CUTTER IT JOURNAL
THE JOURNAL OF INFORMATION TECHNOLOGY MANAGEMENT

August 2001, Vol. 14, No. 8

Enterprise Application Integration

Introduction
William Ulrich, Guest Editor 2

William M. Ulrich is a Senior Consultant with Cutter Consortium's Business-IT Strategies (B-IT) Practice and a regular contributor to the B-IT Advisory Service. He is president of Tactical Strategy Group, Inc. and has more than 23 years of experience in the information management field. His system transformation strategies have been the basis for migration projects worldwide. As author of TSRM, a widely deployed redevelopment methodology, he is viewed as the leading authority on systems transformation for the IT industry. Mr. Ulrich has advised *Fortune* 1000 companies, government agencies, and high-tech companies on information management, organizational change, business continuity planning, supply chain management, market deployment, partner alliance, and Y2K strategies. Mr. Ulrich has published two books and hundreds of articles on information strategies. He writes a monthly column for *Computerworld* and is Chair of the Brainstorm Group E-Business Integration Conference. He was formerly Director of Redevelopment Strategies at KPMG, where he coordinated consulting projects, training, product development, product planning, and business unit profitability.

Overcoming Organizational Hurdles to Business Integration
Lou Russell 5

Integration:
Is It Really Worth the Effort?
Tammy Adams 14

Next Generation Application Integration
From Information, to Process, to Services
David S. Linthicum 18

XML as Glue for Enterprise Integration
Don Estes 28

CUTTER IT JOURNAL

Introduction

by William Ulrich

As companies wrestle with a variety of business and technological challenges in this post-Y2K era, one thing has emerged as an essential requirement: the need to integrate disparate systems, data, business units, and external entities to further e-business initiatives. Enterprise application integration (EAI) is the foundation for e-business integration and serves as the essential lynchpin to all internal and external integration efforts.

EAI requirements are driven by the need to synthesize various systems and data to support e-business, customer, supply chain, and distribution chain requirements. Supply and distribution chain integration falls under a category called business-to-business integration (B2Bi). B2Bi is increasingly being viewed as an extension of EAI, because the border between internal and external systems, data, and business units has been blurred by the emergence of Web services, outsourcing, joint ventures, and virtual corporations. Any EAI discussion must, therefore, consider the broader initiatives driving and relying on EAI.

While EAI can be viewed simplistically as the need to connect front-end and back-end applications and data structures, it must also consider business unit, business process, and external entity integration. This has not always been the case. As a result, EAI efforts have been plagued by quick fixes, technocratic thinking, disjointed projects, false starts, inadequate requirements definition, and a tendency to ignore the basic building blocks of a strategic integration initiative.

Consider, for example, the challenge facing a company streamlining customer service functions. Management has requested a Web-based system to automate customer service. The new system requires access to order, customer, inventory, fraud, and shipping data. Historically, a service representative had to make four or five phone calls or system look-ups to satisfy a customer inquiry. Automating this function minimally requires cross-functional planning, a new front-end system, middleware technology, links to back-end transactions, and data integration.

This type of project is typically justified as a way to cut costs while increasing customer satisfaction. Delivering this level of value, however, cannot be accomplished in a technological vacuum. In this example, as in most high-impact integration projects, integration planning and deployment must also consider the human, organizational, and external factors of such a project.

In this issue of the *Cutter IT Journal*, we have taken a step back to examine the core building blocks of EAI, particularly as it impacts the organization in conjunction with implementing new technological solutions. This includes addressing organizational integration challenges as well as filtering through the value proposition of an integration project. This issue also explores the future of tactical and strategic EAI approaches and looks at a specific method for accomplishing enterprise integration that can also be applied to B2Bi scenarios.

In our first article, Lou Russell writes about "Overcoming Organizational Hurdles to Business Integration," examining the market motivating factors behind integration. Ignoring this key requirements analysis step can leave large projects floundering. Russell identifies the reasons quick-fix solutions not only fail, but also tend to make a bad situation worse. She supports her contention that

 ©2001 Cutter Information Corp.

applying a one-dimensional, technocratic solution to a multidimensional challenge is doomed to fail.

Russell also explains why people resist change and how an integration initiative is, by definition, a major transformative event within an organization. In discussing people and their inherent resistance to change, she outlines how integration projects must have a greater sensitivity to the people working on those projects as well as the people impacted by them. Ignoring the human element is another sure path to failure.

Finally, Russell sketches the role of leadership within a business integration project. By linking ideas on how to establish requirements, deal with human resistance, and establish strong leadership, she provides a firm foundation for any enterprise undertaking an integration initiative — regardless of the size and scope of such a project.

Next, Tammy Adams explores a question that many organizations tend to skip over in their thinking about EAI: Is it really worth the effort? She suggests that organizations engaged in the blind pursuit of application, process, and vendor integration need to question the validity of such projects in the first place — or face serious consequences.

In her article, Adams prompts executives to ask some fundamental questions before proceeding with an integration project. For example, do you have a compelling business reason for integration? In other words, can you tie your motivation back to quantifiable business requirements? She also suggests that companies need to have a clear vision of what must be accomplished or they will not be able to weather the difficult challenges of an integration project.

In discussing process integration, a rapidly growing requirement for almost any EAI strategy, Adams indicates that companies should understand the processes they are trying to integrate. She also believes that companies must be able to measure their success in some quantifiable way before undertaking a project. This discussion is key to developing the necessary level of cost-benefit analysis for an integration project. In closing, Adams outlines the key ingredients needed to succeed in an integration project.

In his article, EAI guru David Linthicum deals with "next-generation" application integration. Linthicum suggests that the benefits of integration are growing in direct correlation to the size of the integration challenge. He identifies key trends in integration, one of which is the shift away from data-oriented integration toward service-based integration. In discussing service-based integration, Linthicum hits upon an important shift in integration that is rapidly emerging: the need to build integration requirements into your application architecture strategy.

In citing this trend toward service-based integration, Linthicum discusses the growing role of Web services within the integration market. He explains how Microsoft's .NET strategy and other industry standards and specifications, such as UDDI (Universal Description, Discovery and Integration), will play an increasingly important role in the evolution of integration projects.

In addition, he emphasizes the importance of deploying integration options that provide near-term value while laying the foundation for more strategic integration scenarios such as those that employ .NET and other emerging service architectures. These options include information-oriented approaches that companies can implement today by focusing on data as the primary point of

Editor Emeritus:
Ed Yourdon

Editorial Board:
Larry L. Constantine
Bill Curtis
Tom DeMarco
Peter Hruschka
Tomoo Matsubara
Navyug Mohnot
Roger Pressman
Howard Rubin
Paul A. Strassmann
Rob Thomsett

Cutter IT Journal® *(ISSN 1522-7383) is published 12 times a year by Cutter Information Corp., 37 Broadway, Suite 1, Arlington, MA 02474-5552 (+1 781 648 8700, or, within North America, +1 800 964 5118; Fax +1 781 648 1950 or, within North America, +1 800 888 1816; Web site: www.cutter.com/consortium).*

Cutter IT Journal® *covers the software scene, with particular emphasis on those events that will impact the careers of information technology professionals around the world.*

Editor Emeritus: Ed Yourdon
Publisher: Karen Fine Coburn
Managing Editor: Karen Pasley
Production Editor: Linda Mallon
Client Services: Christine Doucette

©2001 by Cutter Information Corp. All rights reserved. Cutter IT Journal® *is a trademark of Cutter Information Corp. No material in this publication may be reproduced, eaten, or distributed without written permission from the publisher. Unauthorized reproduction in any form, including photocopying, faxing, and image scanning, is against the law. Subscription rates are US $485 a year in North America, US $585 elsewhere, payable to Cutter Information Corp. Reprints, bulk purchases, past issues, and multiple subscription and site license rates are available on request.*

integration. He discusses how middleware technology can be used to deploy various information-centric approaches to integration.

Linthicum also provides a detailed discussion of business process-oriented application integration (BPOAI) and how this approach differs in scope and intent from integration projects that focus solely on data or systems. BPOAI is a high priority for companies because it connects people and applications from a cross-functional perspective that also reaches beyond the bounds of the enterprise. Linthicum reiterates that application integration is clearly the future, and companies can gain a competitive advantage by pursuing tactical integration efforts now as Web services and other strategic options continue to emerge.

Lastly, Don Estes discusses why the IT industry needs a universal solution to the three-pronged challenge of integrating the Web with client-server, the Web with the mainframe, and the mainframe with the mainframe. That universal glue, according to Estes, is the Extensible Markup Language (XML), which he advocates as a common solution to what until recently have been viewed as separate and distinct challenges. He argues that while middleware has its place in application integration, a more flexible approach can be a better option in many cases. That approach utilizes XML as the glue to integrate the Web with distributed and mainframe-based applications.

Estes provides a good overview of the XML language and how it works. In doing so, he explains why XML is so flexible and can be applied to a variety of information exchange scenarios. This provides the foundation for his discussion on how to integrate applications using XML within an enterprise. He provides useful insights into why and how EAI and B2Bi are quickly being conjoined under a common set of technologies and disciplines.

Enterprise application integration is strategic, has broad impact, and is definitely here to stay. The challenges that many companies must overcome include people's resistance, cost justification, project strategy development, and technology deployment. Our authors provide collective insight into how to overcome many of these obstacles. Coupling these various approaches provides an EAI strategy that spans internal and external integration challenges and the organizational hurdles that are so common in these types of projects.

Next Issue

Testing E-Business Applications

Although high-tech stock prices continue to trend down, the pressures to develop Internet-enabled applications for the general public continue to rise. A key dilemma, though, is how to test these e-business applications to ensure reliable functionality, adequate performance, and solid security. The methods of developing applications for e-business have undergone a sea change in the past five years, but the role of testing has arguably not kept pace with this change. Witness the frequent news reports of denial of service attacks, credit-card database theft, and Web sites that cave in under unexpectedly high traffic. How do we need to think differently in this Internet age about testing security, performance, and reliability of e-business applications? Tune in next month to read more about this critical issue.

 ©2001 Cutter Information Corp.

Overcoming Organizational Hurdles to Business Integration

by Lou Russell

Integrating technology is a complex proposition. Integrating business is even more complicated. If it were only the technology, figuring out how to interface diverse hardware and software would be the challenge. But integrating business requires venturing into the most unexplored and unpredictable territory — people. In this article, you will read:

- Why the market today has made business integration an imperative, not just a market differentiator
- Why quick-fix solutions to business integration not only fail, but often make the problem worse
- Why people resist change so strongly, and since business integration is change, resist it as well
- Why the role of the leader is so critical to business integration success

As you read about these ideas, you will learn ways to grow the effectiveness of your people during a business integration effort, how to define and implement the organizational transformation required to make business integration a success, and how to correctly troubleshoot and manage resistance from people.

BUSINESS INTEGRATION IS AN IMPERATIVE TODAY

The market changes around us have been head-spinning in their quickness. Last year at this time, I would have written that dot-com mania was demanding that businesses integrate their processes through streamlined, Web-based initiatives. Time was the brass ring of business, and cost was no object as long as you could get something to the Web quickly. Today, many of the companies that would have helped you with that objective are gone. Half-baked, poor-quality Web applications are either being rewritten or removed. Speed kills, and in some not surprising ways, it killed some of the e-phenomenon.

Unexpectedly, people at companies well removed from dot-coms are also being laid off. Daily announcements of downsizing numbers with lots of zeros depress the business world. Time is still the brass ring — if the technology solutions do not help the company increase sales, more heads may roll. However, money has become increasingly tight, and quality is an imperative. Bad, cheap, strategic business "solutions" are a signed death warrant for a company, as indicated by the stories of large companies taken down by ERP implementations that became financial black holes.

So, instead of the constraints getting simpler, they have become more complex. The business demands that IT provide business integration solutions that are fast, meet the fluctuating market needs, and create a real return on the investment dollar. There is, ironically, a sense that only IT business integration solutions can save us now, even though the perception also exists that the dot-coms nearly killed us.

Both business and IT staff have to rethink their beliefs about reality. Business has

traditionally been systematic, planned, predictable, stable, understandable, dependable, and enduring. The e-era brought with it a new view of e-business as time constrained, iterative, mission critical, constantly changing, complex, high risk, but also technology based. IT is finally at the table, but the perceptions on both sides must be updated.

Moving quickly with an improvement that is not whole enough can become a disaster much quicker today than ever before.

QUICK FIXES MAKE THE PROBLEMS OF INTEGRATION WORSE

Because of the growing sense of urgency (or now, desperation), people are reluctant to take a moment and think systemically. The problems that businesses are struggling with — brand differentiation, profit, market share, retention, new products — are not new, but they pose a greater competitive challenge for businesses since the Internet has made the world truly a global marketplace. In a matter of minutes, every customer in the world can know all about you. So can your competitors. Moving quickly with an improvement that is not whole enough can become a disaster much quicker today than ever before.

Let's say, for example, that you ramp up your e-commerce site with new product information. Suddenly the sales start increasing through the Web, but the salespeople on the road haven't had time or the ability to integrate their existing customer relationship management (CRM) system with the new product strategy. Misinformation gets out, reviews show up on the Web criticizing the company, the stock falls, the world turns ugly ... all overnight. A good strategy incompletely implemented can destroy a company.

Business solutions must be carefully integrated to ensure that the needs of traditional business — planning, stability, understanding — are considered, but the processes for planning must be streamlined in order to achieve the speed and agility required by businesses today. The only way to grow solutions that can be this adaptive is to transform the core ingredient: the people.

THREE TIERS OF TRANSFORMATION

Awareness

Transforming organizations of people is a three-tier process. The first tier is to make people aware of the current situation and the gap between what is and what must be. For example, before developers can learn a new approach such as Extreme Programming (XP), they must understand the problems they have that XP is able to combat. Some companies either get stuck in this tier, analyzing forever, or get so overwhelmed by the depth of the problems that no one does anything.

Awareness must be achieved at both a team level and an individual level. Improvement, never mind transformation, cannot occur unless each involved person changes. One of the approaches that I like to take with individual awareness is to assess each team member's value and behavioral preferences (see Figures 1 and 2). These strengths and weaknesses can be combined with the assessments of all the other individuals on the team to create a team picture of strength and weaknesses. This is necessary input not only for performance improvement, but also for effective leadership.

As you can see, each of us has individual values, interests, and attitudes that provide the criteria we use daily to make decisions. These beliefs filter reality for us and help us to decide where to place our focus. Once we filter in what we will choose to attend to, our behaviors kick in. These behavioral preferences are much like our intake and

 ©2001 Cutter Information Corp.

	D	I	S	C
	Demanding			Evasive
	Egocentric	Effusive		Worrisome
	Driving	Inspiring	Phlegmatic	Careful
	Ambitious	Magnetic	Relaxed	Dependent
	Pioneering	Political	Resistant to Change	Cautious
	Strong-Willed	Enthusiastic	Nondemonstrative	Conventional
	Forceful	Demonstrative	Passive	Exacting
Behavior: no right/ wrong	Determined	Persuasive	Patient	Neat
	Aggressive	Warm	Possessive	Systematic
	Competitive	Convincing	Predictable	Diplomatic
	Decisive	Polished	Consistent	Accurate
	Venturesome	Optimistic	Deliberate	Tactful
	Inquisitive	Trusting	Steady	Open-Minded
	Responsible	Sociable	Stable	Balanced Judgment
	Conservative		Mobile	Firm
Doesn't measure: learning ethics	Calculating	Factual	Active	Independent
	Cooperative	Calculating	Restless	Self-Willed
	Hesitant	Skeptical	Alert	Stubborn
	Low-Keyed	Logical	Variety-Oriented	Obstinate
	Unsure	Undemonstrative	Demonstrative	Opinionated
	Undemanding	Suspicious	Impatient	Unsystematic
	Cautious	Matter-of-Fact	Pressure-Oriented	Self-Righteous
	Mild	Incisive	Eager	Uninhibited
	Agreeable	Pessimistic	Flexible	Arbitrary
	Modest	Moody	Impulsive	Unbending
	Peaceful	Critical	Impetuous	Careless with Details
	Unobtrusive		Hypertense	

Energy Line

Figure 1 — "DISC" assessment of value and behavior preferences.

processing preferences — we have certain behavioral strengths, such as the ability to influence others or attention to quality, that make us uniquely valuable to the team. It makes no sense to isolate the influencer by having him or her debug code, while sending the detail person to the customer focus group to gather requirements. If individuals instead work on functions that honor their own preferences, the team together becomes a powerful whole.

And don't forget to assess yourself. By understanding yourself, you can adjust your behaviors in the short term to communicate well with others who do not share your profile. When the value and behavior profiles are used as a team assessment tool, the team gaps in terms of behavior ("We have no detail people") or values ("No one cares about the ROI") become evident. Not only does this type of assessment allow the team to honor each individual's unique qualities while building synergy, it also points out holes that must be filled in some way for the project to succeed.

Table 1 (see page 9) lists the descriptions of the four behavioral components, known as DISC (for Dominance, Influence, Steadiness, and Compliance). The strengths and weaknesses of these behaviors contribute to the success of an integration project. In fact, carefully aligning these values and behaviors to a role (e.g., project manager) defined by a set of competencies (e.g., ability to manage using a Critical Path) can enable project success.

Finally, what matters most on a project team is the team composition as a whole. By looking at the values and behaviors of a

Value	Passions	Motivated by
Theoretical	- Solving problems - Objectivity in all areas of life - Identifying, differentiating, generalizing, systematizing - Intellectual process - Discovery, understanding, ordering - Pursuit of knowledge, identifying truth and untruth - Knowledge for the sake of knowing	Rational, analytical, and objective discussion, problem solving
Utilitarian	- Practicality in all areas of life - Surpassing others in attainment of wealth - Utilizing resources to accomplish results - Gaining a measurable return on all investments - Creative application of resources - Producing goods, materials, and services and marketing them for economic gain - Capitalism	Money, ROI
Aesthetic	- Practicality, appreciation and enjoyment of form, harmony, and beauty - Enjoyment of all senses - Subjective experience - Understanding feelings of self and others - Self-realization, self-fulfillment, and self-actualization - Creative expression - Appreciation of all impressions	Subjective things, feelings, harmony, and personal fulfillment (not discomfort)
Social	- Investing self in others - Selflessness - Generosity of time, talents, and resources - Seeing and developing the potential in others - Championing worthy causes - Improving society and elimination of conflict - Appreciation of all impressions	Ideas that help others, harmony
Individualistic	- Leading others - Achieving position - Advancing position - Forming strategic alliances - Attaining and using power to accomplish; purpose - Planning and carrying out a winning strategy - Tactics and positioning	Increased power or position, using strength to strengthen others
Traditional	- Understanding the totality of life - Finding meaning in life - Pursuit of the divine in life - Following a cause - Living consistently according to a "closed" book - Converting others to one's belief system	Rules to live by, causes, systems, beliefs and principles

Figure 2 — Personal values, interests, and attitudes.

 ©2001 Cutter Information Corp.

Table 1 — DISC Characteristics

Dominance Characteristics	Behavioral Style
Results oriented	Acts or speaks before thinking
Desire to win	Impatient
Fast-paced, ability to make decisions quickly	Creates fear in others
Willingness to state an unpopular view	Too high risk
Risk taker	"Juggles" too much at once
Argumentative, quick to challenge	Quick to anger
	Interrupts and will not listen
Influence Characteristics	**Behavioral Style**
Creative problem solver	Talks before thinking
Enthusiastic, natural optimism	Loses track of time, often late and hurried
Humorous	Abandons position in conflict
Fun-loving	Disorganized
High contact ability, trusting of others	Overly trusting
Ability to make others feel welcome or included	Overly optimistic, often superficial
Steadiness Characteristics	**Behavioral Style**
Tenacity for order, stability, and closure	Possessive of things
Need for secure situations	Too low risk
Great listener, calms and stabilizes others	Holds a grudge
Good planner, natural ability to organize tasks	Fakes agreement
Able to mask emotions	Resistant to change
	Too indirect when communicating
Compliance Characteristics	**Behavioral Style**
Follows rules	Requires too much data
High expectations, quality conscious	Hard on self
Able to solve complex problems	Too low risk
Organizes and analyzes, willing to dig for information	Makes excessive rules
	Too critical of others
Natural systems developer	Has analytical paralysis

team overall, it is easy to identify the types of things that will challenge the team.

Performance Improvement

The second tier is performance improvement. Many companies skip awareness to go right to this step, and thus they implement a solution before anyone is really clear on what the problem is. For example, companies may decide to do something about improving project management and dictate that all their employees attend a two-day introduction to project management workshop. Without clearly understanding the problems of projects specific to their company, training of any kind, especially generic training, is very likely to be a waste of time and money.

This is where the core disciplines of both Peter Senge's learning organization [3] and the knowledge-creating company of Ikujiro Nonaka, Hirotaka Takeuchi, and Hiro Takeuchi [2] provide the tools needed to systemically improve performance. The five disciplines of the learning organization create the base to define the issues and intervene through shared vision, personal mastery, team learning, mental models, and systems thinking. The *Fifth Discipline Fieldbook* [4] provides some wonderful tools and techniques to implement these

disciplines. Using tacit (internal) and explicit (documented externally) knowledge processes to create, exchange, and transform knowledge extends an organization's ability to not only improve, but also to continue improving indefinitely (see Figure 3).

Transformation

The final tier is transformation, and it is rare that companies get here. Some dot-comish companies, such as Amazon, Cisco, and Lycos, as well as the Macintosh team years ago, defined their organization by looking at what didn't work in other initiatives (awareness) and skipping right to creating a new type of business (transformation). Certainly, it is much easier to grow a new type of organization than it is to evolve an existing one into it.

But most companies have no choice. They must transform the organizations that exist — it is a market imperative. Even after awareness and some iterative performance improvement through training, new incentives, new processes, and new technology, people still strongly resist. As my friend Michael Ayers once told me, "If you explain something to someone more than three times, it is not lack of understanding, it is resistance." Why do people resist what is necessary to save them?

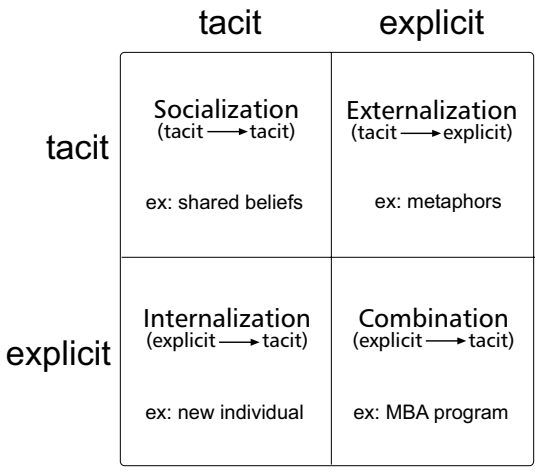

Figure 3 — Tacit and explicit knowledge transfer processes enable organizations to create, exchange, and transform knowledge.

INTEGRATION IS CHANGE, AND PEOPLE RESIST IT

Ayers also wrote a thought-provoking master's thesis entitled "Resistance to Organizational Change," in which he used interviews and Senge's systems thinking to look at how people resist. He chose systems thinking because it let him model the cause and effects between individuals. Ayers found that as one person resists for a certain reason, it influences other people to resist. He discovered that people tend to resist when any of the following five components are lacking:

1. The belief that change can occur
2. A challenging vision
3. A clear pathway
4. Autonomy
5. Congruent values[1]

A person who does not believe that change can occur will resist strongly.

A person who does not believe that change can occur will resist strongly. A leader must convince this person that the change is possible for resistance to decrease. If another individual does not buy into or understand the vision, the leader must paint the picture more clearly and paint this individual into it. Many people feel that they don't know what they are supposed to be doing and what the priorities are when business integration projects are under way. These people need more help with their project planning and more structure to see how they can be a part of the progress. Likewise, some people with older technical skills may feel that they aren't needed, that

[1]Just to make things more complex, any individual may switch between these resistance factors at any time. This creates a fairly chaotic dynamic for a leader to manage, which is perhaps one reason most managers are so unsuccessful at managing resistance and why so many business integration efforts die because of it.

 ©2001 Cutter Information Corp.

other people can do it without them. This lack of autonomy will cause strong resistance, and the leader must clarify the value this person brings to the integration. Finally, if a person perceives that the project is inconsistent with his or her personal values and beliefs, no amount of explanation will lower the resistance. This requires a heart-to-heart, open discussion of beliefs with the leader. This is not very common on business integration projects, but it may occur if people feel technology is being used to downsize or eliminate workers.

By listening to people's language, a leader can discover what is really bothering them, even when they may not be consciously aware of it themselves. For example, if someone says, "They'd never let us do this," it indicates that the person does not believe the change can occur. Ayers has developed a prototype resistance assessment that can be used with a team. All of the team members indicate what they've heard others say and what they think they've said themselves. The contrast is always enlightening to the team as well as the leader. Once the major resistance points are identified, the leader and team can come up with the actions that must be taken. This assessment is also a good tool to use as the project evolves, since resistance points will change.

THE LEADER ROLE IS CRITICAL

Strong leadership is a critical success factor, and leadership is a capacity that most IT organizations have not invested in. Leadership must be grown at all levels in the IT organization, but most importantly in the middle "translation" group of managers. It is these people who must align the vision of the CIO with the vision of the business stakeholders and then translate that into action for the developers and technologists. Essentially, these people play the role of relationship managers, and they often do not have the training, support, or mentoring that will enable them to get that job done. In order to manage the necessary relationships, these managers will require the following leadership competencies:

- Interpersonal and relationship skills
- Strategic business acumen
- Employee development (coaching and motivating)
- Communication skills
- The ability to create and actualize a vision
- Project management
- The ability to influence others and negotiate
- The ability to create, support, and manage change
- Initiative and resilience
- Judgment and decisionmaking
- Technology-management discipline
- Individual leadership self-awareness

Leaders must manage the delicate balance between the business, the workgroup, the culture, and the IT strategy in the short and long term (see Figure 4).

A leader must capitalize on the staff's individual and team strengths. Work must be aligned to the business goal. Tactics and strategies must be balanced. Each individual must be accountable for his or her own clear role. And most important and difficult, the leader must model the behavior expected by others, and when this is not done, he or she must admit the mistake and strive to improve. There is no more powerful message.

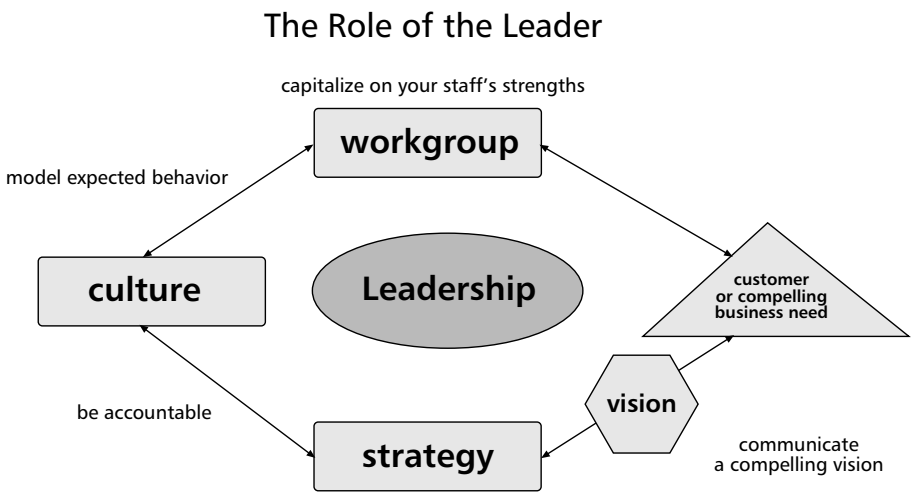

Figure 4 — Business integration leadership.

A compelling company vision creates the link between the business and the strategy and drives IT's prioritization. No business integration project can be successful without a clear, pragmatic business vision specifying what the business is trying to become and what the value proposition is. IT plays the bizarre role of driver when this vision is not in place. Business owns the vision and business need, and IT owns the ability to translate technology-based integration solutions into meeting the business need.

START THE JOURNEY NOW

At this point, you can choose to start becoming the leader your business integration work needs. Here's a brief list to get you started:

- Create and spread an impelling and measurable vision.
- Create business goals and measure the transformation against them.
- Treat all people as unique and leverage their strengths.
- Define your role as a relationship manager and intentionally work on the relationships with the executives, the customers, and the practitioners.
- Help your teams add value in all they do. Teach them to focus on the customer's customer and the value of the integration effort to them.
- Build and continuously nurture a strong and effective sense of team. Encourage and reward openness and integrity.
- Make mistakes safe but incompetence nonexistent.
- Communicate more than you think you should, and encourage others to do the same. View communication as "real work," not extra, less significant activity.
- Model the behavior you want to see in others.
- Find a way to reflect, vent, and be coached so that you can grow yourself.

 ©2001 Cutter Information Corp.

REFERENCES

1. Ayers, Michael. "Resistance to Organizational Change." Master's thesis, College of St. Catherine, 1999.

2. Nonaka, Ikujiro, Hirotaka Takeuchi, and Hiro Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford Press, 1995.

3. Senge, Peter M. *The Fifth Discipline: The Art and Practice of the Learning Organization*. Doubleday, 1990.

4. Senge, Peter, Art Kleiner, and Charlotte Roberts. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization.* Doubleday, 1994.

Lou Russell is a Senior Consultant with Cutter Consortium's e-Project Management Practice. She founded Russell Martin and Associates in 1987 as a consulting and training company. Ms. Russell's Fortune *500 clients rely on her knowledge of information services infrastructures to help manage major efforts in retention, customer service, project management, systems development, and retooling. Her career has included positions at Ameritech and McDonnell Douglas in training and in course development and delivery. Ms. Russell is the author of* The Accelerated Learning Guidebook: Making the Instructional Process Fast, Flexible, and Fun*; her second book,* Project Management for Course Developers*, will be published soon. Ms. Russell has written articles for* Cutter IT Journal*,* Training*,* Computerworld*, Auerbach journals, and others, and currently serves as president of the Society of Information Management (SIM) Indianapolis chapter. Ms. Russell is a frequent speaker at national industry events.*

Ms. Russell graduated from the computer science program at Purdue University and received a master's degree in instructional technology in education from Indiana University.

Ms. Russell can be reached at Russell Martin & Associates, 6326 Rucker Road, Suite E, Indianapolis, IN 46220, USA. Tel: +1 317 475 9311; Fax: +1 317 475 0028; E-mail: lrussell@cutter.com

CUTTER IT JOURNAL

Integration:
Is It Really Worth the Effort?

by Tammy Adams

History has shown us that integration is a tangled web of desires, realities, risk, planning, and work. Whether racial, political, or cultural, every integration effort has come with its own set of complex challenges. Typically opposed by the masses, these efforts were eventually embraced over time in spite of the pain and loss incurred. We would be ignorant to assume that this new flood of integration desires brought on by e-commerce, B2B, and B2C is any different. It doesn't matter whether it's a culture or an application, integration involves conflict. So before you blindly pursue integration of your applications, systems, processes, and vendors, ask yourself one question. Is it worth it?

TODAY'S GOSPEL

If you read the plethora of articles available on the topic, you'll be quickly convinced of the benefits and the overwhelming necessity of EAI, BPI, B2Bi,[1] or any other "i." In his book *Enterprise Application Integration*, David Linthicum describes EAI as "the unrestricted sharing of data and business processes among any connected applications and data sources in the enterprise." Business process integration takes that description one step further to include applications and sources external to the enterprise. Quite an ambitious little task, don't you think?

What are the forces driving us madly to this integration scramble? Shorter lifecycles, quicker responses, lower costs. Now, I wasn't born into IT yesterday. I've heard these same forces used to justify everything from legacy systems replacement to purchasing one of those stovepipe applications we're now trying to integrate.

A HISTORY LESSON

Remember the early 1990s? Data centers were downsizing, tiered architecture was the rage, and members of the Nintendo generation were joining corporate IS staffs. At that time, analysts were propagating the notion that the networked small systems and distributed architectures known as client-server would soon replace mainframe-based systems. Many companies expended significant time and effort, only to be disappointed in the end results. One such story describes a very large and well-known corporation that was slated to go client-server with the following training plan: First, take lifelong COBOL programmers, train them for one week (count it, one week) in C, and then get them going on a client-server application, writing the front-end GUI in native C. Is it any wonder the implementation was designated a "horror story"? Of course there were also many successes, but only for those companies that evaluated the effort, impact, and benefits to make a wise choice.

And lest we forget our more recent experiences, let's talk about the dot-com frenzy. Build it and they will come. Revenue with every click. Funding for any plan that contains the words "Internet," "e-commerce," or "B2B." It was the 1960s revisited — only instead of free love, free money abounded. Companies leapt into the fray by creating their "Web presence" — initially an online brochure with automated glitz. They never stopped to assess the

[1]For the record, these are "enterprise application integration," "business process integration," and "business-to-business integration," respectively.

 ©2001 Cutter Information Corp.

value. Then they added catalogs and online customer ordering without asking if it was worth it.

DO YOUR HOMEWORK

I'm not suggesting that integration is a bad idea. But like the business process reengineering (BPR) of the 1980s, it's not the solution to all business problems. So take the time to investigate the direct value of EAI/BPI to you and yours before jumping in the deep end.

Do you have a compelling business reason for integration?

Research by a number of analyst groups points to four key business drivers behind integration:

- Customer demand for faster and more personalized services
- Customer demand for single-point access to cross-organizational data
- Organizational need for market expansion with cost reduction
- Organizational need to combine merged or acquired systems

Do one or more of these apply to your business? Do you have marketing research or corporate directives to back this up?

Do you have a clear end vision that will carry your organization through the hard times?

Whether you're a CEO, director, or project manager, you need to be able to articulate why integration is necessary. This is not just a quickly recited statement of benefits, but a picture that makes even the skeptics agree that the pain of conflict, potential productivity loss, additional system cost, and human impact of change are worth it.

During the passionate struggle for racial integration in the 1950s and '60s, leaders unified the country behind their vision of equality. President Kennedy expressed his desire for "every American … to have the right to be treated as he would wish to be treated, as one would wish his children to be treated." Martin Luther King Jr. envisioned "that one day on the red hills of Georgia the sons of former slaves and the sons of former slaveowners will be able to sit down together at a table of brotherhood." Your vision, while not as critical to human rights, should be no less moving. Your organization, your project team, and your department need to understand and embrace this vision.

Do you have a strong mandate, actively supported by people in authority, to force the organization forward?

Integration can't be a suggestion. It can't just be the right thing to do. It must be demanded. Imagine if Abraham Lincoln had simply said slavery was a bad idea and suggested it be abolished. Do you think we'd be a free nation today? Integration changes the way we live, the way we work, the way we think. If you think those kinds of changes are easy, you're fooling yourself. Do you have executives willing to state the need, demand the change, and be the first to demonstrate it? If not, don't start down the path.

COUNT THE COST

So you believe integration will be valuable to your organization. Great. You see the vision and are starting to believe. Wonderful. Now it's time to put a price tag on that value. Start by defining your targets and end by doing the math.

Do you understand the business process you're integrating?

According Ian Evetts, head of Cap Gemini's integration practice in the UK, no one buys

Integration can't be a suggestion. It can't just be the right thing to do. It must be demanded.

EAI for EAI's sake alone. Such integration efforts are typically driven by the need to tie together one of the four core processes essential to long-term competitiveness — product/service strategy selection, customer relationship management, order cycle management, or product cycle management. But knowing which process you're focusing on is just the beginning.

In simple terms, not only do you need to know the type and shape of the vehicle, but also what it's supposed to do and how it works. Without that knowledge, you may spend an inordinate amount of time on the windshield wipers and little on the engine. Here's where logical and physical process modeling come into play. Much has been written about the limitations of such tools, but so far no better descriptors have been found. So spend the time to investigate and understand. As you do, a strange phenomenon will occur. People will start to agree that change is needed. Why? Because you've helped them visualize the process.

The old adage "It takes money to make money" applies here.

Do you have end-state process goals that can be measured, both now and later, to demonstrate success?

If you're driving to Las Vegas, you have to know your starting and destination points to plan the best route. But take that one step further — what do you mean by "best"? Do you want the shortest distance, the quickest time, or the most interesting sites along the way? Measuring your integration goals is no different. Step one is knowing what you hope to attain from the integrated process. Are you shooting for overall cost reduction? Increased processing speed? Lower onsite inventory? Greater customer satisfaction? Step two is figuring out the metrics necessary to measure that goal. How will you know when you achieve greater customer satisfaction? Return customers? Higher volume of new customers? Fewer problems reported? Some combination of these or additional metrics? And finally, step three, measure it today. You must have a baseline against which to measure the integrated process in order to determine success.

And what about those intangibles you're hoping to achieve, such as "competitive advantage" or "employee empowerment"? It's impossible to measure such things, right? Wrong. Everything is measurable. As author Douglas Hubbard put it:

- If something is better, it's different in some relevant way.
- If it's different in some relevant way, then it's observable.
- If it's observable, it can be counted.
- If it can be counted, it can be measured.

So every process change can be tracked to some metric. However, in the end it may take more effort than it's worth to gather the information. Again, that's one of the checks and balances in determining whether integration is for you.

Can you afford the investment necessary to achieve integration?

Speaking of investment, the old adage "It takes money to make money" applies here. You've got to spend in the short term to reap in the long term. Marc Buyens of Xpragma, an independent IT research, analysis, and consulting firm, says, "The reality behind EAI is that it is not an easy exercise. The cost of implementing such a solution is easily three to four times the raw product cost. So, it is not only technology that is key, but also the necessary skills to integrate the solution." When evaluating the cost, consider not only the tangible costs (such as the costs associated with providing and supporting intersystem interfaces), but also

 ©2001 Cutter Information Corp.

the costs related to the learning curve for development and operations. Bottom line, a full-blown enterprise-wide EAI implementation can easily cost in the millions of dollars. The cost incurred as a result of retraining, technology transfer, and cultural issues can be just as much. So do the math before committing to the effort.

Can you afford to integrate? Can you afford not to integrate? Are the risks and costs greater than the benefits and opportunity loss? Thank goodness there are tools available to help you in this evaluation, such as GIGA Information Group's Total Economic Impact (TEI), Gartner Group's Total Cost of Ownership (TCO), and other return on opportunity (ROO) models. Of course, they cost money too. Remember to factor that into your thinking.

STILL INTERESTED?

So you've done your homework, estimated your costs, and it still makes sense to integrate? Then may I suggest you take a few lessons from those who have gone before? No, not Caesar, who tried to integrate the world. But rather the learned folks who have implemented SAP, ERP, and other integrated systems. These wise ones recommend the following:

- **Have an experienced project manager in charge of the effort.** Planning is key. Creating and tracking to a well-developed project plan cannot be overstressed. A good project manager will make sure the project stays on target even in the face of adversity.
- **Overcommunicate.** Whether reaching across the organization or extending beyond, stovepipes are no longer allowed. Keep users actively involved by including them in design sessions and reviews. Determine a plan for communication and build trust through regular information exchanges.
- **Focus on one objective.** Don't try to improve everything all at once. This isn't the Big Bang Theory. Choose one of your process goals and go for small wins along the way. Don't waste time on elaborate BPR.
- **Have clearly defined boundaries for consultants involved in the project.** A healthy set of "dos" and "don'ts" is a valuable asset in integration projects.
- **Understand the wave phenomenon.** Yes, any project that relies on teams will have its peaks and troughs. Knowing this up front eliminates the false belief that progress will be well-defined, steady, and incremental. Projects do not evolve, they're circular efforts of revelation and revolution.

History has shown that integration, while costly, can also be rewarding. But it's not for everyone. Implementing and maintaining EAI is not for the fainthearted and may exceed the skills and budget of some. So, is it worth it? That's your decision.

REFERENCES

1. Hubbard, Douglas. "Everything Is Measurable." *CIO Enterprise*, 15 November 1997.

Tammy Adams is managing partner of Chaosity LLC, specializing in business process improvement and group facilitation. Chaosity LLC helps organizations simplify the chaos in their business processes by clarifying, documenting, and improving how they do what they do. Using a team-based, facilitative approach, Chaosity LLC extracts the experience and knowledge of teams to help them figure out how to work smarter, faster, and more competitively.

Ms. Adams can be reached at Chaosity LLC, 8611 S. Kachina Drive, Tempe, AZ 85284, USA. Tel: +1 480 775 8756; E-mail: tadams@chaosity.com; Web site: www.chaosity.com.

CUTTER IT JOURNAL

Next Generation Application Integration:

From Information, to Process, to Services

by David S. Linthicum

Application integration is a complex problem. Although we have witnessed some notable successes in addressing it, these successes have been narrow and limited. The simple reality is that most application integration projects exist just at the entry level. We have yet to see the real-time coupling of thousands of applications. This is not as discouraging as it sounds. As with any complex problem, once it is broken down to its component parts, the solution becomes simply the aggregation of a number of solution sets. In this case, the solution is a combination of a variety of approaches and several types of technology.

The world of application integration is no different from the larger world of technology; it is advancing and changing rapidly. Ironically, as the technology changes, so does the problem it is designed to solve. The application integration problem is morphing from the very simple to the very complex, even as it moves from a departmental problem to an enterprise-wide problem, and, ultimately, to a trading community problem.

As the application integration problem grows, so does the potential benefit of the solution. Technology continues to respond to the perceived need as the problem domains become more complex, and the application integration solution set evolves to address the growing complexity. No sooner is a "traditional" application integration problem solved (such as application-to-application and database-to-database integration), than the newly developed application integration expertise and technology are applied to more complex — but more rewarding — business issues.

Consequently, few companies have been able to keep pace with the "application integration curve." Lacking a complete solution, they have yet to realize the full potential and benefits of application integration. In this context, our pursuit of application integration is like chasing the tail of a growing beast. For now, that "beast" — and a great deal of work — remain a step ahead of us. But rest assured, a solution will be found, and the once unimaginable benefits of application integration will become an everyday reality.

MOVING FROM DATA-ORIENTED TO APPLICATION-ORIENTED INTEGRATION

Another clear trend is the movement away from data-oriented to service-based integration. Data-oriented integration provides an inexpensive mechanism for integrating applications because, in most instances, there is no need to change the applications.

While data-oriented integration provides a functional solution for many application integration problem domains, it is the integration of both application services and application methods that generally provides more value in the long run. This is not a new approach. We've been looking for mechanisms to bind applications together at the service level for years, methods that include frameworks, transactions, and

 ©2001 Cutter Information Corp.

distributed objects, all in wide use today. However, the new notion of Web services, such as Microsoft's .NET strategy, is picking up steam. (Note: there are competing standards from IBM and BEA.)

Web services identify a new mechanism that can better leverage the power of the Internet to provide access to remote application services through a well-defined interface and directory services (UDDI [Universal Description, Discovery and Integration]). The uses for this type of integration are endless. Creating composite applications, or applications that aggregate the processes and information of many applications, is one notable example. Using this paradigm, application developers simply need to create the interface and add the application services by binding the interface to as many Internet-connected application services as required.

The goal of Microsoft's .NET is to provide total Internet-connected integration. .NET defines Internet-connected applications as Web services. You can think of Web services as methods exposed by a company or software program that are both discoverable and accessible by other programs or organizations that are in need of a particular service. For instance, purchasing a product, reserving a flight, calculating tariffs, and so on are discrete business services that have value for many organizations.

.NET is an architectural vision more than an actual technology. However, pieces of technology are beginning to emerge, such as UDDI and a host of tools forthcoming from Microsoft, that will provide the opportunity for total integration. In addition, SOAP (Standard Object Access Protocol) provides .NET with a standard way to expose objects through firewalls, business to business.

The UDDI specification aims to define a common mechanism to both publish and discover information about Web services. The companies that put UDDI together — IBM, Microsoft, Ariba, Inc., and 63 others — hope to create a type of "Yellow Pages" for the Internet. The first generation of the specification is out now, with the second generation due out in 18 months.

UDDI is really just a set of databases where businesses can register their Web services and locate other Web services they may be interested in leveraging. For instance, a company may have a unique program for predicting breakage found in a shipment of glass products, depending on the method of shipping, point of origin, and destination. The company may publish this information in the UDDI databases, allowing other organizations to find this Web service and understand how to access the service as well as the interfaces employed.

The downside, at least with service-based integration, is that this approach requires an organization to change the source and target applications or, worse, in many instances, to create a new (composite) application. This adds cost to the application integration project and is the reason many companies choose to remain at the information level.

Still, the upside is that many enterprises find this "baby step" approach the most comfortable when implementing solutions to integration problems. Service-based solutions tend to be created in a series of small, lower-risk steps. This type of implementation can be successful from the department to the enterprise to the trading community, but never the other way around (i.e., from the trading community to the department).

Information-oriented (a.k.a., data-oriented) application integration gives most organizations a low-risk option for getting application integration under control. After that, as more time and money become available and the appetite for risk increases, they can plot a strategy to "step up" to service-based integration by leveraging emerging standards, such as .NET.

At this time, and in the foreseeable future, one-stop shopping is simply not an application integration reality.

MIDDLEWARE APPROACHES

As we've come to appreciate, application integration is a combination of problems. Each organization and trading community has its own set of integration issues that must be addressed. Because of this, it is almost impossible to find a single technological solution set that can be applied universally. Therefore, each application integration solution will generally require products from several different vendors. At this time, and in the foreseeable future, one-stop shopping is simply not an application integration reality.

Although vendor approaches to application integration vary considerably, creating some general categories is possible. These categories include:

- Information-oriented integration
- Business process-oriented application integration (BPOAI)
- Service-oriented integration

INFORMATION-ORIENTED INTEGRATION

Vendors who promote the information-oriented approach to application integration argue that integration should occur between the databases (or proprietary APIs that produce information, such as business application programming interfaces). That is, databases or information-producing APIs should be viewed as the primary points of integration. However, even within information-oriented application integration, there are many approaches. It is no surprise that each vendor is quick to promote its particular solution. Information-oriented solutions can be subgrouped into three categories: data replication, data federation, and interface processing.

Data Replication

Data replication is simply moving data between two or more databases. These databases can come from the same vendor or from many vendors. They can even be databases that employ different models. The fundamental requirement of database replication is that it accounts for the differences between database models and database schemas by providing the infrastructure to exchange data. Solutions that provide for such infrastructures are plentiful and inexpensive.

Many database-oriented middleware solutions currently on the market provide database replication services as well. Replication services are accomplished by placing a layer of software between two or more databases. On one side, the data is extracted from the source database or databases, and on the other side, the data is placed in the target database or databases. Many of these solutions also provide transformation services — the ability to adjust the schemas and the content so they make sense to the target database.

The advantages of database replication are simplicity and low cost. Database replication is easy to implement, and the technology is cheap to purchase and install. Unfortunately, these advantages are quickly lost if methods need to be bound to the data, or if methods are shared along with the data. If these requirements exist, service-based solutions must be considered (see below).

 ©2001 Cutter Information Corp.

Data Federation

Data federation is the integration of multiple databases and database models into a single, unified view of the databases. Put another way, data federations are virtual enterprise databases that are composed of many real physical databases. Although data federation has been around for some time, the solution set has been perfected only recently.

Data federation software places a layer of software (middleware) between the physical distributed databases and the applications that view the data. This layer connects to the back-end databases using available interfaces and maps the physical databases to a virtual database model that exists only in the software. The application uses this virtual database to access the required information. The data federation handles the collection and distribution of the data, as needed, to the physical databases.

The advantage of using this software is its ability to bind many different data types into a unified model that supports information exchange. Data federation allows access to any connected database in the enterprise through a single, well-defined interface. This is the most elegant solution to the data-oriented application integration problem. Unlike replication, this solution does not require changes to the source or target applications. Still, changes do have to be made to the application that supports the federated database software. This is due to the fact that different interfaces are being used to access a different database model (the virtual database).

Interface Processing

Interface processing solutions use well-defined application interfaces to focus on the integration of both packaged and custom applications. Currently, interest in integrating popular ERP applications (e.g., SAP, PeopleSoft, and Oracle) has made this the most exciting application integration sector.

Message brokers support application interface processing solutions by providing adapters to connect to as many custom or packaged applications as possible. The adapters externalize information from those applications through their open or, more often than not, proprietary interfaces. They also connect to technology solutions that include middleware and screen scrapers as points of integration.

The efficient integration of many different types of applications is the primary advantage of using application integration–oriented products. In just days, it is possible to connect an SAP R/3 application to an Oracle application, with the application interface processing solution accounting for differences between schema, content, and application semantics via on-the-fly translation of information moving between the systems. Moreover, this type of solution can be used as a database replication solution, able to connect to application interfaces.

The downside to using application interface–oriented products is that there is little regard for business logic and methods within the source or target systems — logic and methods that may be relevant to a particular integration effort. In such a case, service-based solutions are probably the better choice. Ultimately, application interface processing technology will learn to share methods as well as information, perhaps by joining forces with service-based approaches. For now, however, you will have to make an either/or decision.

BUSINESS PROCESS-ORIENTED APPLICATION INTEGRATION

BPOAI is the science and mechanism of managing the movement of data, and the invocation of processes in the correct and proper order, to support the management and execution of common processes that exist in and between applications. BPOAI provides another layer of easily defined and centrally managed processes that reside on top of an existing set of processes and data that is contained within a set of applications. (See "Skateboard Integration," below.)

BPOAI integration is a strategy as much as a technology.

The goal is to bring together relevant processes found in an enterprise or trading community to obtain the maximum amount of value, while supporting the flow of information and control logic between these processes. These products view the middleware — the plumbing — as a commodity and provide easy-to-use visual interfaces for binding these processes together.

In reality, BPOAI is another layer of value resting upon existing application integration solutions, solutions that include integration servers, application servers, distributed objects, and other middleware layers. BPOAI offers a mechanism to bind disparate processes together and to create process-to-process solutions that automate tasks once performed manually.

However, by diminishing the importance of the plumbing, it's easy to lose sight of the larger picture. In reality, no single application integration vendor has solved the plumbing issues. Ultimately, the solution to these issues will be delivered by a combination of BPOAI and middleware vendors. That being the case, it is clear that the binding of middleware and process automation tools represents the future of application integration.

BPOAI is a strategy as much as a technology, which strengthens your organization's ability to interact with disparate applications by integrating entire business processes, both within and between enterprises. Indeed, BPOAI delivers application integration by dealing with several organizations that use various meta data, platforms, and processes. Thus, BPOAI technology must be flexible, providing an abstraction layer between the source and target systems on the one hand and the BPOAI engine on the other. Moreover, BPOAI technology needs to work with several types of technologies, including message-oriented and transaction-oriented middleware.

There are many differences between more traditional application integration and BPOAI. Notable differences include:

- A single instance of BPOAI typically spans many instances of traditional application integration.
- Application integration typically means the exchange of information between two or more systems without visibility into internal processes.
- BPOAI leads with a process model and moves information between applications in support of that model.
- Application integration is typically a tactical solution, motivated by the requirement for two or more applications to communicate.
- BPOAI is strategic, leveraging business rules to determine how systems should interact and better extract the business value from each system through a common abstract business model.

As I've said, BPOAI views middleware as a commodity, with the ability to leverage both message-oriented and transactional middleware as points of integration into any

 ©2001 Cutter Information Corp.

number of source or target systems. In fact, most integration servers and application servers are beginning to offer BPOAI tools that support their middleware technology. BPOAI generally provides easy-to-use visual interfaces for binding these processes together.

While some may question the relevance of BPOAI, I would argue that it is the ultimate destination of application integration (acknowledging that we still have a long way to go to perfect the middleware). Despite current shortcomings, many application integration vendors are aggressively promoting BPOAI as a vital component of their application integration technology package. In doing so, their strategy is clear — they are anxious to join the world of high-end, BPOAI modeling tools. They hope that their application integration–enabled middleware, such as integration servers and application servers, will accomplish just that.

BPOAI is best defined as applying appropriate rules in an agreed-upon logical sequence, in order to pass information between participating systems and to visualize and share application-level processes, including the creation of a common abstract process that spans both internal and external systems. This definition holds true whether or not the business processes are automated. For example, processing an insurance claim and delivering a car to a customer are business events that can be automated with BPOAI.

Skateboard Integration

Using a more detailed example of BPOAI, let's say there are three companies that participate in a trading community: Companies A, B, and C. Company A produces parts for skateboards, while Company B assembles and tests the skateboards, and finally, Company C sells the skateboards. Each company has its own set of native processes and internal systems: a production system, an assembly system, and a sales system, respectively.

In order to integrate these applications, the trading community has decided to implement BPOAI by defining a common process model that spans all the companies and internal systems. This process model defines a sequence and logical order of events from the realization of consumer demand, purchase of raw materials, creation of the parts, assembly of parts into a product, product testing, and finally, sale of the product to the ultimate consumer. This common model integrates with local systems through visibility into their internal application processes, if possible, or perhaps through more primitive layers such as the database or application interface. What's important is that the common process model can produce events that are understood by the systems participating in the process as well as react to events that the applications communicate back to the BPOAI engine.

The use of a common process model that spans multiple companies for application integration provides many advantages, including:

- The ability to create a common agreed-upon process between companies that automates the integration of all information systems to react to business events, such as increased consumer demand, material shortages, and quality problems, in real time.
- The ability to monitor all aspects of the business and trading community to determine the current state of the process in real time.

Despite current shortcomings, many application integration vendors are aggressively promoting BPOAI as a vital component of their application integration technology package.

- The ability to redefine the process at any given time to support the business and thus make the process more efficient.
- The ability to hide the complexities of the local applications from the business users and have the business users work with a common set of business semantics.

To this end, there are three main services that BPOAI provides: the visualization of processes contained within all trading partner systems, interface abstraction, and real-time measurement of business process performance.

The use of graphics and diagrams provides a powerful tool for communication and consensus building.

By visualizing enterprise and cross-enterprise processes contained within trading partners, business managers can become involved in enterprise integration. The use of graphics and diagrams provides a powerful tool for communication and consensus building. Moreover, this approach provides a business-oriented view of integration scenarios, with real-time integration of the enabling middleware or points of integration. This gives business analysts the ability to make changes to the process model, implement it within the trading community, and, typically, not involve the respective IT departments.

Interface abstraction refers to the mapping of the BPOAI model to physical system interfaces and the abstraction of both connectivity and system integration solutions from the business analyst. BPOAI exists at the uppermost level in the application integration middleware stack. Those who use BPOAI tools are able to view the world at a logical business level and are not limited by physical integration flows, interfaces, or adapters. What's more, the middleware mechanisms employed are also abstracted and are thus not a concern of the business process analyst, as long as the common process model is interacting correctly with all source and target systems that exist within all companies.

Although each BPOAI tool and project may take a slightly different approach, the internal process of interacting with the physical systems typically consists of the following set of events:

1. The source system that exists inside of a company posts an event to the BPOAI engine; for instance, a skateboard is sold.
2. The event is transformed, if required, so that it adheres to a standard set of business semantics and information processing mechanisms (synchronous versus asynchronous). This will be engine dependent, but there must always be a common set of process semantics and information processing mechanisms defined at the engine level so the analyst can make sense of a business process that spans many types of applications, platforms, and databases.
3. The BPOAI engine reacts to the event, once transformed, invoking other processes in other systems to support the execution of the common process model. For instance, if a skateboard is sold, then the source system sends an order to the skateboard assembler, posting an event from the process engine to the assembler's target system, typically over the Internet.
4. Once it receives that event, the local system reacts per its internal processes and posts an event back to the process engines (say, when the skateboard is assembled).
5. The common process model sequences the master process, sending and receiving other events in support of the common process model. This is an

 ©2001 Cutter Information Corp.

ongoing activity, with information moving up to the process engine from the local systems, transformed, if required, and down from the process engine to the local systems in support of the execution of the process model.

Another way to view the process of creating a BPOAI model is to consider it as defining the hierarchy of processes within the trading community. This means that smaller subprocesses can be linked at the lower tier of integration or remain native to the source or target systems. Building up from the lower-level processes to the higher-level processes, you may link the subprocesses into higher-level processes within the domain of the trading community.

The measurement of business process performance gives the BPOAI team the ability to analyze a business in real time. By leveraging tight integration with the process model and the middleware, business analysts can gather business statistics in real time from the trading community; for instance, the performance of a supplier shipping goods to the plant, and the plant's ability to turn those raw materials into product.

Moreover, BPOAI provides the technology user with the ability to track and direct each instance of a business process; for example, processing individual orders or medical insurance claims through a lifecycle that may consume seconds, minutes, hours, days, or weeks. Finally, the organization needs to measure and maintain contextual information for the duration of a process instance that spans many individual activities.

The goal of BPOAI, and of application integration in general, is to automate the data movement and process flow so that another layer of BPOAI will exist over and above the processes encapsulated in existing systems. In other words, BPOAI completes application integration, allowing the integration of systems not only by readily sharing information, but also by managing information sharing with easy-to-use tools.

In general, BPOAI logic addresses only process flow and integration. It is not a traditional programming logic, such as user interface processing, database updates, or the execution of transactions. Indeed, in most BPOAI scenarios, the process logic is separated from the application logic. It functions solely to coordinate or manage information flow between the many source and target applications that exist within organizations.

SERVICE-ORIENTED INTEGRATION

Service-oriented application integration allows applications to share common business logic, or methods. This is accomplished either by defining methods that can be shared, and therefore integrated, or by providing the infrastructure for such method sharing (such as Web services). Methods may be shared either by being hosted on a central server, by accessing interapplications (e.g., distributed objects), or through standard Web services mechanisms (such as .NET).

There is a long history of attempts to share common processes, one that began more than 10 years ago with multitiered client-server — a set of shared services on a common server that provided the enterprise with the infrastructure for reuse and, now, for integration — and the distributed object movement. "Reusability" is a valuable objective. A common set of methods among enterprise applications invites reusability and, as a result, significantly reduces the

need for redundant methods and/or applications.

While most methods exist for single-organization use, there are times when it makes sense to share methods between organizations. In a new twist on the long-standing practice of reusability, we are now hoping to expand method sharing beyond the enterprise itself to trading partners. For example, we might want to share a common logic to process credit requests from customers or to calculate shipping costs using a set of Web services.

In a new twist on the long-standing practice of reusability, we are now hoping to expand method sharing beyond the enterprise itself to trading partners.

Unfortunately, absolute reuse has yet to be achieved on the enterprise level. It is an even more distant goal between trading partners. The reasons for this failure range from internal politics to the inability to select a consistent technology set. In most cases, the actual limit on reuse results directly from a lack of enterprise architecture and central control.

Utilizing the tools and techniques of application integration gives us the opportunity to learn how to share common methods. More than that, these tools and techniques create the infrastructure that can make such sharing a reality. By taking advantage of this opportunity, we integrate applications so that information can be shared, even as we provide the infrastructure for the reuse of business logic.

Sounds great, doesn't it? The downside might give you pause. This "great sounding" solution also confronts us with the most invasive level of application integration, and thus the most costly. This is no small consideration if you're considering Web services, distributed objects, or transactional frameworks.

While information-oriented and interface processing–level application integration efforts generally do not require changes to either the source or target applications, service-oriented application integration requires changes to most — if not all — enterprise applications to take advantage of the paradigm. Clearly, this downside makes service-oriented integration a tough sell.

Changing applications is a very expensive proposition. In addition to changing application logic, there is the need to test, integrate, and redeploy the application within the enterprise, a process that often causes costs to spiral upward. This seems to be the case no matter how you approach service-oriented application integration, whether with older technologies such as CORBA or with .NET, the latest service-based architecture that's come down the road.

Before embracing the invasiveness and expense of service-oriented application integration, enterprises must clearly understand both its opportunities and its risks. Only then can its value be evaluated objectively. The opportunity to share application services that are common to many applications — and therefore make it possible to integrate those applications — represents a tremendous benefit. However, this benefit comes with the very real risk that the expense of implementing service-oriented application integration will outpace its value.

APPLICATION INTEGRATION: CLEARLY THE FUTURE

Application integration provides a clear competitive advantage for most industries, an advantage that includes the ability to do business at light speed, along with the ability to satisfy customer demand in record time (by using automated processes instead of paper, faxes, and humans). We are truly moving forward into a digital economy where business runs within and between

 ©2001 Cutter Information Corp.

computers, where everything is automated, and customers learn to expect no less than instantaneous access to information.

As corporate dependence on technology has grown more complex and far reaching, the need for a method of integrating disparate applications between enterprises into a unified set of business processes in support of e-business has emerged as a priority. After years of creating islands of automation within each company, users and business managers are demanding that seamless bridges now be built to join these islands together, thereby allowing commerce to proceed in real time. In short, they are demanding that ways be found to bind these applications into unified business applications. The development of application integration allows many of the enterprise applications that exist today to share both processes and data. Thus, it allows us to finally answer the demand for realtime, intra- and intercompany application integration in support of the next generation real-time economy.

David S. Linthicum is the CTO and senior vice president of research and development with Mercator Software (www.mercator.com) in Boca Raton, Florida. He is the author of nine books, including his latest, B2B Application Integration — e-Business-Enable Your Enterprise*, and the groundbreaking* Enterprise Application Integration*, now in its fifth printing.*

Mr. Linthicum can be reached at Mercator, Peninsula Plaza, Suite 250, 2424 Federal Highway, Boca Raton, FL 33431, USA. Tel: +1 561 394 3400; Fax: +1 561 394 3470; E-mail: Linthicum@att.net.

CUTTER IT JOURNAL

XML as Glue for Enterprise Integration

by Don Estes

With the bursting of the dot-com bubble over the last year, some of the pressure for pushing IT departments onto the Internet has lessened. This is all to the good, as many companies were adopting off-the-shelf solutions that really did not fit their needs or were building poorly thought-out B2B or B2C implementations. Not only did these implementations fail to address internal integration issues, they frequently revealed the shortcomings of creating a largely standalone system for the Web that integrated poorly with the legacy systems that run the enterprise. Once again, today's solutions were in danger of becoming tomorrow's problems.

Even though the overhyped and supercharged motivations have fallen back more or less in line with reality, there is a genuine business case for buying or building solutions that scale to the Web. Now that we have the time to think the problem through, we have the opportunity to do it right. If we do so, we can solve long-standing problems of enterprise integration at the same time, while avoiding the creation of new problems.

WEB INTEGRATION = ENTERPRISE INTEGRATION

When an enterprise creates a new Web-based application, it is rare for that application to exist in a vacuum. Integrating that application with existing applications on one or more concurrent platforms is no different from requirements for application integration that have needed filling for some time. What is needed is a universal integration solution that will equally satisfy requirements for Web/client-server integration and Web/mainframe integration, as well as satisfying any other requirement, including mainframe/mainframe integration of 30-year-old COBOL applications.

When the problem is viewed as one of data exchange, there is no significant difference between the requirements of one platform and another. XML comes into the equation because it was designed from the ground up to satisfy all data exchange requirements. Indeed, it does so in a practical manner, once you look past the hype. XML can and should be the glue used to bind multiple applications together. In this regard, my colleagues and I confidently predict that XML will be as revolutionary as the introduction of SQL databases 20 years ago, and it will have as profound an impact as the databases have had, as the technology is absorbed over time.

MAINFRAME AS FOSSILIZED RELIC

There is a common attitude that mainframe organizations encounter when dealing with firms that specialize in creating Web applications. The mainframe is treated at best as a necessary evil, sometimes with thinly disguised disdain. Few Web services firms really understand the mainframe environment, and their preference is to treat any mainframe as a completely static black box.

In some cases, it makes good business sense to get off of the mainframe platform and, therefore, to freeze existing applications. For these organizations, treating the mainframe as a fossilized relic is at least reasonable, if a bit disrespectful of those

 ©2001 Cutter Information Corp.

staff members who have invested a big chunk of their life in the applications on that platform. For these organizations, however, the application integration question is actually more complex. Either legacy system processing is being moved gradually to new applications on new platforms, or the old applications are being ported to the new platform and modernized in the process. Either way, integrating an application in transition is much more difficult, unless the new application or the modernization process includes XML interfaces.

However, for many mainframe enterprises, and for most large mainframe enterprises, the mainframe is not a fossil. Rather, it is the platform of choice for sound business reasons, and it is not static but undergoing regular maintenance on old applications as well as new development. This essential fact must be understood and respected by Web services firms if they are to do the job that is needed rather than the job they perceive or the one that best fits their business operational model.

Interestingly, if you understand both mainframes and Web services well, doing it right may be as easy or easier than current practice. This can mean being willing to make minor changes to programs. Existing products and the services based on them go to great lengths to avoid recompiling a single program, and in the process may add complexity, fragility, and expense rather than reduce it as claimed by their respective vendors. As the saying goes, if your only tool is a hammer, every problem looks like a nail.

These middleware products have their place, and many of them are very respectable implementations, but what is needed is a flexible approach that uses middleware when that makes sense and changes programs when that is the better solution. One implication of this is that the staff members who know the platform and the application well must be closely involved in the design and implementation and respected for their knowledge and contributions.

XML AS GLUE

The latest technical strategies for the future of computing — .NET from Microsoft, e-speak from HP, Open Network Environment from Sun, and similar visions from IBM, Oracle, and others — all involve a key element of platform agnosticism. Mainframes, Unix servers, and legacy client-server platforms can all participate equally well as peers in the democratic world of Web services. The 20-somethings who know only Java and Oracle are just advertising their own ignorance of modern technology when they implicitly consign the legacy platforms to the dustbin of history. Phasing out legacy platforms should be done for good and valid business reasons, not simply to enable Web access to that application.

Two other aspects of these advanced computing architectures are relevant to this discussion: loosely coupled messaging and XML. Like the Internet in general, these strategies all depend on loosely coupled messaging, not tightly coupled remote procedure calls (RPCs). Messaging provides simplicity, which in turn provides robustness and removes the need for the user to have any knowledge of the platform processing the message. Most mainframe transaction processing applications were designed on a messaging model, not an RPC model, so these are usually more or less conformant already. However, this may be a problem for many client-server systems.

The 20-somethings who know only Java and Oracle are just advertising their own ignorance of modern technology when they implicitly consign the legacy platforms to the dustbin of history.

The key to all these advanced designs is XML-based data exchange. When two applications need to exchange data, the most critical problems that arise are always in one of two forms, if not both. First, there is the obvious problem of the format of the data records themselves. One format may have NAME as a single field, while the other may break it out into LAST-NAME and FIRST-NAME. NAME may be one field of 50 bytes for the one format, while the other format may have two fields of 30 bytes each, and so on.

Instead of exchanging files or records, integration with XML involves the exchange of documents.

Second, there is the problem of the *meaning* of the data. We may think we know what LAST-NAME and FIRST-NAME both mean, but what about in Spanish-speaking countries where people commonly have two family names? What about Asian countries where the family name is commonly given first? What about those countries where people commonly have only a single name? Then there is the format problem, best known since Y2K for date fields. For example, one application could have SETTLEMENT-DATE (MMDDYY format) versus SETTLEDATE (ISO format), and so on. More subtly, one application could have settlement as three days after the transaction date, while the other might be one day after. The larger the scope of integration, the more the problem of data definition and translation comes to dominate the design.

A BRIEF INTRODUCTION TO XML

Instead of exchanging files or records, integration with XML involves the exchange of documents. These are not word processing documents (although they could be, given the power of XML), but a formally defined sequence of data elements and attributes presented in conformance with the XML syntax. The HTML pages that we display all the time meet the definition of a document in this sense. View any HTML file with a text editor, and you will see an HTML document that is very similar in form to an XML document.

XML and HTML are both tagged languages. This means that elements are wrapped with indicative tags in angle brackets, such as

```
<BOLD>bold text</BOLD>
```

in HTML, indicating how the text is to be displayed. Similarly, an XML data element is described with tags, as

```
<JOURNAL>Cutter IT
Journal</JOURNAL>.
```

Because an XML document is physically a string of text, it can be exchanged between highly dissimilar platforms without damage by character set translation. Because each data element is presented as a variable length string, it is not tied to the length and format restrictions of the data field or database column from which it was drawn. Because each data element can have attributes, we could even get around the problem of format within a data element, as

```
<SDATE format="ISO">
20010604</SDATE>.
```

Crucially, the document may contain data elements or data attributes that the user is not interested in and is not obligated to handle. XML allows the application programmer to take what he or she needs from the document and ignore the rest. As we shall see, it is these seemingly simple characteristics that make XML a firm foundation for data exchange.

An XML document consists of a header and a series of data elements. A message consisting of an XML document would likely contain the same data as an equivalent

 ©2001 Cutter Information Corp.

fixed-format message, with the contents of each data field from the fixed-format record surrounded by the appropriate indicative tag, perhaps with optional attributes. This document presents the data elements as sequential text, one element after another. This is the "serialized" form of the document used for exchange. Within a program, however, the document is parsed into a tree structure, allowing for random access to the data in the tree. If you think of the directory structure on your hard drive, Windows Explorer allows you to view that structure as a tree, so that you may expand or collapse the view of any branch of the tree at will. A message formatted as an XML document will likely contain only a single instance of the relevant tagged data elements, but a file formatted as an XML document will contain repeated instances of the tagged data elements, one instance for each record in the file. XML does not distinguish between message documents and file documents, although we will want to do so in the design of a universal integration solution.

XML is a general specification that includes a number of subsidiary capabilities. A few are important to address separately as background for the discussion of enterprise integration:

XPATH

XPATH allows a programmer to randomly or sequentially access data from an XML document that has been parsed into a tree structure in the program's memory space. It is this capability that allows a user to select what is needed for the task at hand and ignore the rest.

XSL

XSL, or eXtensible Stylesheet Language, allows a user to take the data from an XML document and present it in a usable form. This might be a screen display or a report. XSL transformation engines, such as Xerces from IBM, can be easily used to take an XML document and an XSL stylesheet as input and give HTML as output. They could also produce as output a text file in fixed, 132-column traditional layout; text formatted in PCL for output to an HP laser printer; or indeed any desired output format. Mainframe interactive programs typically provide both data and presentation characteristics intermixed as a single message. Separating data and presentation has been a long-standing problem in updating mainframe programs to use browser technology. XML solves this problem very neatly by encoding the data as XML data elements and encoding any dynamic presentation characteristics as data attributes. Screen output XSLs can take advantage of those attributes, while report output XSLs may ignore them.

XSLT

XSLT is a subsidiary capability of XSL that is important in its own right. XSL provides the specifications to transform an XML document to HTML or any other visual presentation, into plain or formatted text for reporting purposes, or into another XML document. Although XSLT is the engine that does all three transformations, the third capability is critical for solving the problem of translating between different definitions of similar information.

XML Schemas

An XML schema specifies the definitions for the information that can be coded into a given XML document. Use of schemas is optional, unlike the data definitions for database management systems. As a result, XML documents have no active enforcement of data definitions. However, XML schemas can provide a functional

equivalent, because XML documents can be validated against a given schema using a validating parser. Although not strictly necessary for enterprise integration, it is our opinion that without this facility to enforce specification discipline, the enterprise risks a loss of data integrity.

GLUING APPLICATIONS TOGETHER WITH XML

With XSLT, we can reap the benefits of XML now, without waiting for the standards to solidify.

With that as background, let us now consider how we might address existing and future integration requirements using XML. If we have two applications on the same platform that need to exchange data, we would traditionally build a sequential file output of the one and input to the other to do so, or construct a messaging interface to operate at a record level. However, any time the format changes on one side or the other, all relevant programs must be changed and introduced into production use at the same moment. Sometimes, multiple files will be created, containing overlapping data.

XML solves both these problems. Because XML encodes data without consideration for format, if the format changes on one side but not the other, the XML decoding will automatically adjust to the format changes. Because XML decoding takes only what is needed and ignores the rest, a single file encoded in XML can feed multiple subsidiary processes for the target application. In the largest mainframe organizations, maintaining this interapplication data exchange is often a constant battle for operations, day in and day out.

However, this addresses the fairly simple case where we have only two applications feeding data back and forth. In reality, things are never so simple. Consider a case drawn from one of our clients. This company supports multiple, message-based data exchanges among 17 of its own clients. All of these are point-to-point exchanges, so at last count, there are over 1,000 formats that have to be monitored. Two or three change every single day, sometimes without advance notice. Because these are time-critical financial transactions, there is almost no tolerance for error or delay.

Our recommendation has been to define an XML schema for each of the primary sets of data, and then to replace the existing routines that transmit the data exchange messages with XML encoding and decoding routines that utilize the universal schema definition. In this way, not only do the 1,000 formats collapse into less than 50 or so, but a message can be validated against the schema upon receipt. If it fails validation, then it can be returned to the originating client immediately without the receiving client's accepting any financial liability for failing to execute the transaction in a timely manner.

However, there is one problem with this recommendation. One of the company's clients has begun to create its own XML definitions, referred to as "dialects" in the XML vernacular. Other clients want to adopt a dialect from one of the industry standards-setting initiatives, even though such dialects are insufficient for the current data exchanges. Still others want to wait until everything is settled and a universal industry dialect emerges, even though this will be many years in the future if it ever occurs. The solution here is XSLT. So long as there is a way to map data elements between one dialect and another, XSLT can implement the mapping quite readily, without any programming on the recipients' respective mainframes or client-server systems. Therefore, we can reap the

 ©2001 Cutter Information Corp.

benefits of XML now, without waiting for the standards to solidify.

UNIVERSAL INTEGRATION SOLUTION

This simplest form of application integration involves doing what we do now — sending data back and forth between applications via the exchange of batch data files or real-time messages. XML will definitely address this problem. But the real task for robust application integration is not replicating data in multiple databases, but rather storing it in one place while making it readily available to all qualified users. Note that these users could be on the same platform, on different platforms at the same physical location, or scattered around the world. From this point of view, replicating data might be desirable from a performance-tuning point of view, just as we denormalize relational databases to gain performance, but it would never be necessary for functional reasons.

There is a serious problem with replicating data stores and data queries on multiple platforms. In addition to the obvious problems of databases' getting out of synchronization with one another and the time delays in replication, there is the much worse problem of exactly duplicating the business rules associated with the data for each instance. For example, consider a data warehousing system in which the data from the mainframe database has simply been exported to a database server. The data is all there, exactly as on the mainframe, but when the users attempt to query against that data, they get different results from the server and from supposedly equivalent mainframe queries. Worse, the discrepancies may seem to be intermittent, sometimes right and sometimes wrong, for reasons that are not obvious to the end user. This leads to a fundamental lack of trust in the query against the server.

These discrepancies can always be resolved, of course. The operational issue is how long it takes to diagnose each discrepancy, resolve it, and ensure that other, rarer discrepancies are not waiting to reveal themselves. More importantly, as the business rules evolve on the primary platform, the replicated rules must change in precisely the same way, in synchronization.

Whatever the difficulty in precisely replicating query business rules, there is greater concern regarding the precise replication of update business rules. This is because the damage caused by an incorrect update is usually greater than the damage caused by an incorrect query. For this reason, it is not unusual for a organization to allow direct query access to data but route all update requests through the standard transaction processors that contain complex but trusted validation logic.

All of this complexity devolves from the simple case of lack of trust in the replicated processes. However, whenever we undertake a project to replicate existing processing, we always optimistically presume that we can exactly duplicate what we have now. The reality is that this is much, much more difficult than we ever expect, and it is made more difficult still if the programmers attempt to "clean up" some of the queries in the process.

The practical solution to this problem is the creation of trusted data components (TDCs), a new tier in an *n*-tier application architecture. These components reuse existing queries, obviating the need for duplication of query logic, yet expose them to user requests from any platform via a simple-to-use mechanism. Having a single point of maintenance also reduces

to user requests from any platform via a simple-to-use mechanism. Having a single point of maintenance also reduces maintenance costs while ensuring that the result will be trusted by all users.

The physical implementation may take one of two forms. It may consist of a gateway to the primary host, a gateway to the secondary host once the replicated rules are proven, or both. Having a dual pathway has several interesting implications. First, if there is a valid reason to go to the primary source in any given instance, that pathway can be forced. Second, the component can provide diagnostics, if a variance is suspected, by querying both pathways and comparing the results. Third, in the case of an outage in one pathway or a need for load balancing, the TDC can be switched to the other without causing a disruption at the end-user application. Finally, if there is a maintenance upgrade being applied to the primary source, the TDC can be switched to the primary pathway until the corresponding upgrade of the secondary pathway has been proven.

BUILDING A WEB APPLICATION WITH TDCs

XML comes into this reuse picture as we contemplate building a Web application. We have a very simple Web application example on our Web site (www.forecross.com). It requests an account number and returns the name on that account and the current selling price for the stock symbol associated with that account number.

From the point of view of the Java programmer building the application, each source of information is an XML schema and a URL, together defining a Web service. For the account number look-up, the URL ultimately resolves itself at a component that is in fact a transaction processor running under CICS on a mainframe accessing VSAM files. The other URL resolves to a standard Web service that provides the near real-time stock quotation, from xmethods.com. The programmer formats an XML document to request data according to that schema and sends it to the first URL. The response is mapped into a new XML document according to the second schema and sent to the second URL. The second response is formatted as HTML and displayed on the browser. This is a 21st-century integration solution, and the infrastructure to support it provides a universal integration solution.

This approach to integration provides several advantages. We do not need to concern ourselves with the format of the transaction, only the XML data tags defined by the associated XML schema. The COBOL program on the mainframe decodes the XML into the normal CICS input map area, runs its normal logic just as though the input had come from a 3270 terminal, and encodes the response into an XML document. In fact, had we wanted to, we could have simply taken an XSL stylesheet to format the returned XML document into an HTML document for display on a browser and replaced the 3270 access with browser access.

We do not need to concern ourselves with the location of the data we want, its access method, the hardware or software platform, or any other aspect. All of that is provided by the TDC, accessed by its URL via the associated schema. In fact, the programmer neither knew nor cared that one of the transactions was physically on a mainframe. This would be equally true of a legacy client-server or any other platform following the same approach.

 ©2001 Cutter Information Corp.

messages across the Web, the XML document was an asynchronous message, which provides the robust performance expected of Web applications.

Finally, and most importantly, this design expects a dynamic, constantly changing infrastructure but is ready to continue operating without missing a beat at each change. Processing loads can be balanced between servers, routed by intelligent TDCs, without the Web application's being aware of any changes. As the transaction processor is updated over the course of normal maintenance, as the application changes platforms, as the data moves into a relational database, or indeed as any other of the changes that occur in daily operation or maintenance programming are encountered, the design moves immediately to accommodate them. This is the power of XML applied to enterprise integration.

SUMMARY

The challenge of integrating applications across an enterprise is just a subset of the larger challenge of integrating applications between enterprises on the Web. Integration is based on data exchange, either message based or file based. In either case, XML provides resilience in the data formats and definitions that pass between applications.

Given an XML interface in each application and a ready presence on the Web, either the Internet or an enterprise intranet, any legacy platform becomes a peer in the provision of Web services. Each Web service is defined by an XML schema and a URL, and accessed via trusted data components in an *n*-tier architecture. The TDCs ensure data integrity while providing for operational load balancing and conformance to the reality that both legacy and new platforms in the enterprise operate in a dynamically evolving environment.

Don Estes is a Cutter Consortium Senior Consultant and a specialist in IT management and technical issues, with particular focus on balancing the business and technical risks associated with major IT initiatives and daily operations. Currently, Mr. Estes is applying this experience to products and services for the automated modernization and testing of legacy applications being interfaced to the Internet via XML data exchanges. For the past 10 of his 27 years in the IT industry, he has been involved in the project architecture and implementation of major hardware and software platform migration projects, with an emphasis on developing cost-effective automated regression testing strategies and tools. He is an expert in the disciplines and technologies that support such migration: data flow and logic flow analysis, establishing valid metrics for comprehensiveness and sufficiency of testing, language processing and source code instrumentation, and ensuring optimum performance of project results. Mr. Estes' writings and presentations have made significant contributions to the technical literature, particularly in the area of managing the technical risks of major software projects, automating testing to reduce project cost and risk, and independent validation of testing adequacy.

Mr. Estes can be reached at Tel: +1 781 860 5277; E-mail: destes@cutter.com.

Editorial Board

Edward Yourdon, Editor Emeritus
Edward Yourdon is widely known as the lead developer of the structured analysis/design methods of the 1970s. He was a codeveloper of the Yourdon/Whitehead method of object-oriented analysis/design and the popular Coad/Yourdon OO methodology. Mr. Yourdon focuses on business-IT strategies; mitigating risks of large outsourcing initiatives; auditing of large, risky projects; development and implementation of e-business initiatives; and tracking of critical business/IT "mega trends" in the coming decade.

Mr. Yourdon has been involved in a number of pioneering computer technologies, such as time-sharing operating systems and virtual memory systems. In 1974, Mr. Yourdon founded YOURDON Inc. to provide educational, publishing, and consulting services in state-of-the-art software engineering technology. YOURDON Inc. was sold in 1986 and eventually became part of IBM. The publishing division, Yourdon Press (now part of Prentice Hall), has produced over 150 technical computer books on a wide range of software engineering topics.

Mr. Yourdon was an advisor to Technology Transfer's research project on software industry opportunities in the former Soviet Union and a member of the expert advisory panel on I-CASE acquisition for the US Department of Defense. He is currently a member of the Airlie Council, a group of high-level advisors formulating software "best practices" for the US Department of Defense. Mr. Yourdon has authored more than 200 technical articles and 25 computer books since 1967. His recent books include *Death March* and *Rise and Resurrection of the American Programmer*.

Larry L. Constantine divides his time between Australia and the US. A pioneer of software engineering, Mr. Constantine is the originator of structured design. He developed the concepts of coupling and cohesion, and he introduced such widely used tools as data flow diagrams. His work now focuses on approaches to improve the usability of software to support rapid visual development.

Bill Curtis is one of the most recognized authorities on software development and human factors in computer systems. He is cofounder and chief scientist of TeraQuest Metrics in Austin, Texas, a firm that provides management consulting and training in process improvement. As director of the Process Program at the Software Engineering Institute (SEI), he led the team that published the *Capability Maturity Model for Software* (Software CMM).

Tom DeMarco is a principal of the Atlantic Systems Guild, a computer systems think-tank with offices in the US, Germany, and Great Britain. He was awarded the 1986 Warnier Prize for "lifetime contribution to the field of computing." Mr. DeMarco's career began at Bell Telephone Laboratories, where he served as part of the now-legendary ESS-1 project. He managed real-time projects for La CEGOS Informatique in France and was responsible for distributed online banking systems installed in Europe.

Peter Hruschka is a principal of the Atlantic Systems Guild, an internationally renowned group of software and system technology experts. He is a specialist in technology transfer for software and system engineering. Dr. Hruschka conducts surveys to determine the capabilities of organizations, constructs strategic plans, offers seminars and workshops for both structured and object-oriented methods, provides coaching and consulting, and performs project reviews.

Tomoo Matsubara is an independent consultant who works with software organizations in equipment manufacturing companies. Between 1970 and 1991, Dr. Matsubara was chief engineer and project manager of Hitachi Software Engineering in Japan. He was a primary contributor to establishing a management infrastructure, such as company-wide software metrics, multiperspective project management systems, and software process assessment, in the early stages of the company.

Navyug Mohnot is executive director of the Quality Assurance Institute India Limited. QAI India helps organizations improve software processes and leverage the SEI CMM, ISO 9000, SPICE, and other approaches. Mr. Mohnot's areas of expertise include software metrics, CASE, estimation, and quality.

Roger Pressman is the president of R.S. Pressman & Associates and an internationally recognized consultant and author in software engineering. Dr. Pressman specializes in helping companies establish effective software engineering practices. He is the developer of Process Advisor, the industry's first self-directed software process improvement product, and *Essential Software Engineering*, a comprehensive video curriculum.

Howard Rubin is chair of the Department of Computer Science at Hunter College, and president of Rubin Systems Inc. Dr. Rubin has published major software engineering books and papers on metrics and quality. He is internationally recognized for work in the areas of software process dynamics, software metrics, the business value of technology, and project "navigation." He has provided expert advice and analysis on large-scale project management, oversight, and outsourcing and is the creator of a leading tool for software estimation and planning.

Paul A. Strassmann has served as CIO of Kraft and Xerox and director of defense information for the US Department of Defense. Mr. Strassmann is currently CEO of Software Testing Assurance Corporation, president of Information Economics Press, chairman of Method Software, and a director of McCabe Associates and Meta Software Corporation. His books include: *Information Payoff, Business Value of Computers, Politics of Information Management*, and *The Squandered Computer*.

Rob Thomsett of Australia and Canada is director of The Thomsett Company and has been consulting and educating in the area of project management, teams, and quality since 1974. He is the author of *People and Project Management*. During his 26 years in computing, Mr. Thomsett has continued to explore the relationship between project and team management and the broader issues of the impact of computing on organizational culture.