importantly, he did not yet even have a language with which to describe any of this.

New capabilities enable the creation of formerly unimaginable things, and we do not think that MDA is an exception to this rule

It took another 60 years to get to the point of a standardized manufacturing model, and then about another 70 years to perfect the end-to-end technology needed to support Whitney's big idea across a wide range of industries. By the 1920s, innovators such as Henry Ford had worked out enough of the technical and logistical kinks to support the production of automobiles–a far more complex process than anything Whitney was thinking about in 1790. But once that powerful blend of technology and logistics was generally understood, it ultimately enabled not only the mass production of rifles but of infinitely more complicated and sophisticated mechanisms. This is to be expected. New capabilities typically enable the creation of formerly unimaginable things.

We believe that MDA now provides the foundations for achieving all of these things in support of a true software factory. It has taken 15 years to get from IDL to MDA, so–even giving our industry credit for operating in "Internet time"–we are certainly still less than halfway to our ultimate goal of the universal software factory.

That is, we are still at an early stage in developing and applying MDA technology, and we should not be shocked that we don't yet have highly componentized interchangeable system modules available off the shelf for every kind of application. Fortunately, with the emergence of MDA we are finally at the point where we have a common language for defining true software components and for adequately describing the tools and processes necessary to create and assemble such components.

But look at what happened to the notion of the physical "factory" after it reached a similar point! Today there are many different types of factories: continuous process flow, factories that assemble parts from other factories, "lights-out" factories with little or no human presence, and so on. Visionaries are talking about factories that maintain themselves, and even factories that produce factories. With nano-manufacturing, the concept is being brought down to the molecular and even biological level. So, the general concept of the "factory" has grown along with our overall capabilities, and has been adapted to the specific needs of various industries.

MDA, although very general today, will branch into industry-specific specialties over time

It seems reasonable to expect the same sort of thing to happen to MDA technologies over time. As these case studies show, there are already many flavors of MDA in the real world today, including MDA for real-time systems, for embedded systems, for systems integration, and so on. We see no reason why this number shouldn't grow. Today, the general concept of MDA is still just being introduced to the software industry, along with some general MDA tools. But each industry always finds its own way of applying such things, just as industries applied the notion of a factory in different ways.

This "branching" of MDA was actually anticipated by the OMG, which assumed that many special interest groups (SIGs) and task forces (TFs) would emerge, each with its own flavor of MDA. As of this writing, there are already nine

such MDA SIGs and TFs formally operating within the OMG: Business Enterprise Integration; Consultation, Command, Control, Communications, and Intelligence (C4I); Finance; Healthcare; Life Science Research; Manufacturing Technology and Industrial Systems (ManTIS); Software-based Communications; Space; and Transportation.

The case studies in this book are another type of real-life demonstration of this branching of MDA. They illustrate and inform us about all of the elements that go into using MDA to guide the building of business solutions in various environments. Because we are still in the relatively early stages of MDA's development, these case studies are perhaps the best way to present the emerging big picture, including which parts of that picture are not quite yet in focus.

Modeling will someday be second nature–to practitioners and *tools*

When you read these case studies, you begin to see what people have to think about when they try to apply MDA to real-life problems. Someday, this will be second nature. Everyone will be attuned to modeling, to the process of modeling, and even to modeling the process of modeling, and so on. As the standards mature, more and more support will be built into the supporting "machine tools." Today, people still have to work their way through the overall approach and adapt it to their specific needs "by hand." But even so, these case studies show that they can still gain great advantages and benefits in the process of doing so.

So this is where we are with respect to MDA today. The industry is still in the early introductory period of MDA adoption. We are at the point where, if you are willing to fill in some of the blanks yourself, there are some very interesting proto-tools available and great benefits to be gained by applying them correctly.

Eventually, the early adopters who take this approach will not only be continuously rewarded (that is already happening as we type) but will be moving faster than their competition in the global race toward a brave new model-driven world. They are already gaining hard-earned knowledge about what it takes to develop using an MDA-based approach. So, as the MDA tools and standards continue to improve these people will be in the best position to rapidly exploit those improvements as well.

These are exactly the types of things the people in our case studies have told us: "We realized the important things were people and process," and "We realized this would allow us to sell a whole different kind of product to our customers." More than anything else, these case studies are examples of organizations that in going through the process of adopting–and adapting–MDA are coming up with innovations that not only give them a competitive advantage but help drive the global MDA revolution.

Each case study showcases not only a satisfied end user but a dedicated consulting organization

Here's one final thought before you proceed to the case studies themselves. In this early phase of MDA's own development, its current rapid dissemination must be credited not only to the OMG, to the emerging class of MDA tool vendors, and of course to the brave end users but also to the many consultants who are now introducing their clients to the benefits of a model-driven approach. In each

of our case studies, there is not only a generally satisfied end user but also a dedicated professional services provider helping that end user learn the ropes and avoid the pitfalls that inevitably come with transitioning to a new approach.

At a higher level, these MDA consultants are playing a key role in making MDA work in the overall marketplace. Fortunately, very early in MDA's life cycle a group of such consultants took the step of banding together to create an OMG-sponsored program–MDA FastStart–specifically designed to help end users new to MDA begin their transition to a model-driven approach.

At this writing, the MDA FastStart program has grown to include more than 30 consulting organizations, each of whom the OMG recognizes as a Qualified Service Provider (QSP). Every one of the consultants involved in our case studies comes from this group of QSPs. As much as anyone else, their dedication to helping others learn about and correctly apply MDA is making a major contribution to the upcoming MDA revolution.

We want to personally thank those consultants, and their end users, for contributing their very valuable time and attention to this book. As you can imagine, none of these folks have a lot of free time to spare. Yet, for each case study they agreed to sit through several hours of interviews over a several-week period. In addition, each participant agreed to review the resulting transcripts and provide additional comments and clarifications. We hope you will agree with us that, judged by the results, it was time well spent.