



MDA Journal  
September 2003

**David S. Frankel**  
David Frankel  
Consulting

## Introduction

The MDA Journal will appear every month in Business Process Trends. In it I'll explore issues pertaining to the OMG's Model Driven Architecture (MDA) and from time to time I will invite guest contributors to present their perspectives.

Our focus will be how MDA relates to Business Process Management (BPM). This relationship is still being hashed out, but I'm convinced it's important.

I wrote a white paper on this subject entitled "BPM and MDA" for the June issue of BPTrends. Howard Smith, co-author of the flagship Business Process Management book, responded in the July issue. This column is a response to the interest those papers evoked.

I kick off the MDA Journal this month with an article about the role that MDA is playing in the industrialization of software, and the juxtaposition of that role with that of BPM. This article reflects my latest thinking on MDA and BPM, and is written from a macro rather than technical perspective.

Here are some samples of the subjects of some other upcoming articles:

- The relationship of MDA to Domain Engineering and Product Line Practices. These are software development approaches coming out of the Carnegie Mellon Software Engineering Institute, the same group that brought us the Capability Maturity Model.
- How MDA and BPM will enable model-driven runtime systems management based on highly formalized Service Level Agreements
- Model-driven metadata management

I sincerely thank Publisher, Celia Wolf, and Executive Editor, Paul Harmon, for affording me this opportunity to communicate with you monthly about MDA.

This month's paper, **Software Industrialization and the New IT**, begins on the next page.

MDA™ is a Registered Trademark  
of the Object Management Group.  
The logo at the top of this page is a  
Trademark of the OMG.



## Executive Overview

The thesis that Nicholas Carr outlined in his recent article for the *Harvard Business Review*<sup>1</sup>, wherein IT has reached the end of its build-out phase and is now a commodity, is right in important ways. The OMG's Model Driven Architecture™ (MDA) is partially a response to the pressures of this commoditization. Drawing on lessons learned from industrial manufacturing industries, MDA seeks to industrialize the production, deployment, and operation of software.

However, Carr is also missing a key point: The convergence of computer technology, business process management (BPM), and MDA is a qualitatively new technological leap that fundamentally changes the role of IT in the modern, value chain driven enterprise.

## Commoditization—Where Carr is Right

In his recent, controversial paper in the *Harvard Business Review*, Nicholas Carr asserted that IT is following the path of other technologies such as railroads and electricity. Early in their history new technologies go through periods of massive infrastructure build-out. During the build-out phase, these technologies confer special advantage to companies that use them. However, as such technologies become commonplace, they no longer provide advantage. They become commodities.

Carr says that IT's build-out era is now largely complete and that IT is becoming a commodity (Figure 1). Therefore, he argues, IT no longer has the power to confer special advantage. A business expects IT to simply be there, functioning reliably and efficiently while bearing increasingly heavy loads. Disruptions in service and security breaches are costly, so IT must think defensively. Carr says that the era of investing in new, groundbreaking information technology that bestows competitive advantage is over.

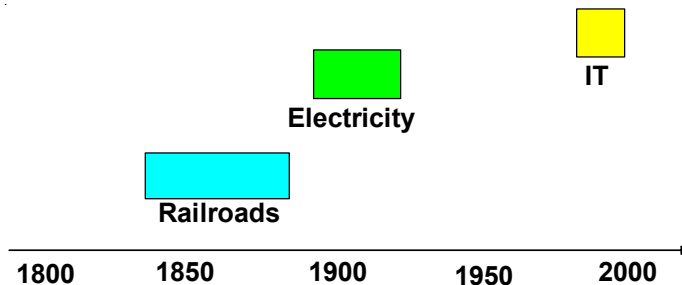


Figure 1: The Carr Thesis—Relatively Brief Build-Out Periods

There are indicators that suggest that forces of commoditization are indeed at work in the world of IT. An infrastructure of Internet technology, servers, PCs, middleware, application servers, ERP systems, and Enterprise Application Integration systems is now in place in all large organizations. The major computer systems vendors are working on creating utility grids that offer this infrastructure

### The Demands of the Value Chain Driven Business

via access and pricing models similar to those used in the delivery and pricing of electricity and natural gas.

At the same time, demands on IT departments have increased markedly in recent years (Figure 2). It is no longer sufficient for IT to automate business operations within the boundaries of the organization. IT is expected to support consumer access to its systems, and handle increasingly diverse user access devices.

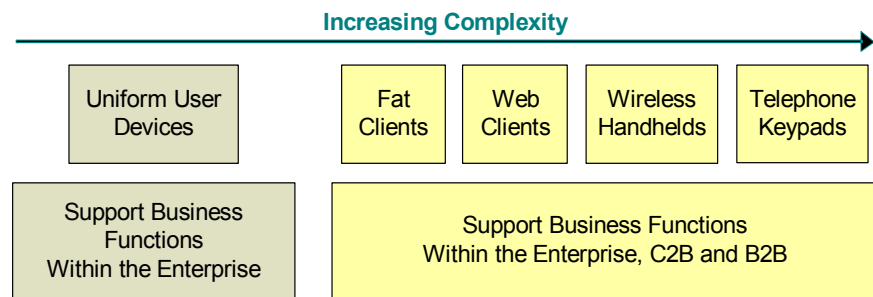


Figure 2. Increased complexity facing IT

More importantly, IT must make it possible to readily combine its systems with the systems of other IT organizations to make larger systems, so as to enable multiple organizations to flexibly combine and recombine their business processes into value chains (Figure 3).

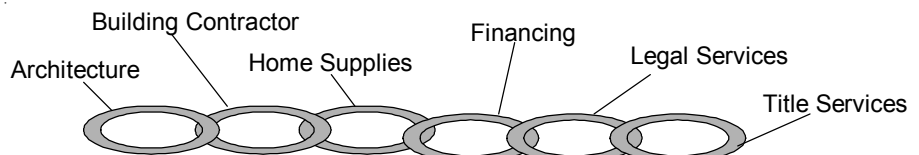


Figure 3. Value chain driven business.

It is difficult to build the distributed systems that satisfy these expanded mandates. Even with the help of powerful application servers and middleware, it is easy to build systems that do not scale. Many projects fail because applications exhibit poor performance and consume too many resources. It is also excessively labor intensive to create and maintain such systems. These pressures undermine the economic viability of the way that we develop software.

To deal with the pressures to do more and do it more efficiently, the software industry is pursuing a path that leads to industrializing the process of developing, deploying, and operating software.



## MDA: Industrializing Software Production, Deployment, and Operation

Modern industrialized manufacturing operates on a number of principles, including formal design, componentization, and the use of manufacturing patterns. It deals with labor-intensive processes by learning to automate them.

Model Driven Architecture® (MDA) brings these industrializing forces to bear on the software industry.

### Formal Design

Industrial manufacturing makes heavy use of formal blueprints and engineering models. Such practices are entrenched to the point where it would be unthinkable to construct a machine or building without such formal design artifacts.

#### Resistance

There has been a strong tendency in the software industry to view formal design as superfluous to the production process, or as something that would be nice to do but that is unrealistic given the realities of short-term time pressures. Despite the negative ramifications of the lack of formality in software engineering for the quality and longevity of software, the negative disposition of the people on the front lines of IT toward formal modeling has been tough to break.

#### Models as Production Artifacts

MDA tends to break down this resistance. It promotes the use of models that are not simply design artifacts but are actually production artifacts that drive code generators or are directly executed by virtual machines. Gradually, IT people stop looking upon MDA modeling as competing for time with production activities and start to see it as a productivity booster.

#### Standards

MDA uses standardized modeling technology based on the Unified Modeling Language (UML™) and a sister standard called the Meta Object Facility (MOF). The Object Management Group (OMG), a prominent software standards body, manages both of these standards.

MOF is actually the more fundamental MDA technology, as it makes it possible to define different languages for modeling different aspects of systems and to integrate models expressed in different languages. UML is simply the most well known and widely used of the MOF-based languages.

#### The Role of Formal Models

Models that are production artifacts have to be formal enough for computers to understand them. A model that is merely an informal sketch for the benefit of humans isn't sufficiently precise to serve as input to a generator. Thus, MDA is having the effect of increasing the use of formal blueprints for software.



Using formal models as production artifacts is actually not entirely new to the software industry. The time has long passed since we programmed basic data management manually. Instead, we create formal data models that database management systems use. Similarly, we don't program GUI dialogs predominantly by hand anymore. Rather, we create WYSIWYG (What You See is What You Get) visual models of dialogs, from which GUI development environments generate code that we enhance manually.

Models as production artifacts and code generation are thus accepted techniques for developing the front and back tiers of systems. MDA follows up on this success by bringing formal models and automation to the intermediate tiers of systems and also to B2B integration, making it possible to apply them to application development more holistically.

### Commoditization of Low-Level Programming Labor

Carr's thesis bears up well when we look at what is happening to the programming labor required in intensive doses to manually program modern IT systems. The software industry is following the industrial manufacturing script as it comes to grips with the high cost of such labor. MDA tools generate code, but even "traditional" integrated development environments (IDEs) now have "wizards" that support extensive code generation.

The movement of time-intensive programming labor offshore is also part of the industrialization trend, and parallels what has happened in other industries when dealing with high labor costs. There is a great deal of variation among the various advanced industrial societies as to the dominant attitude toward the social costs of treating labor as a commodity. In some societies, socio-political realities restrict the option to move labor offshore. Businesses in such societies deal with such restrictions by accelerating the drive to produce more with the labor that they have. In the United States the industry is moving programming labor offshore at a faster rate than in other advanced countries.

Some U.S. companies are explicitly coordinating MDA techniques with the use of offshore programming labor. Architects and designers work "onshore" to produce formal models, MDA tools generate a good deal of the code, and offshore programmers fill in gaps that generators leave. Such an approach requires a strong development process to achieve proper coordination.

### Components and Patterns

Manufacturing industries assemble finished products from reusable components. Well-tested manufacturing patterns govern the manner in which production machinery does this.

The use of component-based development and design patterns in software pre-dates MDA. However, it is now evident that requiring programmers to manually code each instance of a pattern is usually inefficient. For example, a generator



can replicate the popular Value Object<sup>2</sup> pattern almost completely from a simple model of an entity, supplemented by a small amount of additional work by a programmer (Figure 4). Some modeling approaches even allow a generator to perform 100% of the replication.

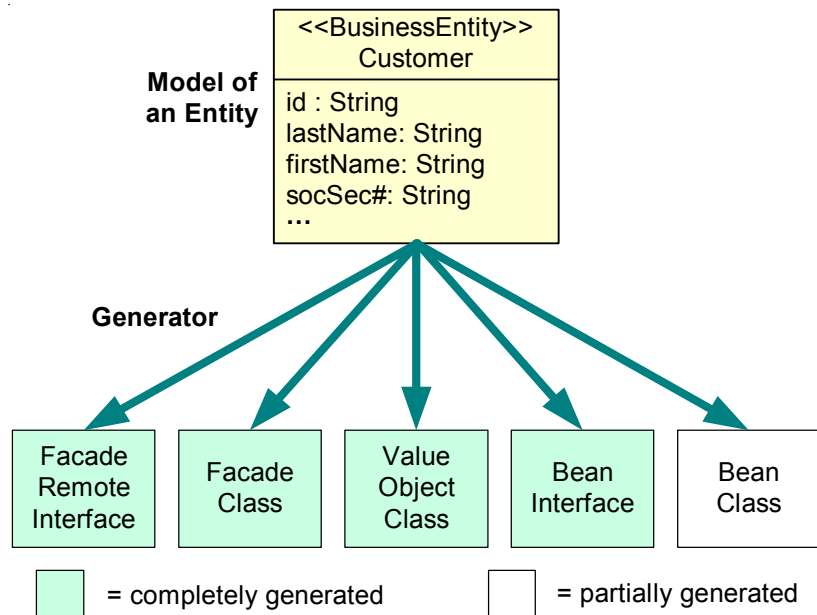


Figure 4. A generator applying the value object pattern.

The software design patterns community is thus reaching the conclusion that has long been central to industrial manufacturing, namely that automated pattern replication is a key to production efficiency.

### Raising the Level of Abstraction

Over the history of the computer industry, the level of abstraction at which we program software has risen several times. (See Figure 5.)

#### 1s and 0s

We started out writing programs in 1s and 0s, which is ultimately all that a computer understands. The tedious nature of such programming cried out for an easier way.

#### Assembly Language

Assembly language made it possible to program using mnemonics instead of 1s and 0s. For example, the instruction, MOV AX,BX, replaces a lot of 1s and 0s. MOV is an abstraction of the combination of 1s and 0s that represents the true form of an instruction to move a value.

The increase in the level of abstraction that assembly language afforded made it economical to computerize some business operations in ways that would be



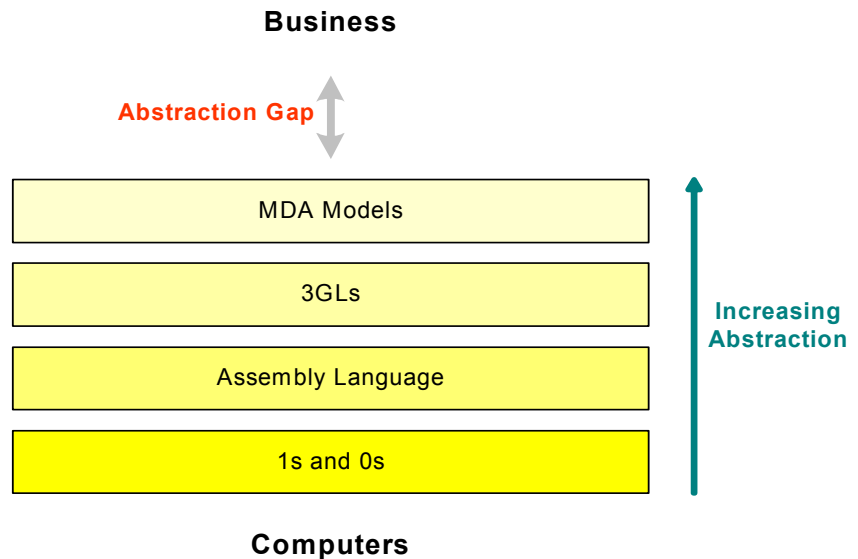


Figure 5. Increasing the level of abstraction.

impractical working directly with 1s and 0s. Over time, though, even assembly language proved to be too close to the language of the machine to support efforts to expand the reach of computerization. Assembly language instructions, although more abstract than 1s and 0s, still correspond to the native instruction set of the machine. They require the programmer to think about the architecture of the CPU and to program in terms of it. Each native instruction to the computer is very fine grained. A lot of instructions are required to do even simple things, such as calculate the extended price of a line item on an invoice.

### Third Generation Languages (3GLs)

Third generation languages (3GLs), such as COBOL, C, C++, and Java, thus came along to raise the level of abstraction further. One instruction in a 3GL might correspond to a dozen or more in assembly language. 3GLs improved productivity again, and moved the language of programming farther away from the computer and closer to the domain of the problem being solved.

3GLs made it economical to computerize more operations of the business. But today even 3GL-centric programming is too low-level to deal with the new demands on IT.

### MDA Models

MDA models are even more abstract, in that they are farther away from the computer and closer to the business point of view. It still takes a person with technical knowledge to construct them, but a small model can correspond to hundreds or even thousands of lines of 3GL code. The models bear enough resemblance to the business information and processes that they automate to make it possible to train some business analysts to read them.

MDA is thus the latest in a series of moves to raise the level of abstraction at which we develop software. Unlike previous efforts to raise the abstraction level, however, MDA seeks to continually push it higher. It is not satisfied with having narrowed the gap between the business and IT. It seeks opportunities to narrow it further.

### **Model Driven Architecture vs. Model Driven Development**

MDA includes model driven development, but is not limited to it. MDA also seeks to rationalize software deployment and runtime management (operations).

There are tools on the market that automate deployment to some extent. However, there is little if any standardization of the metadata that drives such tools. MDA's mandate includes standardizing the language for deployment modeling.

In the runtime management sphere, MDA has the potential to automate instrumentation of code, based on models of Service Level Agreements (SLAs) that define the responsibilities of a party in a value chain with respect to performance, accuracy, and other technical measures.

### **The New IT—What Carr is Missing**

The infrastructure build-out of mainframe, server, PC, and Internet technology is following the classic path of railroads and electricity. Carr's thesis is valid only up to a point, however, because qualitative technological leaps in IT are still ahead of us.

The malleable nature of digital technology—which Carr himself points out—means that whole new technologies can emerge in the form of software. A number of far-reaching technological advances in the future are likely to take place on the digital playing field. One that we can clearly see on the horizon is computer assisted business process management (CA-BPM).

### **Computer-Assisted Business Process Management (CA-BPM)**

The push to raise the level of abstraction in the way that we develop software is converging with new thinking from business process reengineering forces to produce this new technological quantum leap. CA-BPM may be digital, but that doesn't mean it is fundamentally the same technology as what IT has been over the past twenty-five years.

CA-BPM uses software tools to manage business processes directly. It closes the abstraction gap between business processes and IT systems. The main difference between the Old IT and the New IT is that the Old IT is driven by software design and the New IT is driven by business process design.

Value chain based business strategies are a big factor pushing CA-BPM forward. When several organizations combine their business processes, those processes must be more transparent than ever before. Partners must have

clear visibility into each other's processes and be able to measure whether each is living up to its part of the bargain.

#### **Origins: Shortcomings of Business Process Re-Engineering**

CA-BPM has its origins in the Business Process Reengineering (BPR) movement of the 1990s. A number of people from the BPR community learned about what worked and what did not work with BPR and recognized the value chain imperative. Specific problems with BPR that they identified include:

- Radically redesigned business processes were no more flexible, readily combinable, or visible than their predecessors.
- BPR, being largely a manual process, did not exploit the power of computerization to support designing, combining, tuning, and measuring business processes.
- BPR was too quick to discard existing resources, such as legacy computer systems and business processes.

#### **Formalized Business Process Models**

CA-BPM addresses these shortcomings. Business processes are visible in the form of formalized business process models that value chain partners use directly. Business process models are no longer informal artifacts that only humans can use. They drive business process simulation, enactment, monitoring, and fine-tuning, without having to be translated wholesale into IT-oriented languages.

Such models, expressed in a standardized business process modeling language, are a key to B2B collaboration. Note that, while we use a standardized modeling language, the models we create using such a language are themselves proprietary.

As CA-BPM brings computer assistance to business process management, and MDA employs formal modeling to raise the level of abstraction for specifying computer systems, the two trends converge at the intersection of business and IT. (See Figure 6.) The convergence thus acts to close the abstraction gap between business and IT.

#### **EAI is Old, CA-BPM Is New**

At a superficial level, CA-BPM resembles Enterprise Application Integration (EAI) tools that use process flow models to automate integration of disparate software. However, CA-BPM differs from EAI on a crucial point: The motivation for CA-BPM is the need to manage business processes. The need to integrate preexisting software is important, but is not the primary driver. CA-BPM comes from the business world, not from the IT world. It needs IT, but will drive IT, much as formal design models drive industrial production machinery in CAD/CAM environments. This is ultimately the role of the New IT.

The New IT will make it possible to hone and enact business processes in a more direct fashion than has been the case up to now. Competitive advantage will flow from the computer-assisted ability to define, enact, monitor, and fine tune business processes and value chains. This will be a technical leap as significant as the introduction of railroads, electricity, or the Old IT.



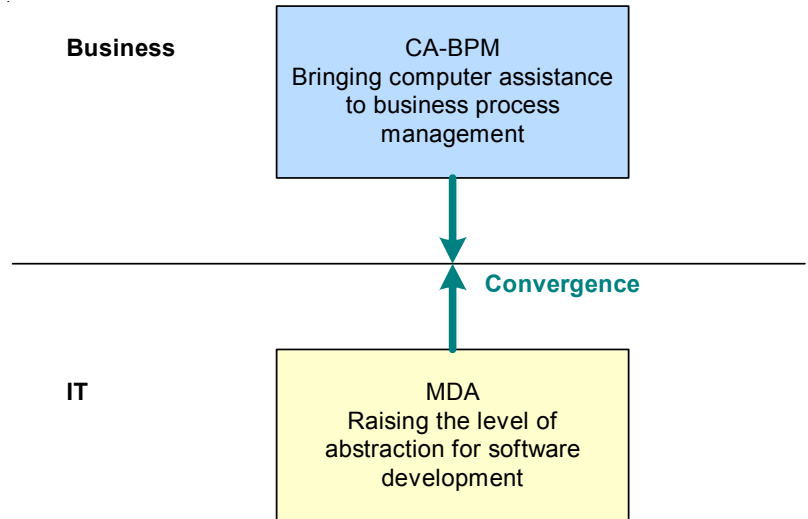


Figure 6. CA-BPM and MDA converge at the intersection of Business and IT.

### The Business Process Management System (BPMS)

Business process management systems form the technological basis of CA-BPM. In order to understand the role of a BPMS, it is useful to compare it with a database management system (DBMS).

As mentioned earlier, in the initial years of IT each application performed data management in its own way. Later, the industry moved the responsibility for data management into the database management system (DBMS). A DBMS supports data modeling, data integrity checking, multi-user access mediation, transaction management, and more. The DBMS manages data models that are reusable across multiple applications. With a DBMS we don't refine a data model into a software specification; we reuse the model as is. (See Figure 7.)

Analogously, CA-BPM moves responsibility for business process management out of applications and into the business process management system (BPMS). A BPMS performs functions analogous to those of a DBMS, and also supports the simulation, combination, enactment, monitoring, and tuning functions described earlier. The BPMS manages business process models that are reusable across multiple applications. Ideally, with a BPMS we don't refine the process model into a software specification; we reuse the model as is. (See Figure 8.) In practice, we will fall short of this ideal, as discussed further, later.

### MDA's Function in the New IT

MDA on its own, without the CA-BPM context, is an effort to rationalize software in Old IT terms. MDA is necessary, but we must see it in its proper perspective. The commoditization of the Old IT is real. A lot of the Old IT technology will remain in place, but as Old IT morphs into the New IT, business process management and design will drive it.

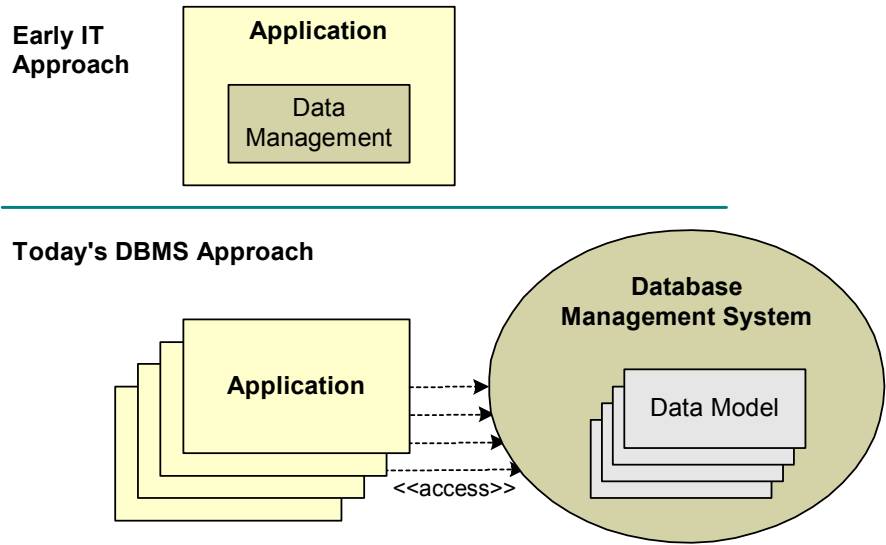


Figure 7. The transition to Database Management Systems.

**The Limits of Direct Execution**

Much of what is in a business process model will typically be executed by a virtual machine that uses adapters to execute various IT systems. However, there will still be some parts of some business processes that we can't execute directly from the business process model for practical rather than theoretical reasons. For the foreseeable future, if we try to directly execute instructions at the granularity of 1+1=2 written in the BPM language, performance is a real problem. In other words, it is not realistic to directly enact, via a BPM virtual machine, all of the fine-grained logic that appears in the model.

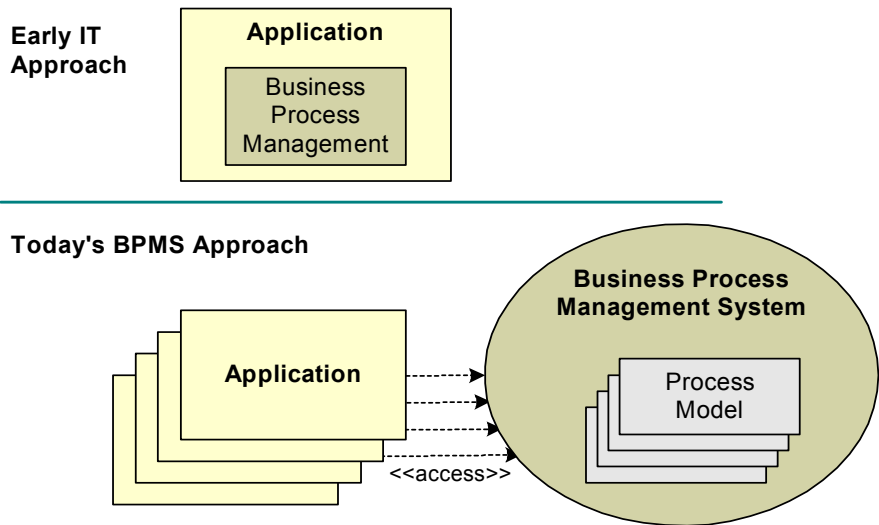


Figure 8. The transition to Business Process Management Systems.

Therefore, it will still be necessary to write special software applications and components that enact some parts of business process models. The need for this will decrease gradually, but is not about to go away completely. Software engineering will still be necessary. BPM models and MDA's abstract software models will interact in this context. The more that MDA successfully raises the level of abstraction for software specification, the easier this interaction will be.

#### **A Supportive Role**

The MDA community must not lose sight of an important fact about the software that BPM virtual machine performance constraints compel us to produce: Such software will run in an environment where basic business process management functionality has been partitioned out of the application and into the BPMS. One of the greatest risks of lack of coordination between MDA and BPM is that MDA efforts might unwittingly undermine this partitioning, by generating code that takes basic aspects of business process management into its own hands.

#### **Specialized Adapters**

Software engineering will also still be necessary to generate specialized adapters that enable legacy systems to enact business processes. CA-BPM tools come with adapters to popular systems, but adapters that tie to proprietary in-house systems will need to be generated.

### **A Cautionary Note**

Paralleling what happened with earlier technological breakthroughs, software industrialization's penetration of industry will be gradual. It is not possible for an organization to adopt this technology wholesale in a short period of time. Significant organizational and cultural changes accompany the technical changes that MDA and CA-BPM bring. Furthermore, the tools and standards supporting these technologies are still maturing. There will be no big bang, just steady progress.

### **Logical Conclusion**

The convergence with CA-BPM is the logical conclusion of MDA, much as the constant push to rationalize manufacturing led to the direct production of machinery from models via CAD/CAM. Just as the manufacturing process became design-driven, the New IT will be driven by business process design.

The complexities and vast potential of value chain business are the economic motivators pushing both CA-BPM and MDA. Companies that fail to see the limits of Carr's thesis will eventually find themselves behind the curve. Their business processes will be incapable of being effective links in value chains, and they will be marginalized. On the other hand, there are companies that "get it." They understand that they have only begun to digitize, and are investing heavily in the New IT.

The MDA community needs to "get it" as well, lest it inadvertently work at cross-purposes with CA-BPM. This is the main reason that the CA-BPM community cannot afford to discount MDA; the lines of communication need to stay open.



### Credits

Portions of this paper are adapted from my book *Model Driven Architecture: Applying MDA to Enterprise Computing*, John Wiley & Sons, 2003.

I draw quite heavily on Howard Smith and Peter Fingar, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, 2003. I have also benefited from some conversations with Peter Fingar about Carr's *Harvard Business Review* article.

Howard Smith and Peter Fingar have just released a new book called *IT Doesn't Matter—Business Processes Do*, which critiques the Carr article. I found their Web site for the new book, at [www.bpm3.com/hbr](http://www.bpm3.com/hbr), to be a very helpful source.

I credit Howard Smith and Peter Fingar for the phrase “at the intersection of business and IT,” although they do not use it in a context that includes MDA. Similarly, I claim no credit for the BPM acronym, and credit Howard Smith and Peter Fingar for the insight that, in the new wave, BPM is computer assisted in a manner that is analogous to CAD/CAM. However, I did formulate the extended acronym *CA-BPM* in order to emphasize the parallel to CAD/CAM.

MDA and “Model Driven Architecture” are registered trademarks of the Object Management Group. “Unified Modeling Language,” “UML,” and “MOF” are trademarks of the Object Management Group.

### Footnotes

<sup>1</sup> Nicholas G. Carr, “IT Doesn't Matter,” *Harvard Business Review*, May, 2003.

<sup>2</sup> [http://java.sun.com/blueprints/patterns/j2ee\\_patterns/value\\_object/index.html](http://java.sun.com/blueprints/patterns/j2ee_patterns/value_object/index.html)

### Author

**David Frankel** is the author of the book *Model Driven Architecture: Applying MDA to Enterprise Computing*, published by John Wiley & Sons, and has his own consulting company, David Frankel Consulting: [www.DavidFrankelConsulting.com](http://www.DavidFrankelConsulting.com). Mr. Frankel served several terms on the OMG Architecture Board, and has been in the software business for 25 years. You can send him comments and questions at [df@DavidFrankelConsulting.com](mailto:df@DavidFrankelConsulting.com).