

Maximize Online Application ROI through Efficient Integration

By: Kathleen Erickson and Jim Plamondon
Midnight Coders, Inc.

Table of Contents

Maximize Online Application ROI through Efficient Integration	1
Executive Summary	3
Why an Integration Server is Needed	4
<i>Presentation (Client) Tier</i>	5
<i>Business Logic (Middle) Tier</i>	5
<i>Database Tier</i>	5
<i>Challenges</i>	6
The Secret Sauce for Integration Success	7
<i>A Practical Example</i>	7
<i>Calculating WebORB's Basic ROI</i>	8
Maximize Your ROI	10
<i>Eliminate Waste</i>	10
<i>Sustain Knowledge</i>	10
<i>Plan for Change/Mass Customization</i>	10
<i>Deliver Fast</i>	11
<i>Empower the Team</i>	11
<i>Build Quality In</i>	11
<i>Optimize the Whole</i>	11
Concluding Remarks & Next Steps	12
Works Cited	13

Executive Summary

Return on Investment increases for businesses developing online applications when the cost and time it takes to develop them are significantly reduced.

By some estimates, more that 30%¹ of IT budgets are spent on integration, which is why IT organizations are increasingly adopting Agile and Lean software development methodologies. “Agile” and “Lean” are not new concepts, but rather tried and true practices established decades ago in manufacturing by such well-known thought leaders as D. Edwards Deming and Joseph Juran (fathers of Total Quality Management) and Taiichi Ohno (father of Toyota Production System). Ohno was the one who really put the manufacturing industry on the “Lean” path stating, “The most important objective has been to increase production efficiency by consistently and thoroughly eliminating waste”.² Eliminating waste happens to be the #1 goal of Lean Software Development and Lean Integration principles.

Upon reading this whitepaper you’ll discover how your IT team can maximize its ROI by saving your company thousands of dollars and getting your application to market faster (potentially enabling your company to earn revenue sooner) using integration server technology.

The principles of Lean Integration include:

7 Lean Integration Principles
Eliminate Waste
Sustain Knowledge
Plan for Change
Deliver Fast
Empower the Team
Build Quality In
Optimize the Whole

Why an Integration Server is Needed?

We live in a world that is changing the way people create, consume, learn and interact not only with data, but also with each other. The complex interlinking of computing and other information technologies, media content, communication networks and people is changing how applications are developed, deployed and accessed. Thin-client, fat-server architectures are morphing as smart phones with as much computing power of the prior decade's laptops emerge as a medium for social networking and work in the field (mobile computing). This client-side fragmentation of the software market and the limitations of server-side heavy architectures has prompted IT organizations to adopt new, cross-platform application architectures and approaches. The greatest challenge is adopting these new approaches when IT teams are increasingly under fire to prove their value and do more work while keeping costs low.

Given that the world communicates by and large through online applications, whether on a computer screen or smart phone, companies can no longer avoid establishing an effective online presence. Online applications, also known as Rich Internet Applications (RIAs), deliver the rich interactivity of desktop applications, but are accessible over the Internet (or intranet) from any Internet-enabled device. Deploying well-implemented online applications has been shown to increase operating efficiency, brand loyalty, and profitability.³ However, online applications can be expensive to implement, due to the complexity of integrating the technologies that provide:

- Better Visualization of Data
- Better Navigation
- Engaging User Experiences
- Cross-Platform, Cross-Browser Availability
- Real-time Synchronization of Data
- Real-time Collaboration
- Server Independence
- Better Bandwidth Utilization

These technologies are typically structured into three or more—that is, 'n'—logically separate (or “loosely-coupled”) tiers, which is only the beginning of the integration dilemma - complexity. The benefit of using this type of structure is that it enables any of the tiers to be upgraded or replaced independently as requirements or technologies change (as they always do). To facilitate the loose coupling between tiers, the system's hardware, software, and data must be integrated through well-defined interfaces.

Presentation (Client) Tier

The Presentation Tier enables rich interactivity on client devices. To maximize both IT and “Business” ROI, the Presentation Tier should:

- Run on the most possible combinations of client hardware/OS/browser
- Leverage the client’s processing power
- Deliver intuitive and engaging user interfaces
- Support single screen delivery of content without a full-page refresh
- Render dynamic content (audio, video, images and text) seamlessly
- Enable asynchronous content retrieval
- Work on- and off-line

Potential Presentation Tier technologies include Adobe Flex, Adobe Flash, Microsoft Silverlight, HTML5, AJAX and JavaScript, running on any client device that can be connected to the Internet—laptop, desktop, terminal, smart phone, etc.—even if that device is temporarily disconnected from the Internet.

Business Logic (Middle) Tier

The Business Logic Tier includes the application domain-specific knowledge (as distinct from “data,” which we’ll get to shortly). It performs the domain-specific processing (for example, in the insurance domain, it contains the logic that describes how to process an insurance claim). To maximize Business and IT ROI, the Business Logic Tier of an online application should:

- Integrate with legacy applications and systems
- Provide access to multiple middle tier services and backend stores
- Conserve bandwidth and execute well across high or low bandwidth connections
- Enable the incremental addition of new functionality
- Enable server-side data push and client-side synchronization
- Enable client-to-client, client-to-server, server-to-client and server-to-server communication
- Support a range of communication standards, such as AMF, XML and SOAP

The Business Logic Tier typically utilizes .NET, Java, PHP, Cold Fusion or Ruby on Rails servers. It may run on in-house servers, hosted servers, or cloud-based servers.

Database Tier

The Database Tier includes both the database and a program to manage, read and write access to it. Common databases include Oracle, Microsoft SQL Server, and MySQL. The Database Tier may also include a reporting server and storage server. Like the Business Logic Tier, the Database Tier may run on in-house servers, hosted servers, or cloud-based servers.

Challenges

The n-Tier architecture brings many benefits—but integrating across these tiers is highly complex. Attempting to manually code all aspects of integration is risky, time consuming, costly and fraught with performance problems. Do you want to risk having to re-architect your application or worse yet, have your project fail?

The Secret Sauce for Integration Success

John Schmidt and David Lyle probably said it best in their book *Lean Integration*⁴, "Integration code falls into different classes, whether it is a 'slowly changing dimension' class, a 'change-data-captured event stream' class or a 'publish-and-subscribe' class, the exact makeup of the data being integrated may be different for each instance, but the general processing pattern and structure are the same for each class." It is these processing patterns and structures that can be automated and made repeatable for a much more efficient and less costly integration effort.

A Practical Example

For this example, we created a simple Adobe Flex application that is integrated with a .NET service and a MySQL sample database consisting of four tables. You can see the process our engineer followed to create this application below. Our engineer used WebORB for .NET as the integration server, because his server-side environment was .NET.

Using WebORB, our engineer was able to create all of the client-side ActionScript and server-side C# code with a single click of the mouse. The generated code enables our engineer to do full CRUD (create, read, update and delete) and data management between the Flex client and .NET backend. (Available for Java and PHP environments as well.) WebORB also created a sample reference application for the same database. The code in the reference application is used to see how the data management APIs can be used as well as to test the client-server integration often before a client application is even built. The time it took to complete this whole process is defined below. Note that our engineer did not have to write a single line of integration or server-side code. The only code he had to manually write was the client-side Flex interface, as well as, the business logic. Most functions were executed by a single mouse click. As a result, there was no waiting time (waste) using WebORB.

As you can see this example is so simple that it's easy to underestimate what it has accomplished. The resulting sample online application implements full CRUD (Create, Read, Update and Delete) for each table in the sample database, with secure transaction processing and client synchronization. WebORB enables this implementation using an architecture of design patterns and best practices that encapsulate the "wisdom of the ages" in Enterprise software development —and it makes this wisdom available at the push of a button.

"Our development team was spending a lot of valuable time just defining and documenting the XML structure. This made communication between both client-side and server-side developers quite burdensome." Debugging programming errors in calls to the middle tier was a nightmare as well.

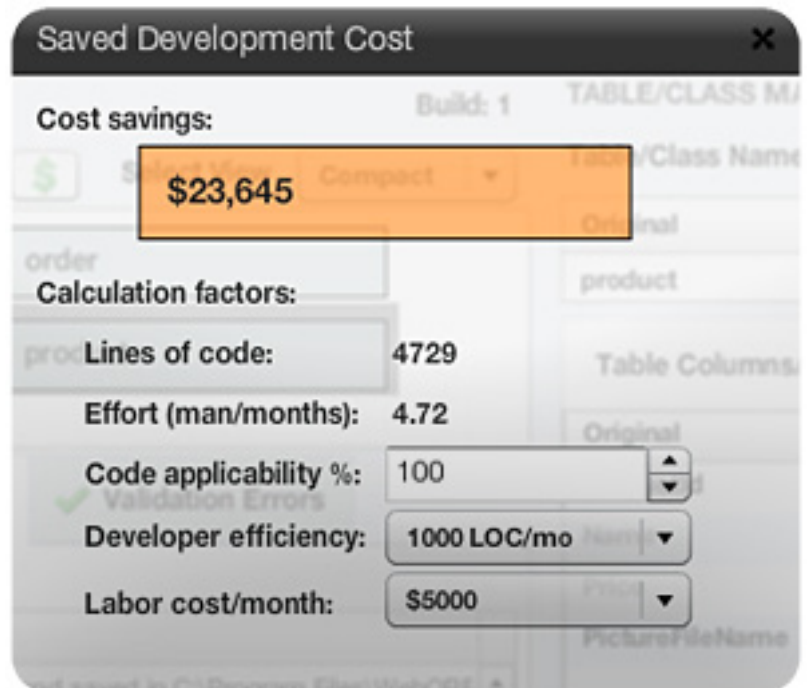
-Ahn Lam, introNetworks

Tasks	Wasting Time	Working Time
Load the FoodandDrinks database into MySQL		60 sec.
Connect WebORB to the FoodandDrinks database		30 sec.
Build the data model (drag and drop)		30 sec.
Explore the data model		Varies - 300 sec.
Validate the data model (single click)		2 sec.
Generate the client and server integration code (single click)		2 sec.
Auto-Deploy the services into WebORB (single click)		2 sec.
Verify service deployment (single click)		2 sec.
Deploy client-side integration code		120 sec.
Run Test Drive		30 sec.
Total	0 Waste	~10 minutes

Now that you've seen how much time can be saved by using an integration server, let's take a look at how much that integration effort would cost if it were manually code.

Calculating WebORB's Basic ROI

Return on investment can be a difficult IT factor to calculate. WebORB makes it a little bit easier as it has an ROI calculator built in. Using this ROI calculator (at right), one can easily see that in less than 10 minutes of work (no waste), WebORB generated 4729 lines of code making it possible to not only integrate the client application with the server side logic, but it also created a sample test application which is used to verify that the integration between the client and server works (before the client is even built) enabling the server-side developer to move on to the next task without having to wait on the client-side developer. Given the basic assumptions (average developer that writes 1000 LOC/month and paid \$5000/month) in this calculation, WebORB provided a cleaner more streamlined (reduced) code base of fully tested, debugged and documented integration code at a savings of \$23,645.



Now we know that estimating savings is not quite the same as actually comparing the manual ap-



proach with the automated approach, so we tasked our engineer with writing this same application manually, without using WebORB. Given that our engineer is highly efficient and has been working for Midnight Coders for several years, it still took him 81 hours and 6,692 lines of code to manually write this application. While WebORB generated code is documented, our engineer did not document his code for this example because the case for WebORB was made without adding time for documenting the manual code.

Let's compare the manual labor time and cost numbers with WebORB's.

Process	Manual Integration	WebORB Integration	WebORB Savings
Total LOC	6,692	4,720	Less code = better performance
Total Time	81 hours	10 minutes	80 hours
Developer Cost (based on \$52/hour pay rate and rounded up to nearest whole hour)	\$4,219	\$52	\$4,167

Even though this is a simple application that requires client to server integration with 4 tables in a database, it is easy to see there is a noticeable difference in time and savings. But what happens when you are dealing with larger databases and have to generate thousands more lines of code to integrate larger projects? The table below gives you an idea of cost and time savings for larger projects.

WebORB Savings in Time and Money							
Project Size (LOC)	5K	10K	20K	30K	50K	100K	1MIL
Time Savings (hrs)	80	120	241	361	603	1206	12,067
Cost Savings (\$)	\$4,160	\$6,240	\$12,532	\$18,772	\$31,356	\$62,712	\$627,484

As important as money saved is, the time saved may be more relevant to business profitability. The example above shows that even a trivial application can take significantly longer to code, debug, and document using manual methods.

Clearly, an online application can be brought to market faster using an integration server to automate many repetitive processes, thereby enabling a business to start earning revenue sooner. Hence, using an integration server can increase the "return" side of your ROI, as well as reduce the "investment" side.

Now that the example above has set the stage, let's dig down into the seven Lean Integration principles and how WebORB can be used to maximize your ROI.

"WebORB has been a great solution for us and has solved our middleware issues. We evaluated build vs. buy and found WebORB to be incredibly easy to use. It cut down our development cycle in that we literally had to drop it in and we began using it. If we went with a build solution, we would have had to spend months building the component and also there would have been the cost in maintaining it."

– Adam Arakelian, EMC Corporation



Maximize Your ROI

The following describes WebORB's built-in features and benefits, which are mapped to the 7 Lean Integration Principles. Interestingly, when waste is eliminated and work made more efficient, many other favorable outcomes result, such as improved developer morale and better communication between disparate teams.

Eliminate Waste

- **Service Browser** - Saves time. DLLs are instantly made available to both client-side and server-side developers from within a common user interface – the WebORB Management Console.
- **Code Generators** - Saves time. Integration code is automatically generated for the selected assemblies with the click of a button. Reduces the amount of code that you must write, test, debug, document, and/or maintain.
- **Invocation Test Drive** - Saves time. Server-side and client-side developers can test the integration independently of the client-side code, thereby increasing the likelihood that all code will pass the integration test before getting to the QA server.
- **Data Management** - Saves time. Not only does automatic generation of data models and code save time directly, but also-by encapsulating the “wisdom of the ages”—it saves the time and cost of hiring a database integration expert.

Sustain Knowledge

- **Asset Protection** - Insures completion. By making all integration-related information accessible from an easily learned central management console, an integration server insures project continuity despite personnel changes.
- **Commercially Supported** – Maximizes developer value contribution. Outsources the development, testing, debugging, and documenting of boilerplate integration code to the integration server provider, enabling your developers to focus on product-differentiating features.

Plan for Change/Mass Customization

- **Extensibility** - Reduces risk. Provides multiple points of extensibility that support easy expansion or customizations at no additional cost.

“We have been using WebORB for a couple of years and integrating the client code with the server code is way easier than it used to be. The client-side pieces are structured very similarly to the server-side pieces so a developer can go back and forth between the two very easily. WebORB saved our development team quite a bit of time in that we didn’t have to define our XML structures for our APIs anymore. I would recommend Midnight Coders to anyone.”

- James Wahlgren, Software Architect, introNetworks

- Loose Coupling - Reduces risk. Makes it possible to update an online application's different components independently.
- Modular Integration Process - Accelerates development Provides a clean interface between tiers, enabling tier-focused teams to specialize on what they do best, independently of the other teams. Eliminates wait time and increases efficient development workflow practices.

Deliver Fast

- Repeatable Process - Saves time. Integration servers provide a natural workflow that keep developers working, not waiting. This process can be repeated over and over from project to project regardless of environment, because integration patterns and structures are largely the same even across heterogeneous technologies. (NET, Java, PHP, Rails, etc).
- Automation - Saves time. An integration server's service browser, code generation, invocation test drive, data management and single-click deployment all work together to reduce waste, speeding time to deliver.

Empower the Team

- Superior Team Results - Improves morale. Outsourcing the mind-numbing tedium of writing error-prone boilerplate code, and the highly-specialized task of developing integration services, enables your team to focus on developing the features that exceed expectations and delight customers, while beating deadlines and maximizing revenue.
- Informal Team Processes – Improves morale. By providing a clean interface between tiers, inter-team communication and trust levels are improved. In fact, some businesses have fun with the process by creating a competition to see which team can get to the integration stage first, which encourages continuous process improvement.
- Liberating Creativity – Improves team value. Being able to leverage the power of an integration server makes a team more valuable, by freeing them to implement features that would otherwise be unimaginable.

Build Quality In

- Integrate it Right the First Time – An integration server enables testing before deploying to ensure that each integration is generated correctly. "Show stoppers" are identified early, so that quality is built-in.
- Data Quality – WebORB also supports scrutinizing the data model right from within the WebORB console. This means that full CRUD testing and performance tuning can be done earlier in development to ensure better data quality.

Optimize the Whole

ROI Calculator - Measurable process improvements, such as the above-cited example's cost-savings, can help prove IT's value..

"While our department had the technical skills to implement this solution, our short timeframe caused us much grief. We decided to use WebORB, because it was easy to deploy, flexible to develop with and customizable. We saved 3-4 weeks of development time using WebORB."

– Caius Swopes,
Wilson Sporting Goods

"WebORB gave us a 20% gain in productivity, improving our data load times by 83%. This allowed us to process 143% more reservations than before."

– Raphael Zbili,
Hilton Grand Vacations

Concluding Remarks & Next Steps

This paper showed that using an integration server could significantly reduce the cost and time-to-market of an online application, thereby also potentially increasing its revenue—thus addressing both sides of the ROI equation. We presented a scenario to show you just how much money and time you could save using WebORB for integration and then showed you how you can further increase your ROI using some of the other functionality that WebORB provides.

About Midnight Coders, Inc.

Midnight Coders created WebORB – the most robust integration/runtime server the world over, along with RIA AppPuncher – used for RIA testing. Midnight Coders products destroy traditional development bottlenecks, enabling rapid, cost-effective development of the most creative, engaging and responsive online applications built with Flex, Flash, AJAX, Silverlight and JavaScript. Visit www.themidnightcoders.com to learn more. Midnight Coders is a commercial, US-based company successfully operating since 2003.

About the Authors

Kathleen Erickson and Jim Plamondon are employees of Midnight Coders, Inc.

Works Cited

1. Bitpipe. Bitpipe Research Guide: EAI and Web Services.
[Online] <http://www.bitpipe.com/eai/eai-web-services.jsp>
2. Ohno, Taiichi. Toyota Production System. Oregon. Productivity Inc. 1988
3. Duhl, Joshua. Rich Internet Applications. IDC, November 2003
[Online] http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf
4. Schmidt, John G. and Lyle, David. Lean Integration.
An Integration Factory Approach to Business Agility. Massachussets.
Pearson Education, Inc. 2010