CHAPTER THREE

Why Aren't My Systems Smart Enough Already?

The cost of information technology in today's organizations is substantial and is growing as a percentage of total operating cost—but today's systems still aren't smart enough. Why this is the case can only be understood by taking a historical perspective on business computing and how information technology (IT) developed as a discipline. In the past, IT departments were thought leaders in their organizations, setting the pace for the application of technology. Today, the needs and sophistication of the people who work in organizations are colored by an outside influence—the Internet—and IT organizations struggle to keep pace, burdened by accelerating demands and the drag of maintaining legacy systems. The tools, technology, and techniques for enterprise decision management (EDM) are ready, but impediments are rooted in the history of business computing and the resultant IT cultures. To understand why, after years of spending on information technology, so many of your systems aren't already smart enough, you need to take a walk back through the evolution of business computing in practice. With that in mind, this chapter is a brief look at the history of a typical IT infrastructure. It's not a general-purpose history but a brief outline of how we ended up in this mess.

How Did We Get Here?

A history of business computing would require volumes. However, a brief overview of the evolution of information technology (IT) in business decision making is useful in understanding why certain aspects of information management are still misaligned. Certain approaches that are taken for granted today are based on situations that disappeared long ago; certain disciplines that are considered unrelated are actually quite similar and synergistic. Some approaches aren't being used in a comprehensive enough way. Only through

understanding the basis of these inefficiencies and, often, dysfunctions can you construct a rational remedy.

Computing in business is roughly 60 years old, although the deployment of computers for processing business information didn't start to catch on until the late 1950s. There were many computer manufacturers at that time, and each one developed its own proprietary operating systems and application software, although software was usually limited to a language compiler or two. Some manufacturers developed a reputation for general-purpose computing based on their programming languages (Assembler originally, but expanding to higher-level languages such as COBOL and FORTRAN); others came to specialize in certain applications, such as back-office work for banks (Burroughs, for example). At the time, general-purpose computing was what was needed for computing to take hold across industries.

Because general-purpose computing took hold, subsequent development of the technology proceeded rapidly, which created a constant lag between optimistic expectations and delayed delivery of useful output. Applying technology upgrades and innovations took considerably longer than producing them, a phenomenon that has only gotten worse over time. In the early days, the delivery of a new model meant reprogramming everything, because there was no upward compatibility. Today, that amount of effort seems almost ridiculous, yet according to a survey[1] by Forrester Research, 75 percent of the IT budget is spent on maintenance of existing software and infrastructure. Organizations' ability to absorb new technology and put it to work is just as constrained today as it was at the dawn of computing. Unlike many other industries, the rate of change has neither slowed nor been adapted to.

People responsible for computers have been a unique breed in business. Their skills with abstract concepts or mysterious codes are in stark contrast to the formal, businesslike demeanor of accountants, salespeople, and executives. From the beginning, they were a separate group, in a career that had little or no trajectory into the "real" parts of business. This chasm between IT and the rest of the organization exists to this day and is still a major cause of dissonance between IT efforts and business requirements. In fact, in the past, IT managers had all they could do to manage IT software, hardware, and staff. They had no time to worry about the business itself, which was left to business professionals. Today, the stereotypes of the computer wizard in the basement with the pocket protector and the florid, cigar-smoking CEO are cartoonish and dated, but in reality, the gap is still severe.

---

[1] Phil Murphy, "APM Tools Will Reach $500 Million to $700 Million by 2008." Forrester Research, July 22, 2005.

> **Note** *There was no academic path for computer specialists, either. The first PhD in computer science was awarded in 1965. Today's colleges and universities are addressing this gap between IT and the rest of an organization, leading to a new kind of IT professional and offering new majors that are combinations of marketing and IT or finance and IT.*

In general, the issues discussed in this chapter are related to problems with data, problems with programs, and problems with people. Data first.

Problems with Data

For a long time, computers were used strictly as calculating machines. Data was fed into them manually from punch cards, a holdover from tabulating machinery. The computer crunched the numbers and generated, with limited flexibility, reams of reports on the familiar "greenbar." Data in computers was typically lost because there was nowhere to store it. Storage wasn't considered important anyway, as punch cards and paper reports, usually retyped into forms with a typewriter, were considered the official system of record. The computer and its programs and data were merely tools to speed up some work.

Both greenbar and official reports were stored for recordkeeping. This is how it had always been done, and the appearance of a "big calculator" didn't change anything right away. In fact, many firms, well into the 1990s, kept paper records long after the appearance of magnetic tape and later nonsequential storage devices, such as magnetic disk drives. Perhaps the tradition of keeping important records, such as legal documents, on paper contributed to this habit. It wasn't until total quality management (TQM) and business process reengineering (BPR) took hold that the inefficiencies of large organizations were exposed. Unfortunately, the attitude that computing was just a calculator and not to be trusted for real work would take much longer to break down than it took to get started.

The Reporting Gap

With the introduction of permanent storage devices, however, computers began to take on a role as archival devices as well as number crunchers and report generators. Although paper was king at the time, the economy of microfiche was easily understood and accepted because it didn't alter the concept of computers outputting information in a format understood by businesspeople. Microfiche just compressed information and made it more durable. However, magnetic storage devices, especially tape, ushered in a completely new way of thinking about computers in business. Tapes stored everything—data, report

streams, and programs. The tapes even substituted for core memory, allowing programmers the luxury of writing programs that used more resources than were actually available. Computers gradually became bona fide business tools.

Computers gained speed and flexibility through the advancements of disk drives, solid-state memory, and virtual memory operating systems. Eventually, interfaces or "transaction monitors," such as IBM's CICS, opened a new discipline: online systems. Instead of batch machines that ran serial scheduled jobs, transaction monitors allowed programmers to create applications for real-time, instantaneous processing of multiple processes (although most small "transactions" were stored for later batch processing). Soon, transaction processing was born, the polar opposite of batch computing. Online transaction processing (OLTP) emerged as the next big thing and never really faded. The bulk of corporate computing, even as it moves through generations of technology, is still transaction-oriented.

The advent of formal data methodologies and software (Chen, Codd, Computer-Aided Software Engineering [CASE] tools, and relational databases) also led to better appreciation and use of the information stored in computing technology. So we moved from computers for processing to computers for information. One thing was missing from these transactions, however—decision making. All the "brainpower" in transactions came from people. The computer's role was to capture and store the data needed to record the transaction. If a judgment had to be made, a person made it. If the transaction had to be approved, the system waited for someone to approve it. These transactions were increasingly fast and essential to business operations, but they remained unintelligent.

Additionally, the more transaction-oriented corporate computing became, the more alienated it became from its original purpose: reporting. As a result, a new industry emerged in the late 1970s that was designed to address this gap. At the time, this industry was called decision support systems (DSSs) or sometimes fourth-generation languages (4GLs), but its goal was to enable users, mostly businesspeople, to create models and generate reports without the IT department's involvement. This industry later became known as business intelligence, but almost all the early providers of these products fell by the wayside before then.

Mainstream systems evolved from batch to OLTP to client/server and finally, by 2000, to Y2K/enterprise applications, as shown in Figure 3.1. Business intelligence and analytics developed separately, supported by functional areas, particularly finance and marketing. As computing in general became more advanced and complicated, it ended up back in IT. It was called "data warehousing" and "business intelligence," but it was still a separate discipline. The original motivation for this separation was mostly IT's neglect of reporting and analysis, but as the need for and volume of business intelligence grew, IT took control to protect its existing resources. The gap has continued to grow and is, with today's technology, largely artificial and unnecessary.

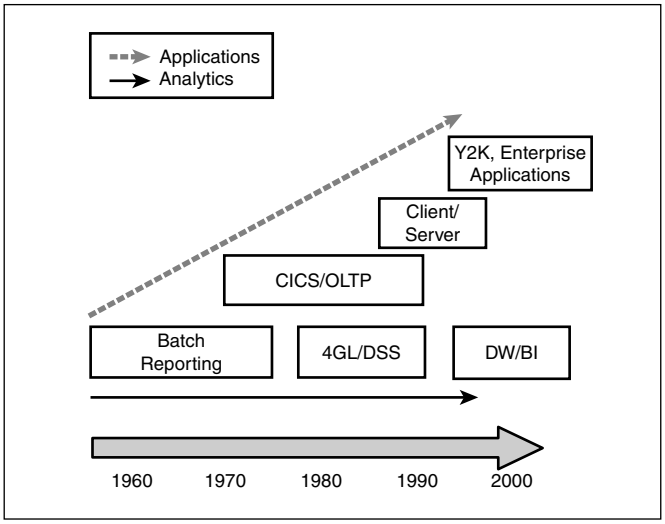

Figure 3.1 The divergence of applications and analytics as computing evolved

Many of the early DSS tools were quite ingenious, with features that current analytical tools would envy, such as nonprocedural modeling, multidimensional orientation, goal seeking, statistical and stochastic routines, graphics, and tools for development and collaboration. Their weakness was data. Without the data in transactional systems, these tools were limited to whatever data businesspeople could generate on their own, such as budget, planning, and forecasting data. In many cases, getting existing data required reentering data from printed reports, a time-consuming and tedious process. Nevertheless, these tools flourished, especially in planning and budgeting roles where the need for external data was low.

The IT department wasn't supportive of these tools. There was a constant refrain that they would "bring the mainframe to its knees," but in practice that rarely happened. What did happen was that for the first time, "power users" emerged—non-IT people who could actually develop applications and generate reports from computers, a privileged group that endures to this day. In fact, there are so many of them, and their output is so important, that this group has a name of its own: shadow IT.[2]

Early DSS vendors met their Waterloo in the early 1980s when a small product was launched: Lotus 1-2-3. It turned out that a PC with a Lotus 1-2-3 spreadsheet was capable of doing much, if not all, of the functions of mainframe- or mini-based DSS tools. Connectivity was actually worse, because PCs weren't connected to the mainframe, but the freedom and novelty was so great that, within a few years, most DSS vendors disappeared.

[2] "Shining the Light on Shadow Staff: Booz Allen Hamilton," *CIO*, January 2, 2004.

A bigger drain on IT was the constant flood of requests for data extracted from operational systems, both decision support systems and PCs. Many alternatives were tried:

- Formal programming requests with highly specific requirements
- Generic abstracts to flat files that could be shared by more than one request
- Nonprocedural programming languages, such as A Programming Language (APL), that allowed a less programmer-centered approach
- Ad hoc query tools, such as Query Management Facility (QMF), so that users could create their own abstracts

In the end, none of these approaches worked well because, for the most part, they couldn't be maintained. The number of extracts expanded geometrically, as shown in simplified form in Figure 3.2, and changes in programs, requirements, and tools overwhelmed IT's ability to respond to requests. There were no standards or best practices for non-IT people to develop libraries of extraction and importation routines, so the "self-service" aspect suffered, too. The sheer number of extracts was also a burden on often heavily used mainframe and minicomputer hosts.

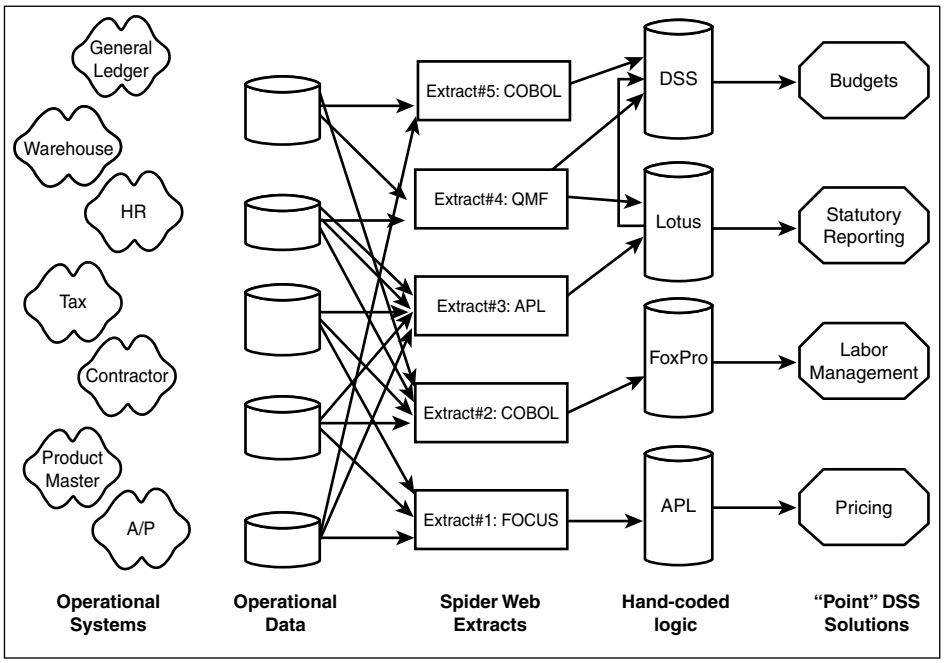

Figure 3.2 Simplified representation of extracts from various systems for diverse reporting needs

Enter the Data Warehouse

Originally, the data warehouse (or information warehouse, as IBM called it) was designed to solve a problem for IT departments, not end users. The goal was to build a database to provide in-place access to physically distributed databases and serve as a repository for satisfying report requests from end users (because individual silo databases couldn't do so), not to support self-service or direct access. This key point explains why even current data warehouse approaches often use a layered, after-the-fact batch architecture with a substantial investment in an enterprise data warehouse that knowledge workers never, or rarely, query directly.

The term "data warehouse" has an uncertain provenance. The most influential person who shaped the concept's development is probably Bill Inmon. For more than 20 years, he has dominated as a thought leader, with dozens of books and countless appearances and publications. However, the practice of data warehousing has surged ahead on many fronts, with broad adoption of Inmon's original work, Ralph Kimball's architecture and methodology, and the contributions of many others. Data warehousing has become a well-established and vibrant strand in the IT mainstream.

However, in the early to mid-1980s, a typical organization owned an IBM mainframe that struggled to perform all scheduled jobs. The computer often simply ran out of computing cycles to do all the work, necessitating an upgrade to a newer, more powerful machine. This solution was expensive and temporary, because new requests were already lined up to absorb all the new computing cycles. This management-from-scarcity approach resulted in a sort of triage process in which requests for new applications were evaluated. Most were put into a queue, the so-called backlog, where they languished sometimes indefinitely. This backlog was largely made up of business departments' requests for reports, because preference was given to operational programs.

The original concept of data warehousing was quite simple: Buy another mainframe, install a more "friendly" operating system (such as IBM's VM/CMS), and provide a set of utilities to build a relational database of data culled from various operational systems. This solution would relieve the other mainframe's load of numerous and overlapping extracts and allow IT to service users' reporting needs with tools of its choosing. How reporting needs were met took different forms. At the simplest level, the new machine simply took copies of data from various systems, in whole or in part. From these files or databases, programmers could draw their own extracts for reporting programs without affecting the "main" mainframe's performance.

This method is very different from the current concept of data warehousing. For one thing, there were no business users in the data warehouse; it was designed for IT. Business units, with or without support from IT, developed their own reporting and analysis solutions to compensate for the weak reporting operational systems offered. None of these solutions, however, was completely effective, and all were expensive and tedious to build and maintain. Eventually, power users of DSS tools and PCs got access to the early data warehouse, and it became apparent that the current arrangement wasn't satisfactory. Combining information from multiple extracts, the data simply did not line up. Whether the problem was caused by coding errors, sloppy data entry, or timing errors, merging information across the various disconnected source systems or "stovepipes" was impossible. Solving this problem was the beginning of data warehousing as it is known today: the integration of disparate data into a common logical model. In time, data warehousing became a structured solution to address these intractable reporting and analysis problems:

- **Performance**—Systems designed for transaction processing performance, especially in an era of expensive, proprietary hardware, were unable to tolerate the added load of query and report processing.
- **Data quality**—OLTP system data integrity was limited to the application's needs, so integrating data across systems was difficult to impossible.
- **Access**—Initially, connectivity issues were paramount, but security, licensing, and organizational boundary disputes also restricted access to information.
- **Stewardship of historical data**—OLTP systems didn't maintain the historical analysis needed for variance reporting, trend analysis, and even statutory and regulatory requirements.

In the 20 or so years since this concept emerged, the data warehouse has taken on far more responsibility, some of which it still hasn't met completely. For instance, many see the "enterprise" data warehouse as the "single version of the truth," a comprehensive repository of the one true set of data and semantics that everyone can get behind. Having absorbed the role of business intelligence, it's also the architecture for all forms of end-user reporting, query, and analysis. It's the plug that fills the gap between operational and analytical processing, a gap that has been steadily widening since the advent of online processing. As Figure 3.3 shows, a data warehouse is the hub for spokes that lead to operational systems, analysts, customers, performance management, and more.

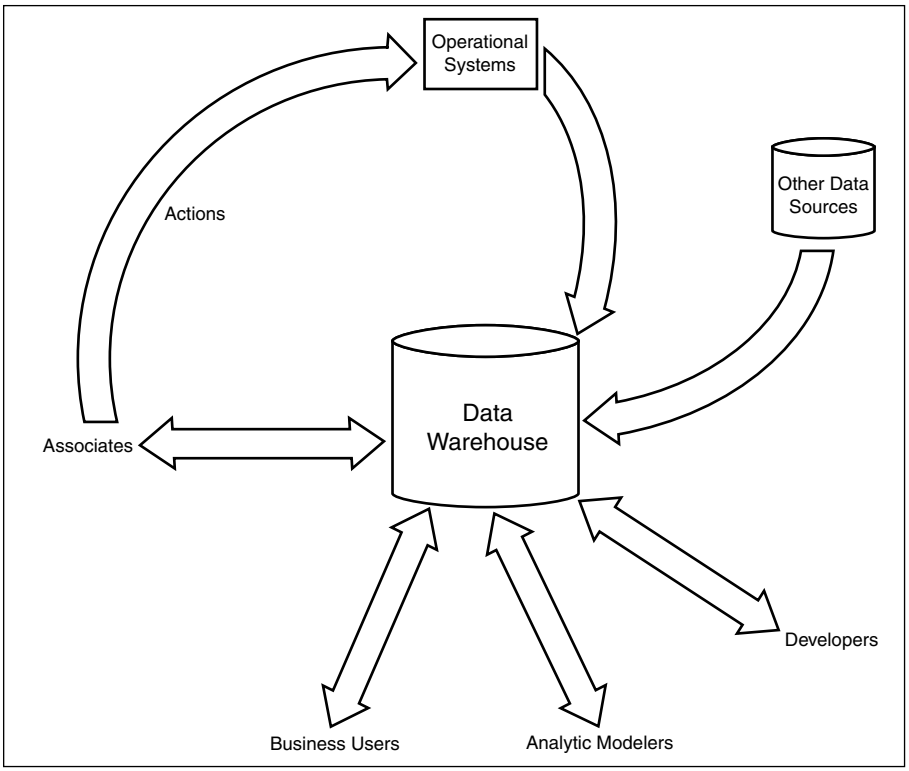

Figure 3.3 A modern data warehouse plays a more central role in information systems (teradata)

A functioning data warehouse is a collection of data, models, metadata, processes, practices, standards, and services. At the most fundamental level, a data warehouse gathers data from other systems and transforms it into a merged model of sorts. This process can range from preserving almost all the original data at the most detailed level to compressing and summarizing at a much higher level of aggregation or, in most cases, both. What's most important is finding a way to preserve as much of the data's original intent as possible yet fit it into a single schema, at least initially. This process often involves "cleaning" the data because data in various source systems might be of poor quality, especially historical data. Another unique characteristic of data warehouses is the extent of historical data they retain. This data provides the basis for tasks such as trend analysis, statutory and regulatory reporting, and variance reporting.

Keeping historical data presents a new set of challenges, however. In today's fast-changing world, shoehorning three-year-old data into current data models can be exceedingly difficult. Data warehousing has developed some ingenious methods for pulling off this process, but keep in mind that these innovations were added to the original concept.

Data warehouses have also prolonged and extended the gap between operational and analytical processes. Because data warehouses and the business intelligence tools used with them were separate disciplines from other enterprise applications, even the small amount of reporting and analysis that occurred in operational systems gradually migrated to data warehouse/business intelligence systems.

The data warehouse was a simple idea, but in practice, its implementation caused, and still is causing, problems. Extracting data from source systems and transforming it into an integrated whole is a monumental effort. The latency caused by the load and refresh cycle devalues data. The scale of data warehouses grew exponentially, leading to spiraling costs and performance problems. The industry divided over polarizing issues of database design, architecture, and methodology. Most of all, the data warehouse's purpose changed dramatically from a tool for programmers to an analytical environment for knowledge workers. Enter business intelligence.

Business Intelligence

The terms "data warehousing" and "business intelligence" are like binary star systems: They orbit each other, but from a distance, they appear to be a single entity. Because the subjects of data warehousing and business intelligence are somewhat aligned with EDM, understanding the differences is important, as is knowing how each evolved, what problems they were designed to solve, and how well they performed.

In its purest form, business intelligence is supposed to be part of decision making in an organization. Phrases such as "better data for improved decision making," "getting the right information to the right person at the right time," and "the single version of the truth" are common. The focus is on informing people with data, who then go about making decisions. Business intelligence is designed to inform people and provide a way for them for evaluate information in retrospect. A branch of business intelligence now called performance management also has provisions for forecasting and planning, but in practice, most of performance management is still retrospective analysis and reporting.

Some people refer to the process of people getting and using information as business intelligence, but the term was coined when many providers, concepts, and methods for today's business intelligence didn't exist. Some fuzziness in the definition of business intelligence is unavoidable. In fact, the term "business intelligence" has more or less subsumed the practice of data warehousing and has become synonymous with a range of functionality that business intelligence vendors offer.

Of all the types of business intelligence, reporting, ad hoc query, online analytical processing (OLAP), dashboards, scorecards, visualization, and even data mining, one common thread is delivering information to people. With transaction systems still relying on

human intelligence for decision making and judgment, business intelligence has sold itself to a new group of users by promising to improve decisions. If people must make a decision because the system can't, at least they should make one informed by all the facts or data in the data warehouse. Even business activity monitoring (BAM), which catches data on the fly and applies some simple rules to process data or let it go, ultimately alerts people through visual devices. Some BAM products are capable of bypassing people in the loop and alerting applications, services, or processes directly, but this feature is unusual. Most implementations update digital dashboards for people to view and navigate.

Operational Business Intelligence

As the gap between operational and analytical processes closes, thinking about the role of business intelligence in this new landscape is natural. Latency in business intelligence is an issue because data warehouses are updated monthly, weekly, and often daily, but rarely within a business day. Operational reporting or monitoring that requires fresher data is a problem. The concept of a data warehouse is based on layers of schema that are updated in batch offline and then conformed, indexed, preaggregated, and otherwise tuned and dispatched. Trickling in new data in real time disrupts this model.

Another problem is that after data is culled from its source and integrated into the data warehouse, the connection to its source can be lost. Although vendors have invested heavily in maintaining these links, persistent connectivity problems make data warehouses a poor candidate for single-source reporting. The solution was a sort of kludge called the operational data store. It violated many data warehouse principles, even though it was still considered part of the data warehouse. In time, operational data stores were also integrated, so they lost their capability to tie back to the original systems. Because of the transformation that happens with integration, they inherited much of the same latency as data warehouses.

The need for more timely analysis of data is strong and growing, however. The business intelligence industry is working toward solutions of operational business intelligence, but that definition is typically limited to daily or "intraday" data.[3] So for the time being, operational business intelligence is characterized as having the same output (to people) and using the same data model (the data warehouse) but taking place a little more often than daily. True real-time data monitoring has been scaled back in business intelligence to a watery concept called "right time," which means as fast as the data warehouse can handle it. Some vendors can trickle data into a data warehouse, but the answer to operational business intelligence clearly lies in the ability to get closer to business

[3] Colin White, "The Next Generation of Business Intelligence: Operational Business Intelligence." *DMReview*, May 2005.

processes as composite applications or from message queues. In any case, the lingering shadow of the data warehouse plus people-based delivery of business intelligence will keep business intelligence from participating in making your systems smarter, but it can be a valuable contributor.

Data Mining, Predictive Reporting, and Operations Research

To improve the value of reporting and analysis tools for knowledge workers, some organizations have adopted data-mining technologies. Data mining involves using powerful mathematical techniques to analyze huge volumes of data, as from a data warehouse, and extract meaningful insight. This insight might mean finding out how two products are related in their sales patterns or what customer characteristics make them susceptible to an offer. Association rules, likelihood scores, and trending are possible outcomes of these insights.

Although much data mining is aimed at understanding historical data, some is focused on predictive analysis. What customer actions predict whether they will cancel a contract? What kinds of customers are likely to respond well to a store redesign? Which suppliers will have credit issues in the future? This kind of predictive reporting can increase the value of analysis by giving knowledge workers a sense of data's future implications. Generally, it doesn't prescribe actions or lend itself to causing change in systems. Most organizations have applied it only to making reporting and analysis more sophisticated. Predictive reporting improves the value of your data but doesn't make your systems smarter.

Operations research grew out of work in the United Kingdom during World War II and has been applied to business problems more over the years. Using a mix of statistical and mathematical techniques to find solutions to problems, operations research is more focused on finding the right action to take, given current constraints and historical data. Using mathematical techniques and optimization and simulation tools, it allows a modeler to set up a problem and then "solve" it to find the best outcomes, given existing constraints. Although some models can be embedded into systems to make them smarter, the range of solutions for which it's sufficient is limited. Therefore, operations research has focused on making people understand possible solutions instead. Many problems have been targeted and solved with operations research, but it hasn't generally been used in programs that run organizations day to day.

Although data mining and operations research have roles to play in solving these problems, problems with data aren't the only ones organizations have. You need to consider problems with programs as well, as discussed in the following section.

Problems with Programs

The gap between analytics and operational or transactional systems explored in the previous section explains why organizations have such a hard time deriving insight from data and applying that insight to their operational systems. A second major flaw with current approaches is problems with programs. Fundamentally, the programs used to run most organizations today are built without any embedded intelligence. The "peopleware" of an organization, not the software, is what makes decisions.

The inherent inefficiency of these programs has long been obvious to those working with information systems. Attempts have been made to address this limitation. All these attempts, from handwritten code to failed attempts at artificial intelligence, from enterprise applications to business process management and service-oriented architecture, have made little difference. Most organizations have software programs that just aren't smart enough.

The Weight of Legacy Code

As more custom code is written, organizations find that it lives longer than they intended. Code that's 20 or even 30 years old is still in use in many organizations—still running core business processes and processing vital transactions. Companies have discovered the hard way that they can't code their way into the future. This deadweight of legacy applications has created two problems.

Part of the problem comes from a mind-set that systems, like other enterprise assets, should be built to last. This focus results in detailed but largely static requirements and huge investments in system architecture and design. However, it also buries critical code in complex systems. To make applications robust and complete, a huge amount of business expertise must be embedded in the system, but there are problems with this approach:

- Embedding business expertise in the system is hard because those who understand the business can't code, and those who understand the code don't run the business. Business users can't explain to programmers what they need, and the result is systems that don't quite work the way they were intended.
- Generally, custom code isn't well documented, or the documentation is allowed to get out of date rapidly. The promises of "self-documenting" code notwithstanding, new generations of programmers struggle to amend code to face new challenges.

- Showing a regulator or auditor how custom code works is nearly impossible, and demonstrating compliance with policies or regulations is extremely difficult, costly, and time consuming.
- New challenges and requirements emerge constantly because the world doesn't stop changing, and organizations can't afford to stand still. Therefore, changing and managing custom code is difficult.

All these factors contribute to another aspect of the problem: the maintenance backlog. The maintenance backlog is the list of projects not progressing because of a lack of time and resources or other projects having higher priorities. Most projects don't even make it to the backlog unless they have a positive potential—that is, the project's business value exceeds its cost. An organization that could magically complete all projects in its backlog would add tremendous value to the business. For most organizations, this backlog represents a sink of resources and time that could add significant value.

Organizations have a maintenance backlog for many reasons, but one of the most persistent is that a huge percentage of IT resources is spent on systems maintenance—75 percent or more, as noted at the beginning of the chapter. So much old code is used to run businesses and must be constantly updated (to reflect new regulations, competitors, and products) that this work dominates the IT department's responsibilities. The systems were originally built to specification but no longer do what the business needs. Perhaps the specification was wrong, or perhaps the business has changed. Maintenance takes so long and uses so many resources that little or nothing else can be done.

Even if maintenance work isn't consuming a large percentage of your IT resources, traditional approaches to embedding logic in systems create rigid and unwieldy systems. This lack of agility causes problems if you need to respond quickly and cost-effectively to a competitive issue, new regulations, a new channel, or another major change. Coding business logic into legacy systems perpetuates the separation between those who know the business and those who run the systems and makes it hard to update systems as business needs change. Expert systems and 4GLs were two programming approaches intended to address these issues.

Artificial Intelligence and Expert Systems

In the 1980s, the IT industry and organizations made a major investment in various forms of artificial intelligence (AI), which was designed to bring the power of human intelligence to computers. Expert systems vendors promised that their software would perform tasks just like the company's most experienced employees. These vendors used intelligence and best practices painstakingly collected from industry experts to power their systems, but most expert systems didn't succeed in practical application.

Expert systems were typically designed as "closed systems," designed to solve a problem on their own. They didn't support or integrate with the programming models prevalent in OLTP systems. Specialized hardware and software were required. At the time, most corporate computer systems were written in COBOL and ran on IBM mainframes, but many expert systems packages required high-end workstations using artificial intelligence languages, such as Lisp or PROLOG. Organizations couldn't easily integrate these systems into their production environments and had no personnel trained in maintenance or programming techniques for them. They had to rely on special training and support from software vendors. Generally, expert systems came with predefined rules for accomplishing specific tasks. Specialist programmers at vendors, working from interpretations of interviews with industry experts, crafted these rules as compromises between the different methods their sources used.

Organizations purchasing expert systems software usually needed to go through laborious tuning sessions to understand how the rules functioned and to modify them for their business preferences. Organizations found it impossible to use expert systems to automate other tasks because they couldn't modify their underlying processing flow and structure. In the end, the organizations that experimented with expert systems became wary of computer software promising "intelligent processing."

In addition, AI software consumed huge amounts of computing resources at a time when these resources were still at a premium. Other factors prevailed as well, especially rampant paranoia that Japan's burgeoning economy would overwhelm the United States and that Japan's government-funded "fifth-generation computing" initiative, largely about developing AI, would pose a bigger threat to the U.S. economy than all the Toyotas it could produce. This mentality created a rush among small AI entrepreneurs to market immature and nonperforming products. Ultimately, the Japanese fifth-generation initiative failed to produce much more than factory (and toy) robots, and many AI vendors returned to the university labs and defense contracts from whence they came. To be fair, some firms survived and, having learned a painful lesson, transformed their products into more commercially useful offerings, such as business rules management systems, data-mining tools, logistics optimization software, and embedded intelligence, such as fraud detection.

Ultimately, even though the outcome was less than desirable, everyone learned from the experience. Lessons included the value of keeping knowledge (rules) in a repository where it could be managed and the power of a declarative approach, compared to procedural programming, to simplify some problems. Expectations for AI were rolled back to more reasonable levels, the buying community became more careful about the next big trend, and AI entrepreneurs learned that they have to embed their inventions in useful applications as well as cooperative architectures. In the meantime, the power and relative

cost of processing is dramatically more favorable, and open standards and ubiquitous communication provided the basis for AI-like technologies to have a second chance.

4GLs and Other User-Friendly Tools

If business know-how is hard to embed into information systems, even with an expert system, can you make maintaining the code easier? To bring a higher level of abstraction to programming problems, 4GLs were developed. In theory, abstraction makes it easier to see what's happening in code, engages businesspeople in the process more effectively, and makes IT development and modification of sophisticated processes easier and quicker. Many 4GLs came and went, and not much changed in the problems of application maintenance.

Although 4GLs do offer some productivity gains and many are easy for less technical staff to use, they fail to address the core "build to last" problems. Code representing core business logic is still procedural and embedded in code that's "plumbing" or otherwise highly technical. Few business users become fluent enough to write code themselves, and those who do often create programs and scripts that aren't managed or controlled well enough for a company to rely on.

Stretching the metaphor, you could consider a spreadsheet a 4GL (at least its scripting or macro creation capabilities). The problem of maintaining spreadsheet applications is well documented and pervasive. Names such as shadow IT, spreadmarts, and spreadsheet hell are used. The problem is that spreadsheets are helpful tools for composing a piece of analysis, but as shared applications, they are a disaster. They lack version control, a repository, and collaboration features. They are consistently applied to problems for which they were never intended, which is their biggest weakness—in fact, the biggest weakness of almost any piece of technology.

Business Rules

The use of business rules as a way to specify how an organization behaves began to gain ground in the mid-1990s and was popularized by Ron Ross[4] and Barbara von Halle,[5] among others. Many early adopters regarded business rules primarily as a tool for describing and understanding how an organization wants to act. In one sense, business rules were a way to design an organization. As technology support for this approach increased, business rules

[4] Ronald G. Ross. *Principles of the Business Rule Approach*. Addison-Wesley Professional, 2003.

[5] Barbara von Halle. *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. Wiley, 2001.

began to separate into a business-design approach and a higher-level, more declarative way to develop code. Declarative approaches allow managing each piece of code or logic separately instead of in the procedural sequence typical of regular code.

Another major impetus for the business rules approach was to bring the idea of business rule management to problems that didn't involve extremely complex decisions (such as diagnosing cancer). Instead, this approach was used with everyday operational transactions that occur in high volume and involve decisions with low to medium complexity. Instead of using an expert system to handle very complex problems, organizations could use business rules to automate 80 percent of less complex cases and, therefore, improve throughput. Additionally, this approach enabled nontechnical businesspeople to state business rules rather than provide fuzzy requirements that IT translates into the system's real logic—sometimes without enough business input and often with a lot of confusion.

Despite the early and repeated proof of the effectiveness of business rules, their use has been somewhat limited. Most organizations using this approach at a strategic level do so only to describe and understand their business. Organizations using it to make specific systems smarter often do so only in a localized way. The rules are extracted from limited sources and largely ignore the insight that can be gained from an organization's data. Without an overall approach, few organizations have become proactive at finding decisions and automating them with business rules, although this approach is clearly possible. Business rules, as you'll see, are a critical component of a solution to the problem of making systems smart enough, but their potential has been untapped so far.

Buying Solutions

In the past, IT departments had no alternative but to build their own software. As the industry matured, however, more packaged applications became available that offered quicker time to market and less need for specialist programmers. In theory, you could buy a package for inventory control, for example, and install it and be up and running quickly.

In fact, many packages were overly rigid, based on a single interpretation of how a certain business process might run. Configuring and installing these packages, especially as they grew into today's enterprise applications, were time consuming and costly. In addition, enterprise applications still assumed that people were the motivating force behind systems. They had a data model, captured data through various generations of user interface technology, and stored it in an operational database. They provided reporting or integrated with business intelligence/data warehouse products so that data could be transferred to an analytic environment and used to help people make decisions. Attempts to customize or extend these applications resulted in custom code with all the problems described previously. IT departments still spent a lot of money and time on

maintenance, organizations still had backlogs, and systems still didn't do what businesses needed them to do .

Processes and Services

At the end of the twentieth century, a new class of software became perceived as a way to solve many problems with IT: workflow or business process management (BPM) software. These products generated high initial ROI because they integrated many disparate systems, linked people in different departments into a coherent process, and made management and reporting of an overall business process possible, often for the first time.

BPM systems, however, still assumed that intelligence and decision making in processes come from people or are embedded in systems. The use of worklists, alerts, and the paraphernalia of integrating people into processes is widespread. Most BPM systems allow limited replacement of human decision making with automated decision making. Those that do tend to focus on routing and other kinds of simple decision making, not making decisions about operational transactions.

In parallel with the growth of business process management, service-oriented architecture (SOA) started making inroads in IT departments. SOA promised a new level of agility and flexibility and reduced maintenance backlogs. To some extent, it's delivering on these promises. However, an SOA approach doesn't change how business expertise is turned into computer code, nor does it address the issues in delivering analytic data discussed previously.

Although BPM systems and SOA don't offer a way to make systems (or even processes) smart enough for today's business environment, they do offer a framework for using the approaches and technologies discussed in this book. Integrating business rules engines and analytics with the process automation and orchestration capabilities they offer can make smart enough systems more possible.

One common factor in problems with different approaches to programs is people. Relying on people to make decisions has a consequence: You can move only at the speed of people.

Moving at the Speed of People

The third major problem with the prevailing attitude toward information systems involves people. Because of the reliance on people to provide the "intelligence" in a system and the lag between data analysis and operational processes, new ideas move only at the

speed of people. You can see this problem in the training and adoption lag, the use of policy manuals and intranets, and the prevailing focus of knowledge management on supporting people.

The Training and Adoption Lag

When new approaches, techniques, and policies must influence people's behavior, typically there's a noticeable lag. Explaining to people how a new process should work takes time, and they must take time out from their jobs to attend training. These formal processes take time and can also create an adoption lag. Adopting a new policy is hard until everyone has been trained, for instance.

For example, suppose your company is concerned about customer retention, so you develop a new policy for making retention offers. You might put it in writing, e-mail it to all your customer service representatives (CSRs), and arrange for training sessions for them. If you have CSRs in a number of locations (to provide 24x7 coverage, for instance) or have outsourced them, training might require several separate training events. In addition, you might not want the new policy followed until everyone can follow it consistently. Eventually you have trained everyone, given notes on the new policy to everyone, updated the new CSR training, and can roll out the new policy. How long might this process take? Days or weeks—perhaps months.

Procedure Manuals and Intranets

A consequence of using people to make decisions in information systems is the widespread use of procedure manuals, often made available on an intranet. Policies and procedures are in writing to ensure that a manual process is carried out in a consistent and compliant way. This information is widely distributed, often in electronic as well as written form. Incentives for using the policy or punishments for failing to might be developed to increase adoption.

In practice, however much those who understand the regulations try to write a manual explaining how to make a decision, adoption is limited by the extent to which experts can explain what must be done and by the typical problem that not everyone will follow the manuals every time or perhaps even read them. In addition, relying on auditing and random checks to enforce compliance is an expensive and inefficient way to ensure consistency of operation. Attempting to make sure a decision is made consistently and accurately by publishing a manual is an uphill struggle, no matter how good the distribution mechanism is.

Knowledge Management

The final consideration for the challenges of moving at the speed of people is knowledge management. It's often proposed as a way to capture an organization's knowledge to make it accessible to others and increase the predictability and quality of decision making. The challenge with knowledge management is twofold. First, it still assumes a manual decision-making process, which might not be practical for many decisions in a fast-moving world. If you need to deliver a decision in seconds, no amount of knowledge management or training will help. You must automate the decision. Second, it focuses on managing documents electronically to automate and improve a process designed around paper. Like the obsession with using reports, managing knowledge as though it belongs in documents is limiting. Breaking down large blocks of knowledge into pieces is hard, so linking them to systems that need them, regulations that drive them, and circumstances that might affect them are also difficult.

Knowledge management is not useless in an enterprise decision management approach. Many knowledge management techniques can help capture the information you need to adopt enterprise decision management. The problem is focusing on capturing and managing knowledge, not on operationalizing knowledge in the information systems and processes that make your business function.

Unfortunately, an organization can invest heavily in IT and yet still find that it doesn't have systems smart enough to support it as it moves into the future. The lack of focus on decisions, especially operational decisions, and IT's focus on reporting for managers might be the primary culprits, but the characteristics of most application development have a role, too.

SmartEnough's Experience

So far, you have learned about the general history of computing, but in this section, we consider a specific example. Instead of picking a familiar organization to illustrate the problems, the imaginary company SmartEnough Logistics is used again. Like most companies, SmartEnough Logistics bought into the major IT trends of the past couple of decades. Like most, it got some return from these investments but found that its systems infrastructure wasn't smart enough. In no particular order, then:

- **Enterprise applications**—SmartEnough bought both enterprise resource planning (ERP) and CRM software but found the systems had overly rigid processes when used as is. Any time SmartEnough wanted to add functionality, it had to write custom code, which turned out to be hard to maintain and update. Some

modules were modified so much that they couldn't even be upgraded. When SmartEnough eventually gave up on modified code and replaced some modules with new versions (which was painful), it found that it no longer had any differentiation from other companies using the software. If only 5% of its business was unique,[6] clearly that 5% wouldn't come "off the shelf."

- **Business intelligence**—SmartEnough was an early adopter of data warehousing, reporting, OLAP, and indeed the whole business intelligence stack. Status meetings were enhanced with colorful, detailed reports. Executives got pretty dashboards. Management got a grip on what was actually happening. However, being better informed about where packages and shipments were didn't improve SmartEnough's on-time delivery much. Managers went from complaining about no data to complaining about too much data. Dashboards, it turned out, were requested and developed but often not used. All the reporting gave analytically minded people the power to manipulate and understand the data but offered little improvement in operations.

  When "Operational BI" was tried, pushing reports and analytical tools down to front-line workers, SmartEnough found that perhaps only 20 percent of those workers could use the tools. Even the ones who could often didn't have time to review a report or graph before deciding how to treat a customer. Customers wanted answers, not just information, and even the best graphs couldn't be displayed well on drivers' handheld devices. Simply having more data didn't help.

- **Custom applications**—SmartEnough recognized early that IT and custom systems matched to its business could offer a competitive advantage. It focused on marketing systems (both direct and to distributors) and the pricing engine. SmartEnough discovered that each channel needed its own technology stack and built several engines: one for the call center, one to support the ERP system, and so on. This code was hard to update and suffered from rampant inconsistencies that upset customers when prices varied inexplicably. Ultimately, the maintenance difficulties caused delays in introducing new products and pricing policies and made it hard to offer custom services to more demanding customers.

  When the Internet took off, SmartEnough added a Web site so that customers could track packages. Nevertheless, customers ended up calling constantly because they could track packages but do nothing to influence delivery. On top of that,

[6] Comment attributed to Shai Agassi, formerly head of the product and technology group and on the executive board of SAP AG.

alerting and tracking were fine for customers with a few packages but overwhelmed those with many (SmartEnough's best customers). In the end, multiple platforms and expensive maintenance used up so much of the IT budget that they prevented new development.

- **Customer experience**—At the time SmartEnough bought the CRM system, its focus was on using technology to improve the customer experience. Most of the technology had the opposite effect—acting as a barrier to good customer service. Customers hated the interactive voice response (IVR) system because it never remembered them and gave the same mindless options to everyone. Most customers learned how to jump past it and speak to an actual person. Wait times increased, and customer satisfaction fell. The Web site, kiosks, and call center software failed to make customers feel known or appreciated.

  Deciding that the problem was a lack of best practices and high turnover in the call center, SmartEnough tried outsourcing some call center work, but it found the resulting lack of control over decision making unacceptable. Focusing back on its internal staff, SmartEnough initiated a major knowledge management exercise to capture policies, procedures, and best practices. Many documents were produced and stored; many were e-mailed to drivers and call center staff. Little changed, however; employees had too many e-mails to read and too little time to read them.

- **Automated package scanning**—Realizing that packages were at the heart of its business, SmartEnough has continually invested in package scanning and tracking technology. It has bought everything from handheld computers and barcode scanners to radio frequency identification (RFID) and global positioning system (GPS)-enabled trucks. Despite major improvements in visibility and tracking, SmartEnough still found that drivers couldn't act for their customers; they couldn't cross-sell or up-sell effectively by offering higher-value services when a customer wanted urgent delivery, for instance. The drivers didn't understand the trade-offs and were moving too fast to spend time reviewing detailed analysis graphics on their handheld devices.

  Meanwhile, the managers tracking overall results appreciated the data they got but still found that everything hinged on people responding to the alerts and reports the system generated. As staff became inundated with reports and notifications, alert fatigue set in, and more alerts were ignored. Simple decisions were held up until someone in another time zone got to work, and the whole edifice relied on a few experts juggling data and analysis.

- **SOA, BPM, and BAM**—Most recently, SmartEnough has tried to upgrade its IT infrastructure to become more agile, which has helped. SOA has made composite applications easier to develop, but many services are still hard to change, and reuse of services has been slower than expected. SmartEnough worries that it has JABOWS (just a bunch of Web services), not a real architecture at all. Making legacy applications available as services seemed to help, but the same number of maintenance requests come in, and legacy systems are no easier to change than before.

  BPM has helped control and monitor some processes and given SmartEnough more flexibility in some dynamic processes. Problems have arisen, however, in processes customized for major customers. Managing all these different processes is complex, even though almost everything in a process is the same. SmartEnough also found that the benefits, which were published in a major IT magazine, wore off quickly after major inefficiencies were eliminated, because there were still too many manual referrals and too much waiting for decisions. Adopting BAM solved some of these problems, but too many people were still responding to too many alerts, and critical activities were often held up until the next morning.

Perhaps SmartEnough Logistics sounds like an artificial nightmare to you. Perhaps it sounds like your company. Probably your company falls somewhere in between—no one has tried everything and had it all go wrong, but most organizations have had at least some of these experiences. If, like SmartEnough Logistics, your systems aren't smart enough, you probably want to know how enterprise decision management will help and how you actually go about it. The next few chapters introduce the core concepts you need to understand, describe an easy path for adopting enterprise decision management, and show you how it might affect your IT environment. Read on.