CHAPTER TEN

EDM and the IT Department

The impact of EDM on the IT department can be profound and positive. EDM builds on some of the most important trends in IT, solves some of IT's most persistent problems, and helps power some of the capabilities most in demand from IT departments. This chapter covers some of these impacts and discusses how decision services are deployed. The impact of EDM on software development life cycles and methodologies is also considered.

Complementing, Solving, and Enabling

In many ways, the time is right for EDM, and it probably wasn't right until fairly recently. Although some organizations, many used as examples throughout this book, have been using these technologies and their predecessors to build smart enough systems for some time, the widespread adoption of EDM technologies and concepts wouldn't have been possible without today's IT ecosystem. Indeed, a major advantage of EDM is that it complements other technology adoption trends. It's additive rather than competitive.

A typical IT architecture has many components that relate to two aspects of EDM: data infrastructure and operating infrastructure. Some components relate to both.

The data infrastructure contains more data that's better understood, better organized, more timely, and more integrated. Many technologies, from enterprise information integration (EII) and enterprise application integration (EAI) to customer data integration (CDI) and master data management (MDM), contribute to integrating and organizing information. Business intelligence/data warehouse (BI/DW) and corporate performance management (CPM) technologies help manage and understand it.

All these technologies build on an operational infrastructure structured around a backbone of enterprise applications—such as customer relationship management (CRM), enterprise resource planning (ERP), and sales force automation (SFA)—that store, manage, process, and maintain data as part of running the business. These enterprise

applications also provide an electronic backbone for delivering information and, therefore, decisions to front-line staff and associates who need them.

Enhancing and supplementing these applications with business process management (BPM) software makes them easier to use in support of complex business processes, and the move toward a service-oriented architecture (SOA) makes building, managing, and reusing components and services easier. Ever-improving Web and client user interfaces, including those labeled "Web 2.0", and their capability to make more systems accessible to a wider audience continue to push this electronic backbone closer to those concerned with these processes.

Meanwhile, IT departments struggle to develop new applications while being submerged in maintenance requests to upgrade and enhance existing systems. Many are asked to deliver business activity monitoring (BAM) and event processing to make businesses more responsive and to mobile-enable their workforce, customers, and associates. Providing more self-service applications that work in multiple channels is a persistent problem, as devices multiply and the Internet changes communication styles. IT departments have legacy platforms of all types, many of which must be coordinated, yet the pressure to introduce new technologies, such as social media, and new approaches, such as model-driven development, continues.

EDM offers opportunities to build on and complement your IT architecture, solve some problems your IT department is facing, and provide some of the most demanding functionality on your to-do list. When you think about a modern IT architecture, you probably do so in one of two ways. Perhaps you're an optimist and a follower of technology trends who sees the new capabilities and technologies revolutionizing enterprise IT or perhaps you are a pessimist who can think only about your organization's hodgepodge of aging technology and the limitations this imposes. Table 10.1 compares these two contrasting views of enterprise IT.

Table 10.1 Comparing Views on IT Architectures

Optimistic View	Pessimistic View
The state of the art is moving to a world that's "digital, mobile, virtual, and personal," as Carly Fiorina, former CEO of Hewlett-Packard, said in 2004.	Most organizations have an IT architecture that's static, complex, messy, and impersonal.
All functionality is made available through well-defined services running on a robust SOA platform that provides a strong repository and high performance registry.	Although new projects take an SOA approach, much of the functionality the organization uses isn't available as services.

Optimistic View	Pessimistic View
Services are used in composite applications. Mostly they are defined by using a BPMS that orchestrates them into effective business processes supporting the way the organization needs to operate.	Although business process management is increasingly important, few core business processes have been reengineered into a BPMS. Some new applications are composite applications, but most are not, and business logic and other functionality are coded into each one.
Processes are monitored and tracked to see how the organization is doing against its key performance indicators, and those involved in the process have a rich set of information available to them.	Most reporting is from a data warehouse that's not current or complete. Analytics are rudimentary, with power users building their own reports and many standard reports of dubious value. Data mining is scattered and piecemeal.
Business activity monitoring is tightly integrated into processes to ensure that those managing them are immediately aware of issues or bottlenecks.	Monitoring is mostly by reports, and dashboards are limited in scope and are not as up to date or useful as needed.
The data used in services is integrated so that all information about customers or products is available in a single request, and data is available for summary and analytic reporting without performance implications.	Data is stored in several databases, and integration remains a problem. Technologies such as EAI and EII bring together some data sources in an application, but despite CDI and MDM initiatives, integrating metadata remains a problem.
The organization uses packaged enterprise applications for standard functionality—functionality that supports the organization but doesn't differentiate it.	Much of the functionality used is available only through large legacy applications running on a wide variety of platforms.
All this functionality is available as services and is easy to integrate with custom services the organization develops.	Enterprise applications for CRM or ERP are used, but various collections of functions are at different release levels, and few or none of them are available through services.
Some processes are outsourced, and integrating them is managed easily by using an SOA approach. Their performance is managed effectively alongside the performance of internally run, but related processes.	Some departments use software as a service (SaaS) providers to plug gaps in the information architecture, but there's little or no integration with internal systems. Similarly, outsourced processes are managed at arm's length, with reports going back and forth and no integration.

continues

Table 10.1 Continued

Optimistic View	Pessimistic View
Outside core processes, the organization is highly event-driven. Data on inbound events is processed, aggregated, and acted on immediately to notify staff and systems to act. Event-driven functionality reuses the services and service architecture that process-driven elements use.	Event-driven applications are simple and notify people to take action only through worklists and simple e-mail notifications. Few, if any, parts of the business are managed by using a combination of events and processes.
Customer and employee-facing applications use dynamic Web interfaces on thin and mobile clients seamlessly. Wireless and other location-aware technology is used with customers and mobile staff.	Blackberries are the only mobile device of note in the organization that are connected to enterprise systems. The Web site has some features available to mobile device users, but management of mobile, location-aware devices as a channel is rudimentary.
Social networking technologies allow associates to use different aspects of the organization that matter to them in a unique mashup of functionality and information.	Some small projects are adopting Web 2.0 technologies to build more engaging and dynamic Web sites, but they don't affect core enterprise applications, and early attempts at social networking aren't integrated with anything else.
The IT architecture runs on a hardware platform that's interoperable and closely managed.	Every conceivable kind of hardware and operating system seems to be in use. Processes and applications often require more than one platform.
Requirements are managed in tools and combined with models. Systems development and maintenance work come from these models.	Projects use models, such as those defined in Unified Modeling Language (UML), in theory, but the models have little or no long-term value. Models are used in initial design and construction, but ongoing maintenance and modification are manual. Requirements are poorly defined and managed.
IT is a source of innovation and focuses its energy on new systems that add value to the business.	The application maintenance backlog is large and consumes the majority of IT resources to change existing systems to match new and revised requirements.

If your organization looks more like the optimistic view, EDM can add value to your architecture. If your IT architecture looks more like the pessimistic view but you're trying to move toward a truly modern IT architecture, EDM can build on the architecture you *actually* have as well as the one you're building. Before considering the impact EDM can have on the IT department's concerns and issues, reviewing a brief summary of how decision services are deployed can be helpful.

Decision Services and the EDM Ecosystem

Deploying a decision service into the production environment where other applications and services can access it is the final step in bringing EDM solutions into operation. Typically, this step means deploying decision services into an existing IT infrastructure and linking those decision services to business processes, enterprise applications, and Web sites. Figure 10.1 shows how one particular organization uses centralized business rules and analytic model infrastructure to deliver effective decision services to multiple operational systems. A number of key concepts, discussed in the following sections, are involved.

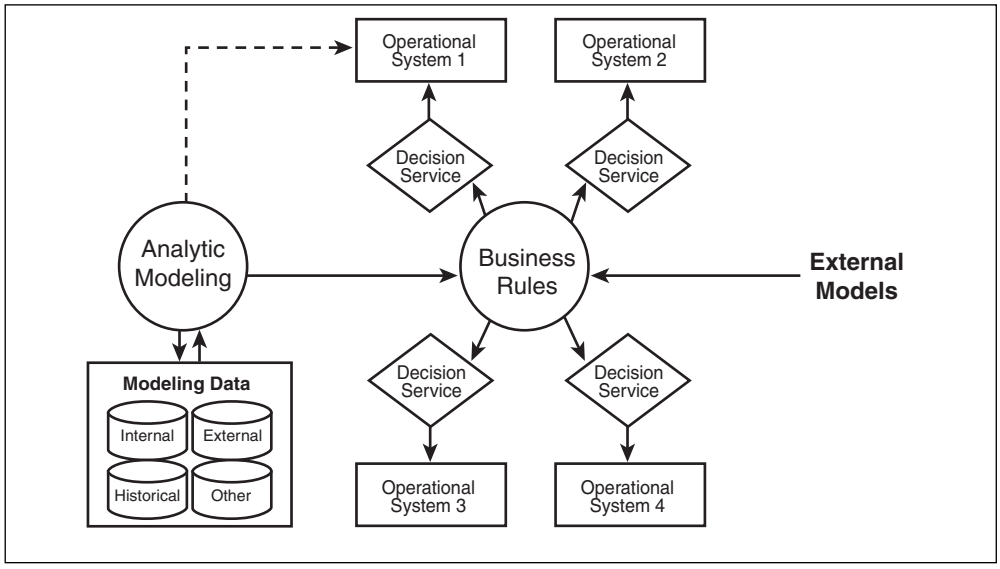

Figure 10.1 An architecture showing how rules and models deliver decision services in a major credit card issuer

Role: Developers

Developers (programmers) build information systems and the "plumbing" for decision services. They need the following:

- A willingness to give up some control of their systems to improve them
- An understanding of business users' perspectives so that they can build effective rule maintenance environments for them
- An understanding of the value of analytics and the challenges in using them
- Communication skills and an ability to work with both analytical and business users

Developers do much of the construction work of a decision service and the setup to allow business users to be involved. On top of usual developer skills, they need a willingness to partner with business users on development.

Concepts in Deploying Decision Services

From the point of view of an IT department, there are relatively few new concepts that you need to consider when deploying decision services. Data must be mapped, live updates must be managed, and the additive nature of decision services understood.

Data Mapping

Decision services execute rules and models against information from other information systems. At deployment time, the rules and models used in a decision service must be able to access this operational data without any runtime overhead. At runtime, the decision service must access data by using standard, high-performance application programming interfaces (APIs) and not require data type conversion (casting or mapping) that imposes a performance burden.

Live Updates

With most decision services, you must be able to deploy new rules and models to a running service without having to interrupt or restart it. Therefore, you need a way to schedule updates or specify an on-demand update mechanism that allows deploying tested rules and models. When these **live updates** happen, running transactions must be able to finish, using the rules and models they started with (although this capability might be

overruled for some updates), and all new transactions must use the new rules and models. This capability is widespread in today's decision technology, and for highly agile systems it can be crucial.

Additive Services

A key characteristic of decision services is that they tend to be additive. They aren't replacements for existing systems or services, nor are they **systems of record**—they don't manage the data that documents your business, such as orders and customers. They don't run the business directly; instead, they allow the systems you have running your business to run more effectively. Decision services typically don't *replace* existing systems but *enhance* them or replace only part of the existing system—hard-coded decision logic. When considering how to integrate decision services into your IT infrastructure, keep this characteristic in mind.

Deployment Process

Figure 10.2 shows how a decision service responds to requests for decisions from production applications by executing rules and models against information, all in the context of your enterprise IT architecture. Getting decision services to this point involves quality assurance (QA) and test processes, deployment, and integration.

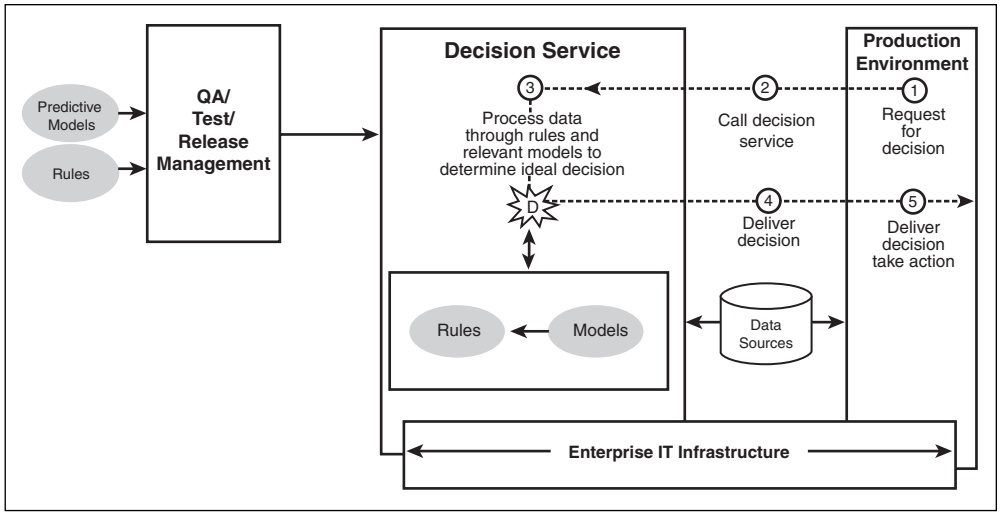

Figure 10.2 Deploying decision services in EDM *©Fair Isaac Corporation, reproduced with permission*

QA/Test

Each version of a decision service is put through a quality assurance (QA) and test step that's usually the same as the QA/test for other services in the application portfolio and, like them, is concerned with integration and system test issues. If the rule maintenance environment enables business users to change rules in a running decision service, this capability probably isn't covered in standard QA/test plans. You need to develop suitable tests and checks to make sure that authenticating business users, limiting their edits, and redeploying rules to running services all work correctly. A separate QA/test step, focused only on rules changes, takes place *after* business users have made changes to the test system and before those changes are applied to the production decision service.

As discussed in Chapter 7, "Adaptive Control," champion and challenger strategies should be run through the QA/test process, and the whole environment needs to be tested to make sure the right percentage of transactions flow through each challenger. If the decision service will check for new rules or models automatically, this process needs to be tested, too.

Deployment

How you deploy a decision service into a production environment depends largely on the production environment. If a decision service is deployed into several distinct environments without using an SOA approach, each deployment is separate, and part of the testing should ensure that all deployments stay synchronized correctly.

Many BRMS products support automated deployment of new, tested rules, and more modeling environments now include this capability for models. Deploying decision services that monitor for new rules or models typically requires deployed services to have access to repositories where these rules or models are stored.

Integration

Decision services must be able to receive data from applications asking for answers and be able to pass information back. In addition, many decision services access internal and/or external data as part of making a decision, and these data sources need to be integrated with the service. Technologies such as enterprise service buses (ESBs) and business process management systems (BPMSs) might also need to be integrated. Integration usually isn't complex; modern decision-making technologies are built with integration in mind, but it must take place and be suitably robust. Figure 10.3 shows one form of decision service integration where decision services are deployed as Web services and accessed using standard interfaces. The applications calling the decision service treat it like any other service.

Integrating decision services into your IT architecture can have many benefits in terms of complementing and strengthening it, solving some persistent problems, and delivering some needed improvements in IT capabilities. These benefits are discussed in the following sections.

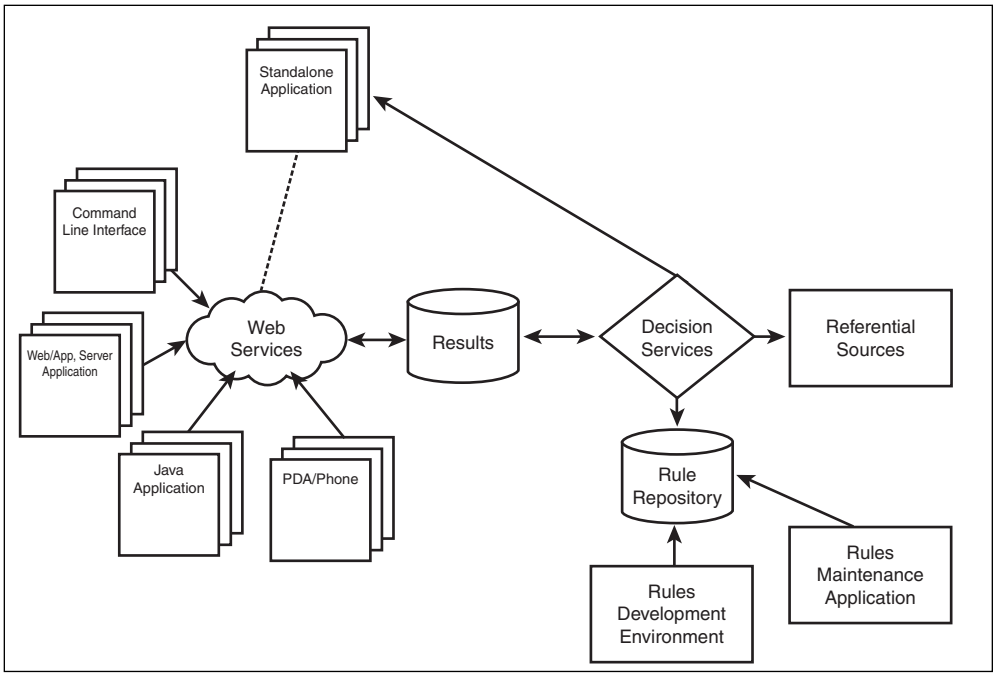

Figure 10.3 An architecture for preventive maintenance decisions that uses Web services to integrate with systems

Complementing Your IT Architecture

The first area of IT architecture to consider is what EDM *complements.* EDM builds on and takes advantage of some current IT architectural trends:

- Service-oriented architecture (SOA)
- Business process management (BPM)
- Data integration (including CDI, MDM, EAI, and EII)
- Web 2.0 and social networking (tagging, mashups, wikis)

EDM builds on each trend in different but complementary ways, as explained in the following sections.

Building on SOA

A major benefit of adopting an SOA is supposed to be an increase in business agility, mostly because of the reduced time, cost, and difficulty of making a change. The definition of functionality as coherent components or services with well-defined interfaces helps limit a change's impact to a single service, which makes change easier to control and implement. Well-defined services are loosely coupled—they use service contracts to allow services to interact without having to depend on interaction. These services change independently, and as long as the interface to the service doesn't need to be changed, independent service changes shouldn't affect other services. SOA contrasts with the typical result of changing monolithic applications—a change is likely to cause a ripple effect throughout the application stack. SOA also supports a more iterative approach to defining services because of this control over the impact of change, which also helps in agility by eliminating the need to define a complete set of requirements upfront. SOA makes more agile development possible.

When you define business services with SOA, you can decouple the business from automation of the business. Business services are independent of a particular process; they perform a business function you can use in many processes. In this way, you can define new composite applications and business processes that use existing business services, which increases reuse as well as agility. Now you can assemble a new process—such as for handling a new channel, for example—mostly by orchestrating existing business services, especially with entity-centered business services in which functionality is associated with a defined entity or set of information, such as customers or accounts.

In addition, using an ESB to implement an SOA can increase agility by providing an integration layer and enabling you to assemble services on different platforms and perhaps with different interface semantics. By making it easy to add new services, transform messages to allow services to interact, and so on, an ESB can increase the level of agility beyond what service orientation alone can offer. Figure 10.4 shows how one organization used publish/subscribe interfaces with a decision service to ensure that specific process steps and messages on the ESB could trigger the same decision. This infrastructure also enabled the decision service to publish additional messages onto the ESB, which allowed for easy integration with other services connected to the ESB.

Clearly, some services implement a business function that must change more often and be more capable of adapting to change than others. Some services increase value when changed. The costs of failing to change some or doing so in a way that can't be audited for compliance might also vary in services. These services are defined as decision

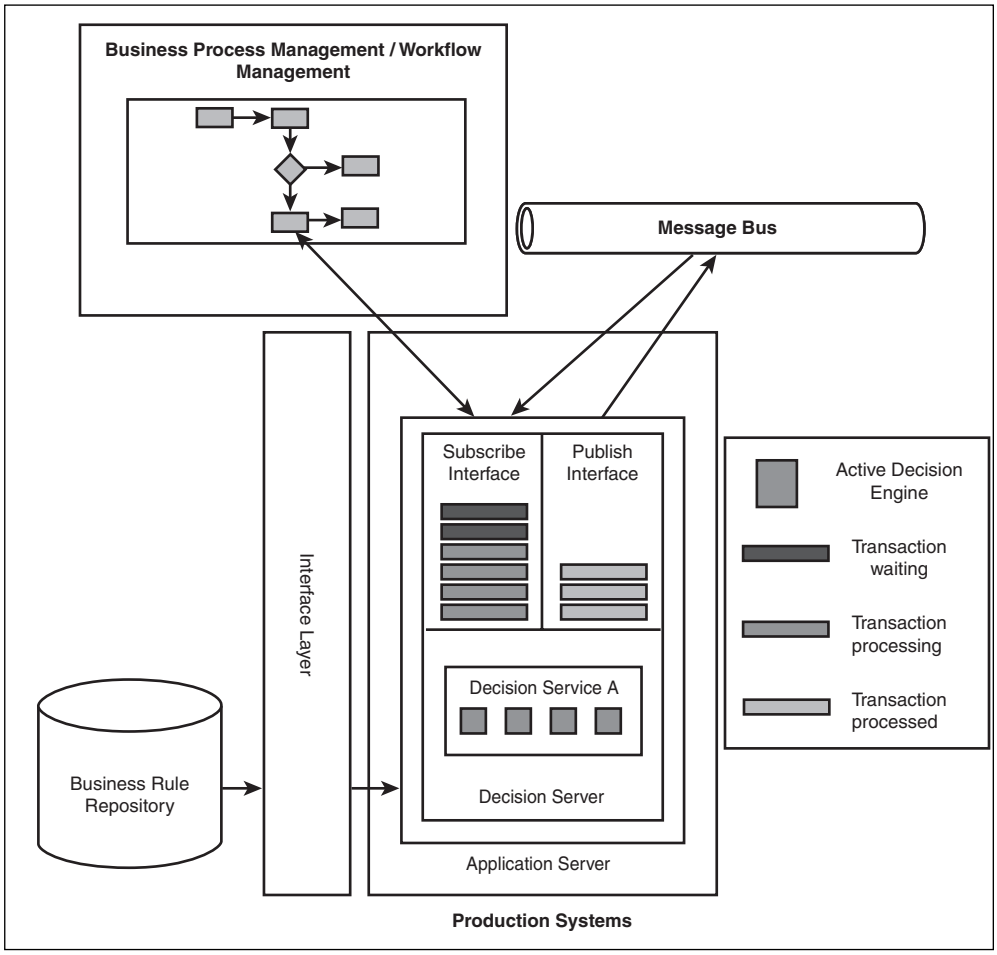

Figure 10.4 How a decision service can communicate decisions with BPM or workflow software and a message bus by using a publish/subscribe approach

services and usually implement business functions that are in constant flux, complex or voluminous business logic, or business functions that aren't easy for programmers to understand or for which business user control is critical. With decision services, business logic can be changed and shared more easily between services in the SOA and non-service-enabled applications in the portfolio. As shown in Figure 10.5, decision services are a subset of all possible business services as well as legacy services available for reuse.

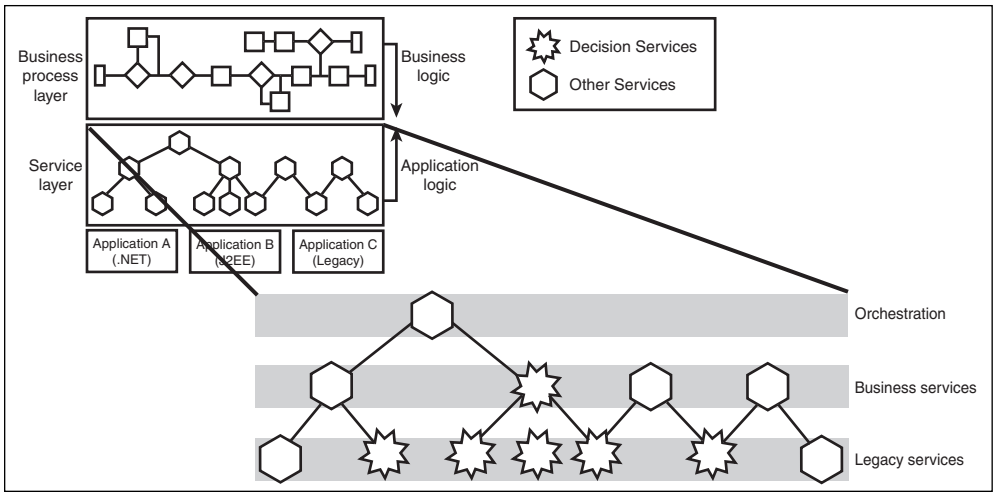

Figure 10.5 Decision services are a subset of all business services

Figure 10.6 shows an SOA implementation of decision services for a mortgage lender in the United States. Various services provide credit retrieval and summarization, secondary market analysis, customer scores (based on risk models), and product and pricing information. The core decision service then handles the mortgage origination decision.

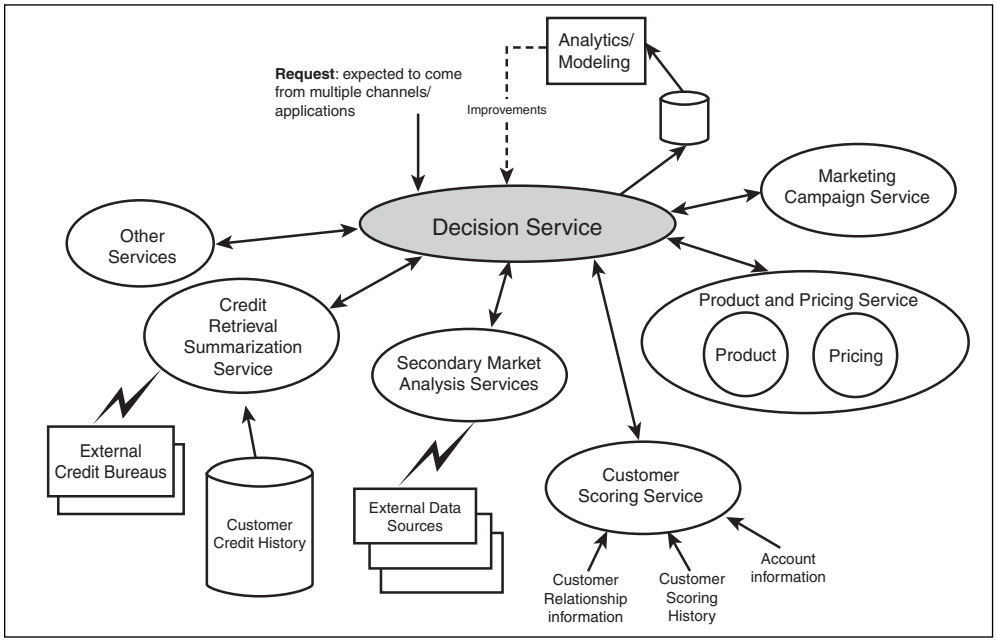

Figure 10.6 An SOA for decision services in mortgage lending

Decision services also allow a more effective "build or buy" decision. Organizations can buy services based on best practices and standards where the functionality of those services is not critical to the organizations' competitive differentiation. They can then build services with competitive potential and use an SOA infrastructure to compose them into effective applications and processes. Many services that differentiate an organization—that is, that define how it acts differently within a standard process framework—are decision services. Focusing on decision services can, therefore, make it possible to construct composite applications mostly from standard services that still deliver a unique and competitive customer experience. In addition, integrating analytics in decision services is a more effective way to apply data to improving processes than trying to "service-enable" traditional business intelligence (BI) tools.

Completing Application Decomposition

If you embed decisions in your applications, you hide these decisions from view and delegate details to the wrong people—systems developers, not business users. Traditional application development techniques hide decision logic deep inside software, making development time-consuming and costly. Developers have to translate business requirements ("If this condition is encountered, respond in this manner") into abstract representations in programming languages—a laborious process full of possibilities for error.

By embedding decisions in applications, your decisions become a liability. By managing decisions, however, you can enable business users to make their own changes, which reduces the time to make changes and reduces maintenance expenses. Focusing on decisions as a separate component is, in many ways, the last step in the decomposition of traditional applications.

Not so long ago, applications were monolithic, containing data, user interfaces, business logic, and process flow in one block of code. Then the process of decomposition shown in Figure 10.7 began. With the advent of databases, managing and reusing data became easier if it was removed from applications. For the first time, data was defined so that people knew what it represented, which allowed business users to access data for themselves. Next, client/server, thin clients, portals, and rich interfaces improved and separated the interface from the application. The same interface could access multiple applications in more sophisticated ways. Most recently, BPMSs have been adopted, which makes it possible to externalize work flow and build cross-application flows effectively. All that's left in applications is technical code and business logic. Decomposing applications one more step by separating business logic into its own managed environment makes more sense now and could be the most important advance to date.

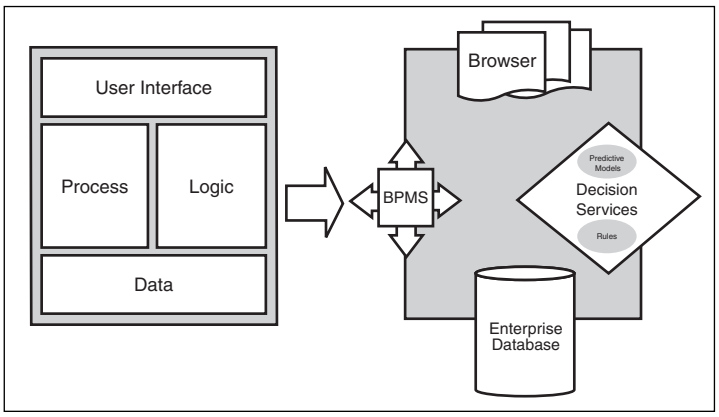

Figure 10.7 The evolution of applications from monolithic to completely decomposed

Avoiding Brain-Dead Processes

As the monolithic application of old has been decomposed gradually, organizations are using BPMSs to design and build more processes. A BPMS focuses on *how* a process should be carried out. It helps standardize processes, facilitates collaboration and compliance, defines and manages workflow, automates steps, and provides activity monitoring, alerts, process reporting, and integration. What it doesn't do well is decide what should be done. A BPMS alone doesn't help standardize operational decisions, facilitate decision automation and maintenance, centralize business rules, or support straight-through processing in any but the most simple situations.

Adopting a BPMS without also adopting an EDM approach to decision automation has a number of risks, as described in the following list. You have probably spent time thinking about processes you're implementing in a BPMS but not as much about decisions in those processes—the "diamonds" in your process diagrams.

- A BPMS doesn't manage business rules or decisions properly. It manages process orchestration and process flow design, but not rules or policies. A BPMS does have some support for rules, but usually only as part of the definition of orchestration or composition. As a result, business rules and the decisions they automate are an afterthought.
- Without explicit management, business rules are reburied in the new process, which makes the process complex. Routing rules aren't business rules; decision-centered business rules are about the organization's underlying behavior, not its processes.

- Inconsistency in business rules is likely. This inconsistency is a problem, particularly if you need several kinds of BPMSs, and embedding policy rules in each BPMS means duplicating them and failing to manage them as an asset. Ensuring enterprise consistency in processes is hard unless you manage the decisions in them separately.
- Problems with consistency and rule management can cause trouble when regulators ask you to explain how you picked a particular branch in your process. Being able to explain just the process is not enough. Noncompliance caused by faulty business rules is likely, leading to fines.
- Although you can add process analytics to a process, you can improve the process only manually. Someone must examine and redesign the process. If you have automated decision points and manage them, you can use analytics to improve a process by adding analytic models to aid automated decision making.
- Personalizing transactions for customers is hard unless you make transaction-centered decisions in the process. You probably don't want to create a personalized process for each customer, but personalizing the decisions you make about customers as they run through a standard process might be just as effective.
- You might not get the business agility you're looking for. Although some problems require a change in process definition, others do not. Especially in a core process that doesn't change much, agile management of decisions could matter more.

Compliance Issues

Some compliance is about processes—whether you follow certain steps or keep certain data—and some is about rules—whether you enforce certain rules or take only allowed actions. Often both types of compliance are required. In a healthcare claims process, for example, you might have to show how the claim was reviewed or referred for a second opinion, and show when you saved information and what information you saved. You might also have to show that the rules you followed for deciding to decline a claim were legitimate and appropriate. You can't get compliance correct without the right mix of flexible process automation and effective decision automation.

European Health Insurer: Claims Handling

Old Way

Claims were processed with a 30-year-old system running on a mainframe combined with client/server systems. Business users had no understanding of the decision logic used in the systems and couldn't adjust it, so it became out of date. Any changes required costly IT projects.

EDM Way

Three decision services are integrated with a BPMS and a data integration hub. They handle claims adjudication and payment, among other decisions. Business users manage business rules for adjudication, for example, which ensures that rules are current and makes future product additions and enhancements straightforward. The centralized decision services provide consistent decisions and benefit calculations for all claim types and enable a high rate of auto-adjudication for maximum efficiency. Decision services can apply additional rules to route claims to the correct departments. These same rules deliver Web-based self-service for claimants. Figure 10.8 shows the architecture with three decision services.

Benefits

- Flexibility and agility in making changes and adding new products
- Consistency in decisions and benefit calculations
- Adjudicating claims automatically

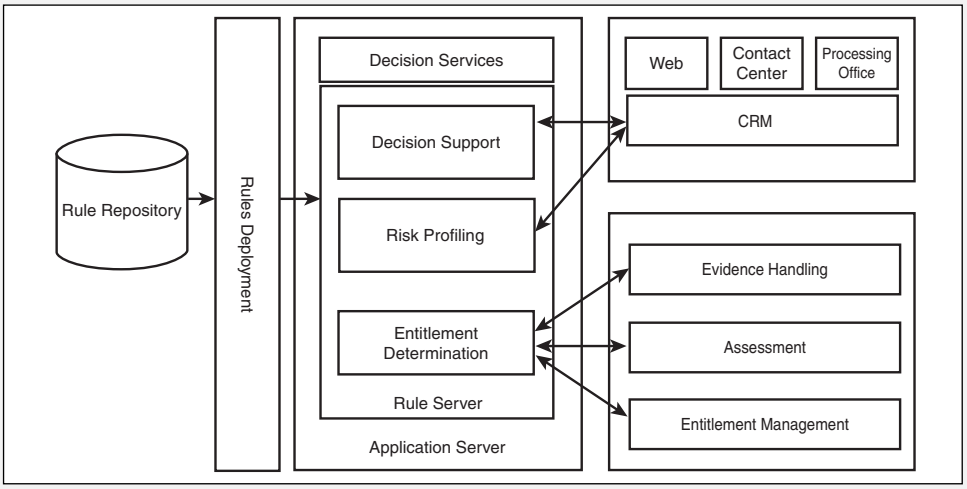

Figure 10.8 An architecture for a claims-processing system

Similarly, adopting a BPMS and EDM in parallel has several potential benefits:

- Using EDM to automate decisions in a new process can simplify the process dramatically. You can often eliminate several steps to have a single decision node in the process. When the automation percentage is high, the main process becomes the one without manual intervention, and the more complex one becomes the exception. Even if you can't reduce the steps in the design environment, the implementation complexity of a typical process is reduced. Figure 10.9 is an example of two decision services used to reduce the complexity of a process.

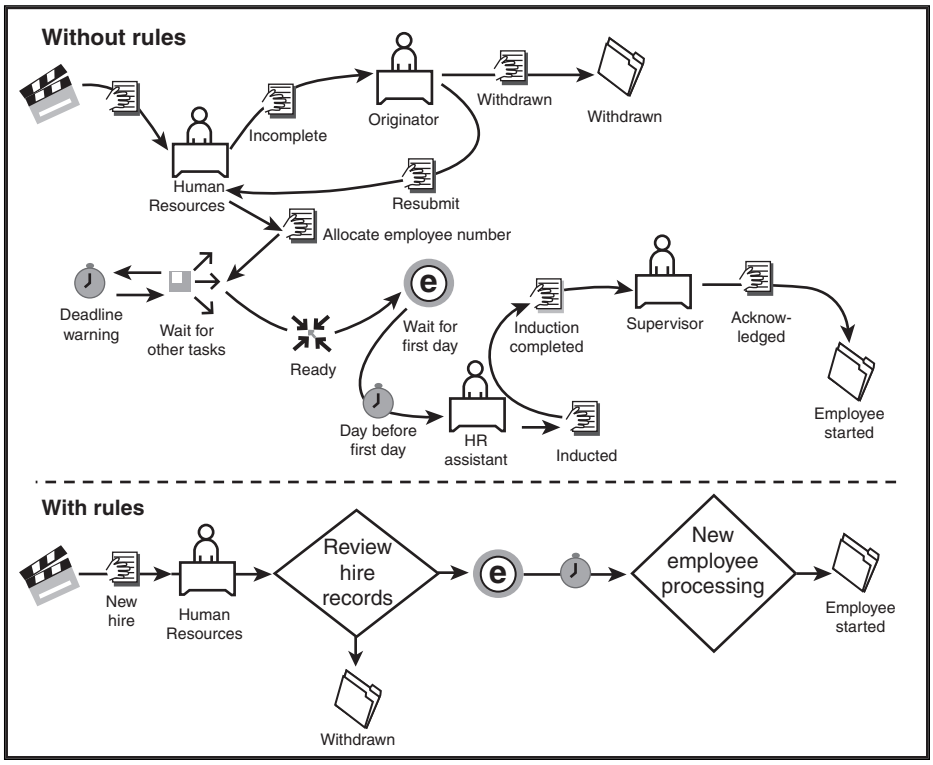

Figure 10.9 A process simplified by automating critical decisions

- Using business rules in an EDM approach to manage rules in a process's decisions makes management more effective. You can tie rule sets to business objectives and monitor them, and although rule sets are tied to a new process, your rules aren't buried in it. You can tie these same rule sets and decision services to other systems not built with a BPMS so that you can use the same decisions in several applications. For example, business rules for dealing with an order might affect your call

center, processes for problem resolution in your BPMS, self-service applications on your Web site, and legacy systems for bill production.

- You can manage and deploy process and decision changes independently. There's no reason that the need to change a decision should correspond with a need to change process steps. When you have long-running processes, keeping decisions separate allows you to change work in progress. The process definition to be used to process an item of work is fixed for that item at the time it's instantiated. For long-running processes, a fixed process definition that embeds decisions is a problem because business rules and analytic models might not be current when they're evaluated. If you manage decisions separately and retrieve them when a process needs the decision, the decision is always current.
- If you want to use BI with business processes, especially those you're automating, you need to be clear what it is you want to do. To improve a process, do you want to analyze how you run the process, or do you want to use your store of information about products, customers, suppliers, and so on? Most BPMS tools handle analysis for you: They help you see how you run the process and what trends are identified and help you use that information to improve ongoing execution of the process. To use your store of information, you have two choices:
  - Use traditional BI tools to deliver information to someone who performs a manual step in the process.
  - Use an EDM approach to apply data insight to decision services.

  If you use only reporting-style BI tools to apply analytics to your process, you're limited to investigating and understanding your process. With EDM, you can embed data-driven and scientific decision making in your processes using decision services.

Using EDM to avoid brain-dead processes is common when a business process reaches the point of needing a complex automated business decision before continuing, such as origination, underwriting, fraud detection, or precision marketing. To do this, the process calls a decision service to examine applicable data and recommend actions, such as which products or services to offer or who should be notified of the current status. The BPMS uses the decision data to continue its flow through the process.

Types of Agility

When focusing on agility, you need both process agility and decision agility. Some kinds of change require changing a process in response. A change in core processes usually has a major impact on the entire organization and could require organizational change and perhaps new audit procedures. Processes around a company's "edge"—those with lower transaction volumes, less repeatability, and more manual steps—must be easy to change, because they're likely to change often. To do this, you need **process agility**.

Sometimes the change required in a process isn't about the process itself. For instance, a change to rules for determining price discount eligibility doesn't change the process—it changes the decision of what discount to offer. To achieve **decision agility**, you must be able to change decisions in a process quickly without changing the process. This agility matters most in core business processes that are stable in steps and outcomes but can vary in decision making over time.

EDM can also take advantage of a BPMS when a decision service reaches the point of needing additional data in the decision process, which requires human intervention. The service initiates a BPMS process to bring the right users into the flow and step them through the required tasks. When the BPMS process finishes, it reinitiates the decision service with a saved state and new data. Similarly, a decision service can call for a complex business process to be started. It calls the appropriate BPMS process as the rule action, often while continuing to run the rest of the decision.

Better Decisions, Not Just Better Data

Many organizations deploy a vast array of technology to better integrate data and deliver it more effectively to the part of the business where it's useful. These technologies usually fall into these broad categories:

- **Enterprise information integration (EII)**—EII technology creates a virtual object from a variety of data sources that other applications can use without having to worry about the original data source. EII can result in much looser coupling between data sources and the services that need the data.

- **Enterprise application integration (EAI)**—Similar to EII, EAI enables integrating enterprise applications to support a business process. Unlike EII, it's not just about integrating data in those applications; it's also about the process that runs through them.
- **Customer data integration (CDI)**—CDI technology gives companies a 360-degree view of their customers by reconciling different data sources with customer information so that all information the organization has about customers is accessible after the customer has been identified.
- **Master data management (MDM)**—In many ways a superset of CDI, MDM is an attempt to bring a company's reference or master data under control. This data is typically spread over many data sources and is hard to access coherently.

The challenge with these technologies is that you must *act* on the information to get value from them. Overestimating the value of making more information available is easy. No matter how easy you make it to use information, you still assume that users can put it into context and use it. For instance, if your doctor has an electronic medical record of your entire history, will she make a different treatment decision because of it? Will she have the time to read it all or be able to spot the crucial piece?

Transaction-Centered Processes

Using EDM for decision automation in processes allows you to build processes that are driven by transactions. The data or metadata in the transaction determines what scores models generate, and the combination of scores and data determines which rule fires in your decision service. This in turn decides which steps to take to complete the transaction.

You have now "inverted" the process—it flows from the customer to the organization. An example of an inverted process is an origination process, in which data the customer enters affects the models and rules for determining which products are available and the process then executes to offer and fulfill those products. This kind of analytically driven, transaction-based process is a key component in customer focus and personalization strategies.

Making information more readily available is important, but making better decisions based on information is what pays the bills. Bill Gates was quoted recently as saying:

> *"Resolving the information overload and underload problem will take more than just better search tools. What's required is a comprehensive approach to enterprise information management that spans information creation, collection, and use and helps ensure that organizations can unlock the full value of their investments in both information and people."*[1]

You also need ways to turn better information into better decisions and, therefore, better outcomes. Better-informed organizations don't perform better automatically; they perform better if they can make better decisions with that information. In other words, having a 360-degree view of a customer results in better customer treatment only if you can, and do, use that view to *improve decisions.*

Another challenge in using data integration and management technologies is latency.[2] You can divide latency into three categories: capture latency, analysis latency, and decision latency, as shown in Figure 10.10. When many organizations talk about real-time data, they focus on capture latency—how long after the business event data is available for analysis. In reality, analysis latency (the time to analyze data) *and* decision latency (the time to decide how to act in response to analysis) also matter. EDM builds on the latency reduction of data integration and management technologies by reducing analysis and decision latency as well and adding business value, as shown in Figure 10.10.

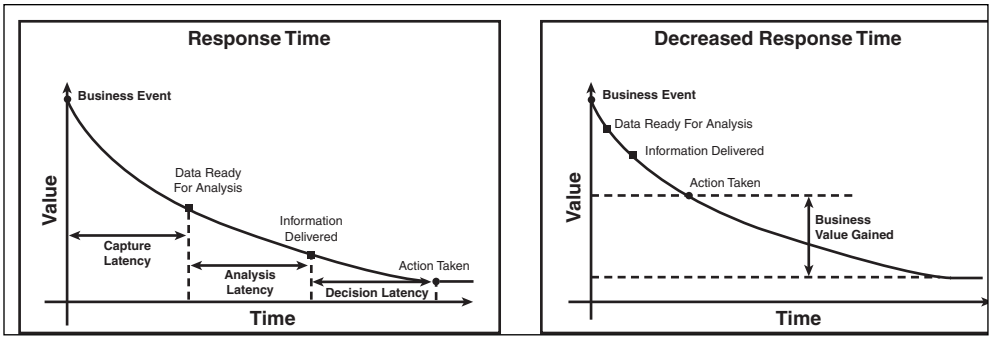

Figure 10.10 Types of latency between events and responses

[1] Bill Gates, "Beyond Business Intelligence: Delivering a Comprehensive Approach to Enterprise Information Management," 2006.

[2] Richard Hackathorn, "Active data warehousing: from nice to necessary," *Teradata* magazine, June 2006.

U.S. State Tax Authority: Nonfilers

Old Way

The tax authority had more than 200 million pieces of information from federal and state sources on tax returns. It used this information to identify citizens who haven't filed all the required tax forms or who have made major mistakes in filing. A 30-year-old system failed to identify nonfilers correctly, missing some and misidentifying other taxpayers as nonfilers. Tax revenue was lost, and taxpayers were angered. Millions of nonfilers had to be handled, and changing the system to respond to new tax laws or data sources was hard. Hundreds of millions of dollars was at risk annually.

EDM Way

A decision service contains business rules that match filing information to other data sources to identify potential misfiling or nonfiling citizens. Business and technical staff collaborate on rules to make processing more accurate. Those who understand the tax system can edit rules. Changes can be made to rules whenever regulations or policies change, and all rules are stored and managed in a central repository. A second decision service contains rules for managing the process of contacting nonfilers (such as generating customized letters) and supporting self-service capabilities for taxpayers to help them correct filing problems.

Benefits

- More than $30 million annually in new revenue
- Mistakes in identifying nonfilers reduced by more than 50 percent

Business Agility and Data Integration *When a decision service uses complex data, business agility is lost unless the integration is easy to manage in the face of change. An EDM approach's agility depends mostly on a stable object model. To improve business agility, integration technologies can ensure that the decision service "sees" a stable object model, even if underlying data sources change.*

One final note on how EDM builds on data integration technologies: Proving an ROI for some MDM/CDI initiatives can be difficult. How do you show the value of better decision making when you don't have a good definition of what "good" decision making is? Using an EDM approach, particularly the champion/challenger strategy, can be effective in proving an ROI. Comparing an account-based strategy with a customer-based one, for example, can show whether a new strategy is better and, if so, how much better. This comparison enables you to put a value on data integration, but only if the decision has been automated, of course.

Using Customer Interests and Social Media

One trend made possible by the Internet is the growth in customer (and other associates) participation. Customers can write blogs, contribute content to wikis, develop their own mashups that include your content or reference your products, tag and review your products, or change the rules you use to interact with them. All this information tells you something about customers, but using the information often seems impossible without manual intervention, which isn't possible either if many people are involved. With EDM, however, you can get more value from this type of customer interaction.

When you're trying to use this kind of customer information, however, remember that the act of participation tells you something about customers. You can include information about how much they contribute (number of postings, number of reviews, average ratings, number of contributions to a wiki, and average length of contributions, for example) as part of their information profile. These characteristics might be highly predictive of a certain class of customer. Wikipedia, for instance, finds that some contributors make a few important edits (they could be considered subject experts), and others make many minor edits (they could be considered content stewards). Understanding your customers in this way can be helpful. You can also use these characteristics for rules-driven treatment of customers. For instance, you might route a call to more knowledgeable customer service representatives (CSRs) if the customer contributes regularly to your product wiki.

You might also be able to analyze what customers say to infer their opinions. This analysis might be as simple as using their ratings or as complex as text analysis of postings to look for competitors, product names, positive or negative words, and other details. With this analysis, you could separate customers into those who like your products and those who like your competitors' products, for example. In reality, only a small percentage of any community participates in this way—perhaps a maximum of 5 to 10 percent. If you can use participation to understand a group of your customers, however, you could find attributes known for all customers that predict the identified behavior. For instance,

if analyzing your wiki identifies customers who like your products, you might find other aspects of your customer data, such as buying patterns, that are highly predictive of this behavior. You could then infer that customers showing those buying patterns probably think the same way, even though they aren't participating in the wiki.

> **Use Data Holistically** *Remember that all customer data should be used together. Information about how customers interact with you (by phone, over the Web, in person), the way they change their preferences (and whether they do so), and their participation in social media tells you something about your customers. Using this information to affect how they're treated by rules and in analytic models can improve their experience and help you. The same advice about using data together holds true for other kinds of associates—employees and partners, for example.*

EDM and deploying decision services can complement current IT trends, but it can also help solve some of IT's most persistent problems, as discussed in the following sections.

Solving IT Problems

EDM also has an impact on IT architecture by easing a number of well-established problems, such as maintenance backlogs, channel inconsistency, heterogeneous platforms, and the general level of "stupidity" of enterprise applications. Although solving these problems has value for the whole organization, many IT departments regard these problems as "their" domain, so they are often described as "IT problems."

Ending Maintenance as You Know It

Most IT departments are drowning in maintenance work. This work, including changing systems to meet new or revised requirements, is often perceived as low-value work and a burden on the IT department that prevents it from doing more useful work. However, change is a constant, so maintenance might seem to be, too. After all, despite IT professionals' jokes, most change requests don't come from users too stupid to get it right or from users failing to decide what they want. Most changes are caused by users' business needs changing—new regulations, policies, competitors, products, and market opportunities, for example. Taking an EDM approach to new systems and to modernizing existing applications can dramatically reduce IT's maintenance burden.

An EDM approach allows you to fix the applications you already have incrementally to reduce maintenance. In many applications, the majority of the change requests come in a small area, such as the pricing or eligibility module. In these applications, the majority of the code typically has few change requests and is largely stable. The change requests are usually requests for changes to the business rules embedded in the code. Renovating this one piece by using EDM, developing a decision service to replace it, costs much less than replacing the whole application. The new decision service, built using business rules, will require fewer IT resources to maintain and will allow business users to make many, if not all, of the business changes they need themselves. The application remains stable, because much of the code is not edited, and the robust nature of business rules-based components minimizes the impact of rule changes on the system.

Fixing existing systems is part of the problem, but with EDM, you can prevent the problem from continuing to grow. Using an EDM approach means building decision services that encapsulate rules an application must run. These rules represent the single largest source of change requests, so an EDM approach minimizes the impact of these change requests. Indeed, using EDM to build decision services makes it possible to build changeable applications and design flexibility into the system. By enabling business users to maintain some rules, you can use user-configurable components that require fewer programmers to maintain. These new systems, like your old ones, will be in use much longer than you expect (perhaps 10 to 15 years) and will continue to evolve and change to meet new business needs. EDM can help make sure they don't contribute to your maintenance burden.

Another consequence of neverending maintenance work is that most IT organizations have a huge backlog of projects they're unable to start, let alone finish. With reported maintenance spending sometimes reaching 75 percent of software budgets, perhaps any new projects being taken on is more surprising than having a backlog. The ROI for eliminating maintenance comes in part from your "value backlog."

Projects in your backlog aren't progressing because of a lack of time or resources; other projects have higher priorities. Your organization has probably estimated the potential business value of projects in the backlog. Indeed, most projects don't even make it to the backlog unless they have a positive potential—the project's business value exceeds its cost. So being able to complete all projects in your backlog would add tremendous value to your business. By reducing maintenance work, you free up resources to work on the backlog, and the total net business value of completed projects is your value backlog and represents a potential return on an EDM investment.

Large Field Services Organization: Managing Field Representatives

Old Way

A network of more than 1,000 contractors performed property management activities for the company's clients. In a typical month, 150,000 properties were managed. A largely paper-based process involved faxing and shipping work orders to contractors. Ensuring compliance with federal and state regulations involved manual review of binders full of regulations. Updates to regulations were frequent and time-consuming to make. These regulations influenced what work should be done, the time frame for the work, and payment of contractors. A lack of integration resulted in frequent rekeying and manual review of information in legacy systems. Clients often wanted a customized service, but customizing a client process meant initiating a major programming project and producing specific instructions and workarounds.

EDM Way

A decision service is integrated with a workflow engine and EAI software to support field representatives. Integrated data from several systems is fed into the decision service, which uses the workflow engine to route work orders and contains rules for work order decisions. Some rules implement state and federal regulations, account managers implement other rules representing a client's standard operating procedure, and contractor managers enter rules based on arrangements with contractors. Business users manage all rules. Regulatory, client, and contractor rules are applied in real time to generate instructions and orders for contractors, pricing and invoicing for clients, escalation, and new orders prompted by completing previous orders.

Benefits

- Annual operating costs reduced by $1 million
- Time to market for a new client reduced from six months to a few weeks
- IT support staff reduced from 50 to 5

The Requirements Tar Pit

Much time and money are spent trying to improve the process of gathering and managing requirements for information systems. Widely perceived as a serious problem, requirements have the potential to bog down projects. The InfoWorld 2005 annual Programming Research Report[3] had a "Getting Applications Right" section containing this quotation:

> *"This gap [between user requirements and developer specifications] was one of the two principal [sic] challenges developers complained about in our survey, with 40 percent of respondents reporting that it was a major problem at their site."*

This gap has always been seen as a major cause of project failures. The graph in Figure 10.11 shows that although project failures are in decline overall, failures caused by requirement errors have remained steady.[4] At this rate, we might soon reach the point at which almost all project failures are caused by requirement errors. The rate of change is the prime culprit in this persistent failure to reduce the number of failures due to requirements errors. As Kulak and Guiney say:[5]

> *"The major difference between developing systems 20 years ago and doing it today is that change is much more pervasive now. Changes to business processes and rules, user personnel, and technology make application development seem like trying to land a Frisbee on the head of a wild dog."*

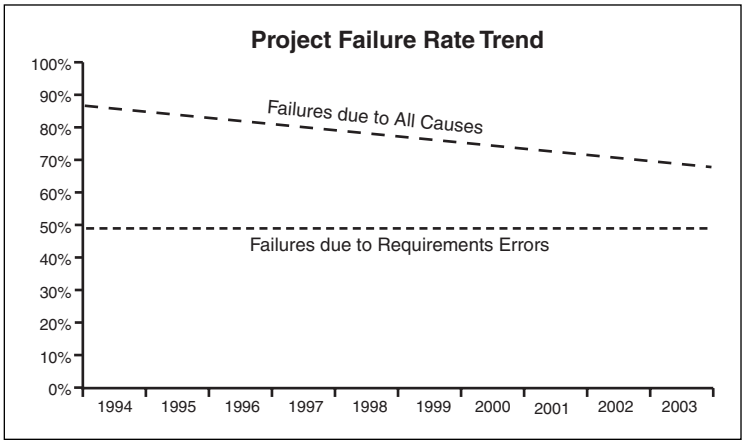

Figure 10.11 The increasing importance of project failures caused by requirement errors, as overall failure rates drop

[3] InfoWorld Programming Research Report, IDG, 2005.
[4] Joe Marasco, "Unraveling the Mystery of Software Development Success," www.sandhill.com, 2006.
[5] Daryl Kulak and Eamonn Guiney, *Use Cases—Requirements in Context,* Second Edition, Addison-Wesley, 2005.

However, not all requirements change more rapidly. In fact, requirements don't change very rapidly at all, but business rules do. Rules are "requirements" only if they are to be transformed into another format, such as a rule about how long to store information. Business rules—pieces of business logic that can be automated—aren't requirements; they're business statements.

Business rules change constantly because competitive pressure, market movements, and regulatory requirements change so often. Requirements for rules in a system never really stabilize. There's nothing you can do to stop changes in the way systems need to work, especially in how they need to make decisions, unless you can stop the world from changing.

If you enable business users to make their own changes to business rules in a controlled environment with EDM, the burden of this constant change can be reduced, however. The IT department's role then becomes one of supporting business users and focusing on technical requirements. So instead of investing in more detailed requirements, IT departments can solve many of their requirement problems if they invest in identifying what the system must do (business rules) and make it possible for business users to create, modify, and delete business rules. EDM, with its focus on business rules to manage this logic, can make a big difference in the scale of your requirement problems.

Channel Consistency

Most organizations today do business through several channels: on the Web, by phone, in person, and through third-party agents. These channels are quite different in form and function and have widely varying levels of automation. Some, such as ATMs, are completely computerized, some are completely manual (personal advisors, for example), and many are a mixture, such as an interactive voice response (IVR) system that can refer you to a person for some activities or a person supported by information systems. The people involved in these channels can work for different employers; when you buy a cell phone plan, for instance, the person selling it might work for the phone retailer, not the company providing the phone service. Some customers always use one channel, but most use different channels at different times or for different purposes.

Different systems support this variety of channels. These systems are from different vendors and often run on different hardware and software platforms. Product pricing and availability, service eligibility, and other decisions critical to customer experience must be delivered to several channels, however. With an EDM approach, you can ensure consistent decision making in all channels without having a single system to support them. In addition, EDM lets you reuse some standard rules and models across channels while including channel-specific rules and models. For instance, you might apply special Internet pricing rules on your Web site in addition to the standard risk-based pricing rules you use in

other channels. A multichannel environment is a reality for most organizations, and EDM can help ensure consistent decisions in all channels. Figure 10.12 shows a typical financial services organization's architecture, where many channels and business processes must deliver consistent decisions.

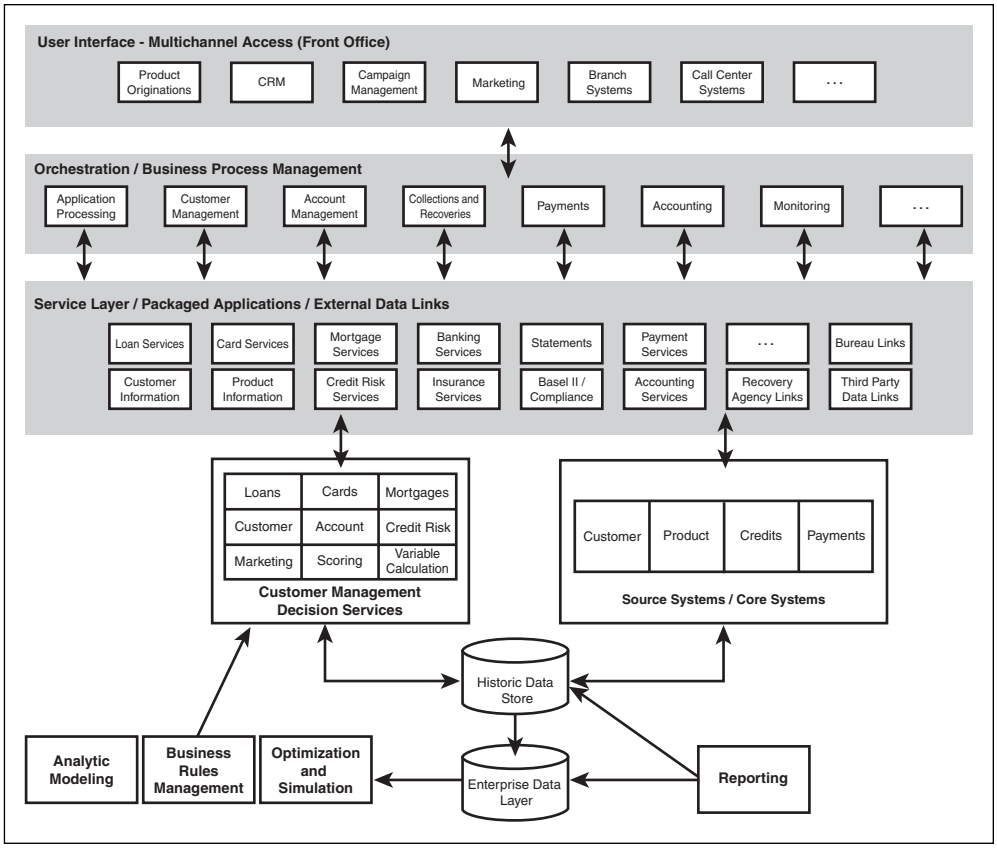

Figure 10.12 An architecture showing the wide range of channels and business processes that need customer management decisions delivered by a core set of decision services

Multiplatform Consistency

Closely related to channel consistency is the problem of supporting heterogeneous platforms yet delivering a consistent experience in systems running on those platforms. A typical organization might have mainframes, UNIX servers, Java application servers, Windows PCs, and a modern SOA. It might own applications written in several languages and use both Java-based and Microsoft .NET approaches. Ensuring that different

platforms run the same logic, calculate things the same way, and can be changed as a set is extremely difficult. Many organizations adopt an SOA approach to address this issue but still struggle with systems that aren't on SOA-enabled platforms or are hard to make available as useful, shared services.

Just as high-maintenance parts of systems can be, and often are, decision services, so are the pieces that need to be made available for cross-platform consistency. With the EDM approach of using a decision service as the point of platform consistency, or even using rules in services to ensure a level of consistency between services designed for specific platforms, you can address the multiplatform reality of your organization.

Commodity Enterprise Applications

Organizations adopting enterprise applications for ERP, CRM, or SFA can find limited opportunity for differentiation in these applications. The standard processes and best practices these applications embody might be efficient, but they are mostly the same for everyone. Enterprise applications are, largely, a commodity. Indeed, Shai Agassi, formerly the president of SAP's Product and Technology group, once asserted that "more than 95 percent of business is common across all companies, in all industries."[6] The 5 percent difference is what provides strategic differentiation.

In a typical business process or application, almost every step has a best practice or template that is widely applied by your competitors. In contrast, how you make decisions in that process is unique to you for the following reasons:

- Customer segmentation is based on *your* data and profiles.
- *Your* policies and procedures are unique, even if they build on regulations and external rules.
- The data used for personalization is *yours*; no one else can duplicate the insight from that data.
- *Your* business users have experience that uniquely informs the rules they write, the way they treat customers, and so forth.

The 5 percent difference between companies *might* not be best represented in decision services, but decision-making approaches certainly aren't common across companies.

[6] Shai Agassi, in a keynote speech at SAP® TechEd 2006.

Resisting Commoditization

The relentless commoditization of processes through process templates and outsourcing makes it harder for organizations to offer unique services. Drill into the problem, however, and you find two kinds of differentiation:

- Process differentiation means performing steps in a process in a radically different order or at a radically different pace. Adopting a build-to-order process could fall into this category.
- Decision differentiation means performing the same steps in a process in a similar order but choosing when to use which steps/branches or whether to price/approve transactions differently. Essentially, the transaction content determines how the process proceeds.

Decision differentiation is a way to give customers different experiences and outcomes yet use a standard process. For example, an organization that approves low-income customers more readily than its competitors represents decision differentiation, even if the organization and its competitors use the same loan paperwork-processing vendor. If the key decision points (approve/decline, complete/incomplete, treat as a good/average/poor customer) in a process are automated by using EDM, changing decision services differentiates the process, even if the process steps are standard.

Automating and improving them by using EDM to apply rules and analytic insights gives you strategic differentiation while still allowing you to purchase commodity components for the other 95%.

The other challenge of enterprise applications is, to quote Butler Group,[7] that "enterprise applications tend to be pretty dumb. They collect data, store it, and produce reports on it." Using an EDM approach to build decision services can make these applications smarter, as in these examples:

[7] Butler Group, "Exploiting Enterprise Applications," 2006.

- Using EDM to improve up-sell decisions in a call center system
- Using EDM to plan maintenance work based on predicted failure risk and maintenance schedules and to use a maintenance, repair, and operations (MRO) system to manage maintenance schedules
- Using EDM to devise optimal staffing plans for an HR system
- Using EDM to generate complex product masters and load them into an ERP system

Automating and managing decisions can make your enterprise applications much smarter.

Large Chemical and Gas Supply Company: Streamline the Supply Chain

Old Way

A largely manual process was used to create materials masters (used in complex supply-chain processes) in the company's ERP system. Because of the high number of products and manual handoffs, the process was slow, which created customer issues (delays to products) and transactional problems caused by inconsistent data. Teams had members from the United States, Europe, and Asia and relied on functional experts to handle issues such as bills of material, costing, inspections, and more. Excel-based forms were reviewed and forwarded manually until enough data was assembled to create a first version of materials masters, and then different groups updated the ERP system's records until the definition of the materials master was complete. Demand for new materials masters was anticipated to grow to 100 to 200 requests per day, resulting in the need to hire 15 more people. The average creation time should have been just 5 days, but it was running up to 45 days, causing unacceptable delays.

EDM Way

All business rules for creating materials masters are captured and stored in a rule repository. These rules are then applied by using a decision service at the point of data entry. The service ensures that valid and complete data is entered, even though different data is required, depending on the type of materials masters and circumstances. Rules are used to derive the right questions to ask to complete the data, based on the kind

of request, its status, and so on. Additional rules are applied in a second decision service to route applications for approval and review using the ERP system's work flow and to ensure conformance with regulations and internal policies.

Benefits

- All work is now handled online, with automatic validation and routing for approval.
- Data is controlled at the point of entry by rules.
- Rework is down and productivity is up, with many requests being completed the same day—a more than 95 percent reduction in elapsed time.
- Knowledge previously held by a few has been transformed into a corporate asset.

Enabling IT Capabilities

EDM helps solve many IT problems inherited from the past, but it can also help the IT department offer some capabilities it's being pushed to deliver in the future, such as support for mobile devices, event-driven and model-driven architectures, outsourcing and business process outsourcing (BPO), performance management, consumerization, and location awareness.

Self-Service

A large part of the workload for many IT departments is building self-service applications in response to pressure from associates. Sometimes these applications are simple forms or reports. Often, however, true self-service applications require decision automation.

IT departments focus on the infrastructure that's needed for self-service applications: portals, intranets, Asynchronous JavaScript and XML (Ajax), forms management software, and so on. However, decisions can be a bottleneck if they aren't automated. If you automate a form to request a service but the request still gets routed to someone's work queue, customers might not think the form is helpful. If your suppliers can request a delivery date extension on your Web site but the decision on allowing it or the cost implications can't be determined immediately, suppliers might phone instead.

By focusing on automating and improving associate-facing decisions, you can make self-service more rewarding and extensive. Customers who want to self-serve appreciate being able to do more without the need to seek approval from an employee. In general, nothing frustrates customers more than not being able to get things done. Creating an "always there for you" environment, in which the company makes rapid decisions about customers' needs, helps lure customers to self-service. Automating decisions helps meet the self-service mandate and in a way likely to reduce the demand for staff. Studies have shown that answering a question online can cost 4 to 40 times less than answering it on the phone. Imagine how much you could save by automating a decision.

Making Mobile Matter

As the consumerization of enterprise IT continues, IT departments struggle with integrating mobile technology into their organizations. Mobile, consumer-friendly technology creates many issues, not just connectivity and security. How do mobile device users want to use them? What access to the enterprise IT infrastructure do they want and need?

An EDM approach to automating decisions can clarify what mobile device users want. When they want to go beyond simple notifications, they often want to make decisions. For instance, when they're notified of a delayed order to a major customer, they can see what options are available for rerouting and pick one, ideally having been recommended the most effective one. Simply displaying user interfaces of existing systems doesn't get it done, however, for these reasons:

- Using BI tools on mobile devices is a problem. How can you "slice and dice" data or view reports on a cell phone?
- Although seeing all notifications on your PC might make sense, you should see only urgent or critical ones on your mobile device.
- How many options should you display on devices? Being able to spot the best one easily is helpful for mobile users.
- Application users might use different mobile devices with varying capabilities, so their opinions on acceptable numbers of options or what requires an urgent notification might differ.

With EDM, an IT department can develop decision services to power mobile applications as readily as enterprise ones, and users and others can interact with the rules for an application, making it possible to personalize and configure mobile devices.

Government Bureau: Ship Inspection

Old Way

The bureau must certify many different kinds of vessels in several locations, so it needs a lot of inspectors who must be trained and given a list of regulations. Manual inspections could take a long time. Consistency and accuracy of inspections were hard to ensure, given the paper-based process.

EDM Way

A laptop-based decision service was created and kept up to date with the latest regulatory and safety rules. Experts who understand the regulations and have experience in inspecting vessels are responsible for entering rules. The service prompts tasks and questions to complete inspections quickly.

Benefits

- Consistent application of inspection rules
- Faster inspections, reduced paper and materials
- Verifying and recording status reduced from 15 days to 1 day

Smart Event Processing and Business Activity Monitoring

IT departments are beginning to transition to an event-driven architecture or at least one that combines service orientation with event-driven styles of development. **Event-driven architectures (EDAs)** allow loose coupling between elements of IT architecture and real-time assessment of how to respond to incoming events. Sometimes called **complex event processing (CEP)**, these approaches analyze inbound events and then prompt a response based on those events. Similarly, organizations are adopting business activity monitoring (BAM) to assess their state in real time and use that assessment to alert people, or occasionally systems, to the need to take action. Organizations using a BPMS or an SOA to manage business processes might also use an EDA to link processes or trigger processes in response to events while processing some events directly. They might also use BAM to monitor processes they have automated.

Telecommunications: Self-Repairing Network

Old Way

All network faults were reviewed manually, and an engineer was dispatched to make a repair. Meanwhile, an engineer in the control center reviewed the existing network and traffic and reassigned traffic around faulty equipment. Major customers with service-level agreements (SLAs) might be affected by a failure, but the company didn't know until the end-of-month analysis, which caused customer service problems.

EDM Way

A decision service handles network errors and alerts without requiring staff intervention. To respond to system failures, the service uses business rules to assign field service engineers based on region, product expertise, and urgency. The rules-based system ensures accurate assignment of field service engineers and allows easy modification and deployment of rules about new engineers or products. The service tracks and correlates systemwide alarm information to determine uptime and downtime for equipment on the network and routes calls through equipment that's running. The rules can also prioritize major customers.

As new equipment is added or company experts learn more about equipment, experts can add new rules to the service. Another decision service determines what compensation is owed to major customers in the case of system failure. The rules analyze system outage information against signed SLAs to determine whether an outage violated the agreement and what compensation is due to the customer. This proactive approach has increased customer satisfaction.

Benefits

- More effective assignment of engineers
- Faster response time to network faults and outages
- Proactive management of major customers

The essence of these approaches is to monitor events within and external to operational business processes. They are often used to link BI systems to everyday operations

more. The role of decision automation and EDM in supporting these approaches is clear. Although these specialized contexts—business process management, event processing, and activity monitoring—can take advantage of rules-based technology, it's not the same as using EDM to support them. Using rules for routing, activity monitoring, or event processing doesn't replace true decision automation.

First, if you embed rules in a CEP/BPMS/BAM solution, you make it impossible to reuse those rules in other parts of this group of solutions and in legacy applications. Rules for what makes a good customer or a fraudulent claim, for example, are the same throughout the enterprise and should be managed as the enterprise asset they are. Managing decisions in this way also allows you to invest in improving them and maximize the ROI. Some rules belong in each environment, but core rules for decisions don't.

Second, rules for routing and handling processes are synchronized with process definitions in the same way that rules for event handling are synchronized with event definitions. Rules for making business decisions in these processes or as a result of identifying an event are independent of the event or process definition. You shouldn't oversynchronize business decisions and processes or business decisions and event identifications. You must be able to change how your business responds to an event separately from how it processes and identifies the event. Similarly, you must be able to change how you manage a decision without having to change the processes that include it.

By adopting EDM, you can make the right parts of your architecture more event- or activity-oriented and still control, manage, and reuse your decisions.

Model-Driven Engineering

Model-driven engineering (MDE) is the systematic use of models as primary artifacts throughout the software engineering life cycle. MDE involves building a platform-independent model (PIM) and translating it into a platform-specific model (PSM) automatically when deployment is required. (Chapter 6, "Business Rules," introduced PIMs and PSMs.) Ongoing maintenance and modification are performed on the PIM, and design is kept separate from architecture and implementation technologies. The design and implementation platforms can, therefore, evolve independently. Design addresses functional requirements without incorporating infrastructure issues, so ideally, round-trip engineering is possible, in which the model is always synchronized with the implementation.

Organizations adopting MDE should also adopt EDM. Not only do business rules represent a way to define business logic in a platform-independent way, but they also can be used in many different projects in a way that domain-specific languages (DSLs) might not. Rule management is also well established, so rules can be shared and modified coherently; most DSLs don't allow effective management of atomic units of logic. Many BRMSs support deploying the same rules to several platforms, which is critical to the MDE

approach, and almost all rule syntaxes are platform-neutral. EDM and business rules extend MDE's power by making definitions of what the system should do more accessible to business users, as described in Chapter 6. Most PIMs aren't user friendly, but business rules are.

Additionally, predictive analytic models normally aren't considered in MDE. Using predictive analytics is important in many decisions, so this omission is a weakness in MDE. By formalizing decisions as separate but equal components alongside other models, you can include analytics more coherently. Finally, decisions can be considered somewhat orthogonal to objects in an object model. With EDM, you can treat decisions this way and still manage them, making impact analysis, for instance, much easier. MDE tends to wrap decision making into objects and lose the capability to manage decisions properly.

Smart Outsourcing

IT departments are involved in outsourcing in two ways: They are often responsible for managing technical interactions with outsourcers running the organization's processes, and they are under pressure to outsource IT development or maintenance work.

BPO vendors can deliver innovation and optimize processes only if they make the key decision points in their processes available so that customers can control them. Organizations have their own way of doing business, and even a standard process varies in decision making that outsourcers need to support. With BPMSs, outsourcers can make these decisions available to clients yet still run standard processes. Although IT departments must care about visibility into processes in terms of data and reporting, they should also insist on visibility in how certain decisions are made.

In addition, for some regulated processes, organizations might be legally required to know the inner workings of a BPO vendor's system. Business rules are ideally suited to showing that a decision has been made in a compliant way. If you're audited on what you did and why, you should make sure your outsourcer is using business rules, and you can integrate their logs about how decisions were made into your own compliance infrastructure.

When outsourcing development, IT organizations worry about intellectual property (IP) infringement. Not all code in a system is core IP, but some is. Outsourcing development and maintenance of nondifferentiating code can be cost-effective, but outsourcing code representing your IP is a big risk.

The code that business rules replace is core IP, so one option is to develop core IP by using business rules, focusing on ownership and reuse of this IP, and to develop other code with the most inexpensive resources or tools. This way, business contributors to core IP can manage it directly, and you can still take advantage of low development costs for other code. Working with an outsourcer to take control of an existing system—while reengineering key business logic into a decision service that can be managed in-house—can increase cost and efficiency savings and business control.

Global Logistics Company: Freight Management

Old Way

The company's clients wanted to customize their shipping requirements, but often divisions of the same client had different freight-forwarding methods, for instance. Allowing clients to control freight management required custom-built applications modified for each customer whenever customers requested changes.

EDM Way

A decision service handles data validation, event management, and logistics logic. The company's experts created baseline business rules, and each client sets up its own additional business rules. Clients can modify their rules directly within the bounds set for them. The complete set of relevant rules verifies each shipment as it's entered, based on the client's shipping requirements, goods requirements, and so on. Business rules for personnel allocation and material configuration are also applied to automate the shipping logistics definition.

Benefits

- Flexibility and customer satisfaction increased
- Reduced IT costs

Corporate Performance Management

Corporate performance management (CPM), an important initiative for many IT departments, is a systematic, integrated management approach to linking enterprise strategy with core processes and activities. By providing planning, budgeting, analysis, and reporting capabilities, CPM lets you run your business "by the numbers" and use measurements to make management decisions. CPM typically takes the form of dashboards and other visualization tools for past results and perhaps trends and predictions, and these tools are designed for analytically minded people. What CPM doesn't do, in general, is help you take action in response to information. Many organizations use BPMSs for this purpose; when their CPM systems tell them things are going poorly, they can use their BPMSs to adapt and change processes to respond. When things are going well, they can use BPMSs to standardize best practices.

Software as a Service and EDM

One form of software as a service (SaaS) is called a decision service provider (DSP), which offers on-demand delivery of analytic business decisions. In addition to the usual SaaS capabilities, a DSP also needs to

- Build and maintain analytic models and business rules
- Pool third-party and customer data to build models
- Access third-party information at runtime
- Ensure that regulations and industry guidelines are followed

A DSP offers value in part because many possible data sources are available for decisions, and an organization might have difficulty orchestrating access to them without a DSP. In addition, analytics and decision making can be based on pooled data and expertise rather than an understanding of the organization's own (potentially limited) experience. DSPs are already common in areas such as fraud, but they could become important in areas such as marketing, where shared data sources (such as customer panel data) can play a role in retailers' and Consumer Packaged Goods (CPG) firms' decision making.

EDM adds value to a CPM/BPMS environment when you start thinking about decision change rather than process change. For example, suppose your CPM environment tells you that bad debt in a customer segment is rising. The reason is unlikely to be a process—after all, you have standardized the process—so it's probably related to decisions for how you segment and treat customers. If these decisions are manual, managing them effectively is hard, because you must find all those responsible for the decision and retrain them. If you have automated them in your BPMS, however, you probably don't have as much control over the decision as you would like, because BPM environments are centered on processes, not decisions.

With EDM, you could develop decision services and make them readily available to processes implemented with a BPMS and to other legacy environments, Web sites, partners, and so on. By using business rules to develop these services, you can ensure that the business users who have access to the CPM environment could manage the rules to eliminate the time lag between CPM insight and BPMS action. What's more, as your CPM environment helps you learn more about your business, you can identify the potential to use embedded predictive analytics. If your CPM environment shows you how to predict

that a customer has a high risk of cancelling her service, you could develop a model for this purpose and embed that prediction into decision services. Then your operational decisions and hence your operational processes are run by the numbers. Using EDM to automate key decisions acts as the glue between the insight CPM provides and the BPMS/SOA environment.

Extending Your Software Development Life Cycle to Support EDM

Most IT organizations have a preferred software development life cycle (SDLC) approach as well as a formally documented methodology or group of methodologies. To reflect EDM adoption, these methodologies need to be adapted to include business rules, analytic models, and adaptive control. This section contains some general notes on these approaches, as well as specific guidance on adapting Rational Unified Process, Unified Modeling Language (UML), and agile methods.

General Approach to Adapting an SDLC

When you're adapting an SDLC to include EDM, you need to consider four areas: identifying and managing decision services, managing business rules separately from requirements and as a model, integrating analytic models, and recognizing the impact of adaptive control.

Decision Services in an SDLC

Adopting EDM means building decision services and managing them over time, so this means treating decision services as a class of components or services in your SDLC. The classification scheme and design guidelines you use for services or components should be extended to include a decision service type, and you should document guidelines for creating and managing them.

In many ways, decision services have the same characteristics and requirements as other business-focused services or components. In addition, you need to document how rules are maintained and what rules in the service might be shared among decision services. Release management tasks should be updated to include releases that just involve an update to logic in a decision service and to handle updates to shared decision services.

> **Finding Decision Services** *Your methodology should include activities to analyze reporting, worklist, and other services to see whether you have hidden decision services. Becoming more decision-centered should be a long-term goal of any methodology change.*

Rules in an SDLC

The most important change to your SDLC is the separation of rules and requirements. Most methodologies don't make this separation, so rules and requirements are intermingled. This mixing is always a bad idea, but it's particularly unhelpful when you're managing rules as explicitly as you do in EDM.

The first step is managing "source rules" and mapping them to the scenarios and design requirements you're documenting. Source rules are generally written in the language your business users use—in terms of being in English, for instance, and using business terminology. These rules are designed to be atomic and manageable, but they can't be implemented. Effective management of terms and vocabulary for these rules makes checking them and mapping them to analysis artifacts easier. Source rules should usually be kept general. Some source rules become production rules or sets of production rules; others map to templates for production rules. For instance, a source rule might describe typical knock-out (decline) rules for an applicant, which map to a rule template that constrains a rule set for eliminating applicants.

The second step is managing platform-independent but object-specific rules for your decision services. As you develop your object or data model, you can start to develop rules that run against that data. These rules must be managed in a repository, versioned, and structured to allow for reuse in other decision services. More production-oriented rules map back to source rules, although not usually in a simple fashion, and are collected into rule sets.

> **Rule Maintenance** *You need to develop a separate rule maintenance process outside your regular SDLC, especially if business users maintain some rules in your decision service. Although you want to go through many of the usual test and QA steps, realistically, you need a different process, because the time frame is shorter and changes are more frequent than in a typical IT project.*

Logging rules as they execute to create a record of how a decision was made is a common requirement of decision services. They can be managed like a typical logging requirement, except that using business rules makes it easier. Using business rules also makes it possible to use the logged information in both customer-facing and regulatory conversations, because the rules are more user-friendly. Figure 10.13 shows a real, and typical, architecture for decision services with a strong compliance requirement. The architecture manages development and production environments, logging, reference data, and test scenarios to ensure that compliance can be demonstrated.

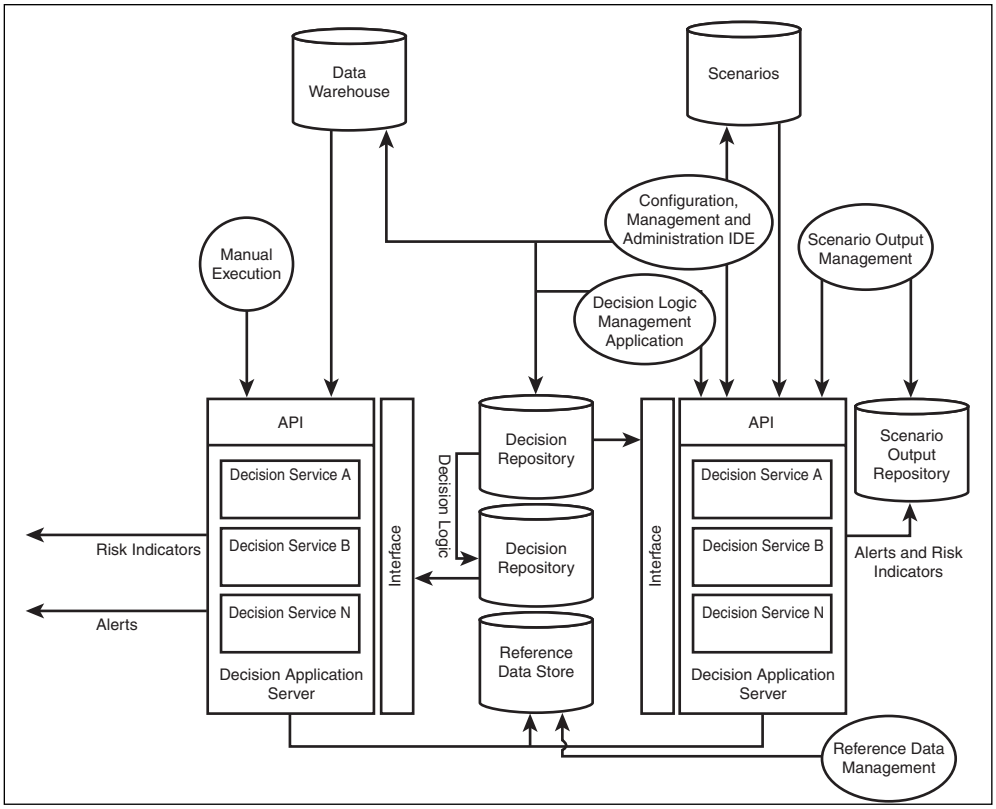

Figure 10.13 Production (on the left) and development environments (on the right) for decision services handling regulatory compliance, showing scenarios, reference data, and alerts for compliance follow-up

Analytics in an SDLC

Most IT departments have made little effort to integrate analytics into their SDLCs, even in organizations that use predictive analytics. The need to write code to implement a predictive model has often meant that a predictive model was regarded simply as a kind of specification, and a complete programming project was followed to implement it. As predictive analytics become more automated in your organization and you use them more, this viewpoint needs to change. The need to synchronize models and rules also motivates change in an SDLC, because previous projects are likely to have used a model simply as a block of code.

Besides changing how model deployment is handled in the SDLC, IT departments must consider how predictive models can enhance their data or object model. The methodology should encourage the possibility of adding an attribute that predicts something about an object—retention risk for a customer or likelihood of return for a product, for example. The full benefit of predictive analytics can be realized only if those developing models for the operational system start asking these questions and working with analysts to understand what's possible.

IT departments must also be aware of additional data requirements that come with a focus on predictive models. In particular, you might need to keep and make accessible far more historical detail and more specific time-series data. Additional requirements for reporting and monitoring as well as data access at runtime are also likely. Without meeting these requirements, organizations can't ensure that these models can be improved over time.

Finally, most organizations that make extensive use of EDM and predictive analytics also use external data at some point. Integrating external data at runtime and delivering historical external data to modelers are additional considerations.

Adaptive Control in an SDLC

The use of adaptive control in EDM has an impact on testing and operations. Several versions of rules and models need to be assembled and tested, often in parallel. In addition, the operational system runs some transactions through the champion approach and others through challengers. This testing requires understanding the adaptive control approach on the part of those developing and operating the production application. Adaptive control requires logging information for subsequent analysis, which affects data storage and APIs for the operational system.

The development process, when it's adapted to rules and analytics, also needs to support developing champion and challenger approaches simultaneously. It must also allow developing new challengers for a decision service already in production, promoting a challenger to champion status, and adding and removing challengers. Developing champions and challengers isn't a full-blown development project; it's a smaller, more focused effort. It does affect a running operational system, however, so you should schedule, manage, and track them.

Rational Unified Process and UML

Rational Unified Process (**RUP**) and, more recently, **Enterprise Unified Process** (**EUP**) are methodologies designed to apply UML and development best practices in a formal way. In the past, these approaches haven't managed business rules in a way suitable for EDM but have considered them part of use cases or requirements. Generally, a more

managed approach to business rules as separate artifacts is required for EDM, but adding this approach is easy.

RUP outlines six best practices, and business rules clearly support all six:

- **Develop software iteratively**—Declarative business rules are separate, testable decision-making units. EDM allows incremental development and testing of business rules, separate from the other software components. Software development is divided into manageable and measurable changes.
- **Verify software quality continuously**—Business rules are easier to verify, because they are close to the business. Rule management encourages constant validation and verification, which improve software walkthroughs and reduce test cycles.
- **Control changes to software**—Traditional software systems are "brittle"—meaning small changes can break them—but change is guaranteed and must be managed. Business rules are expected to change and can be traced to software more easily and updated separately, even in 24x7 environments.
- **Manage requirements**—Business rules shouldn't be managed like requirements, so separating them from requirements makes rule management and requirement management easier. Rule management replaces some requirements management with the management of explicit business rules, throughout and beyond the application life cycle.
- **Use component-based architectures**—Identifying and managing decision services separates business logic into reusable, manageable components.
- **Visually model software**—If a picture is worth a thousand words, business rules could be worth many lines of code. Business rules and their interrelationships can be visualized and managed more easily than code, so more of the application can be managed more visually.

Integrating Rules

To bring business rules into RUP, several changes are required:

- Add a business rule-modeling activity alongside business modeling that emphasizes collecting and organizing business rules. This activity replaces a single focus on use cases with a more balanced view of use cases as a means of rule discovery. However, it doesn't provide a place to manage them (see the section "The Requirements Tar Pit" in this chapter). This activity also has highly compressed interactions between rule modeling and analysis, design, and implementation.

- Formally identify decision services as components that implement business rules.
- Consider additional software components and processes for future rule maintenance and testing.

In UML, business rules must appear at three different levels: business model or computation-independent model (CIM), PIM, and PSM. At the business model/CIM level, you need an unambiguous representation of business policies, procedures, and constraints as business rules in natural language and independent of assumptions about the platform on which an information system is delivered. At the PIM level, you need a representation of business rules targeted to a business rule engine—perhaps a format that's independent of a particular vendor.

Modeling business rules in UML is a broad topic, as you can see in these examples:

- Business rules defined as formal texts for documentation or requirement purposes are covered by the SBVR standard.
- Business rules, such as simple data relationships and constraints, have simple counterparts in a UML model. For example, the business rule "Orders must have at least one line item" is typically represented as a multiplicity constraint on an association. Other rules translate into constraint expressions easily. For example, the rule "An account can never have a negative balance" can be expressed as an invariant by using Object Constraint Language (OCL).
- Process-oriented business rules define conditional action, behavior, and state changes—those that OCL doesn't handle—as production rules. Business rules expressed by production rules are common, and representing (modeling) production rules in UML isn't standardized.

The two UML mechanisms for defining constraints and behavior are OCL and action semantics (AS). However, neither provides an out-of-the-box solution for representing production rules.

Integrating Analytics

Analytic models and their integration into systems are a good conceptual match for the model-driven development promoted as part of a RUP/UML approach. That said, little or no clear guidance for bringing analytics into a UML model is available. Object attributes can be derived analytically, and model components can be manipulated in an interaction diagram to show when a real-time scoring service is used. These methods have no specific support in RUP/UML, but you can manage them by using existing artifacts.

Agile Approaches

Describing agile approaches in detail is beyond the scope of this book. The basic principles of iterative development, collaboration, developing working software regularly, and test-driven design are well known and well documented. Using agile approaches with EDM requires integrating rules into the process, integrating analytic modeling, and considering adaptive control.

Rules in Agility

Two key agile principles[8] apply to EDM:

- Welcome changing requirements, even late in (or after) development. Agile processes harness change for the customer's competitive advantage.
- Business users and developers must work together throughout the project.

Applying these principles can improve business agility by making sure you respond to changes that are needed because of a changing environment or competitor. There's a challenge when a new requirement isn't really a requirement but a new or changed business rule, however. For instance, new legislation might force new requirements on a project, but it's more likely to generate new rules. Business rules aren't the same as requirements, so your agile development processes must work as well for business rules as they do for other requirements. Furthermore, a core practice in agile modeling is **single source information**—that is, information required for development should be recorded once in a suitable artifact. Business rules should also be treated this way.

Scott Ambler once said that

> *"Agile software development teams embrace change, accepting the idea that requirements will evolve throughout a project."*[9]

To embrace change in this way, agile projects focus on the highest-priority requirements in each iteration. To manage business rules separately from requirements, you must also manage business rules alongside your other requirements. You use three key artifacts in this process: decision services, rule templates (or definitions of kinds of rules), and actual rules. These artifacts usually require business users, developers, and business analysts to work together. The best way is for developers to define decision services, developers and business analysts to define rule templates and how they fit into processing the

[8] Agile Alliance, "Manifesto for Agile Software Development," www.agilealliance.org.
[9] Scott Ambler and Ron Jeffries, *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*, Wiley, 2001.

decision service, and business analysts and business users to manage rule instances. This method supports agility, because rule instances reflect the business users' understanding of the business at that time and the target range of test cases.

Test-driven development means writing the test before the program—thinking about how to use the component or what it's for *first,* and then thinking about how to implement it. For developing a decision service, the agile approach of test-driven development is ideal. Most decision services have a simple interface—a modest amount of data is passed in, and a simple answer is passed back. All the complexity is inside the decision service, so developing a test or set of tests for the decision service when it's first defined is straightforward. Business users can provide real examples and explain what decision was made in each case, which then becomes a test case. Typically, only a core set of cases is automated to start with, as described previously, and new test cases must be known before new rules are written (because rules are defined to handle specific cases not handled before). Therefore, adding test cases before making changes is completely logical.

The process of defining templates is iterative, and the process of defining rule instances is highly iterative, so these processes are well suited to agile approaches. An important benefit of using business rules in this way is that real collaboration is possible. When rules are written in a syntax everyone can understand, agreeing on them and editing them are much easier. In addition, domain experts can develop rules independently, so some change management during an iteration is in their hands. They can change as quickly or slowly as they like, and cost is controlled because there's a defined difference between change within expected boundaries (requires only new or changed rules) versus unexpected change (requires new decision services or templates). Modifying a system means targeting a certain percentage of decisions and then gradually increasing the percentage by automating exception handling, identifying patterns, and so forth.

By taking this approach, essentially you record a business rule once as human-readable but implementable code so that you can reuse it. This method helps improve business agility and matches the Agile Alliance's Manifesto for Agile Software Development, which has these key tenets:

- **Individuals and interactions over processes and tools**—A key interaction is between developers and business users. Using business rules facilitates this conversation.
- **Working software over comprehensive documentation**—Business rules can deliver working software that's easier for business users to read and change, which makes software more "self-documenting" and lessens the pressure for documentation.

Pair Programming for Rules

An interesting approach to bringing rules into an agile development process is "pair programming." In this method, rules are created by a pair—one member from IT and one from business. Because a rules environment allows both technical and business users to read and write rules, this kind of collaboration is possible. For rules that are too technical for business users to maintain on their own yet require a high degree of business content to manage effectively, pair programming can be successful. Building a majority of your rules quickly with this technique is often possible. Organizations unwilling to go this far might still find value in pairing a technical rule developer with a "pure" code developer.

- **Customer collaboration over contract negotiation**—Both developers and business users being able to read and understand business rules allows true collaboration in implementing business logic.
- **Responding to change over following a plan**—Business rules provide business agility by making the code you write easier to change during and after the project.

Analytics and Agility

Both analytic development and agile development are highly iterative. Most analysts developing models develop a "working" model at the end of each iteration, but it might not be particularly useful, because it generates poor lift or worse lift than a previous version. Actually deploying and integrating a model each time might not be worth the effort. The most effective way to bring iterations together is after the decision service has stabilized in its integration with the rest of the system. At this point, subsequent iterations involve only changes to rules and/or models in the decision service, which maps well to analytic iterations in model development.

Test-driven development is by far the most common approach to building analytics, with validation and test data sets created along with training data sets as part of the core process of developing a model. While the analytic staff focuses on these data sets, IT staff might find it helpful to similarly define their tests for the deployed model.

Adaptive Control and Agility

Adaptive control should be integrated into agile approaches by ensuring that part of iteration after initial deployment is a focus on the champion/challenger strategy. This strategy might run at a different pace than more technical iterations because you might have a noticeable lag time before results can be measured and compared between the champion and challengers.

The EDM Wiki

You can find more information on how EDM fits into the IT ecosystem in the EDM wiki at www.smartenoughsystems.com.