How to Bring Your Business Applications to the Web

By Heather Gately

Executive Summary

This article will take you through the migration of your current or legacy business applications to a Web environment. It focuses on the benefits of choosing this direction, and the pros and cons of the four most popular methods of accomplishing this including Screen-scrapers, In-house Development, Outside Consulting, and Development Tools.

"If it ain't broke."

Regrettably, it often seems when it comes time to address IT, outdated business systems are the elephant—make that mastodon—in the room. If you talk about it, you might have to actually do something about it.

And so much time and money and energy has been invested in these expensive business systems bought eons ago, and often the software running on them is completely unsupported, written by defunct companies, or long-lost employees. Big businesses are being run on the backs of ancient machines, and there is a very noticeable shortage of mechanics.

Why not just scrap everything and start over?

According to the Gartner Group, between 60 and 80 percent of an average company's IT budget is spent on maintaining existing systems and the applications that run on them. So, why not just use that money toward brand new systems?

Darcy Fowkes, Research Director of Internet Business Practice, Aberdeen Group:1 "The fact that there is two trillion dollars' worth of mainframe applications in corporations today housing approximately 70 percent of all critical business logic and data make them valuable sources to leverage."

So, even if the IT world could make a decision on what all future technologies are going to be, and conform everyone to one path, it would take billions of dollars to recreate all of the applications. It's also taken years to get all the bugs out of these "current" applications. Starting over from scratch in a new environment runs the risk of major revenue losses due to business incontinuity and customer frustrations.

If you haven't figured out the perfect solution yet, you're in good company. No matter how "ahead of the game" your company may appear to the outside world, technology is approaching such a pace that it is almost impossible to keep up. Somewhere, in some system, you are already out of date.

Making the move: Justification.

Since you can't win every battle, the only way to win the war is to do a proper evaluation and address what areas would best benefit your business to modernize. Here are a few common areas of scrutiny:

Let's say you are a manufacturer…Production is your bread and butter…but inventory might be kept on various systems in different plants running a decade-old MRP software package that demands constant maintenance.

For example, let's say you have an application that has the ability to set a separate and unique price for each of your products for every single one of your clients. How much time did it take to fine tune that application? What would it take to start over and get it right again?

How would you go about updating it efficiently while going on with your business without interruption? How much of your day-to-day applications are home-grown? What if the vendor that brought you your MRP went out of business, and was unable to support its products? Where does that leave you? Is that your situation? Will it be tomorrow, or next year?

Maybe your main focus is not production, but instead you are working toward streamlining your accounting department. But, comparing reports between different plant or office locations is completely apples to oranges. Reports in such situations are often written for disparate systems, by individual IT departments with different expertise and/or agendas. To compound the problem, by the time you get the information, it might be two weeks old or more because of an IT application backlog. How stale are the numbers your decision-makers are basing their decisions on? Do all the necessary reports relate to one another? Are your company captains being forced to navigate by guesswork and old coordinates?

Are your company captains being forced to navigate by guesswork and old coordinates?

Then there's CRM. CRM, or Customer Relationship Management, is one of the most-talked about buzz-terms of the day, but the concept is age-old: it all comes down to service. If your customer service representatives are not knowledgeable, helpful, and efficient, your customers will go elsewhere. It's as simple as that. Do you find that good Customer Service Representatives (CSRs) are hard to find, and even harder to train? CSRs often have to be trained in complex interfaces that are non-intuitive and can take months to master.

Even with a client/server solution, management often asks CSRs to navigate the ever growing, twisting and turning, forest of directory

trees, extensive business rules, and query limitations. And, at the same time, try to relay information to the customer with a smile.

Even if you are only trying to answer one of the previous hypotheticals, it remains mission-critical to leverage your existing systems. Your best bet is to modernize your current applications by bringing them to the Web with a platform-independent solution such as Java servlets. Browser-based applications present you with the most flexibility in distribution, data accuracy, and ease-of use, while Java servlets protect your future, allowing all database and platform decisions going forward to be yours to make.

Here are some of the positives to bringing applications to the Web with Java.

- You can serve them directly from the hardware you've already invested in, so there is no added hardware cost.
- Distribution and operational costs associated with mailing, faxing and printing reports, invoices and bills of lading will go down dramatically.
- Business decisions will be based on accurate numbers made from reports and queries accessing live data.
- Remote users such as sales people in the field can securely access live data including inventory, pricing, and order tracking from wherever they are, including from their cell phones. Imagine the time savings!
- Users and IT alike can focus on tasks that are mission-critical by giving authorized users secure access to online reports and data instead of having to make requests through IT and tying up development staff.
- You will realize a fast ROI from so-called soft costs including reduction of IT man hours, training for end-users and developers, lost orders, "temporary" software fixes, paper, ink, pre-printed forms, etc.

So, what is the best way to get to Java and the Web?

Well, you have four basic choices if you are trying to bring your current apps into a browser-based environment via Java. You can:

- elect to use a "Screen-scraper" software
- train current, or hire new staff to re-write your applications in-house in Java
- hire outside consultants to re-write applications or customize package software for the Web in Java
- purchase a software tool to modify your applications yourself without a Java prerequisite.

There are pros and cons to each. Here's the run down.

Screen-Scrapers

"Screen-scrapers" are software products that allow PCs to intercept character-based data, older usually, and present it in a Web-like interface. In basic terms, it automatically turns what looks like text on screen (remember programs where you needed to use the F-buttons, like F7 or F10?) into what looks similar to a Web page. Screen scrapers can often present themselves as the easiest way to convert older applications into a Web application. The popular claim for screen-scrapers is that with the push of a button, the application is translated to an Internet Language, usually Java, and ready for the Web.

And, if you're not talking about a complex business application, it usually is. Once purchased, though, developers often realize that it is necessary to go in and hand-code many features of their applications in order for the screen-scraper to work. But, by then it's too late. Also, because they sometimes rely on older Java, they can require Java to be downloaded onto the PCs of those users that want to access the application. These versions are traditionally very Java-heavy which means that they tend to run very slow.

Common problems with screen scrapers include: server time-outs, browser-compatibility problems, software plug-in warnings, and complications with firewalls. They also don't look much like they belong on a Web page, and are not very intuitive in navigation because they are based entirely on an old character-based application.

Screen-scrapers are a good solution if you have an extremely simple application, are looking for a short-term solution, and are prepared to spend in the neighborhood of $100,000.

In-House Java Development

Use your highly skilled IT staff to your advantage. After all, isn't that what you are paying them for? Why pay out extra money for contracting or hiring a team of Java developers, when you can train your current IT staff on a popular open source language, like Java, and get them working for you!

They have priceless business expertise building your applications and they know the ins and outs of your business.

This seems like perhaps the most suitable solution. After all, these people know the original applications inside and out, many being built by their capable hands. And, learning a new skill set like

this would be an absolute up-side to working all those late hours doing double-duty maintaining the current backlog of applications and reports while building all new applications and reports for the Web.

The problem is…time. According to the Butler Group2 , it is estimated that to retrain a developer in Java takes a minimum of nine months and has a significant failure rate. This is compounded by significant turnover when recently trained staff take their new skills elsewhere. The cost of this training needs to be taken into account, in addition to the expansion of the application backlog, the learning curve, development time, experience in taking a Java project from the early planning stages to a successful close within a reasonable timeframe, and the cost of staff turnover.

Also, keep in mind that the learning curve could be company-wide. According to another study reported by the Butler Group: "Further evidence is provided by the forecast that for over 65 percent of companies, their first generation e-business development work will be scrapped and re-built within two years, often at a huge cost."

In-house Java coding is your best bet if you can say without a doubt that your IT people won't take the training and run, and they have extensive C++ or Java project management experience. Most importantly you need to also say, without a doubt, that this intricately-designed, expensive, "custom-built car" that you're aiming for is not on its way to obsolescence within two years.

What happens if you put all your eggs in this Java basket, and something new comes along?

Consulting

Okay, consultants will charge you more than a salaried employee for the privilege of contracting with you, but they ARE the Java experts specific to what you need. Would you want your general practitioner performing your brain surgery?

You just need to make sure you're hiring the right consultant. Consultants come in all shapes and sizes, and like surgeons; they sometimes have a bit of a God complex.

Just remember the following: if you go to the doctor to cure your cold, and he wants to cut out your brain, you might want a second opinion. Same with consultants.

It's estimated that to retrain a developer in Java takes a minimum of nine months and has a significant failure rate.

There are several things to watch out for in working with consultants. The first is of course, references and expertise. It is your responsibility to know that they know what they are doing because you're letting them loose on the nerve center of your business. It's a dirty little secret in the consulting business that $150/hr will often get you a fresh-faced college graduate who is "learning as he goes" on your dime.

Make sure that no matter whom you select, they stick to their schedule. In the old days, consultants would sit down with clients and charge high hourly rates to cover time and materials, but only for a limited period of time. In today's massive IT marketplace, "time-and-materials billing" can be a financial albatross when numerous consultants are working at a customer site on an "open-ended" project. These consultants get paid for every hour they put into a project, regardless of whether the project is successful or on time. Make sure you cover yourself in your contract.

Additionally, some corporate consultants are rewarded for selling follow-on work to customers so that once the company gets its foot in the door, it can stay there as long as possible. KPMG calls this its client-for-life strategy, according to Mark Lee, senior vice president of product solutions for KPMG in McLean, Virginia.3 *CIO* article, July 15, 2002, "Take Control of Your Consultants"

Also, consider this, they may know Java, but they probably do not know a thing about the languages and applications you are trying to modernize. You may find that they are re-inventing the wheel on your dime. Make sure you keep tabs on project goals and plans and timelines at all times.

Hiring IT consultants will likely cost a bundle, but that solution will get you to where you want to go, and you don't have to pay for internal training, worry about turnover, or take time away from your busy schedule. Of course, you have limited control over how long it will take, and since many of you choose consulting because you don't have the expertise that you are paying a premium for, do you feel comfortable questioning their methods? Do you feel comfortable maintaining the final solution? Once it's completed, if you ever have a problem with it, who else will know how to fix it?

In that sense, some other vendor-based consultants, once they get their product's foot in the door,

also try to "remain a part of their client's lives forever." They attach themselves to the wallet of a perfectly healthy corporation, and over time purposely create dependencies on certain products. Whenever a product needs to be updated, or a solution changed, that company's consultants are called in yet again.

If the product your consultant is recommending has the potential to create a real dependency on them, either through the amount of money spent (as in an ERP) or in the knowledge of a language, make sure you weigh all of your options before you sign on the dotted line.

Development Tool

A development tool is a solid option for quite a few reasons. It allows you to capitalize on your staff's expertise, and at the same time speeds up the development process. But, there are many different categories and things you need to consider when looking at development tools including cost, ease of use, flexibility, and extensibility. When purchasing any Web development tool, the latter two become increasingly important.

In the past, the level of flexibility available today was not an option, and proprietary solutions were the order of the day. And this is precisely why so many businesses find themselves with outdated systems, and are nervous to move forward. Don't make the same mistake twice by going with a proprietary solution.

Flexibility and extensibility are both critical when looking to the future. Where will you go, and how will you grow? Flexibility comes into play in terms of application deployment. Where will you be able to use your solution? What if you want your solution to be deployable to wireless devices? Would that ever be a need? These questions are important because as the technology evolution moves forward, it is important not to place these applications on another dead-end path, or they will have to be rebuilt yet again.

A good Web development tool should also build applications with growth in mind. This is something to take a good hard look at if you are looking for a long-term solution, and want to allow for your company to be global. So, if you're looking for flexibility, you need to examine architecture…and if you're looking for architecture, you need to go n-Tier.

A good Web development tool should build applications with growth in mind…the best method for allowing growth in your systems is with n-Tier architecture

n-Tier Architecture

The best method for allowing growth in your systems is using "n-tier architecture." The original concept with roots in the days of client/server, had two components, and was therefore "two-tiered". This was the client/server version. The server was one tier (the business logic tier), and the client was the other (the presentation or graphic user interface tier). "Three-tier architecture" incorporates the first two-tiers and then adds a third-tier (the database tier). n-Tier architecture provides flexibility for variable combinations of the above tier types.

This architecture, in a nutshell, breaks Web applications into three components: a database component, a presentation component (GUI), and a business logic component and the advantages are many. To give you a better idea of how it works, let's look at an example. Let's say you wanted to develop a reservations system for a chain of hotels that had rates and taxes varying from state to state and city to city. Well, the designer could make that report look like the rest of your site in the presentation component. The database programmer would know which files to link in the database component. The business logic expert would program the pricing structure calculations into the business logic component.

If the pricing method changed next year to double the state tax as a resort fee in California and Massachusetts the business logic component could then be updated without ever touching the other two components, and the application would remain seamless. The changes would not have to be duplicated for each state's application. Instead, the changes made in the business logic component would only affect the screens using that particular component. Or, perhaps the designer decides to add a blue bar across a page of the Web site. He could change only the presentation component, and the changes could be made across the board. The same applies to the database component.

This "n-Tier" architecture is ideal for maintaining your applications going forward. The set up eliminates errors, simplifies change, and saves time because each component is chaired by persons with expertise in that tier. It also allows for future changes such as new languages, and new looks, to be added, and it can save you money on hardware.

In the past, a program was self-contained, and would need to be duplicated for each of those applications, taking up space, and it would be required to be on each machine where the application was running. Now, the components of an application could all reside on one Web server, as in the past, or one component could be in a box in Arizona, one could be in Paris, and one could be in Seoul, and still work together. If the hotel headquarters in Seoul wanted its own presentation component (GUI) to allow for the Korean language version, and different database component with Korean tariffs, it could still use the same business logic component as its Arizona division, if it wanted. This architecture provides for the ultimate global possibilities.

Not all tools use n-tier architecture in their design. Make sure you look for one that does.

So, what kind of tool should I use?

There are many different Web development tools to choose from, and keep in mind that there is no perfect solution. But, using a development tool that will deploy to any platform puts the power and control back into the hands of your IT department, where it belongs.

In the category of Web development tool products, there seem to be two main software camps: products using vendor-specific languages, and products that are menu-driven that can be used without learning another software language. So, what are the pros and cons to each?

Pro-proprietary Tools

Those vendors who offer products requiring you to learn and use their proprietary development language, offer developers the bonus of putting another language on their resume. These languages are challenging to learn, and in some circles can be a source of pride among programmers and a bartering chip in salary negotiations. For the vendor, it provides a viral marketing of sorts. If you, as a developer, spend 6-9 months in training to learn a vendor-specific language, and you move companies, you will likely encourage your new company to purchase the software tool that uses that language, and shows off your skill set.

The downside to tools built in (and requiring) proprietary software language, is that they force the user to have a certain dependence on their vendor. Their time in training is invested, the amount of money they have in the software is invested, and then any additional "specialized" modules they need to purchase for later projects or for later upgrades must come from the same company because the language is incompatible with other software pieces.

These modules are also priced individually so any future needs that crop up that aren't covered may require additional system purchases.

If you want to handle the project internally, you enjoy the challenge of learning a new language, and you have the timeframe to account for staff training and learning curve, then this is the solution for you.

However, you also have to be willing to accept that any expert-level work in that software will require an expert in that language, and if the software vendor goes out of business, you have no safety net, and will need to look for another solution. More and more companies using proprietary software are migrating toward open-source code solutions to prepare for the future.

Pro-Productivity Tools

The other tool option is to go with a specifications-based, menu-driven development tool. The downside to this option is that if you are a hard-core coder, you never really have a need to tinker with the code (although you usually can). And, there is no proprietary language to learn, so no bartering chip in your salary negotiations, no gold star on your resume.

The plus side, though, is that these software tools have an extremely short learning curve, with no languages to learn at all, so you can be up and running in a matter of days. Look for tools that write applications in 100% open languages like Java servlets. Choosing a tool like that means that if the software vendor ever goes out of business, any number of companies could support the software. It also shortens the project's time to market. Imagine being able to develop an application for your Web site in Java servlets simply by choosing what you want off a menu and hitting the enter button.

An additional bonus feature to look for is a tool that will insulate you from technology changes going forward. If Java goes by the wayside, a code generation tool can insulate you from those bumps in the road, and allow you to automatically move (or with very few changes) to the next level—without a lot of fuss or muss.

This is the best solution for you if you want to handle the project internally, avoid vendor dependencies or language-incompatibility in the future, and you want to get to the Web extremely quickly.

To sum up…

In summary, there are four basic methods to leverage your current legacy applications by bringing them to the Web. You can use a screen scraper method, you can build your solutions in-house, you can hire IT consultants to do the job for you, or you could use a tool to build your applications for you.

Whatever solution you choose, you should look for one that fits your budget, fits your schedule with training and learning-curves, is flexible in terms of platforms, and databases, and allows for growth on a global scale (we recommend an n-tier architecture).

You also need to consider whether or not you want to commit to a proprietary language tool, or if you'd feel more comfortable moving to a menu-driven tool written in a flexible language like Java.

Regardless of which method you choose, ultimately, it is most important to know what direction you are headed toward, and why. Whether you are moving your legacy apps over to the Web because you would like to save money on maintenance, gain a competitive advantage over another company, improve internal communications, or increase employee productivity; make sure you have a clear goal in mind before entering into the Web world.

The more you clarify your own needs, the more software vendors and consultants can present you with the clearest solutions for your business.

***

Featured product

m-Power™ is Java-based and can run on virtually any operating system, and access any database. It accelerates development by eliminating routine infrastructure programming, giving you ready-to-deploy n-tier Java Web applications in minutes…

mrc's m-Power development tools hit the **Productivity Sweet Spot** between professional developer tools and power end-users tools, and have been helping developers be more productive for 25 years.

Why developers love it:
Among developers, m-Power and the mrc-Productivity Series span wide and varied abilities. Since applications are created in **100% Java**, expert developers win because they can avoid tedium, developing their applications quickly and easily-and they can customize any application to suit their needs.

- **No languages to learn, and no manual coding:** Unlike most development tools, you are not required to learn Java, JSP, WebSphere or any 4GL.
- **All of the capabilities, none of the hassle:** Whether you are a Java expert or an expert in RPG, COBOL or VB-you can call previously built logic, work with complex analytics and calculations, build robust enterprise-level business application systems, and customize ERPs and the like...without any of the tedium of manual coding.
- **Teach it to code the way you code:** Developers can even teach m-Power to code Java custom to individual development styles. At the same time, developers who are Java newbies and wish to learn or are in the process of learning Java, can begin learning it as they go using m-Power as a study tool.
- **Shorten the process and deliver the best solution:** Free up your time for crucial IT matters by teaching end users to create their own reports or applications. And to make sure that you are delivering the right solutions and better fulfilling executive needs, you can just send them a URL to mark progress as you go.

Why end users love it:
If you are an end user— you can easily create all of the reports or data inquiries they could ever need, plus the capabilities to develop more complex applications down the road.

- **The interface is easy to use:** If you've ever been on the Web, using m-Power and the mrc-Productivity Series' development menu is a piece of cake. You just follow a series of screens that take you through the wizard-like Web interface from start to finish.
- **All you really need to know is your business:** You really just need to know what kind of application you want to build, and where to find the data files that you want to access. That's all. So, once users become accustomed to the software, the capabilities are there to create more complex business applications down the road.
- **Shortest learning curve in the industry:** mrc recommends a 3 day training class to get a strong base, but you'll be building applications your first day! Each tool is entirely menu-based, so end users can create new applications in technologies that even developers may not have had time to learn, such as Java.