Microsoft Dynamics AX 2009

Programmability in Microsoft Dynamics AX 2009

White Paper

December 2008
Contents

Introduction .................................................................................................................... 4
  Scenarios ..................................................................................................................... 4
  The Presentation Layer .............................................................................................. 4
  Business Intelligence and Reporting ....................................................................... 4
  Business Logic and Data Base .................................................................................. 4
  Web Services ............................................................................................................ 5
  Workflow ................................................................................................................... 5
  Future Directions of Programmability in Microsoft Dynamics AX .......................... 5

Programmability Scenarios ........................................................................................ 6
  Customization ......................................................................................................... 6
  Add On Functionality ............................................................................................... 6
  Exposing business functionality through Services .................................................. 6

The Presentation Layer ............................................................................................... 7
  Rich Client .............................................................................................................. 7
  Web Browser Client ............................................................................................... 10
  Other Client Types ............................................................................................... 13

Business Intelligence and Reporting ......................................................................... 14
  Microsoft SQL Server Reporting Services Reports ............................................... 14
  Analysis Services ................................................................................................. 15
  Native Reporting Tool ........................................................................................... 16

Business Logic and Data Base .................................................................................... 18
  MorphX ................................................................................................................. 18
  X++ ....................................................................................................................... 19
  CLR-Interop ......................................................................................................... 20
  Service References ............................................................................................... 21

Web Services and the .Net Business Connector ....................................................... 22
  AIF (Application Integration Framework) Enabling Features/Data as Web Services .. 22
  .NET Business Connector ..................................................................................... 23

Workflow .................................................................................................................... 24

Future Directions of Programmability in Microsoft Dynamics AX .......................... 25
  Will X++ become a managed language? ................................................................. 25
  What will happen to MorphX and the way to design application objects in the Future? 25
  How will Business Intelligence and Reporting develop? ........................................ 25
  What will happen with Microsoft Dynamics AX in relation to Service Strategy? .... 25
**Introduction**

This white paper is aimed at Technical Decision Makers and is written to provide the reader an overview of the wide range of possibilities offered by the programmability provided in Microsoft Dynamics® AX 2009.

Creating new features or changing the existing ones can be done in several ways. The collection of development tools available for building applications in Microsoft Dynamics AX 2009 is the same regardless of whether the developer is from Microsoft, an independent software vendor, a Microsoft Dynamics AX partner or even a local developer in the customer’s IT organization – this means that any developer with the skills and background necessary to use these tools, has exactly the same possibilities when it comes to writing powerful applications in Microsoft Dynamics AX or customizing an existing one.

This document includes the following areas:

**Scenarios**

Programmability in Microsoft Dynamics AX is interesting when customizing, adding new features to the system or exposing business functionality through services – the most common scenarios will be described in this section.

**The Presentation Layer**

Users can access functionality in Microsoft Dynamics AX from various types of clients, like the Standard Windows® Client (Rich Client), the Web Browser based Client which requires nothing but a web browser or other types of clients like the Microsoft® Office Client (also known as Microsoft Office Snap Ins).

**Business Intelligence and Reporting**

Data can be extracted for reports using the native tool in Microsoft Dynamics AX, but more extensive possibilities for analyzing and displaying data graphically are available using external tools such as Microsoft SQL Server® Reporting Services or Microsoft SQL Server Analysis Services.

**Business Logic and Data Base**

Developers design the database from MorphX®, the native development environment of Microsoft Dynamics AX regardless of whether the database backend is Microsoft SQL Server or Oracle. Business logic is typically written in X++ the native object oriented programming language of Microsoft Dynamics AX however it is also possible to create business logic as a .NET assembly or a web service and call it from X++. This means developers are not confined to only using X++ for creating or customizing applications. The .NET support provided in X++ allows the programmer to leverage any .NET code stored in an assembly. Finally Web Services (implemented in any language by any vendor) can be called from Microsoft Dynamics AX.
Web Services

Data and functionality in Microsoft Dynamics AX can be accessed from other applications using Web Services. In this way service operations can be made available across applications including selected parties outside the companies (for instance customers could access price list and place orders in the system). The Application Integration Framework inside Microsoft Dynamics AX contains a wizard that quickly publishes data and classes as Web Services without having to write any code. It is also possible to consume Web Services from inside Microsoft Dynamics AX as mentioned in the previous paragraph.

Workflow

Business processes can be modeled as workflows inside Microsoft Dynamics AX – this feature was enabled in Microsoft Dynamics 2009 by embedding Windows Workflow Foundation. From a programmability point of view one of the advantages is that developers only need to design the “building blocks” of the workflow. The actual configuration of the workflows – assigning users to different steps of the process and setting up the rules for when a step should be executed or by passed can be done by administrators/or super users without developer skills.

Future Directions of Programmability in Microsoft Dynamics AX

This section of the whitepaper discusses some points regarding the future directions of programmability related features in Microsoft Dynamics AX, including the future of the programming language and integration with Visual Studio®.
Programmability Scenarios

Customization

Microsoft Dynamics AX comes with a rich set of features out of the box. However, in most cases, customization of the standard application is required in order to support the specific business process needs of the company using the system. Customizations usually include changes in the business database design as well as changes in the existing business logic. Microsoft Dynamics AX comes with full accessibility to application source code and database design. Using the native development environment and programming language of Microsoft Dynamics AX, the business functionality can be changed.

Add On Functionality

In this scenario, rather than changing the existing set of functionality, new functionality is added to the solution. This could either be increasing the horizontal functionality or adding industry-specific features to the solution. Regardless of what type of functionality needs to be added, the approach is usually handled in one of two different ways.

The add-on can be developed inside the native development environment and programming language of Microsoft Dynamics AX (similar to the customization scenario described above). Some of the advantages of this approach are the solution only needs to be maintained from one place and the user experience will be similar to working with the rest of Microsoft Dynamics AX.

The other approach involves the integration features of Microsoft Dynamics AX more so than programmability. With this approach, the add-on is located outside of Microsoft Dynamics AX. The add-on is usually a .NET application. Integration is then made between the add-on and Microsoft Dynamics AX. The advantages of this approach are if the add-on already exists as a .NET application (perhaps also used together with other ERP systems), it doesn’t need to be rewritten to Microsoft Dynamics AX – hence the developers of the add-on do not need to learn how to use the development environment or programming language of Microsoft Dynamics AX. The only thing that needs to be developed additionally is the interface between the add-on and Microsoft Dynamics AX.

Exposing business functionality through Services

Any business functionality in Microsoft Dynamics AX can be exposed as a Service using a special framework (Application Integration Framework) that quickly can enable desired functionality as Web Services regardless if the functionality is out of the box, added, or customized. This way Microsoft Dynamics AX enables well for SOA, any part of the application can be exposed as a service, interconnected and composed with other applications.
**The Presentation Layer**

Rich Client

Changing or creating new UI or other types of application objects such as forms, menus and reports is done from the native development environment called MorphX.

Inside the AOT (Application Object Tree, see Figure 1), it is possible to customize or create new objects.

![The AOT](Image)

**Figure 1, The AOT**

One of the powerful features of MorphX is being able to create application objects simply by dragging and dropping items in windows containing the model of a given application object such as a form. Figure 2 displays how a developer in a short amount of time can add data fields to the design of the form by dragging them from the data source to the design of the form.
Figure 3 shows the form at runtime – notice that the developer didn’t have to specify the location of the fields or the layout of the form. This feature is called IntelliMorph. One of the advantages of this feature is saving design time. Developers only have to concentrate on the structure of the form and not the layout. IntelliMorph handles the layout automatically at runtime.

Another benefit of this feature is that if certain fields/functions should not be available to a particular user due to the security setup, IntelliMorph will layout the form/menu/report without those fields/features in a consistent way. There will be no empty spaces where the fields or buttons were supposed to be and the UI will always have a consistent look.
Extended data types are one of the features that make MorphX extremely powerful when customizations are required. With extended data types several features in the system can be altered at once by changing a single property - for example the length of a field. In most other ERP systems changing the length of a field that is used in multiple tables and uses variables for storing values in the business logic several times is a task that can take one or more man weeks to perform. This task can include changing the database design, changing variable definitions in the business logic and the user interface. In Microsoft Dynamics AX this can be done in a matter of minutes and only requires the changing of a single property.

When customizing Microsoft Dynamics AX as described in the Programmability Scenarios above MorphX is typically the development environment.

Role Centers
The new Rich Client and Web Browser Client in Microsoft Dynamics AX contain a new component called a Role Center Figure 4. Role Centers are designed individually for different types of users. They help employees organize, prioritize and access tasks and information from a single window. This could be an information worker who needs quick access to specific data in the system related to their working area such as sales orders. It could be an executive needing to monitor the overall health of the business, in which case the Role Center would likely contain key performance indicators and score cards that graphically indicate how well the business is performing.

Unlike the rest of the Rich Client, the Role Center is generated from SharePoint® and can contain Web Parts from Microsoft Dynamics AX as well as from many other sources. The
technology used for designing Roles Centers is the same as is used when designing Web Part Pages for the Web Browser Client which is described in the next paragraph.

![Figure 4 Role Center on Rich Client](image)

**Web Browser Client**

Like the Rich Client described above the Web Browser Client (also known as the Enterprise Portal) is fully based on Windows SharePoint Services or Microsoft Office SharePoint Server. This means that users will only require a web browser to access features and data inside Microsoft Dynamics AX. Regardless of which client is used the business logic is the same (meaning that the only thing that needs to be maintained individually is application artifacts used to present the data per client type). Hence, there will only be one set of business logic and data base to maintain.

Designing Web Pages that can be used in connection with the Enterprise Portal or Role Centers containing features/data from Microsoft Dynamics AX requires two steps. The first step is to create the necessary Web Parts. The second step is to create the Web Pages if they do not exist already. Creating web parts can be accomplished in two different ways. One way would be to use MorphX Figure 5. This is similar to designing other types of application objects.
The other option when it comes to designing Web Parts is introduced in Microsoft Dynamics AX 2009 using Visual Studio® and ASP.NET Figure 6. With this option, developers can benefit from Visual Studio design time experience. Visual Studio might be more familiar to most developers with little or no experience with Microsoft Dynamics AX. This option also provides a very rich toolset for designing Web Parts.
Once the Web Parts have been created (regardless of which option was chosen to create them), Web Pages can be designed in SharePoint Figure 7. These Web Pages could contain Microsoft Dynamics AX 2009 Web Parts potentially combined with Web Parts from various other systems.

![Figure 7, Designing a Web Part Page](image)

After designing the page, the Web Part Page can either be included in the menu system of the Enterprise Portal or it can be applied as a Role Center available from the Enterprise Portal or the Rich Client.

Microsoft Dynamics AX 2009 utilizes a technology within this area called AJAX (Asynchronous JavaScript + XML). AJAX enables richer, more interactive and responsive applications compared to previous versions of Microsoft Dynamics AX.
Other Client Types

Recent analysis revealed that in average, only 15% of the total number of the employees of a company has direct access to the company’s ERP system – this means others ways of exchanging data to various groups of the remaining employees are required.

In many cases the remaining employees only require access to a limited subset of the functionality of the ERP system and the training required to use the ERP system for a rather small purpose would be too expensive. However the employees might be experienced in other applications. In other cases employees might have to perform redundant tasks by having to enter data into multiple systems.

A large array of .NET based applications that can be customized in a managed language can become a front end for Microsoft Dynamics AX. The Microsoft Office Snap Ins are examples of how this can be done and include features like extracting business data into Microsoft Office Word / Excel and creating expense transactions in Microsoft Dynamics AX from Microsoft Office Excel.

Integrations like those mentioned above have been created (and can be developed even further) in Visual Studio, using Microsoft Dynamics AX business functionalities enabled as Web Services. In most cases this is the recommended option when creating new alternative clients or front ends is to enable the features / data required as Web Services as a described in the section dealing with services.

When creating add ons for Microsoft Dynamics AX as described in the Programmability Scenarios section both the Web Browser Clients and Clients described could be used for the add ons already developed as .NET applications.
Business Intelligence and Reporting

Microsoft SQL Server Reporting Services Reports

Interoperability with Microsoft SQL Server Reporting Services makes it possible to design Reporting Services Reports based on data sources from Microsoft Dynamics AX. In this way reports can be generated leveraging all the Reporting Services features available such as graphics and the ability to consume Web Services. The reports can be executed from multiple places including the Microsoft Dynamics AX Standard Windows Client (just like reports developed in MorphX). The reports are generated in Visual Studio, see Figure 8, where a query from Microsoft Dynamics AX is accessed as an ADO.NET Data Set and works as the data source for the reports. After the reports have been graphically designed within Visual Studio, the reports can be added to the Application Object Tree in Microsoft Dynamics AX.

Figure 8, Designing the Report in Visual Studio
Analysis Services

Interoperability with Microsoft SQL Server Analysis Services makes it possible to extract data from Microsoft Dynamics AX into OLAP cubes that can be analyzed using various tools such as Pivot tables in Microsoft Excel.

The OLAP cubes are designed in Visual Studio displayed in Figure 9. Relationships can be changed, unwanted fields can be removed or calculated members can be defined.

![Figure 9. Designing an OLAP Cube in Visual Studio](image)

When this step is completed, the cube can be processed and the content can be analyzed from inside an analysis tool such as Microsoft Office Excel.
Native Reporting Tool

Microsoft Dynamics AX also contains its own native reporting tool. Reports can either be created using the Report Wizard Figure 10 and modified in the Application Object Tree or they can be created from scratch in the Application Object Tree.

Figure 10, The Report Wizard
Creating reports from scratch in the Application Object Tree Figure 11 is quite similar to designing a form as described in the previous section. Fields represented in the report are simply dragged and dropped from the data source to the design of the report.

![Designing a Report in Application Object Tree](image)

Figure 11, Designing a Report in Application Object Tree

In most reports, the layout is handled automatic using the IntelliMorph feature however it is also possible to create reports with manually designed layouts.

In relation to the Programmability Scenarios, the native reporting tool is normally used when customizing existing reports made with the tool.
Business Logic and Data Base

MorphX

An introduction to MorphX has already been made during the previous chapters on designing forms, web parts and reports. All application artifacts including data base design are accessible in the AOT (Application Object Tree) displayed in Figure 12. The philosophy behind MorphX is that application objects can be designed quickly by dragging and dropping items between windows and entering properties using property sheets as displayed in Figure 13.

The architecture of Microsoft Dynamics AX is object oriented, meaning properties can be inherited in various object hierarchies such as Classes (containing business logic) and Extended Data Types (used for defining variables and data fields). One of the advantages of Microsoft Dynamics AX over other ERP systems is illustrated when the length of a field used in multiple tables needs to be enhanced. An operation like this can take a number or weeks to handle in other ERP systems, but because of the architecture of Microsoft Dynamics AX this can be done by changing one single property in a matter of minutes.

Figure 12, The Application Object Tree

Figure 13, The Property Sheet
When creating or modifying the existing application objects are saved in a specific application layer (Microsoft Dynamics AX contains 16 such layers). Each of the layers is assigned to different groups of developers (Microsoft developers, independent software vendors, Microsoft business partners or the customer’s own developers). The advantage of this structure is to ease the upgrade process. In addition, all new/customized functionality is stored in separate places isolated from the original application.

Another powerful feature of MorphX is IntelliMorph, as mentioned in the section on the Rich Client. This feature enables automatic layout of user interface application objects such as menus, forms and reports. A number of advantages are provided by this feature. From a developer’s point of view, the advantage is that the developer designing user interface objects only has to focus on what data fields and other controls should be present and not the layout. IntelliMorph handles the layout automatically. It is however also possible (but not recommended) to suspend this feature if the developer wants to do the design manually.

**X++**

The business logic of Microsoft Dynamics AX is written in X++. An example of the code syntax is displayed in Figure 14. X++ is object oriented and in many ways similar to other existing object oriented programming languages (such as C#). It is unique for Microsoft Dynamics AX and was created for one single purpose – to create business applications. Various features such as memory management are therefore handled automatically.

![Figure 14, The X++ Editor](image)

Most of the business logic is stored in Classes. Classes can be inherited in hierarchies of Subclasses and called from different applications objects such as methods on tables, forms or even external .NET applications.
In order to increase performance reducing the calls between the Client and Server, it is possible to define X++ Classes by setting one single property to specify if the code should be executed on the Client or the Server.

X++ has many other powerful features such as the possibility to refer to database tables and fields directly in the code and using a number of SQL commands. X++ is self referential and powerful enough to handle some of the technical features in Microsoft Dynamics AX. X++ for example could be used to in making new development tools - like the Report Wizard or the Unit Test Frame Work.

In relation to the Programmability Scenario section, MorphX and X++ can be used for customizing as well as creating add ons for Microsoft Dynamics AX - however if the add on has already been created as an existing .NET application it might be advantageous to use one of the integration possibilities provided in Microsoft Dynamics AX rather than having to rewrite the application in X++.

CLR-Interop

In some cases, the ability to integrate with .NET applications is necessary. This could be because an existing .NET application has features that would be beneficial to utilize in Microsoft Dynamics AX rather than having to port it in X++ or it could be that developers have a preference for using a specific desired .NET language rather than having to learn how to program in X++.

The possibility to make use of .NET assemblies was introduced in Microsoft Dynamics AX 4.0. MorphX contains a node called References as displayed in Figure 16. After making a reference to a .NET assembly, the reference can be referred to directly from X++. This feature even supports IntelliSense and checks for correct data type for input and output.
Similar to making references to .NET assemblies as described above, it is also possible to consume Web Services from various other systems. In previous versions of Microsoft Dynamics AX it was a manual process to make a reference to a Web Service, in Microsoft Dynamics AX 2009 the references can be created in the Application Object Tree shown in Figure 16, the Web Services can be called directly from X++.

Service References enables Software-plus-Services for Microsoft Dynamics AX. It provides the option to combine the features within Microsoft Dynamics AX with external Services such as Windows Azure® deployed Services and hence possibilities to create solutions that provide new capabilities and offer new levels of utility, convenience and flexibility.

Both CLR Interop and the ability to consume Web Services are very useful in relation to creating add-ons described in the Programmability Scenarios.
Web Services and the .Net Business Connector

AIF (Application Integration Framework) Enabling Features/Data as Web Services

The Application Integration Framework serves a number of purposes when it comes to exchanging data between Microsoft Dynamics AX and other applications internally or externally.

The focus on AIF in this document is mainly on Web Services, data and features from Microsoft Dynamics AX which can be enabled using 2 different approaches.

If a Web Service is needed to create/read/update/delete data in a table within Microsoft Dynamics AX, the AIF Document Service Wizard Figure 17 is an easy solution. This Wizard will create the X++ Classes necessary for the Web Service to work (those Classes can be then modified following if additional functionality is required).

Figure 17, The AIF Document Service Wizard

The second approach is to write the X++ Classes from scratch. This approach is of course the one that requires the largest amount of effort.
.NET Business Connector

The .NET Business Connector is an interface that makes it possible for .NET application to access data and features inside Microsoft Dynamics AX. The .NET Business Connector can be considered as a “faceless client” meaning that everything accessible on the Windows Client is also available here except from the user interface standpoint. Menus and forms cannot be accessed with this interface.

The .NET Business Connector is used for multiple purposes in the standard application such as integration with Internet Information Server and Visual Studio.

In relation to the Programmability Scenarios, Web Services enablement is usually recommended to connected external applications like add ons with Microsoft Dynamics AX.
Workflow

For modeling business processes inside Microsoft Dynamics AX, Microsoft Dynamics AX 2009 has divided this task into a Developer Role and a Super User / Administrator Role in order to reduce the costs connected with this task.

Workflows are divided into different categories (usually based on the modules in Microsoft Dynamics AX). Figure 18 illustrates the developer experience. The “building blocks” of Workflows are called Tasks and Approvals. They are strung together in Workflow Templates. Creating these components and writing the X++ code that should be attached and executed with them is a job for a developer.

When the developer has finished the Workflow design, the Workflow can be configured (assigning users to the different processes, setting conditions, escalation rules, etc.) and activated. This part does not require any coding skills and is usually performed by a super user or systems administrator.
Future Directions of Programmability in Microsoft Dynamics AX

This document has mainly addressed current programmability features in Microsoft Dynamics AX 2009. In this section some of the future directions surrounding programmability will be highlighted. From an overall point of view, it is safe to say that Microsoft Dynamics AX will move towards .NET in several ways.

Will X++ become a managed language?

No, however Microsoft will continue to increase the Interoperability between X++ and managed languages. As described previously in this document, we already have CLR-Interop and the possibility to call .NET assemblies from inside X++. One of the future plans for this feature is to make it possible to use the .NET assembly to call back into Microsoft Dynamics AX without having to use the .NET Business Connector Client for this purpose.

Currently, there are no plans to replace X++ with a managed language.

What will happen to MorphX and the way to design application objects in the Future?

There is currently no end date for MorphX, but in the future versions of Microsoft Dynamics AX an increasing amount of application object types are expected to be designed from Visual Studio instead of from MorphX.

How will Business Intelligence and Reporting develop?

Microsoft will continue to increase the number of integration points with Microsoft BI and Reporting software. In addition Microsoft will also provide additional APIs that increases the ease of performing BI related functions from inside Microsoft Dynamics AX.

What will happen with Microsoft Dynamics AX in relation to Service Strategy?

Microsoft Dynamics AX supports consuming standards-based, external web services from within the application. Future releases of Microsoft Dynamics AX will support additional standards to enable additional scenarios. The related programming model will also be extended to further improve the development experience in particular for common scenarios. Moreover, Microsoft Dynamics AX can be exposed through Microsoft Dynamics AX services. Microsoft Dynamics AX can be published e.g. through WCF web services. Additional guidance and mechanisms for standardizing common functionality such as customization and validation will help accelerating the implementation of integrations. Furthermore, future releases of Microsoft Dynamics AX will aim at further simplifying the creation and management of Microsoft Dynamics AX services. Enhancements of existing functionality, which may include better support for digital signatures etc. will aid in deploying Microsoft Dynamics AX in scenarios where compliance is a high priority. Setup and administration of integrations will also be further streamlined.
Conclusion

There are many possibilities available when it comes to customizing or adding new features to Microsoft Dynamics AX. Please refer to the References section below for deeper knowledge around using the tools or when to integrate solutions with Microsoft Dynamics AX or embed them into the product.

References

Developers Best Practices Handbook
Inside Microsoft Dynamics AX

http://www.microsoft.com/downloads
http://msdn.microsoft.com
http://blogs.msdn.com
https://mbs.microsoft.com/customersource

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2008 Microsoft Corporation. All rights reserved.