

WHITE PAPER.

Model-Driven Application Development with Agile Business Suite

Alan Hood

Unisys Systems and Technology

Agility—your survival depends on it. Agile organizations react swiftly to changing market conditions and seize new opportunities. Change impacts every element of your enterprise, so it's critical that your development environment supports your ability to respond. Agile Business Suite can help—connecting business and IT professionals at the highest level to let business needs drive application development.

- > Consulting.
- > Systems Integration.
- > Outsourcing.
- > Infrastructure.
- > Server Technology.

UNISYS

Imagine it • Done •

CONTENTS.

Introduction.	1
▶ Model-driven development.	1
▶ Modeling benefits.	2
▶ The Agile Business Suite difference.	2
Introducing Agile Business Suite.	3
Solution definition with System Modeler.	4
▶ Develop composite applications.	4
▶ Define the complete application.	6
▶ Enterprise-class, model-based application development.	6
Solution generation with Builder.	7
▶ One model. Multiple environments.	7
▶ Generate 100 percent of the application.	7
▶ Derive the database from the model.	8
▶ Maintain multiple configuration.	8
▶ Scenario one: multiple environments to support business continuity.	9
▶ Scenario two: multiple environments to support QA activities.	9
Solution deployment with Runtime.	10
Agility to support change.	11
Summary.	12
▶ One tool does it all.	12
Biography.	14
▶ Author.	14

Model-driven development.

What is a “model”? And what do we mean by “model-driven” development?

In its simplest definition a model is a representation of something that may already exist in the real world, or may be produced at some point in the future. A model is used to help individuals envision what the thing may look like and how it will behave under varying circumstances—and to communicate this information to other people. Models come in many, varied forms. An architect will usually draw a picture of a building with different plans and elevations in order to show a prospective buyer the layout and how it will look. An auto manufacturer might build a three-dimensional scale replica of a new car in order to get opinions on its styling. A physicist might write a set of mathematical equations that predict the effect of high winds and heavy rain or snow on a new suspension bridge.

Models provide benefits in two ways. First, they allow the developer to communicate important information about the object they are modeling with other people—even if that object does not yet exist in the real world. Second, models can be tested and changed much more easily, more quickly, and at far lower cost, than changing the building, the car, or the bridge.

In the realm of information technology, model-driven development is very similar. It allows developers to describe a part of an application, or even a complete system, in ways that make it easier to communicate with other developers and end users. Models help people to envision how the system will behave. And, it is easier to change IT models than functioning systems, so organizations can react more quickly and cost effectively to changing business requirements using model-driven development.

In reality, most designers and programmers engage in some level of modeling, even if they aren’t aware of it. The “model” may be as simple as a block diagram on a white board. It is far easier to describe the problem and make changes to the drawing on the white board than it is to code, compile, and test programs—and then make changes if the result isn’t quite right.

A model-driven development environment places a stronger emphasis on more formal modeling processes. Model-driven methodologies tend to emphasize the design aspects of the process, with the understanding that if the design is done properly, it will be much easier to write the application and the application will tend to have fewer errors. Using modern model-driven development environments, developers find they write less lower-level code, because they can reuse major pieces of functionality identified at the model level. The fundamental goal of model-driven development is to be able to get to the point that all of the lower level application elements can be completely derived directly from the model.

Modeling benefits.

Ultimately, model-driven development improves communication, reduces the amount of rework and cost of developing applications, and speeds the time to market for new applications. When design and development are tied together with modeling, the cost of maintenance can decrease. Plus, IT organizations can become more responsive to changes in business requirements.

The Agile Business Suite difference.

Most model-driven toolsets require users to switch from one component to another at each phase of development. This requires different tools and different skills—and frequently means losing some of the information each time the model is transitioned from one source to another. Unisys Agile Business Suite is different—it employs a single integrated development environment (IDE) in which you can design, test, deploy, and maintain your applications. In fact, Agile Business Suite allows you to generate 100 percent of your application from a high-level model. It is also the only tool that integrates development of user presentation, business rules, database, and more—from that single integrated environment.

Agile Business Suite (AB Suite) is the only development environment that can support such diverse runtime environments as Microsoft® Windows® .NET, Java 2 Enterprise Edition (J2EE), and Unisys mainframe platforms, without requiring changes to the application source. With Agile Business Suite, you can build world-class application solutions that support the needs of your business today—and protect your investment well into the future.

INTRODUCING AGILE BUSINESS SUITE.

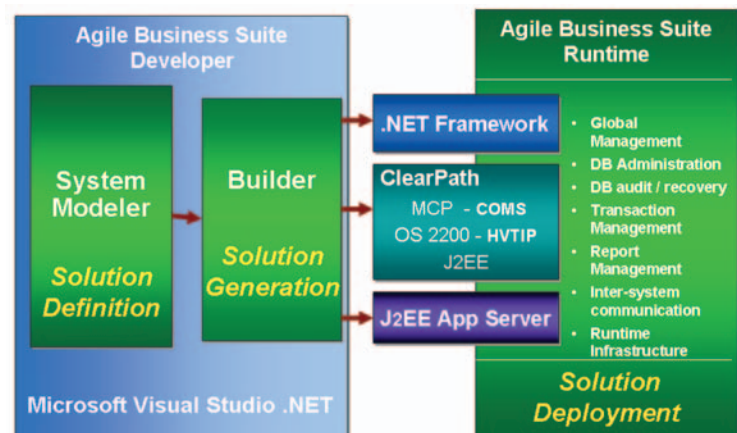
Agile Business Suite is a complete rapid application development and deployment environment that enables enterprises to quickly implement new processes and encapsulate business knowledge as reusable components in a Service-Oriented Architecture (SOA) environment.

Agile Business Suite offers model-based and model-driven development capabilities that allow users to define and modify applications at a higher level of abstraction than traditional 3GL languages, such as C++ and Java. Developers and business specialists can collaborate and think of applications in terms of what—not how. That is, specify what an application should do—and not worry about how the application should be coded to do it.

Once the application requirements are defined in the model, Agile Business Suite translates those requirements and specifications into a fully functional application that is customized for the target deployment environment. Developers are able to design, develop, generate, and deploy applications to a diverse range of runtime platforms—all from a single, high-level application model—and without making changes to that model. Agile Business Suite provides a complete runtime environment to support the solution—customized for the specific deployment platform.

Agile Business Suite is comprised of two major components, Agile Business Suite Developer and Agile Business Suite Runtime (see Figure 1.) AB Suite Developer includes the System Modeler, Builder, Version Control and more. It is installed on a Microsoft® Windows® workstation, and operates as a plug-in to Microsoft Visual Studio. AB Suite Runtime is installed on the target runtime platform, and provides an infrastructure in which the deployed AB Suite components will run. The runtime environment is different for each of the target platforms supported by Agile Business Suite: Windows .NET, Unisys ClearPath MCP and ClearPath OS 2200, and J2EE application servers, including JBoss, WebSphere, and WebLogic.

Figure 1: Agile Business Suite



The Agile Business Suite Developer is used to define and generate applications. AB Suite Developer is integrated into the Visual Studio .NET 2003 environment—providing a rich operating framework that is familiar to developers while enhancing Visual Studio capabilities via model-based development.

System Modeler in AB Suite Developer provides a model-based, object-oriented development environment. It enables a development team to model and build an application from the ground up, including the end-user presentation, classes, and objects that represent business logic, and automatic generation of the associated database tables. System Modeler's core attributes include the ability to rapidly develop complex business applications, make development personnel more productive, and handle changes much more effectively and efficiently. With System Modeler, developers manipulate the model—not platform-specific code and data structures. This platform-independent model (PIM) is then used to generate the entire application.

System Modeler functions as a "plug-in" to Visual Studio. It is used to define a separate project type, just like the various project types that can be developed using such tools as Visual Basic or C#. Within a single instance of Visual Studio, developers can define, view, and maintain many different types of projects. As a result, it is easy for developers to switch between System Modeler and other toolsets within the Visual Studio environment—using whatever capability is best suited for the job at hand.

System Modeler provides an application view that is closely aligned to an object-oriented, model-based development methodology. Users of System Modeler define programming objects via a rich set of pre-defined, stereotyped classes that implement proven patterns for high-volume, business-oriented transactions. The parts of the application that may be easier to express in a more procedural fashion can be quickly modeled and deployed from the System Modeler view, as well. System Modeler also includes the capability to create end-user interfaces and application logic, as well as derive the database structures from its higher-level application model.

While supporting the latest innovations in system definition, modeling, and development, System Modeler enjoys a rich heritage that incorporates the lessons learned from years of experience in application development technology. It is based on a toolset that has been used by thousands of people to create and maintain business-critical application systems in many types of enterprises, including financial services, public sector, transportation, telecommunications, and commercial market segments.

Develop composite applications.

AB Suite Developer provides an ideal platform for development within a service-oriented architecture (SOA) and for the creation of composite applications. By definition, an SOA is comprised of independent services that may come from many sources, including legacy applications, purchased package software, and new custom-developed components created using Agile Business Suite, a Java 2 Enterprise Edition (J2EE)-based tool, or any number of application development products. In an SOA, services are accessed in a standardized manner and may be combined to perform more complex functions. Developers can blend services without having to be

concerned with how each service was written—creating composite applications from a variety of individual components developed by a wide range of tools.

There are many benefits from this approach. Application components, defined as services, are easier to write and debug, because they tend to be smaller and focus on a single function. Once they are written, they can be readily reused, because they are defined with an industry-standard interface that makes it easier for another programmer to include that functionality in the application. And since each service component is autonomous, there is less to test and debug. You don't need to spend a great deal of time testing someone else's components, as long as you test the interfaces you use.

Application development largely becomes a process of assembling services in different ways to achieve the desired result at one point in time. If your business requirements change, simply rearrange the components by adding or replacing them with new ones that perform the required functionality.

The benefits are clear, but so are the challenges. One concern in particular is that with components coming from many resources, it's difficult to find one tool that can model all of them—regardless of how each service was created. Lack of a consolidated view hurts developer productivity and makes the modeling process unwieldy. For example, without the ability to model the entire solution at a relatively high level, a developer can get lost in the details. With some development environments, the higher-level plan of how all of these disparate components fit together may not be available to detail-level developers. It may require different tools from those available on their desktop. And if they do have a high-level model, it is often out of date because of the changes in the application's design over time.

Because Agile Business Suite is completely integrated with Visual Studio .NET, developers can define business solutions that include Agile Business Suite-generated components and services from other environments, including standards-based components developed using third-party technologies. System Modeler has the ability to model all these components within one tool.

For example, if a J2EE component was created using Eclipse, System Modeler developers can import its definition and use it inside the Agile Business Suite model. They may not be able to examine the actual code, but that's not necessary, as long as the interface is known and the necessary service is provided. Developers can include these third-party components in their application definition just as if they had been developed using System Modeler. They can incorporate the details of the methods and parameters from the interface, or they can pull back to see the big picture of how these components fit together to satisfy the business requirements.

Agile Business Suite provides great flexibility and control for environments with significant legacy applications and data. With it, developers can integrate existing functionality into a new composite application—and further extend that application with new functionality created using the toolset. In turn, new services can be defined using Agile Business Suite and then called from a legacy system. Further, composite applications can be built and maintained faster and more easily by combining the strengths of Agile Business Suite with existing resources in the organization.

Define the complete application.

Many modeling tools still require the use of additional tools to fully define the application—such as the user interface and database structure. This lack of integration at the model level makes it difficult to truly visualize the application. And, it creates additional work for developers to move from the model stage to a running application.

Agile Business Suite is different—and makes development of a complete application from a higher-level definition possible. It extends traditional object-oriented capabilities with many additional labor-saving features, such as the ability to define a graphical user presentation for any class and class stereotypes. For example, there are stereotypes that support best practice patterns for online transactions and batch reporting. And, these interfaces can be deployed to a variety of environments, including:

- ▶ Workstation clients, such as Windows Forms and Visual Basic
- ▶ Web interfaces, such as Web Forms and Active Server Pages
- ▶ Programmatic interfaces for custom clients

In System Modeler, any class can potentially have persistent data (database tables and profiles) automatically defined as a part of model creation. This ability to build the database definition from the model is a tremendous improvement over traditional development tools. It eliminates the need to employ specialists to define the database structure and interpret the database design back to the application programmers and end users. The database structure is automatically derived from the needs of the business. And, when requirements change and the model is revised to reflect those changes—the underlying database is automatically modified, as well.

Enterprise-class, model-based application development.

System Modeler incorporates the best of model-based, object-oriented design and development processes. Features, such as encapsulation, inheritance, and polymorphism, are mapped very naturally onto the model.

System Modeler further extends the model-based approach—adding a very powerful scripting language and the ability to define programming objects in terms of the business operations they perform without getting bogged down in the details of how they should be implemented on a particular hardware and software platform.

Once a project has been defined using System Modeler, it's built and deployed using the Agile Business Suite Builder. AB Suite Builder provides a number of unique capabilities, including the ability to:

- ▶ Deploy to multiple operating environments from a single model
- ▶ Generate 100 percent of the application code—including user interfaces, transaction control, and more
- ▶ Automatically define a complete, enterprise-class database from the model

Let's examine each of these capabilities in more detail.

One model. Multiple environments.

Agile Business Suite-created components can be deployed to a wide range of operating environments, including:

- ▶ Microsoft .NET Framework
- ▶ Unisys ClearPath Plus enterprise servers
- ▶ J2EE application servers

The real power of Agile Business Suite is that an application can be built for and deployed to any or all of these very different runtime environments—from one model and without changing a single "line of code" in the model itself.

Generate 100 percent of the application.

Many model-driven tools claim to generate application code, but actually create the skeleton of a program where details are left to be filled in by developers. With most model-driven, UML-based tools, code generation is limited to producing subprogram header definitions, parameters, interface descriptions, and perhaps, small snippets of logic for frequently used constructs.

With Agile Business Suite, 100 percent of the application code is produced directly from the model—no additional programming outside the tool is required (although it is supported, if your technical requirements so dictate.) And, AB Suite Builder is much more than a code generator. It can also produce the graphical user interfaces that developers define with System Modeler. In addition, it defines and creates transaction control, built-in security, batch report processing, and much more. Moreover, the code created by AB Suite Builder is automatically and completely tailored for the specific architecture where the application components will be deployed. Developers do not need to be experts in all areas of a given operating environment or server platform in order to build applications that run well on that platform.

Derive the database from the model.

Unlike most development tools, with Agile Business Suite, the database is automatically derived from the model. And the result is not a "least common denominator" definition of a generic database management system, but one specifically tailored to the platform for which it is intended.

For example, when Agile Business Suite is used to develop an application for a Windows .NET environment, the database tables it creates will be a Microsoft SQL Server or Oracle relational database management system. Agile Business Suite determines, from the model, what tables need to be created, defines the primary and secondary indices, and generates the SQL procedures required to perform the potentially complex queries that are needed by the application. Developers do not need to explicitly define the tables or write any of the SQL.

Now if components of that same application are generated to run on a ClearPath Plus MCP server, Agile Business Suite will use DMS II, a network hierarchical database engine, as the underlying database of choice. It will create the DASDL that is required to define the data structures and the access methods needed by the application.

Even though the database structures, and the way you get to them, are very different in these environments, they can be created from the same Agile Business Suite model. In other words, developers do not need to be experts in database technology to develop database-oriented applications that run well in these very different environments. Agile Business Suite selects the database engine and database structures that are most appropriate to the target runtime environment and produces an efficient database for that platform.

Maintain multiple configurations.

Finally, AB Suite Builder provides the ability to simultaneously maintain different configurations and deploy application components to different environments and hosts—and even different platform types. It specifies the details of a deployment target environment using configuration information that is maintained in Agile Business Suite Developer—along with the other model details. This capability is useful in a number of situations, including the following examples.

Scenario one: multiple environments to support business continuity.

Consider a situation where a company's business continuity strategy changes with an impact on the application deployment environment. Originally, application components were built to provide business functionality on a corporate hub and deployed to an enterprise server in the headquarters data center. Then for business continuity reasons, the organization decided that instant access to these same components in its branch offices is required at times when the network is down. Local servers are deployed to branch office locations to support this decision.

Using traditional tools, the application would need to be rewritten to deploy to an alternate operating environment. However with System Modeler, a new configuration for the local servers is defined and the application generated. AB Suite Builder takes care of the rest—including creating a new database definition specifically for the branch environment.

Scenario two: multiple environments to support QA activities.

The ability to target multiple environments can also be helpful to support Quality Assurance (QA) activities. Ideally, QA analysts want to test in an environment that is nearly identical to production. But they may not require a full-blown database or may need to run on a different server from the production system. This often means that coding changes are made to substitute QA database specifications or host information.

That's not necessary with Agile Business Suite. QA can define the required configuration by establishing an alternate configuration based on production and simply changing the properties as required. When system qualification is complete, the application components are generated using the appropriate production configuration. And because nothing in the model has changed, generation for the production environment is accomplished with the highest degree of confidence possible.

SOLUTION DEPLOYMENT WITH RUNTIME.

The application components generated by Agile Business Suite are deployed to and run in the Agile Business Suite Runtime. Runtime provides an infrastructure that is specifically tailored to the operating environment of the target platform. It is not like a virtual machine, where the generated code runs in an interpretive mode which may not be as efficient on some platforms. Rather, Runtime provides the online transaction and batch services and utilities required to manage the database environment. Again, these services and utilities will be very different depending on the target platform—whether a ClearPath server, Windows .NET environment, or a Linux server running J2EE.

Runtime also includes services and libraries that enable your AB Suite application to communicate with other components, in both proprietary and standard methods and protocols. Where appropriate, Runtime ties into the database management and transaction management of the operating environment. But even if the platform does not provide such management capabilities, AB Suite does through the Runtime environment. In part, this is what allows an AB Suite application to be deployed to different environments without making changes to the application model. The Builder and Runtime are customized to each environment and hide the complexity from AB Suite developers.

AB Suite Runtime will support Windows .NET, ClearPath Plus MCP, ClearPath Plus OS 2200, and J2EE (Java 2 Enterprise Edition) with different application servers, including JBoss, WebSphere from IBM, and WebLogic from BEA.

AGILITY TO SUPPORT CHANGE.

Change is inevitable. And, an application that cannot be easily changed is one that will not continue to satisfy the needs of its end user or the organization it is meant to serve. The larger and more complex an application becomes, the harder it is to modify when requirements evolve. Tools that speed up and simplify initial development—but do little to support on-going change—are not effectively supporting the needs of today’s business environment.

A model-driven development approach fundamentally supports change. After all, it is far easier to modify a model than the Java, C++, SQL, or COBOL code created using more traditional development tools. When business requirements change with a model-driven development approach, the model is updated to reflect the new business processes.

What if you could go directly from the model of the system to the functioning application, without the need to translate the model into user presentation, application code, and database tables? Not only would the development of the initial application be that much faster, but it would also be incredibly easy to modify the application when any business needs change. By simply changing the model, you change the application system.

Agile Business Suite provides just such capabilities. Update the model to reflect changed business requirements or processes and regenerate the application. Agile Business Suite does the rest. Code and database structures are recreated. All of the objects affected by the change are checked and modified if necessary—and only those objects that have changed are regenerated.

The power of Agile Business Suite extends into deployment, as well. When the modified application is redeployed, database structures are changed automatically and data is seamlessly migrated to the new structures. If necessary, the database is automatically reorganized before the new application components are brought online.

Agile Business Suite takes your development efforts from architecture to implementation in one easy step—and extends that simplicity into the future to support ongoing enhancement and maintenance.

One tool does it all.

Agile Business Suite is the one application development and deployment product suite that can define, generate, and manage complete, highly-scalable, real-world composite applications. Applications that are easily adapted to changes in business rules and processes and can be deployed across multiple operating environments.

Agile Business Suite is better than .NET and Java (J2EE) alone, providing such differentiators as:

- ▶ A higher level of definition
- ▶ Platform independence
- ▶ Ability to mask complexity
- ▶ Capturing the what without worrying about the how

And because host operating environment and platform decisions can be delayed until the solution is ready for deployment, Agile Business Suite is superior to 3GL and 4GL environments.

Agile Business Suite offers a wealth of valuable capabilities, including

- ▶ Easily building and managing composite applications
- ▶ Supporting a Service-Oriented Architecture (SOA)
- ▶ Allowing one model (single source) to be deployed to multiple and differing environments
- ▶ Using standards-based structures to provide both development and runtime interoperability
- ▶ Providing a model-based, object-oriented development environment that is based on proven technology
- ▶ Ensuring rapid application maintenance by providing the ability to easily make significant changes to a solution and quickly redeploy to production

In short, Agile Business Suite combines the best of the world-class application development capabilities that have been part of the Unisys core competency for many years with the best the industry has to offer in terms of object-oriented, component-based application development. It blends proven technology with the best of Model Driven Architecture and emerging standards.

To learn more about Agile Business Suite, please contact your Unisys representative.

Or call Unisys at:

1-800-874-8647, ext. 840 (U.S. and Canada)

00-1-585-742-6865, ext. 840 (Other countries)

Or visit us on the Web at:

<http://www.unisys.com/AgileBusinessSuite>

BIOGRAPHY.

Author.

Alan Hood, a Senior Marketing Product Manager, has been with Unisys since 1983. He has spent a large part of his career as a technical consultant and architect, helping customers apply the latest in technology innovations to solve their business problems. Since 2000, Alan has been part of the marketing and program management team in the Systems and Technologies organization. In that role, he has worked on the development of both the Enterprise Application Environment and the newest offering—Agile Business Suite.

NOTES.

**To learn more about Agile Business Suite,
please contact your Unisys representative.**

**Or call Unisys at:
1-800-874-8647, ext. 840 (U.S. and Canada)
00-1-585-742-6865, ext. 840 (Other countries)**

**Or visit us on the Web at:
<http://www.unisys.com/AgileBusinessSuite>**

Specifications are subject to change without notice.
©2005 Unisys Corporation
All rights reserved.

Unisys is a registered trademark of Unisys Corporation.
Microsoft, Windows, and Visual Studio are registered
trademarks of Microsoft Corporation. All other brands
or products referenced herein are acknowledged to be
trademarks or registered trademarks of their respective
holders.

Printed in US America 10/05



4126 5067-100