Modeling: The key to successful integration

by Magnus Christerson
Product Strategist
Rational Software
IBM Software Group

Modeling can play a central role in business and application integration. This article begins by outlining some of the problems inherent in integration, and explains how modeling can help address them. It then examines how the telecom domain has used modeling to achieve integration and interoperability industry wide, and how other domains can benefit from the same practices.

Integration: What does it really mean?

Why do so many application integration projects fail? Some get bogged down in trying to overcome technical difficulties in the systems undergoing integration; others try unsuccessfully to adapt business processes and/or data across systems before the systems are equipped to handle the changes. In such cases, the business value of the project -- in terms of either time-to-result, quality of result, or return on investment -- then becomes questionable. In fact, the failure rate for such projects is so high that typically, even many years after a merger takes place, the original companies’ software support systems remain unintegrated.

With the pace of mergers and acquisitions today, this situation is simply not acceptable in terms of both cost and business efficiency. Application integration projects need to deliver business value quickly by enabling superior, autonomous business processes -- that is, business processes that can stay competitive and adapt in an ever-changing world. For e-business on demand, this quality becomes critical.

Integration is fundamentally a business problem. The experience of one of our customers, a large global bank, underscores this point. The bank's business strategy is to achieve global growth through acquisitions, and it
The Rational Edge -- October 2003 -- Modeling: The key to successful integration

has become very good at it; over the last six or seven years, it has acquired one or two new banks each quarter.

At first, however, the bank's IT people estimated that even the most aggressive systems integration project following an acquisition would require six to twelve months of work. And that would accomplish integration of only the most business-critical systems from the newly-acquired bank. Our customer's executives soon realized that IT integration was creating a bottleneck in the bank's growth strategy. And, as you can imagine, the CIO certainly was unhappy about having to tell the business executives that they simply couldn't grow the business as fast as they wanted to. So what did IT do to get around this problem?

Well, their solution was very pragmatic. When they acquired another bank, the first thing they did was to make models of the acquired bank's systems. They mapped to the most detailed level whatever customer and other business data resided in those systems. Then they did a full-fledged migration of the acquired bank's business data into their own systems. Of course, since they were managing customers' money, they were extra careful not to mess up and lose a checking account here or there. In fact, over the last few years this bank has become so systematic and efficient about their integrations that they can now accomplish them in about six weeks. Today, they are capable of handling one or two such integrations every quarter. So now, it's the business people, not the IT people, who are responsible if a bottleneck develops in the business growth process.

As this example from the banking industry shows, common agreement about what the term integration actually means is central to an integration project's success. Before beginning the project, the CIO and other business executives must reach consensus on questions such as "What do we actually mean when we speak of integrating one system with another?" and "What do we want to achieve with this integration from a business perspective?" To succeed with an integration project, the IT group must understand in painful detail what the data in all systems actually mean, how the business processes that use the data actually work, and how the interfaces of all the systems can interact. This requires that they create accurate models. The business results an integration project can deliver will be only as good as the IT team's understanding of the desired business outcome. Simply plugging two systems together naively -- without modeling and careful mapping -- almost always leads to disaster.

Lessons learned from the world's biggest integrated system

Of course, integration is not a new computing problem. The world's largest integrated computer system comprises more than three billion integrated systems. What is it? Our global telephone system. This system has two billion standard telephones and another billion cellular phones, plus fax machines, voice mail systems, billing systems, and the entire Internet! Add twenty or more different technology standards bodies (each with hundreds of standards), legacy systems that are up to fifty years old, and more than a thousand competing vendors with products they want to integrate, and you have a nice little integration problem.
The most amazing thing about our telephone system is that, despite this staggering complexity, it actually works. And it's quite reliable, especially compared to a standalone PC with one CPU. How often does your phone go down or require a reboot? Furthermore, it has evolved across new technologies, new standards, and new vendors. It has survived both disruptive technologies -- fax machines, cellular phones, and analog to digital conversions -- as well as evolutionary technologies. These include the journey from pulse dialing to tone dialing, and from GSM (Global System for Mobile communication, the world's most widely used cellular system) and GPRS (General Packet Radio Services, a mobile standard for data services), to the 3G (third generation) movement, which gave cellular phones enough bandwidth to receive and display live video. The systems that are connected to these new technologies still continue to work.

Why has integration worked so well in this industry? If we study the industry's characteristics more closely, we can discover some answers.

First, the *business value* of the technology (or user value, if you think about it that way) *lies in integration*. The telephone system lives by integration, and integration is crucial for its operation. If a new phone technology does not work in integration with previous phone technologies, it has very little value. You might develop the coolest new phone, with super features -- crazy things like calendars, cameras, and mp3 players -- but if you can't call anyone who does not own exactly the same phone, then you will not have a successful phone business. You might have a successful PDA business, but then drop the phone feature, please. This imperative for integratable products is, I believe, why the telecom industry is the most advanced systems integration marketplace in the world.

Second, telecommunications has a *long tradition of standards*. Making this vast system work requires standardization across a number of dimensions. Standards organizations in the telecom business are very mature, and the standards are thoroughly detailed and precise. It's not uncommon for a new technology that requires standards, such as GSM, to involve dozens of thick binders with very detailed specifications. Most of the standards also come with highly detailed test suites to validate whether an implementation actually conforms to the standard. These test suites become part of the acceptance tests for operators. And of course, competing vendors try to outmaneuver each other in working their particular products and technologies into the standard.

Third, it's true that telecom has one of the most complex technology domains around; systems are inherently *distributed and heterogeneous*, combining the characteristics of both (hard and nonhard) *real-time systems* as well as massive *data processing applications* (e.g., billing). And they combine synchronous and asynchronous behavior. However, these systems must also meet extremely difficult requirements for continuous *availability* over very long periods of time. These availability requirements place profound limitations on executing upgrades. For example, for normal IT applications, availability measurements allow planned down time for maintenance. But a typical telecom switching system might have an acceptable down time of perhaps thirty minutes.
over a period of ten years! When telecom people talk about "five nine availability" (99.999% up time) they actually mean it; there is no such thing as "planned down time"! These availability requirements are one aspect of what is referred to as Quality of Service (QoS). Various QoS standards are widely adopted in the telecom industry to specify what level of service must be provided. Other QoS metrics in the telecom industry include things such as response time and sound quality.

Fourth, the telecom industry demands that all its technology must have a **global reach**. The expectation is that a call made on a phone served by one telecom company can, without any problem, reach a customer on a different type of phone supported by a different type of network, and provided by another telecom company (that might be a competitor of the first company) -- perhaps even in a different country in another part of the world! Although a myriad of changes are constantly happening in this web of systems, it is still operating with a very high quality of service. And ever since telephone service has become a commodity, customers have been choosing vendors by price and quality of service, which have become unique business differentiators. For customers, the value of a telecom company lies in its ability to supply global reach with high quality of service.

Fifth (and this is why this example is relevant here), how do telecom firms ensure that their technologies can meet such stringent demands? The answer is simple: **They make more advanced use of modeling** than businesses in any other industry, and they always have. Many features in the UML, such as sequence diagrams, use cases, state machines, and subsystems, were adopted from existing practices in the telecom domain. Furthermore, to ensure that they meet standards and achieve total interoperability with other systems, virtually all telecom manufacturers and operators use modeling tools. Many of these organizations are so advanced that they generate up to 100 percent of their systems using models. Also, virtually all the standards in the telecom domain use modeling languages for precise specifications. The telecom industry has been divided by two mostly competing standard modeling languages: the UML, which emerged during the last ten years; and SDL, which has roots back in the 1970s. With UML 2, we are trying to bridge the gap between these two languages so that UML 2 can replace SDL and become a single standard for the industry.²

**Changing demands in other industries**

Admittedly, telecom is a very special case, and telecom manufacturers have always built special systems to support the industry's extreme requirements. But what about other domains, such as financial services and health care? Let's look at how the emerging world of e-business on demand is bringing about changes in every domain that parallel characteristics of the telecom industry.

The first change is that business processes are becoming more dynamic and more automated. And, as in telecom, increasingly the *business value of their systems depends on integration*. This means integration across business processes: from suppliers to providers, customers, partners, and
various stakeholders (sometime invisible) who participate in a global business process. For example, if I can pay bills or maintain my investments using my bank's Web site, it not only cuts operational costs for the bank, but it also increases my customer satisfaction. I get 24X7 availability for business transactions on my terms, and through a higher-bandwidth medium than the telephone. Integrated business processes are also increasing both efficiency and customer satisfaction in the healthcare industry. If my physician refers me to a specialist and suggests a medical service such as an MRI scan, many medical care centers now have the capability to transfer my patient records between facilities in a secure way.

Another change relates to industry standards and regulations. We simply assume that mature industries have them. But in contrast to telecom, in many industries these standards do not directly target the integration of business systems; instead, they focus on regulating business practices or legal affairs. But this is rapidly changing. We are seeing a dramatic increase in the formation and activities of standards bodies, especially around systems integration using XML. Today there are more than 400 standards bodies working on XML alone! And although business integration standards have existed for a long time, primarily around EDI (Electronic Data Interchange) and supply chain integration, standards activities over the last few years have addressed many other integration concerns and new technologies. For example, the banking industry for many years has had SWIFT, which standardizes and operates the network for interbanking transactions and handles the transfer of billions of dollars every day. Today, SWIFT is taking action to leverage XML (and UML). In the healthcare industry, HL7 (Health Level 7), which has long standardized things like patient records, is also adopting XML (and UML).

Business integration is also becoming increasingly complex. Not only must we assume that most companies have distributed systems as opposed to a central data center, but also, since we can't make assumptions about what other vendors and partners are using today or who they will be tomorrow, we do need to assume that we will live by heterogeneous systems. Emergent technologies such as XML and Web Services are helping to breach the gaps between systems by providing a common, widely-accepted and deployed protocol.

As in telecom, availability requirements are also increasing in the business domain, as the business processes we support become more business critical. For example, in the world of e-business on demand, a warning message such as "Planned down time between 8 pm Saturday and 6 pm Sunday" for a Web Service is simply unacceptable.

And finally, increasingly, the business processes of most companies must have global reach. Today, companies are heavily outsourcing and "offshoring" not only development but also services. This trend means that soon, businesses will not have control over an entire automated business process -- whether it is inside a firewall, available through an extranet, or even within their own organization. Automated business processes will flow across companies and institutions, and across geographic boundaries. Companies will, as always, gain competitive advantage through pricing as well as the value of their business processes. However, as business
processes become more automated and increasingly similar, what will differentiate one company from another (as in telecom) is the *quality of service* it can offer. Why is amazon.com still outselling buy.com in electronics and barnesandnobles.com in books? Buy.com and barnesandnoble.com are clearly cheaper. Not only does Amazon have a stronger brand presence on the Internet, but also their quality of service is superior.  

As other industries take on some characteristics of the telecom industry, will they also become heavy users of modeling? Today, although modeling is used in all types of industries, the practice is not as advanced or widespread as what we have observed among our telecom customers. Let's take a look at how other industries can leverage modeling to greater advantage.

**Technology challenges and the advantages of modeling**

Every business integration project faces the challenge of accommodating constantly evolving technologies. Whether the project is dealing with message-based systems such as IBM WebSphere MQ or Microsoft MQ, whether it involves an integration server from a vendor such as Vitria or WebMethods, or whether the project is based on XML or Web Services, the project team can assume that the technology selection will change over time.  

*The best way to handle these inevitable changes is to do what the telecom industry has done for decades: Create an integration model at a high level of abstraction that you can bind to a variety of technologies.* Modeling allows you to focus on the semantics and design of an integration as opposed to the technology constraints. Don't get me wrong; technology constraints are real and need to be dealt with. But putting them in the driver's seat is like putting the wagon before the horse. Instead, if you can decompose business processes into logical services, these services will not only be well designed, but they will also be resilient to technological and architectural changes. Telecom systems have shown this over and over.

So how does modeling help you solve integration problems? Mainly, it helps you understand very clearly what systems and interfaces are involved in your integration, and how they will interact.

If you look at most systems or applications today, at best they have an interface; sometimes they don't even have that, which forces you to try "back doors" when you integrate, such as going directly into the database itself or manually retyping information into the system. But let's assume that you *do* have an interface. Such interfaces often take the form of a set of function calls that more or less make sense to you. Of course, these interfaces can be expressed in a number of technologies, some proprietary, some not. These can be IDocs (Intermediate Document interfaces) or BAPIs (Business Application Programming Interfaces) for an SAP system; they can be CICS (Customer Information Control System) transactions for mainframe systems; and they can be various RPC
Remote Procedure Calls technologies using various Interface Definition Languages (IDLs). CORBA (Common Object Request Broker Architecture), Java, COM (Common Object Modeling), and .Net all provide ways to describe these interfaces. Today we are all trying to figure out how to read WSDL (Web Services Definition Language) specs. Who knows what it will be tomorrow?

Now, these interfaces and APIs (Application Programming Interfaces) are often very complex and difficult to make any sense of. That is why many project teams hire an expensive expert consultant who has "done it before." Just look at Win32, the Microsoft Windows API. It has thousands of entries, created over the last ten years by multiple teams. Of course, it has redundancies -- many ways to do a certain thing. And what the telecom guys figured out decades ago is that a simple interface description list is often the root cause of misunderstandings about getting systems to work well together. So they created sequence diagrams, protocol definitions in state machines, and other techniques to show more clearly how a certain interface was to be used in a certain context.

These techniques make it a lot easier to describe what interfaces and systems you are going to integrate, and how one system will interact with others. And the nice thing about these techniques is that they scale well. You can use them to describe very high-level interactions, such as how the systems will engage in a certain "conversation." But you can also describe the lowest level of detail, showing exact parameters and their error conditions.

That is what a group of high-tech vendors called RosettaNet (www.rosettanet.org) did recently, in an effort to figure out how to integrate their supply chains. How does Dell order processors from Intel? How does Microsoft give Ingram Micro catalog information? These issues often relate to huge orders that have great business importance as well as serious legal implications. "I ordered 20,000 processors but never received them!" Dell might complain. Or Microsoft might declare, "No, I never got that order, so you are on your own if you missed the holiday season!" So of course, if you go to a RosettaNet meeting, you will see lawyers there.

But the technologists are there as well, and they have done impressive work. They have mapped the highest-level business processes relating to supply chain management, such as placing an order, all the way down to the actual XML tags that are sent as part of the order message. And they have specified everything in a big, detailed IBM Rational Rose® UML model that companies can map to their specific technologies. Now, most integration servers ship with built-in RosettaNet processes (called Partner-Interface-Processes, or PIPs) based on this model, and these processes are being deployed worldwide in the high-tech supply chain. If you wanted to do business with Intel, for example, these processes would provide a standard way for you to plug in. Last year, Intel transacted more than $5 billion worth of business electronically using the RosettaNet standard: $2 billion in purchases and $3 billion in sales.

Conclusion
The key to a successful integration project is a deep semantic understanding of how to integrate the set of interfaces that go along with the systems -- which means understanding the business processes, and not just the technology. And that requires creating models that raise the level of abstraction.

New technology platforms should solve more problems than they cause, but even the best of these platforms do nothing to clarify the semantics, or meaning, of integration. That's where modeling comes in. Business value lies in understanding, in a detailed, often visual way, what is required to integrate systems and their interfaces. Technology platforms may evolve, but business requirements stay surprisingly constant. Just think back to our discussion of the telecom industry: A phone call today is very similar to what a phone call was fifty, or even a hundred, years ago. Although underlying technologies will continue to evolve and change, the basic imperatives for most businesses will be the same: Deliver high-quality goods or services at the right cost, and at the right time. Modeling can help businesses both adapt to new technologies and continue to meet those demands.

**Note:** This article is an expanded version of a talk given at the Integrate 2003 conference.

---

**Notes**

1 A [hard real-time system](#) requires that every response time must be below a certain threshold. A [nonhard real-time system](#) requires that the average response time be below a certain threshold. For example, a hard real-time system might require that a connection must be established within 100 milliseconds for a switch to meet its capacity requirements; a nonhard real-time system might require that a dial tone be provided within one second on average.

2 In fact, the UML can be directly traced to SDL. The UML is in many ways a modernized version of SDL that adds support for object orientation and generalizes the full language. In the UML 2 proposal, we walk the final mile and add a few last pieces that allow us to fully replace SDL. This includes concepts such as structured classes and many small but powerful features in sequence diagrams.

3 The telecom industry measures quality of service by looking at parameters such as availability, response time, sound clarity, and coverage. On the Internet those characteristics are still important, but the Web also offers a number of other ways to increase the quality of service. For example, the "other buyers' comments" feature is a big differentiator for Amazon; 24x7 telephone support is a big differentiator for companies such as E*Trade and Wells Fargo.

---

*For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided.*

*Thank you!*