UNISYS
imagine it. done.

Developing Agility

*A Quarterly Newsletter for
Unisys EAE and AB Suite Customers*

Contents

EAE Modernization with ASP.NET Generator Client Tool

By George E. McGowan, Jr. CPA, Founder and President, McGowan Computer Associates

The Real Cost of Modernization

Modernization is a hot topic for many IT organizations. Regardless of how well an application fulfills business or organizational requirements, "green screen" user interfaces send the message that the system is antiquated and in dire need of replacement.

We have seen many organizations take on the challenge of replacing their finely tuned Enterprise Application Environment (EAE) applications with a package solution, or several packages. But, the chance of deploying something that provides equivalent functionality is very slim. Few organizations fully comprehend all the business rules contained in an EAE application that's evolved over the years to meet changing business and market requirements.

The price tag for new business-critical software is big – typically $2 million to start. It's very common to underestimate the cost of migration. Mapping and moving the data from one system to another is the easy part. It's training the users, qualifying the new software, rebuilding interfaces, conducting serious, mixed transaction workload performance testing, and building in "missing" functionality that pumps up the cost. Organizations always underestimate the effort to test the new system and retrain ALL of the application users.

Developing a new custom application is not always an option for today's IT organizations: resources are scarce – and the backlog gets longer every day.

With all things considered, the value of a finely tuned EAE application is nearly priceless.

It's the Presentation, Not the Application

There's no reason to abandon your EAE applications when it's the presentation that's the problem. You can easily protect and preserve (and continue to refine and extend) business rules while improving the end-user experience with Client Tools, specifically the ASP.NET Generator. The ASP.NET Generator helps you create a modern, browser-based GUI with the kind of capabilities that Internet-savvy knowledge workers expect.

ASP.NET forms work remarkably well – much better than the original Active Server Pages (ASPs) that many EAE shops looked at years ago. ASP.NET renders web pages universally to all kinds of browsers and browser levels, as well as in Developer Test. From a support perspective, there's no client software to deploy to workstations, either. Once an organization moves to ASP.NET they can choose to expose all or part of their application via the Internet to external users, too.

Best Practices for Developing Browser-Based GUIs

Using the ASP.NET Generator is straightforward – but like any seasoned EAE developer already knows, stakeholder buy-in is essential. We have used the following five-step approach on many successful projects:

1. Involve

Build an advisory team that includes a variety of end users – from data entry to managers to power users – and develop the screens together.

2. Experiment

ASP.NET forms can be a Pandora's Box if you don't take a measured approach. Work with your end users to define several types of generic screens, including a menu, table, inquiry, and data-entry-heavy form. This allows you to try out the many options ASP.NET offers until you find a set of functionality that works for your environment.   >>

3. Simplify

Just because you can make every field a drop down list box or radio button and add color and graphics galore, doesn't mean you should. Chances are your EAE application is a serious core business system, so avoid the temptation to add too much "flash and dazzle." Once the Experiment phase is complete, take a step back to assess the actual business value provided by the new GUI options and simplify your approach to include those that truly support your needs.

4. Standardize

The majority of end users will readily embrace a modern, easy-to-use GUI. The learning curve is minimized when a consistent, reliable, intuitive feel is applied across all new ASP.NET forms. That's why we recommend you take the time to set GUI standards based on the outcome of the Experiment and Simplify steps. For those users that resist change, allow time to work with them and meet their needs. Moving from a terminal emulator to a browser-based application can be a bit traumatic for existing users. On the other hand, new users of the application will *expect* a browser-based application.

5. Modernize

With a considered approach as outlined above, you'll have the knowledge, standards, and end-user buy-in to successfully update the presentation of your EAE application using the ASP.NET Generator. Even so, keep the first official steps to modernization low impact:

• Consider moving existing transactions to a web browser without any further changes, maintaining the current navigation and functionality
• Identify a subset of frequently used screens and selectively apply your new standards to just those transactions – adding list boxes, radio buttons, and other controls
• Live with the changes for awhile, make adjustments as needed, and expand to more transactions

The development process will change if you opt to use Developer painter for your GUI deployment. The focus of your painted screens will move from the character-based screens to the graphical painter. The ASP.NET Generator renders forms very close to the Developer painter. If a screen will only be accessed via the web browser, there's no reason to invest development time in making the character-based screen look pretty. Position the fields on the character painter in the order you wish to tab between each using the graphical screens. Remember, you do need to be concerned with labels or headings in the character-based painter.

Heads-Down Data Entry

Who wouldn't welcome a browser-based, modern GUI? The data entry people of the world! Be particularly careful to understand the process these folks follow to get the data into your EAE applications – and design a GUI that supports, rather than hinders, that effort. No need for radio buttons and drop downs for this crowd!

Get Started with GUIs

There's never been a better time to modernize your EAE end-user presentation. The ASP.NET Generator is a proven solution that is on par with most ASP.NET development tools on the market today. It's the best way to quickly and efficiently bring your green screens into the 21st century. We have yet to find a need that can't be fulfilled using the standard ASP.NET Generator capabilities. With this powerful Client Tool, we have helped many organizations take the first, small steps to a browser-based GUI, as well as giant leaps to using sophisticated controls that pass data to other applications running on the workstation.

Get modern at a cost and effort that you can afford – and preserve all the value in your current EAE application. Modernizing your existing applications will provide your organization with the highest rate of return on the applications. Protect the investment in your applications by getting started with ASP.NET Generator.

George E. McGowan, Jr. CPA is president and founder of McGowan Computer Associates, which has been providing consulting services for organizations that use LINC, EAE, and Agile Business Suite (AB Suite) for more than 23 years. For more information about McGowan Computer Associates' consulting, training, support, and migration services, contact George at 801.446.7100 or email GMCGOWANJR@aol.com

Engineering Corner:
The Many (user inter) Faces of EAE and AB Suite
Enhancing Standard ASP.NET Clients Using Renderers and Custom Generators

To continue this issue's focus on user interfaces, we are devoting Engineering Corner to a review of several advanced approaches to upgrading the generated ASP.NET clients. We encourage you to start using Client Tools, including the ASP.Net Generator, with your EAE applications to improve the end user experience today – and preserve that effort when you migrate to AB Suite.

To get a full accounting of the many ways you can extend ASP.NET web applications in EAE or AB Suite, we spoke with our resident expert on the topic, John Papachristos from ACUS. John is a senior software engineer and technical lead for Client Tools.

Developing *Agility*: Why would a developer want to do more than what's available using the standard ASP.NET Generator?

John Papachristos (JP): In our development environment, we supply a base set of form design controls in the screen painters. Because multiple client interface options can be generated from this single screen definition – with either EAE or AB Suite – we have to stay within certain boundaries. However, there are times when developers want to deploy a more sophisticated user interface for the specific client environment that they want to use, such as ASP.NET pages. And, that's where more sophisticated customizations can come into play.

For example, with ASP.NET a developer may want to add a Calendar display function, a file upload control, or other controls that we don't automatically provide in our painters. We've always said that our generated ASP.NET client environment is really a starting point for customers that want to do more with their browser interface. Using Microsoft® Visual Studio®, the ASP.NET application can be enhanced to include additional controls from the ASP.NET Framework and custom processing can be added to meet the customer's business needs.

What is a Custom Control?

A control is the object that you place onto a form (such as a web page) using the painter to perform specific functions. Common controls include push buttons, drop down lists, images, and so on.

A custom control is one that is not part of the standard set of controls provided by Microsoft. The Web Form Renderer is a control that provides the container for an EAE or AB Suite application. It handles the communications to and from the application, and it presents or "renders" the transaction forms (presentation) in the web browser.

So, the Web Form Renderer is a way to add more sophisticated controls or objects or perform pre- and post-transaction processing between the client and the application without having to modify the application itself. And with the Renderer, you don't have to be concerned about overwriting because you are always using the Web Form as generated (except with the OnAlternateForm event, of course). It's a great way to introduce some custom processing on the front or back end as data is being passed between the browser and the runtime environment.   >>

DA: What are some of the ways that developers can enhance the generated ASP.NET form?

JP: There are three primary options. Developers can:
1. Modify our ASP.NET Client Infrastructure files
2. Use the Web Form Renderer
3. Use the Client Generator Customization Kit to create a custom generator

Before we discuss renderers and custom generators, let's review how developers can introduce custom processing in various areas of the ASP.NET client framework using our Client Infrastructure files.

DA: Modifying the infrastructure? That sounds difficult – and a bit scary. Is it?

JP: It's not as hard as it sounds. By making changes to infrastructure files in our ASP.NET Client environment, developers can modify the presentation and client logic that runs in the browser on the end user's desktop, as well as the logic that is processed on the Web Server. The infrastructure files are bundled with the Client Tools ASP.NET Generator and form the foundation of our ASP.NET client framework.

For example, we provide a script file called CWFRCommonScript.js., which is a collection of JavaScript functions that are executed in the browser. Depending on the needs of a given organization, developers with JavaScript skills can modify functions in or add functions to this script file to invoke custom processing in the browser. The script could be updated to capture keystrokes and perform certain actions based on what an end user types, or to force an auto tab to the next field when the user reaches the end of the current one. I've seen developers introduce scripting that enables interaction between the browser form and external applications running on the local workstation – such as exchanging data between the form and Microsoft Word or another workstation-hosted application.

DA: Just to clarify, are these changes made after the ASP.NET project is generated? And, because you are changing the "infrastructure," then the next time you generate, you don't have to change anything to re-apply those changes?

JP: That's correct. The CEWFRCommonScript.js file is part of our set of infrastructure files and doesn't get overwritten with each generate. However, developers that choose to modify the files must be aware that new versions are supplied in Interim Correction (IC) releases. So, if you apply a new IC you need to be careful to preserve your updates.

DA: You mentioned there were other ways to modify the infrastructure files.

JP: Another area where changes can be introduced is by modifying the server-based infrastructure files. One of these is IspecView.CS, which defines a class that is inherited by all the generated Web Forms for an application. It includes many of the server-side functions that get executed when a Web Form is processed. This provides an opportunity to introduce some custom changes that affect the processing of all forms.

A developer could make changes to IspecView.cs to extend the dynamic presentation attributes to include some custom attributes. For example, you could perform range checking on numeric fields or change the colors of individual buttons. All the logic for the dynamic attributes' processing is contained in this infrastructure file, which is then extensible by developers according to their unique requirements.   >>

DA: Where can our readers go to find out more about making changes to these infrastructure files?

JP: If you are familiar with form processing and C#, you should be able to easily figure out what's going on. And the infrastructure files contain comments, which provide further explanations about the various functions. There is some information in the customization kit too, but that is a separately orderable item and is not required to make changes to infrastructure files, which are installed as part of the Client Tools ASP.NET Generator.

DA: Are there other infrastructure files?

JP: Yes, there are quite a few others. You can configure the ASP.NET environment to display a log-in page and developers can customize the log-in page we provide to suit their needs. And there are other Web Forms, such as timeout pages and error pages, that can be customized.

Another interesting one is the "right click" context menu, which is a custom control that can be updated to exclude current items or introduce custom processing. I know of one account where the right-click context menu was extended to provide a new menu item that copies data from a Microsoft Excel® spreadsheet to the Copy.From fields on the ASP.NET Web Form. The customer was previously using such a feature with their T27 emulator and found that this could not be done with the standard ASP.NET Client environment. By customizing the right-click context menu and adding some associated JavaScript methods in CEWFRCommonScript.js, they were able to replicate this feature in their ASP.NET Web Forms.

DA: OK, that covers the first option for extending the ASP.NET forms. What about using the Web Form Renderer?

JP: Right. Another approach to customizing the ASP.NET environment is to change the generated files to provide additional controls and processing. Of course, this is where you need to be careful to make sure you manage the customized files because subsequent generates could overwrite them. And that's where the Web Form Renderer "OnAlternateForm" event comes in handy because you can actually trap that event and then display your customized form for a particular ISPEC instead of the "standard" generated form that is defined in the application. Customers take advantage of this event when they have certain forms that require more extensive customization than others in their application.

DA: What is a "renderer" and how does a developer use it? What types of improvements can developers make to end-user interfaces with renderers?

JP: Actually, the 1Q2006 issue of this newsletter included quite a good article on renderers. The Web Form Renderer is essentially a custom control that allows for the display of Web Form controls that are generated from the ISPEC forms – i.e. the forms that are designed using the Developer painter – and allows a programmer to add specific processing based on certain events or triggers.

I mentioned the OnAlternateForm event that allows you to substitute forms that you have customized. There are other events, like the OnPreTransaction, OnPostTransaction, and OnStatusLine that developers can use at certain points in the processing of the Web Form to perform certain actions. For example, you might want to capture the data that comes from the host runtime system before it goes to the browser and provide default values in certain fields.

DA: Any cautions when it comes to using the Web Form Renderer?

JP: The only thing I would raise is that runtime processing could get "expensive" if you implement a lot of form processing using the Renderer. Remember, the Renderer runs on the web server – not on the Client workstation.   >>

ASP.NET Web Forms Runtime Processing
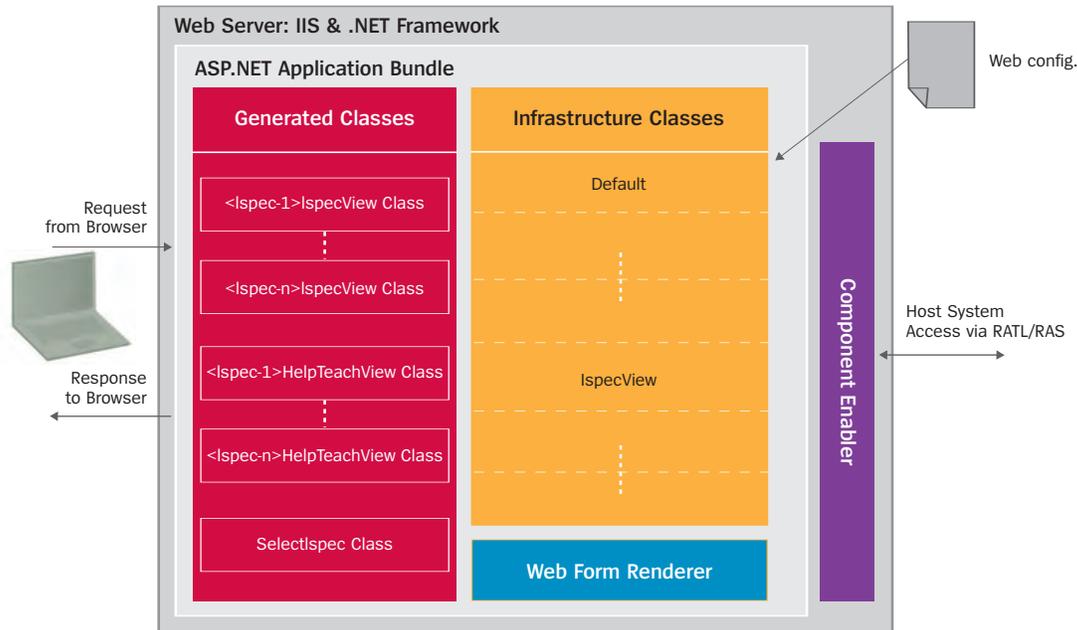
This illustration shows the interaction between the three tiers of a web-based application: web browser running on a workstation to the ASP.NET Web Form running on the web server and access to the host system application via RATL/RAS.

DA: OK, time to move on to the third option – using the Client Generator Customization Kit. What is a custom generator and how does a developer use it?

JP: Developers would use a custom generator when they have numerous Web Form changes, such as custom controls and special processing, and they want to make the process of applying those changes repeatable. If you are concerned with the potential overhead of using the Web Form Renderer for extensive customization, the better option is to update the generated Web Forms directly. Once developers are happy with the way their customized ASP.NET Web Forms work, they can then use our Client Generator Customization Kit to modify the ASP.NET Generator source and create their own custom generator. The custom generator will generate the Web Forms in the way the developers want – creating additional controls and avoiding any runtime processing that the Web Form Renderer would impose.

DA: Creating a custom generator sounds complex.

JP: I hear that a lot, but I can help people get started in a matter of minutes. We did a 40-minute hands-on workshop at the recent Unisys Technology Forum user meeting in Queenstown, New Zealand. In about 10 minutes the attendees were able to add "tooltip" support to fields using the customization kit. Don't be afraid to give it a try.

DA: When would a developer use the Web Forms Renderer and when would they create or use a custom generator?

JP: Use the Renderer if you don't want to make any changes to the generated form but you do want to impact forms processing at runtime. Developers should use the custom generator if they wanted to make changes to the generated forms and wanted that change process to be repeatable.   >>

The Four Phases of the Generate Process
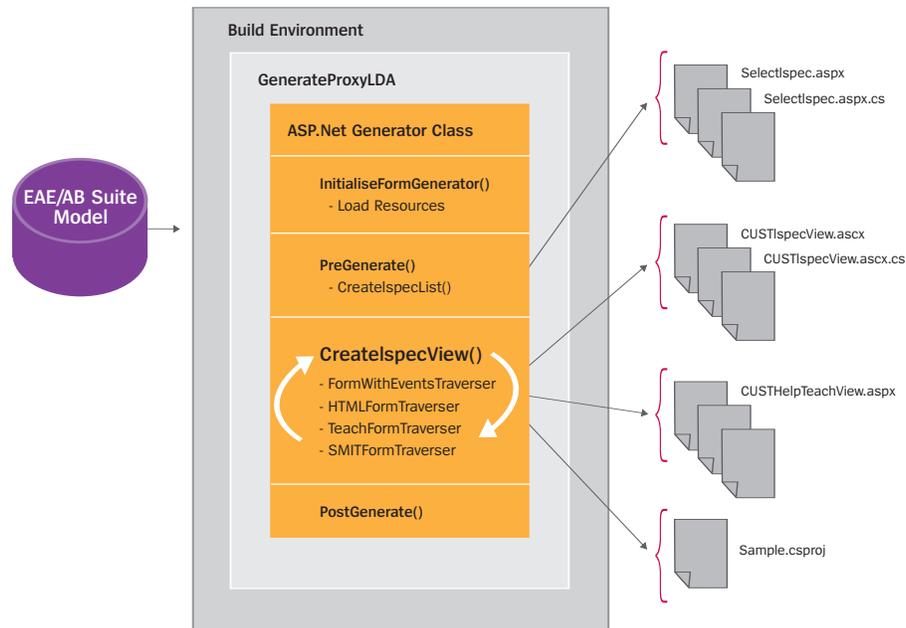
This illustrates the four phases of the generation process. The bulk of the work is done in the CreateIspecView() method, which executes the various "Traverser" classes that generate different parts of a Web Form. A Traverser class processes the data that has been populated in memory by the calls made to the GenerateProxyLDA interfaces from the EAE or AB Suite Builder. A Generator typically instantiates one or more Traverser classes, depending on the output that needs to be produced for a particular client application.

DA: Is there anything a developer can do with Client Tools in AB Suite that can't be done in EAE?

JP: No, and in fact you have the option of using the AB Suite version of Client Tools with EAE. They can be used interchangeably between EAE and AB Suite. Naturally, there are capabilities in the AB Suite form painter that do not exist in EAE. But, the capabilities of the ASP.NET client, whether you use the default, Renderer, or customization kit, are the same. It is the same client.

DA: If a developer uses the ASP.NET Generator are there any restrictions/limitations they should be aware of – especially when the plan is to go from EAE to AB Suite?

JP: No, there aren't any restrictions or limitations. Everything you can do in EAE will migrate seamlessly to AB Suite when you load the model. Customers can migrate to AB Suite, build their bundle folder, and compile their ASP.NET clients, and the user interface on the browser will be identical to the one in EAE.

DA: AB Suite Developer uses Visual Studio. Do customers need to license Visual Studio to use with Client Tools for EAE applications, as well?

JP: No, they don't need to license Visual Studio. All that's needed is the C# compiler, which comes with the .NET runtime framework. That means you can generate the AB Suite applications using the Batch files that are supplied with AB Suite, and then they can be deployed and used without Visual Studio. That said, all of the customization activities that I just reviewed do require the Visual Studio Development Environment.

Many thanks to John Papachristos for providing some great tips on how to customize ASP.NET forms and form processing.

More GUI Options: Custom Generators

A Quick Recap

This issue of Developing *Agility* is all about the many options developers have to provide a graphical end-user presentation on EAE and AB Suite applications.

First, you can use the GUI painters in EAE and AB Suite to define the end-user presentation as a part of the model and generate it for all types of clients, including, but not limited to, web browsers.

Want to do more? You could modify the generated client manually, however we don't recommend doing so because changes will have to be reapplied after a generate. But there are plenty of other options, particularly if you're interested in deploying browser-based clients. As noted in the Engineering Corner article, you can modify Client Infrastructure files, use the Web Forms Renderer to modify the ASP.NET project that has been generated, and/or use the Client Tools Customization Kit to build your own custom generator. Most of these options require some level of C# and/or .NET expertise, but the results are impressive and stay perfectly in sync as your EAE or AB Suite application changes over time.

But Wait – There's More!

Several companies have developed specialized, parameter-driven custom generators. Designed for situations where the standard ASP.NET Generator isn't enough – and an organization isn't interested in building its own custom generator – these tools make it particularly easy for developers who lack web or .NET expertise to customize the GUI. And, they enable a repeatable approach so that changes are automatically reapplied with each system generate.

Built by smart technicians that know EAE, AB Suite, and GUIs inside out – and understand the needs of end users – these custom generators are well worth considering, particularly if you don't want to build in-house web or .NET expertise. The following table lists several of the more well-known sources for custom generators.

In the end, the choice is yours – and there are many options. Which is right for your organization? That depends on two things: how you want your GUIs to look and the kind of technical expertise you have or want to develop. Consider the following:

Custom Generator Name Company/Contact Information	Description
Generator ASP.NET for AB Suite and EAE Available as a service engagement from Unisys Europe by contacting Dominique.Michaut@fr.unisys.com	With this generator, developers can easily enhance GUIs by automatically adding graphical objects not supported by the ASP.NET Generator. This is achieved by entering parameters into the custom generator tool. This tool can be a full client/server solution with generic JavaScript, which could also be customized. The custom control possibilities include: • Calendar object • List of Values • DataGrid with sort and pagination • Use of .css file (cascading style sheets) • And much more Both simple and efficient, Generator ASP.NET for AB Suite and EAE is used by the majority of EAE/AB Suite users in France, as well as other European countries. >>

Custom Generator Name Company/Contact Information	Description
Interface Builder Available from Information Exchange Group (IEG), Inc.	Interface Builder (IB) is a flexible Web Form builder that easily exploits browser capabilities with minimal change or expertise. From its first release in 1999 to the current version, IBdotNET has seen continuing enhancements and is now tightly integrated with AB Suite. IB-generated forms are used by nearly 200 organizations. In addition, three North America-based software development companies use the product. IB's strength is a flexible infrastructure that enables the addition of graphical functionality and customization of client or server processing without the need for significant web-based programming skills. The only limit is your imagination. An evaluation copy of IB can be downloaded from the IEG website. Evaluation trial period terms can be reviewed at http://ieg-ib.iegonline.net/ieg-ib.htm.
Smart Web Client Available from McGavin Consulting Group	Smart Web Client (SWC) enables developers to add many sophisticated graphic items, such as calendar buttons, drop down menus, or tab strips, to EAE graphic forms. SWC's AJAX Client feature allows end users to query and select application data without changing Ispecs. Mature users love the zoom feature, which dynamically expands the size of the window display. SWC also has features, such as function keys, hot keys, and programmable numeric keys, that data entry "speedsters" require for fast-cycle screen operations. SWC is being used by EAE development organizations in New Zealand, Australia, and the UK. It is currently being enhanced to support AB Suite, including automatic upgrade of SWC from EAE to AB Suite. Visit www.mcgavin.biz to get an overview of SWC or email to questions@mcgavin.biz.
CTC ASP.NET WebForms Generator Available from Client Tools Consultancy (CTC)	This custom generator was featured in the 1Q2008 issue of Developing Agility. Using the CTC Configurator, developers can configure the look and feel of every single control on a screen, as required, taking full advantage of ASP.NET capabilities. This includes the ability to leverage third-party controls to meet the demands of modern user interfaces. To download a trial version of the software, visit the CTC web site.

- For a great GUI that goes far beyond the old green screens of the past, most development organizations are quite satisfied with Developer painter and Presentation Client as the display mechanism
- For a fully featured browser-based GUI, the Client Tools ASP.NET Generator is an excellent choice
- Development organizations with C# and ASP.NET expertise – and a need for more sophisticated controls on their GUIs – may want to consider making additional enhancements to the web forms from the ASP.NET Generator as outlined in Engineering Corner
- Organizations that want a customized browser interface – but don't want to build new technical expertise in-house – should look at the Custom Generator options outlined in the table above

Choose the option that best fits your environment and needs – your end users will thank you for it!

Get SOA Smart

In a recent case study, Galvano Groothandel BV describes how creating a SOA leveraging its EAE applications has allowed this wholesaler of high-end plumbing fixtures to expand e-business order volume by 55 percent. SOA is a hot topic and there's never been a better time to start exploring how your EAE applications and ClearPath systems can participate.

SOA helps you derive more value from your existing business applications, including those based on AB Suite and EAE – and do so without putting at risk the revenue streams those applications drive.

Unisys wants to help you take advantage of SOA opportunities while building on the business value inherent in your existing strategic applications. To that end, we recently published a series of white papers about SOA that follow a logical progression from high-level SOA concepts to a discussion of the middleware available to allow your ClearPath applications to participate in a SOA.

The papers can be downloaded from the eCommunity:

- Service-Oriented Architecture: Delivering for Business
- Service-Oriented Architecture: ClearPath Systems in SOA
- Middleware Products and Strategies for ClearPath OS 2200
- Middleware Products and Strategies for ClearPath MCP

The Galvano case study can be read in full on the Unisys website.

Calendar

There are several upcoming opportunities to meet with fellow AB Suite and EAE users and Unisys personnel around the world. Be sure to check the eCommunity for the latest information.

What	Where	When
AB Suite April 2008 Update Webcast Recording	Available on demand	Any time
AB Suite and EAE User Meetings (Choice of two dates)	St. Paul de Vence, France	September 22-24, 2008 September 24-26, 2008
UNITE Annual Technology Conference	Caribe Royale Orlando, Orlando, FL	October 19-23, 2008