Tools for Change: Enabling Effective Application Integration Projects

A White Paper

Published: May 2003

Abstract

This white paper presents an overview of integration projects and the unique environment in which they exist, as well as Solstice Integra Enterprise, a suite of tools that enables organizations with complex application integration projects to manage and complete those projects on-time and on-budget. By providing the tools to simulate, test, and validate an integration project at the beginning of the project—rather than the end of the project, when identifying and remediating problems is complicated and costly—Solstice Integra Enterprise enables an organization to identify issues while they are still easy to fix, which improves an organization's ability to bring the project to completion on time and without budget overruns.

This white paper is intended for individuals who want to understand the business benefits of Solstice Integra Enterprise. The overview presents the components of the solution and examples of real-world deployments in which the solution provided significant benefits.

Contents

Introduction

Complex integration projects—those massive projects that span multiple systems, applications, services, groups, even companies—are fundamentally different from application development projects. Stated this way, this sounds intuitively obvious—yet many organizations continue to approach complex integration projects as though there were no differences. They use the same tools and methods that they would use for an application development project—and all too often end up with an integration project that is over budget and off schedule.

The fundamental difference between these two types of projects is this: an integration project involves far less actual coding than an application development project, but much more assembly and testing of individual components. This remains true whether the integration project involves the automation of a new business process across an existing group of applications or services or whether it involves the integration of Web and legacy system functionality to create an entirely new business function. In the traditional application development cycle, all the testing takes place after the coding is done, but testing an integrated application at that same point in the project cycle leads to problems and complications that are difficult and costly to resolve. These, in turn, lead to cost and schedule overruns.

Approaching complex integration projects with tools specifically designed for these types of projects can help deliver these projects on time and on budget. Solstice® Integra Enterprise® enables a project management team to simulate, test, and validate an integration project at every step of the project—rather than the end of the project. This enables integrators to test the flow of messages through individual components *before* trying to bolt these components together. Integrators can test and validate that the output from one system or application is clean before it goes into the next system or application. Then they can validate that the output from that system is clean before it goes into the next system—and if there is an issue they know exactly where they need to focus their remediation efforts.

As a result, when all the applications and systems are finally linked together, the integration team has already validated the message flow through the individual components and there are no surprises. The new application works, and it is available on schedule and on budget.

The Challenges Posed by Complex Integration Projects

Application integration projects are different from standard application development projects. This should be self-evident, yet IT organizations around the world have long approached application integration projects as though the differences were minimal, something that they could overlook. Unfortunately, taking a typical application development approach to a complex application integration project does not work, and the many projects that have not met expectations—to say nothing of coming in buggy—stand as a painful testament to this reality.

Wherein lie the critical differences? Both projects involve coding, assembly, testing, and implementation. Critical differences, though, lie in the weight that each of these activities carries. An application integration project typically involves far less actual coding and far more assembly than does a typical application development project. Indeed, in the case of an application integration effort, the assembly involves existing applications and systems, not individual files of compilable source code sitting in a carefully-guarded repository. These systems and applications—usually numbering no fewer than three and often numbering in the double digits—are likely to reside in different locations and be managed by different teams. They are likely to use different communications protocols and rely on a wide range of open or proprietary interfaces.

Critical Business Success Factors for Integration Projects:

1. Build a different team
   - A core group responsible for the big picture
   - Liaisons with each application group responsible for component communication and delivery
2. Build requirements focused on interconnections in the dimensions of both flow and function
3. Create an assembly phase of the project
4. Use tools that support integration process and technology
5. Combine business and technical people in requirements assembly and testing

In addition, each one of the components in an application integration project may act upon the input it receives. It is not merely the case that the CRM system will accept input from a self-service Web portal and create a new customer account, but that the CRM system will then add information about the new customer account to the message snaking its way through the network of integrated systems. Each application that ingests the message, from a Web server to a CRM system to an ERP system and on to a legacy financial system, may return something different as output, and one of the critical tasks of application integration is to make sure that the information going into these systems is appropriate at each step.

A History of Misses

The mid-1990s saw a dramatic increase in the number of vendors trying to simplify the creation of integrated systems and applications through the use of centralized middleware solutions—Tibco, Vitria, webMethods, and SeeBeyond, for example. Still more recently there have been entries from larger industry players, including IBM's MQSeries and BEA's WLI. Industry standards such as Common Object Request Broker Architecture (CORBA), Extensible Markup Language (XML), Java 2 Enterprise Edition (J2EE), and Web services have also evolved in an attempt to bring order to this need for integration. While these tools have greatly improved the integration infrastructure—from a technology perspective—the integration projects themselves still have not delivered on their promise.

These middleware solutions and architectures, elegant though they may be, have not addressed the unique organizational makeup of integration projects. It is essential that the development, testing, and support teams that contribute to the delivery of integrated applications—including the integration broker itself—work effectively and efficiently. Yet none of these solutions provide the tools to accomplish this, and the teams become hopelessly dependent on each other.

Let us examine the structure of an integration project and then analyze what needs to be in place to enable an effective integration project.

Anatomy of an Integration Architecture

To integrate independent applications, virtually all organizations take a layering approach (see Figure 1) to establish a workable level of commonality while leveraging their existing technology. While most integration projects leverage vendor integration brokers on the market, some organizations have created this layered approach internally.

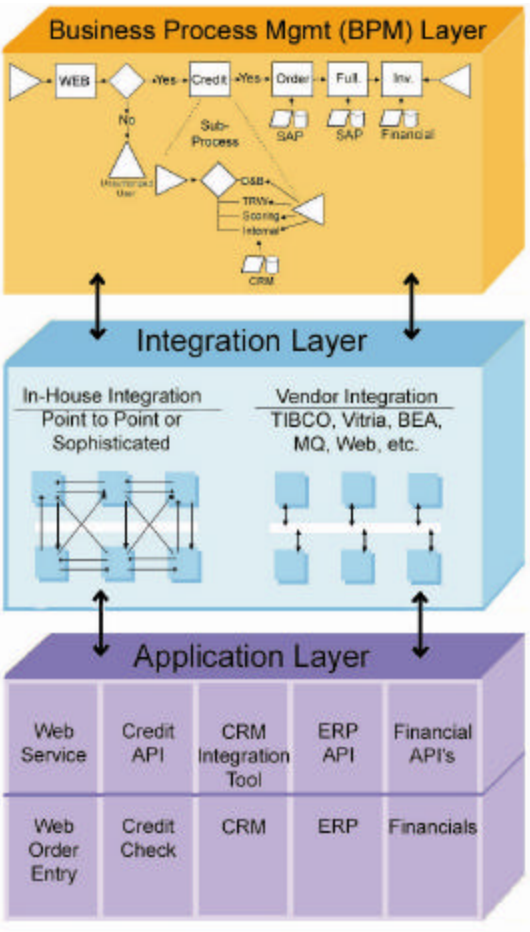

Figure 1: The Main Layers of an Integrated Application Architecture

Most integration efforts have a similar architecture with three main layers:

- The **Business Process Management Layer** (yellow) provides the ability to use business logic to determine the appropriate sequence of events. This layer is only aware of the integration layer.
- The **Integration Layer** (blue) provides for inter-application communication. This layer is aware of the application layer (purple), but it has no ability to access applications directly. The integration layer could use an integration broker, standards-based protocols such as SOAP or HTTP, or proprietary protocols.
- The **Application Layer** (purple) identifies not only the applications themselves, but also the critical data/event communication and transport mechanisms that exist between individual autonomous applications. These applications have no knowledge of each other; rather, they use the integration layer to communicate. If a point-to-point solution is being used, each interface/API must be handled individually.

Imagine an integration effort to incorporate a new Web order entry system into an infrastructure in which there already exists a credit check system running on a third party vendor's mainframe system, a CRM system running on Siebel, numerous SAP modules performing ERP, and a separate financial system running on Oracle. These components run on different platforms, have different APIs, even run in different locations.

In mapping this imaginary example to this three-layer architecture, we would see the following:

- The business process layer contains the business logic, the how and the when of a business process. Major changes in business logic would be modeled and tested in this layer first.
    - An order comes in through the Web order entry system, which performs a credit check and, if the credit is approved, checks for an account in the CRM system. If no account exists, the CRM system creates one; if one does exist, the CRM system updates the account with the information about the new order. The order then passes to the SAP system, which handles all aspects of manufacturing and order fulfillment, and then passes over to the Oracle Financials system for accounting.
- The application layer connects the applications with the integration layer using the available APIs.
    - The Web order entry system is linked in as a Web service.
    - The mainframe-based external credit check system is linked in via MQSeries (WebSphereMQ).
    - The Siebel system is connected using XML over HTTP.
    - The SAP ERP modules use a packaged SAP Adapter.
    - The Oracle Financials system uses SQL.
- The integration layer is the delivery, transformation, and routing engine.
    - Messages are delivered to the various application layer APIs on the various platforms by the integration layer.
    - Messages from each independent application are “translated” into a format that the receiving application can understand. The integration layer transforms the complex messages so that the output produced by one application can be consumed by the next application in the application layer.
    - The integration layer also performs message routing. It must support logic that streamlines complex function paths and enables high volume systems to function effectively.

The Challenge of Testing a Layered Integration Architecture

This architecture can span the length and breadth of an organization, even go beyond its boundaries. It involves many systems, many application owners, and many technical challenges. The complexity of testing an integrated application in such an environment is clear. Integrated application developers need to test and refine the business logic; they also need to test the individual component themselves. If a component needs a new connector to bring it into this integrated environment, a developer needs to build and test the adapter to ensure that it performs properly with regard to both input to the system and output from the system. Finally, developers need to have an integration layer in which to test the flow of messages among the different components.

Lining up all these elements is time-consuming at best and nearly impossible at worst. The integrated application development team can find that critical components—the Oracle Financials system in the production environment, for example—are simply not available to them during their development

efforts. The Oracle Financials team may, for any number of reasons, say that they will make the database available only when the integrated application is ready for final pre-production testing. Some organizations do have development and testing environments that mirror their production environments, but these are expensive to maintain and time-consuming to set up. Moreover, the existence of such a development and testing environment is no automatic guarantee that the environment will be available when the integration team needs it. The SAP or Oracle Financials group may already be involved in a development effort that requires the use of these systems.

In many integration efforts, the requisite systems may not become available until system owners believe that the developers are ready to go live with the integrated application. This is especially true if the integrated application is external to your department or company, like the credit checking system in the example above. Only then does it become possible to test the application from beginning to end, but without prior testing of all the components, message flows, hand-offs, and transformations, the application quite often fails to perform as anticipated. The developers must then discover where the problem lies, yet they have no way to test for failure conditions without tracing the message back through multiple applications, groups, and departments—a process that consumes still more time and financial resources.

Any one of these challenges could derail a project. Taken together, they present a clear explanation of why these projects so often run late and over budget.

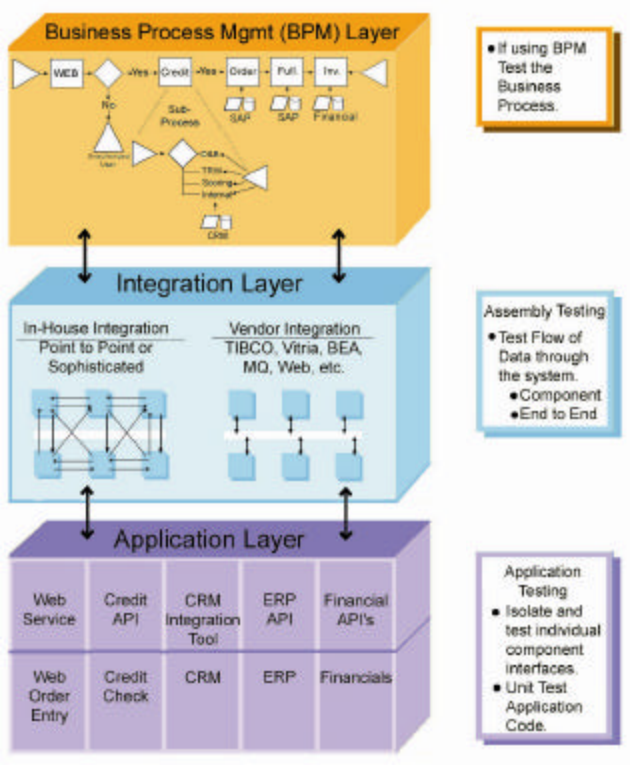

Figure 2: The Challenges of Testing an Integrated Application

Tools for Enabling Application Integration

Testing and validating the flow of messages through these layers—through multiple applications and potentially across multiple transport protocols—poses one of the greatest challenges to the success of an integration effort. Application integration projects need tools that will enable developers to test and validate the movement of messages between individual components, and standard application testing tools are not designed to do that. Standard application testing tools are designed to simulate a user load on a graphical user interface (GUI)—but they provide no capabilities for analyzing the flow of information *behind the screen*, which is precisely what really matters in an application integration project. In the absence of standard application development tools that can test and validate the flow of information behind the screens, the process of identifying and remediating problems becomes very time-consuming and expensive—which is the primary reason these projects encounter cost and time overruns.

To help application integration teams model, test, and validate the flow of information behind the screens—with the net effect of enabling the on-time and on-schedule completion of these projects—developers and project managers need an entirely different set of tools than those found in the traditional application development toolkit. Such a tool would:

- Isolate and automate step-by-step testing of interconnections and message traffic early in the application integration development cycle, when problems are easier and less costly to fix.
- Validate that the business logic is handled correctly in the business process layer.
- Simulate the diverse component systems—and their message transport protocols—that are unavailable in order to create a viable testing environment early in the development process.

Each of these capabilities helps an application integration team overcome the key challenges facing an application integration effort and increases the likelihood that the team can accomplish its goals on-time, on-budget, and, ultimately, with a cleaner integrated application—one that is easier to extend, enhance, and build upon at a future date.

Isolate, Automate, and Validate

Because the real challenges to an application integration effort reside in the complexity of the components and their interactions behind the screens, integrators need a tool designed to isolate the individual components. This makes identifying the root cause of a problem exponentially easier and eliminates most of the problems typically discovered late in the development effort—when the integrated application has been built and developers are trying to get it up and running in the production environment.

A tool that provides exacting analysis of what is going into and coming out of a given application or system component enables developers to isolate problems quickly and efficiently. Instead of realizing that the output at the end of a long string of interactions is not what they had expected—but not knowing where things went awry—developers can see exactly where the issue arises. They can then take precisely the steps required to fix the application.

A tool to automate and optimize message testing also helps ensure the cleanliness of the integrated application code. Too often, when the message at the end of the sequence of transactions is not what was expected, a corrective patch may be applied to rectify the problem. Perhaps a value that should be

positive is showing up consistently as a negative value, so a patch is applied to convert the negative value to a positive value. But the patch may not address that aspect of the integrated application that is creating a negative value. And if that aspect of the application is used later for another application, then that new application may have problems with negative values also.

With a tool that can test and analyze interfaces step by step through the integrated applications, it would become very clear very quickly where the problem of negative numbers originates—and the problem could be fixed at its source rather than by an inappropriate patch. This creates code that developers can reliably build upon and reuse in the future, without having to rely on a succession of patches to compensate for internal problems that were never isolated or properly fixed.

Simulating Unavailable Systems

As noted earlier, some or all of the production systems that the integrated application will touch may not be available. A tool that can simulate the interfaces to all the different systems, applications, and message transport protocols that will play a part in the integrated application enables the development team to begin the testing and application analysis early in the project, regardless of whether the actual systems and applications are available. This makes it more likely that the project will be completed on-time and on-budget.

Solstice Integra Enterprise

To facilitate the success of complex application integration efforts, Solstice Software has built Solstice Integra Enterprise. Solstice Integra Enterprise is specifically designed to provide comprehensive support for the assembly of integrated applications. It provides the key capabilities and features required by application integrators:

- The ability to simulate all the applications and systems involved in an application integration effort.
- Automated, optimized message-oriented testing capabilities.
- Reporting and collaboration capabilities that encourage the full participation of stakeholders.

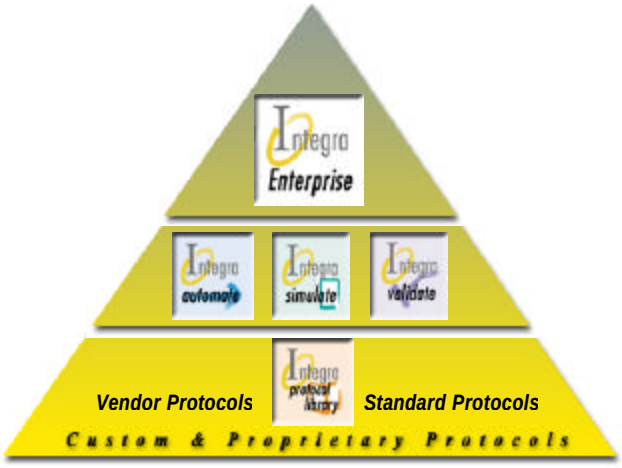

Figure 3: Solstice Integra Enterprise provides automation, simulation, and validation tools, all built on a comprehensive library of industry standard, proprietary, and custom-built message transport protocols

With Solstice Integra Enterprise, application integrators can easily model an integration environment by simulating all the systems that the integration effort will involve. The system requires no coding; its wizard-driven interface enables a user to create the test bed environment quickly and easily. The automated testing capabilities enable developers to see just what happens to messages as they pass from system to system and application to application. The reporting tools enable them to discover errors and problems quickly, which makes it possible to isolate and fix those problems quickly. Solstice Integra Enterprise even provides validation tools that can monitor and validate the integrity of messages as they pass through the application. All test assets—including stubs, scripts, scenarios, and cases—reside in a secure central storage repository, enabling easy access to development information for any authorized stakeholder.

Delivering Integration Layer-Based Testing

Solstice Integra Enterprise embodies a best practices approach to the development, testing, and validation of integrated application development effort. It enables the development team to test and validate each component in the integrated application both in isolation and then in sequential groupings—long before the full set of components are expected to run in a production environment. With these capabilities, Solstice Integra Enterprise enables integrated application developers to develop stable message-based applications quickly and effectively.

Testing the Individual Applications

The business logic of the integrated application determines which individual applications need to be integrated. Solstice Integra Enterprise enables project teams to simulate each of these applications for testing purposes—which, again, makes it possible to begin thorough testing of the entire integrated application long before the production systems become available. Once the team has created the simulated test bed, it is in a position to perform baseline testing on the applications themselves, including:

- **Positive Tests** – which test all the ways that the application is expected to be used
- **Negative Tests** – which test calls that do not match the expected formats

Once the positive and negative tests have yielded a thorough understanding of what the application is doing to an input stream, the testing team can use Solstice Integra Enterprise to ensure that key application interconnections are valid, which in turn ensures that the messaging system does not begin passing garbage forward to the next application in the queue.

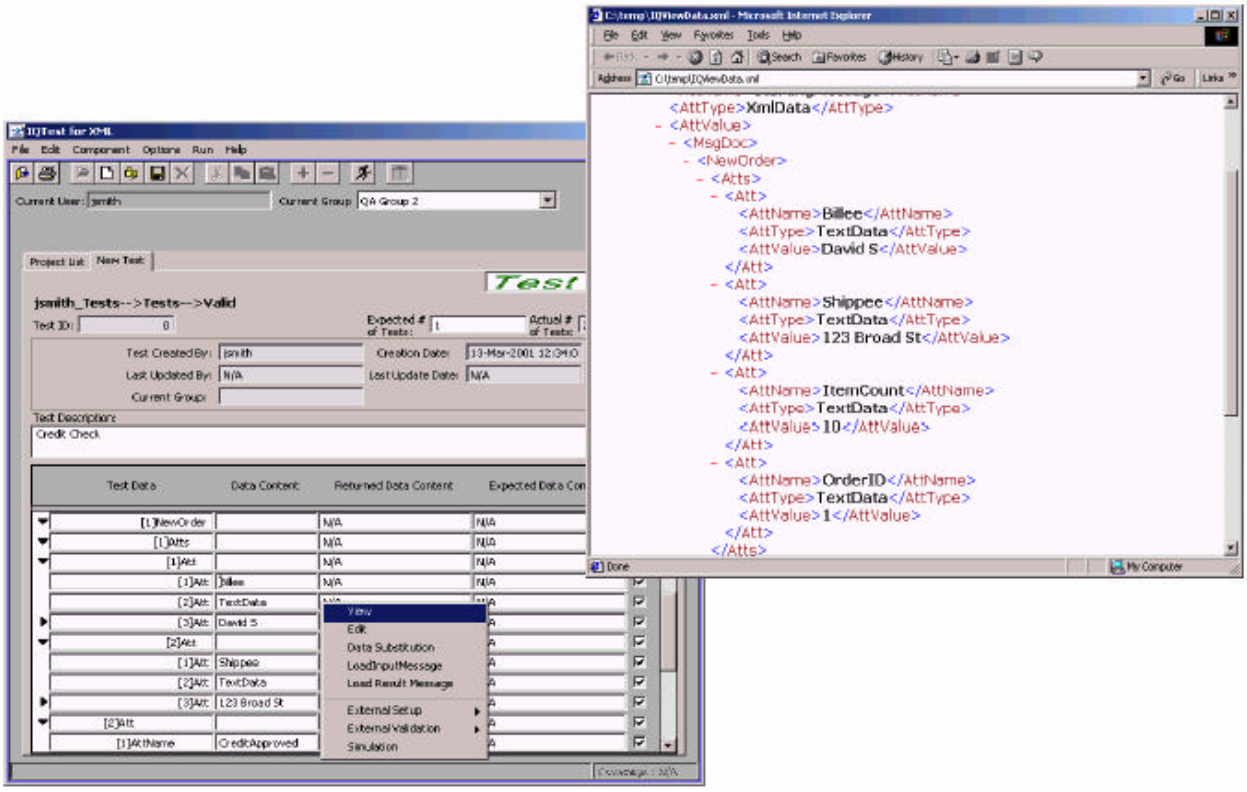

Figure 4: Solstice Integra Enterprise enables developers to test expected vs. actual responses on a component-by-component level to ensure that messages are moving as expected through the integrated application

Testing the Process Logic in Isolation

The ability to test process logic in isolation is essential to the success of an integration project. With this capability, integrated application developers can test and validate each step and each component in the process—before all the components are brought together. Solstice Integra Enterprise enables testers to use a building block approach: They can test the initial block of process logic and validate it. Then they can test the second block and validate it. They can then combine the first *and* second blocks and

validate both together. Solstice Integra Enterprise provides capabilities to store and replay testing scripts, which makes it easy to test and retest process blocks as they are stacked one next to the other.

This ability to test process logic in isolation makes it much easier to retest and revalidate an integrated application when either the overarching business logic has changed or the applications that the integrated application touches have changed (due to an upgrade, for example). Developers can isolate the logic or the application that has changed and run regression tests to determine quickly whether those changes have any impact on the integrated application.

Isolate the Integration Layer

The third area that benefits from Solstice Integra Enterprise's testing capabilities is the integration layer itself. Whether the integrated application is to rely on XML over HTTP, Tibco, MQSeries, or any other transport protocol—or any number of different protocols in combination—the developers need a way to ensure that messages moving through the integration layer are moving correctly. While individual applications may transform the content of a message—and developers can test for that in the application and process logic testing stages—transformations are also possible within the message transport layer itself.

Solstice Integra Enterprise includes a comprehensive message protocol library that developers and testers can use in the course of testing the integrated application. By enabling developers and testers to isolate the integration layer and the underlying transport mechanisms, Solstice Integra Enterprise makes it possible to test and validate the movement of messages through this layer. This can be done with or without engaging the business process logic, making it possible to test and validate the integration layer in a variety of ways.

Solstice Integra Enterprise provides development teams with a comprehensive message transport protocol library, including standard industry protocols, vendor-specific protocols, and proprietary protocols. The application provides built-in support for:

- MQSeries / WebSphere MQ
- TIBCO
- webMethods
- JMS/CORBA Standards
- Other Vendors
- HTTP/HTTPS
- FTP & TCP/IP
- SQL & PL/SQL
- Web Services (SOAP / WSDL)
- Custom Development (SDK)

Solstice Software also offers a Flex-Adapter Protocol Builder that enables the rapid creation and incorporation of additional protocols—whether an aging proprietary protocol or one completely new to the marketplace—so there is virtually no protocol that an application would encounter that cannot be incorporated into Solstice Integra Enterprise.

The bottom line? With Solstice Integra Enterprise, integrated application developers can manage and test complex application integration projects quickly and cost effectively—indeed, many organizations using Solstice Integra Enterprise report a return on investment of 500 to 800 percent—and they can bring these projects to completion on time and within budget.

Summary

Solstice Software improves an organization's ability to deliver integration projects on time and on budget. Our flagship product, Solstice Integra Enterprise, streamlines integration projects by enabling the early testing, simulation, and validation of business processes that cross applications, departments, locations, and companies. By enabling integration teams to test an application at the beginning of—and continuously throughout—the development cycle, Solstice makes it possible for an organization to bring a new integrated business processes to market without budget or schedule overruns.

Critical Technology Success Factors for Integration Projects:

- Having the ability to work, test and validate across all the diverse technologies in an integration project
- Having the ability to test process/business logic early in the development stage of the project
- Having the tools (and processes) to support integration-centric requirements-gathering, assembly, and testing

For more information about Solstice Integra Enterprise, visit www.solsticesoftware.com or contact your Solstice Software representative today.