



Developing Agility

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

Contents

- Page 2 **Customer Q&A with Dutch Insurer Movir**
System Engineer, Ron Thakoer, talks about his organization's move to Agile Business Suite.
- Page 4 **Engineering Corner: Taking Migrated Applications to a New Level by Modeling for Reuse**
Grant McCauley reveals four ways to boost reusability in existing systems once the move to Agile Business Suite is complete.
- Page 9 **Silverlight Introduces New User Interface Option**
Get a quick overview of how this new technology can be used in an EAE or Agile Business Suite application.
- Page 12 **In Memoriam**
We mark the passing of two well-known EAE experts.
- Page 12 **Calendar**
Check our calendar for the latest information about upcoming events.

*Not a subscriber of Developing Agility?
Don't miss the next issue – sign up in the eCommunity*

Customer Q&A with Dutch Insurer Movir

One of the key goals of Developing *Agility* is to keep EAE customers updated on how some of their peers have fared planning and implementing their migration to Agile Business Suite. This article continues that effort with a discussion about the activities and accomplishments of Movir, a specialized insurance company located in the Netherlands that provides disability coverage to Dutch professionals in business and healthcare.

We recently spoke to Ron Thakoer, System Engineer at Movir, about the company's recent migration to Agile Business Suite targeting the ClearPath MCP operating environment. Read on to learn about what went into the process of moving Movir from EAE to Agile Business Suite and to get Ron's thoughts on the main benefits he hopes to realize as a new Agile Business Suite user.

Developing Agility (DA): Please tell us about the IT environment at Movir. How many applications do you have? What business functions do they support? About how big are they? Also, what's the size of your development team?

Ron Thakoer (RT): Movir operates three main EAE applications, two of which directly support the business, while the other is a relatively small, custom-built version control tool. Our main application, MOSYS, contains 120 ISPECs and 200 REPORTS, maintains a production database of around 10 GB, and has roughly 110 end users working with it.

The second, much smaller application closely resembles the MOSYS system, but is specifically designed to support insurance services for small business owners.

We have five Agile Business Suite developers.

DA: What business benefits do you expect Agile Business Suite to help you realize?

RT: We went live in June 2009, so the first step following the migration has been to give our IT team time to familiarize themselves with the new environment. Following this introductory period, we expect to leverage Agile Business Suite to optimize certain processes – for example, tightening the integration between Agile Business Suite and our front-office applications.

Additionally, we have plans in place to use Agile Business Suite to extend the scope of a capability we developed in Microsoft® Word to automate correspondence production. We also use Infolmage, a Unisys document management system, and we expect to simplify the process of integrating this tool with Agile Business Suite applications, as well.

DA: What about the technical benefits?

RT: Of course, we expect Agile Business Suite to help us improve developer productivity, but more importantly, we're looking forward to working in an object-oriented (OO) environment. With OO, we can optimize the structure of our applications, which will improve our ability to quickly and efficiently maintain and improve them.

DA: Prior to moving to Agile Business Suite, you migrated the graphical front-end on your EAE application from PowerClient (Graphical Interface Workbench) to Microsoft Visual Basic® .NET. Why did you select Visual Basic .NET? Did your end users notice any changes after you implemented Agile Business Suite?

RT: We had been using an integrated PowerClient and Visual Basic 6 end-user interface for our EAE application. However, with primary support for both products at an end, we knew it was time to upgrade to the latest technologies. And because we were already using Visual Basic 6, it made sense from an operational perspective to move to Visual Basic .NET using the generator included in EAE and Agile Business Suite Client Tools. We also added custom code using Visual Basic .NET to the generated end-user interface. >>

We've made some great gains in efficiency on the end-user side. For example, when a Visual Basic .NET function is invoked, our users can now stay in the same window. By modernizing this interface first, our users were able to use it with the EAE applications for a few months before we moved to Agile Business Suite. In addition, we were able to migrate our updated Visual Basic .NET GUIs to Agile Business Suite without any issues, and users saw no difference.

DA: *What features of Agile Business Suite played a critical role in influencing your migration decision?*

RT: The biggest determining factor was the robust integration functionality in Agile Business Suite. These capabilities are certainly valuable now, and will become more and more important for our company as we address future challenges.

DA: *You went live on Agile Business Suite 1.2 in June 2009. What have been your experiences using Agile Business Suite in a production environment? What do your developers think about Agile Business Suite Developer?*

RT: The move to Agile Business Suite has enabled us to maintain the quality of our operations without compromising performance. Everyone on the various Unisys teams we worked with provided valuable support throughout the project. We especially appreciated the responsiveness of the Agile Business Suite engineers, who delivered the quick fixes that helped us avoid major delays.

From a development standpoint, some of our developers are already up and running and familiar with Agile Business Suite Developer, and we expect the others to begin experiencing the full benefits quite soon.

DA: *What did you learn from the migration project that you think other IT organizations would find useful? What were the most important activities during the migration project?*

RT: We treated the migration the same as we would all other projects, and I think that's an important mindset to have. Thinking in this way helped us compile the internal resources, make the business case, and contract the necessary Unisys support personnel to guide the process.

Overall, the most important activity was allowing our end users to test the migrated applications. We also used the Unisys Business Application Test Manager (BATMan) services and toolset to replay days' worth of EAE application transactions in the migrated Agile Business Suite application, which provided the peace of mind that we touched most areas of the application prior to going live. Obviously, testing takes time, but it's an extremely crucial step to ensure a smooth migration.

DA: *What are your plans for new development with Agile Business Suite?*

RT: As our most important back-office application, MOSYS is constantly being updated and extended to address emerging end-user requirements. As such, our first activity with Agile Business Suite is using Agile Business Suite Developer to create innovative new features for MOSYS.

BATMan Ensures a Smooth Transition to Agile Business Suite

Delivered as a service, BATMan automates the development of test scripts – which capture real-life EAE or Agile Business Suite transactions – to help you test online applications before they go live. BATMan replays these transactions in a target test environment and offers the ability to confirm a successful playback by validating that the results match the original transactions. In doing so, BATMan delivers an extra measure of confidence as you move applications from test to production environments.

Taking Migrated Applications to a New Level by Modeling for Reuse

By Grant McCauley, Agile Business Suite Model Technical Lead, GTC Australia, Unisys Systems & Technology

The benefits of UML modeling capabilities, such as those offered by Agile Business Suite, are quite clear when you're developing a new application. But there's value to be gained with existing applications, especially for those who are migrating from EAE to Agile Business Suite. This article is a review of simple and straightforward ways to use the modeling features in Agile Business Suite to make your migrated EAE applications easier to support and your migrated EAE code easier to reuse. By taking advantage of these tips, you'll not only boost productivity in the long run, you'll get some hands-on modeling experience while working with your familiar (former) EAE systems.

First things first. When you migrate to Agile Business Suite, your EAE model becomes an Agile Business Suite model – and many of the basic building blocks are automatically created for you. Objects, classes, methods, attributes – all the good object-oriented stuff – will be there when you migrate. We've covered object-oriented basics in previous Engineering Corners, so I won't go into detail here. But, I do encourage you to revisit those articles when you have time.

Starting with the Agile Business Suite model of your application, there are many ways to make it easier to understand, maintain, use, and reuse. I see the opportunities divided into the following broad areas:

1. Analyze
2. Organize
3. Document
4. Develop

Let's take a look at each one in a bit more detail.

Analyze

Once your model is migrated to Agile Business Suite, a good first step is actually to take a step back – and look at what you have. There's no need to make changes simply for the sake of change, so take a bit of time to look at your model from “all angles.” Use the tabs (properties, dependency, etc.) to explore the different model views available via Agile Business Suite Developer. Check out the dependencies that come across, as well as how properties, such as kind, owner, inherits, and multiplicity, have been defined for the objects' constructs. You can also try out the Search dialog (Alt+S) and Quick Navigator (Alt+G) features to further analyze your migrated system and model.

Organize

Once you've got a good feel for how your application model looks in Agile Business Suite, it's time for some organizing. By organize, I mean taking steps that make the model easier to understand – and to eventually reuse and restructure. Once again, you aren't starting from scratch because your EAE functional areas and activities are carried over to Agile Business Suite during migration.

There are three tools I recommend you use to organize: folders, dictionaries, and diagrams. >>

Object-Oriented Refresher

For more information about object-oriented concepts, take a look through these prior Engineering Corner articles:

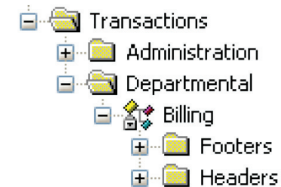
- [When Applying a Stereotype is a Good Thing](#)
- [Encapsulation](#)
- [Polymorphism](#)



Folders

The folder metaphor is an excellent means of bringing more order to your model and should be the first step you take in organizing. Use Agile Business Suite folders as you would use a physical folder – for grouping related items together. For example, you might want to:

- Place all the transactions related to a particular business area into a folder
- Create a folder that holds all your external classes or all reports or all administrative transactions
- Use folders within a Report to organize the Heading and Footing frames



Folders are so flexible that the same entity can be used in multiple folders and folders can be organized within other folders.

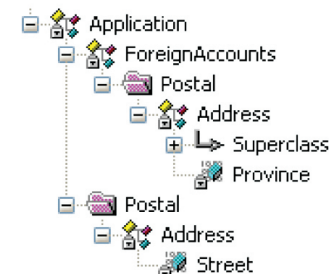


Dictionaries

In Agile Business Suite, the dictionary is a special type of folder used specifically for organizing objects that serve as definitions for other objects – such as classes that you want to easily reuse in many areas of the application. Unlike EAE, you can have as many dictionaries as you need with Agile Business Suite – and they can exist at multiple levels in the model (where they apply to all objects defined below that level.)

Why would you want to have more than one dictionary? Well, for exactly the same reasons as a folder, but with the added benefit that dictionaries facilitate reuse by automatically setting the “inherit” relationship when the name of an attribute or variable matches an object within a dictionary.

For example, multiple dictionaries might make sense if you are combining functionality from two or three applications or want to be able to redefine certain terms or definitions in a local area of the model. In one place you could have a dictionary that defines an address with street, apartment, city, state, and zip. In another part of the model, which deals with other countries, you might redefine address to include a long post code, a country, and a province. In each place it would be legitimate to refer to an address as a single object and automatically attach to the appropriate one.

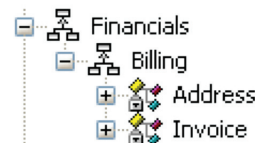


With EAE, the definitions in a dictionary were limited to just primitive types, such as string and number. In Agile Business Suite they can also be classes that include attributes, logic, and even presentation. By making entries in a dictionary, you provide a pattern for commonly used items in your model – which may end up being your first step toward real reuse.

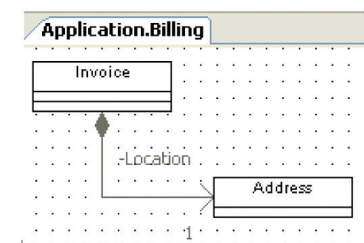


Diagrams

Diagrams are another tool for organizing. Like a folder, you can drag related items onto a diagram – creating a UML picture of the contents. Even if there are no lines (dependencies/relationships) initially connecting the elements on the diagram, you’re still ahead of the game because you have a logical grouping of related items. As they say, “a picture is worth a thousand words.”



Finally, using any of these organizing tips will do no harm to your application. They are simply tools to help developers bring a bit more order to their world. >>



Document

Document is just that – actions that help make it clear to anyone who comes into contact with your system what the intent is for any and all objects, logic, and so on. UML diagrams are just one way to describe a system – creating views into an application and showing structural relationships between classes.

Agile Business Suite also has great “document as you go” features that make it very easy to enter comments and descriptions as you are designing and developing. These same features can be used with the migrated system – think of it as “document as you discover” – to formalize your institutional knowledge about the various functions in a system you already know well. And, you can add documentation for various audiences. Information about what business function an object supports and future plans to extend a function could be helpful to designers. End users benefit from documentation that explains how to use an Ispec or the source of list data.

Start by entering documentation at the folder level and work your way down to more detail as time and interest permit. Or if you are working in an area of the model and you discover something interesting, open up the documentation window at that point and jot down your findings. A WordPad-like function makes the process almost painless.

Finally, with Agile Business Suite comes a new ability to extract application documentation. Comment pages, which are HTML-based navigable reference guides, provide a more robust and user-friendly solution when compared with the standard EAE Print Specification output. Online documentation is easy to maintain when it's automatically generated from your system whenever needed.

Develop

Up to this point, every action I've recommended has not actually changed your application model. Analyze, organize, and document simply make it easier to understand. Developing an existing system is about refining and extending the application, which gets to the heart of modeling for reuse. And, here are four steps you can take to do so: eliminate, encapsulate, decompose, and generalize.

Eliminate

Eliminate is about refining the roles of entities in an application – removing the unnecessary or redundant capabilities that no longer serve a purpose. Because EAE offers just high-level entities, there will inevitably be opportunities for refinement. For example, Groups may have been used to define structures without requiring the string serialization capability or Insertables may have been used for forms without the need of special macro code insertion. Eliminating unused capabilities not only simplifies the definition of an entity, but also enhances its ability to be reused.

So keep an eye out for objects in Agile Business Suite that are not utilizing their stereotyped behavior – and set them back to a (no stereotype) class, which allows your object to take full advantage of object orientation.

Encapsulate

Encapsulation facilitates reuse by grouping common functionality and controlling access to it, which eliminates unintended side effects. When an object is well encapsulated, it becomes a “black box” – providing an interface to use it, but keeping the details of implementation hidden inside. With this approach, you can reuse the object anywhere in your model and know exactly how it will behave. More to the point, any internal change to the object will have no external impact. >>

You can begin to encapsulate objects by changing the visibility of their members. By default, most data items are migrated from EAE as “public” attributes and are accessible from any other part of the application. If you want to protect the data in an Ispec class from being modified outside of that class, set its visibility property to “private.” Changing visibility will quickly identify logic outside of that class that is trying to access the now internal data, because you’ll get an error as soon as you validate the model. The journey of encapsulating continues by moving this logic to a method within the same owner as the data.

Start by doing this with one or two attributes to see the impact. As you go through your model you may see patterns emerging resulting in fewer methods, the elimination of code duplication, and the growth of more powerful classes.

Here is an example: If your model has an Ispec class called ACCT and you make the BALANCE attribute private, when you validate the model you may now find that you have many places outside of ACCT where you have logic that looks something like:

```
Subtract TRANTOTAL ACCT.BALANCE : Subtract total from the account balance
```

You may also see that nearby you have logic to check the balance and report an error if the balance falls below a certain limit. The conditions that determine whether the account balance is allowed to be changed should be governed by its “owner,” the account class. By placing that logic inside the class and making it available through a single method, you not only protect the balance from being updated incorrectly, you can simplify the code in other parts of the model and make it easier to change the update logic if the business requirements change at some point in the future.

Decompose

Now, I’m not talking about decay here. In computer terminology, decomposition is the process of creating smaller, more manageable objects from larger objects. In EAE, it was easy enough to copy logic from one Ispec to another – and from one application to another. With Agile Business Suite and the object-oriented concepts, the approach is different. Common functions should be defined as classes and methods, which can be shared across one or more applications. This reduces complexity and promotes reuse. Decomposition combines repetitive logic into a small number of methods – and groups these methods into classes from which they can be reused.

Consider the example of managing inventory. When your system is processing a cash sale, credit sale, or return, inventory for the item is updated the same way (with a plus or minus thrown in). So it makes sense to define a new class called StockLevel that has a method UpdateStock to do the plus-ing and minus-ing anytime inventory levels must be changed. And with this approach, should there be a fundamental change in how stock levels are managed, there’s just one place that logic changes – but the functionality impact will be felt across the application.



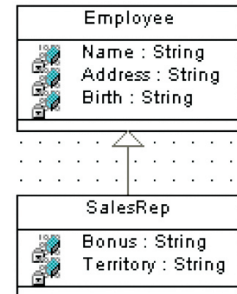
You can see clues for commonly used patterns if you take the time to organize your system into a UML diagram. The relationship there will show you common data elements, logic, and more – which are all opportunities to decompose. >>

Generalize

The last technique to employ is to generalize, which is about finding ways to use inheritance to simplify your existing model. Look for common characteristics and behaviors that are shared by more than one class and define them in a new superclass. Then, you can allow your existing objects to become subclasses that inherit and extend their parents' class. And, when requirements change, you simply change the superclass and – through the magic of inheritance – those changes are realized in the subclasses.

The classic example of generalization is an employee. Employees have a defined set of characteristics and behaviors: name, address, birth date, and so on. From there, you can define certain types of employees, such as sales representatives. These inherit all the characteristics of employees but they also have their own special attributes, such as bonus structure or territory.

Looking for opportunities to generalize allows you to group common data items, logic, and even presentation into a generalized group – and then define specializations to create subclasses underneath.



Taking Your EAE Model to New Levels

Moving to an object-oriented development environment stands to deliver productivity gains for your organization. Following the steps outlined in this article is a great way to build your familiarity with Agile Business Suite – while moving your existing models further down the UML continuum.

Silverlight Introduces New User Interface Option

In July 2008, we devoted a [major portion of Developing Agility](#) to the topic of user interfaces, featuring articles about the many options developers have to make their EAE or Agile Business Suite applications easier for people to learn and use. At that time, your options included the GUI painters in EAE and Agile Business Suite, Client Tools, such as the ASP.NET Generator, and custom controls, like the Web Form Renderer. We also highlighted the ability to create a custom generator using the Client Tools Customization Kit – or use one built by a knowledgeable third party.

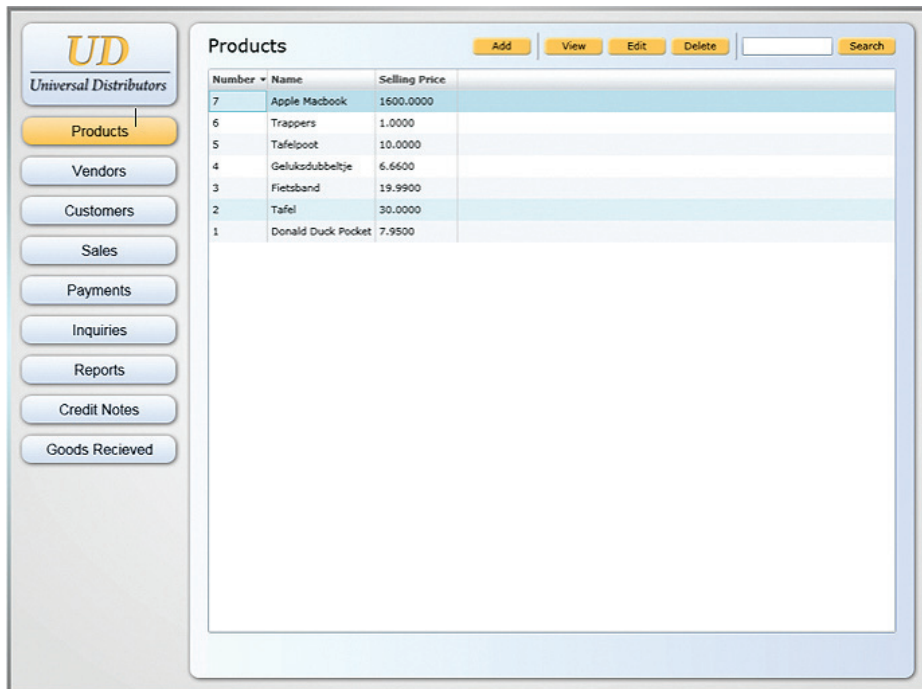
Now there's a new kid on the block – Microsoft® Silverlight™ browser plug-in development tool – and that means there's another tool you can use to build a better GUI.

You can read all about [Silverlight](#) on the Microsoft website and even download a copy of the development kit. Silverlight offers a lot of cool capabilities to further enrich the end-user experience. It's important for EAE and Agile Business Suite developers to know that it takes nothing more to make Specs available for enhancement via Silverlight than it does for other GUI options. Simply generate those transactions you want to enhance as Web Services and get to work. Silverlight is a Visual Studio plug-in, so if you're developing with Agile Business Suite, the two environments are very closely aligned.

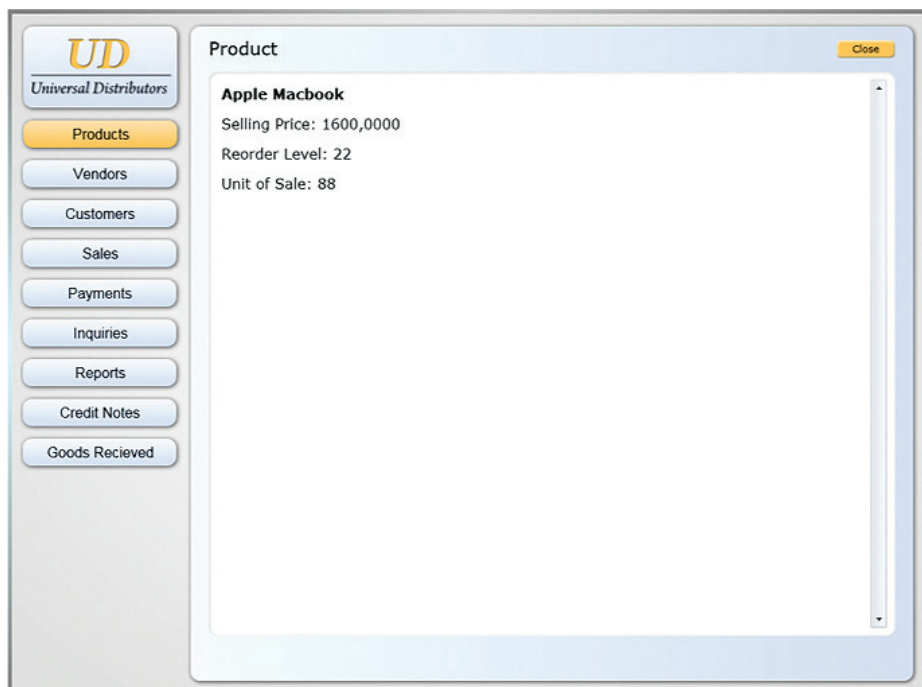
Interestingly, a group of university students in the Netherlands recently completed a project that demonstrated the use of Silverlight with our old friend the “Sample” application. Using Agile Business Suite and Silverlight 2.0, the students were able to go from a basic screen presentation to a great, new graphical front-end in a matter of weeks – and without detailed knowledge of either tool (but with some Visual Studio familiarity).

The team started off on exactly the right foot by evaluating the old interface and whiteboarding ideas for updating the presentation. Of particular note is how they used Silverlight to combine the Product List, Details, and Maintenance screens onto one form. >>

In the sequence of screen shots below, the user clicks the Product button to get a list of all products, using a nice grid control.



Another click on a particular product provides the details, as shown in this next screen shot. >>



Finally, authorized users can maintain that product information using the same form.

The screenshot shows a web application interface for 'Product Maintenance'. On the left is a vertical sidebar with the 'UD Universal Distributors' logo and a list of menu items: Products, Vendors, Customers, Sales, Payments, Inquiries, Reports, Credit Notes, and Goods Received. The 'Products' menu item is highlighted. The main content area is titled 'Product Maintenance' and contains two sections. The first section, 'Product Information', has a yellow header and contains four input fields: Name, Selling Price, Reorder Level, and Unit of Sale. The second section, 'For Price Changes Only!', also has a yellow header and contains two input fields: Your Name and Reason. At the bottom of the form, there are three buttons: 'Submit', 'Cancel', and 'Reset'.

In each of the above steps, only that part of the screen that is directly affected is refreshed – rather than the full screen. In addition, the students used the same metaphor for vendor and customer maintenance, which provides a very consistent experience for users.

At the end of the project, the students concluded that Silverlight was fast and simple to use and provides a good way to refresh the look of older applications.

Whatever tools your organization chooses to leverage – Silverlight or one of the many other options – we encourage you to improve the usability of your high-value EAE and Agile Business Suite applications with new user interfaces. It's an investment that's sure to pay off in the long run.

In Memoriam

We are saddened to relate news of the tragic passing of André Paridaens, who lost his life in a motorcycle accident on May 1, 2009, in his native Belgium. Andre brought a tireless enthusiasm and unparalleled expertise in many technology areas, including EAE and Agile Business Suite, to the benefit of customers and his fellow Unisys employees. His unique style and approach set him apart from all others – and his loss is keenly felt by all who knew him. Andre was 55, and is survived by his wife and two children.

We have been remiss in alerting the EAE community of Colin Zealley's passing following a short illness on October 13, 2007. Colin worked for Unisys for 36 years, beginning his career with Sperry. He was a highly respected technical architect, an exceptional advocate for EAE, and prolific contributor to newsgroups on a very wide range of topics. Colin brought sincerity to all that he did, perhaps most notably so when representing his colleagues as a member of the Unisys UK Works Council. Colin's outspoken personality and offbeat humor are still missed today.

Calendar

There are many invaluable learning opportunities available to you. Please be sure to check the Webcasts & Events section of the [eCommunity](#) for the latest information.

What	Where	When
OS 2200 Best Practices Workshops	Brussels, Belgium	September 24, 2009
MCP Best Practices Workshops	Houten, Netherlands Brussels, Belgium Milton Keynes, UK Paris, France	September 29, 2009 September 30, 2009 October 7, 2009 October 13, 2009
EAE/Agile Business Suite User Meeting	Paris, France	October 13, 2009
ClearPath Briefing and OS 2200 Best Practices Workshop	Stockholm, Sweden	October 15, 2009
DACH User Event and OS 2200 Best Practices Workshop	Basel, Switzerland	October 22-23, 2009
UNITE Annual Technology Conference	Hyatt Regency Minneapolis, Minneapolis, MN	November 8-11, 2009
Business Information Server (BIS) Symposium	Unisys Roseville, MN	November 12-13, 2009

Specifications are subject to change without notice.

© 2009 Unisys Corporation.
All rights reserved.

Unisys and the Unisys logo are registered trademarks of Unisys Corporation. Microsoft and Visual Basic are registered trademarks and Silverlight is a trademark of Microsoft Corporation. All other brands and products referenced herein are acknowledged to be trademarks or registered trademarks of their respective holders.