Enterprise Software Packaging Practices, Benefits and Strategic Advantages

Nelson Ruest

A Report by Wise Solutions, Inc.

Abstract

This white paper provides an overview of enterprise software packaging practices and procedures. Its intended audience is IT executives, system administrators and packaging technicians who belong to organizations that want to implement a structured approach to system management, and especially, software packaging in the enterprise. This paper presents a comprehensive look at the reasoning behind the enterprise packaging process and how it integrates into the overall enterprise systems management strategy organizations strive to put in place.

In addition, this paper outlines a selection of best practices for software packaging and general systems management. While it is not product specific, this paper is focused on the management of Windows-based systems. Its processes and some of its best practices can also be used in non-Windows environments. Managing distributed systems is a complex process that must be comprehensive. This paper provides a general discussion on the topic with strong specific recommendations in terms of gaining control over a distributed network within a corporate environment. It also paints a broad outline of how enterprise software packaging practices can help gain a rapid return on investment for most organizations.

About the Author

Nelson Ruest is an IT professional specializing in systems management and design. He is coauthor of "Preparing for .NET Enterprise Technologies — People, PCs and Processes Interacting in a .NET World", published by Addison Wesley, ISBN 0-201-73487-7.

He is currently working on a new book for Osborne McGraw-Hill entitled "Windows .NET Server 2003, Tips and Techniques for Enterprise Deployments" due to come out early next year, ISBN 0-07-222343-X.

Table of Contents

1. Enterprise Software Packaging

Software packaging is often one of the most overlooked processes in IT. Yet, because it is the process that controls software installations, it is one of the most important processes in any software management strategy.

Few people in IT today haven't performed an interactive software installation. The installation experience can range from good for an advanced user to very bad for a novice. The simplest installation will ask only a few basic questions but some of the more complex installations are enough to stymie even the most sophisticated PC technician.

The best installation is the automated installation — the one where no user input is required. And the best of these is the customized version of the automated installation — the version that is designed to work within the specifications of a corporate network.

Many enterprises have leveraged Windows 2000 and XP migration projects to begin migration to Microsoft's Windows Installer as a consistent method of packaging and in an attempt to gain the promised control and consistency across applications. What has become clear is that enterprises are recognizing that this effort requires dedicated expertise. Enterprises that do not dedicate resources to packaging applications will fail at software distributions through 2004 (0.8 probability).
– Gartner Group

In comes Enterprise Software Packaging (ESP). Enterprise Software Packaging deals with the preparation of standard, structured automated installations for deployment within a specific corporate environment. These automated installations or "packages" must take into consideration all of the installation requirements for the corporation: corporate standards for software usage and desktop design, multiple languages, regional issues, and especially, software-related support issues. In addition, packages must be prepared for both commercial software as well as corporate applications.

Most commercial software products include the ability to automate their installation. Unfortunately, there are no official standards in the plethora of Windows-based commercial software products for installation automation. Some ad hoc methods exist, but they are rarely satisfactory. This means that if an organization undertakes managing their ESP processes using the generic automated installation tool included with the software product, it will need to support "pockets" of knowledge; i.e., groups of technicians that are specialized in one or another installation automation method.

Fortunately, this state of affairs is changing. Microsoft® has introduced a standard installation and software management tool for Windows systems: the Windows Installer Service. This service is available for Microsoft's own as well as other commercial software products. It is also available for the preparation of automated installations for corporate applications that are developed in-house. This means that it is now possible to aim for a standard, consistent installation methodology for all software products within an organization. This standard approach is at the heart of Enterprise Software Packaging.

1.1 Software Management Issues

Organizations today have embraced the distributed computing model because of its inherent benefits to the end user. It's a simple principle: with a processor, a hard disk drive and memory, each PC is a powerful tool with local processing capabilities that have little or no impact on any other networked component. But as organizations grow and add more and more of these "independent" components, they soon discover that managing distributed environments is not without its issues.

For one thing, enterprises, whether small, medium or large, cannot afford to have personnel go around from computer to computer to install software. In the same vein, organizations cannot expect their users to have the knowledge required to install software properly on their own. Software installations in the enterprise must be automated, controlled from a central source and must support updates as well as removal or uninstallation.

This is the reason for the emergence of software packaging tools. These tools are designed to assist IT personnel in the preparation of automated, interaction-free software installations. There are several models of software packaging tools

on the market, but each and every one offers the same basic feature set: allow IT personnel to perform a software installation and configuration, capture this customized installation and ideally, reproduce it successfully on every desktop or server in the enterprise.

But software packaging must be structured if it is to succeed within the enterprise. Standards must be set and maintained. Software packages must be documented and detailed in the same manner no matter who performs the packaging process. Quality assurance must be high; installations must work in every instance even if the configuration of destination computers varies. Software package repositories must be maintained and updated according to organizational software acquisition policies. All of these activities are part and parcel of the Enterprise Software Packaging process.

In addition, organizations are faced with centralization versus decentralization issues. Should the software packaging process be centralized? If so, how can central IT personnel know and understand regional issues? Should it be decentralized? If so, how can the organization maintain standards and avoid duplication of efforts?

These are some of the questions organizations must ask themselves when they prepare an Enterprise Software Packaging policy. These are some of the questions this paper tries to address.

1.1.1 Software Installations

Software installations are a well-known source of frustration in systems management. This is partly due to the success of Windows. There are literally hundreds of thousands of software applications designed for the Windows operating system. Each one has its own particularities when it comes to installation and configuration. Though most come with some form of automated installation, few are standardized.

Microsoft and other organizations have been challenged to develop distributed infrastructures that can support controllable centralized management. Software installations often cause either conflicts with existing software products on a system or even conflicts with the operating system itself. In fact, the instability of software installations has led organizations and individuals to mistrust the Windows operating system. Few are the users who have never experienced the "blue screen of death" — the blue screen Windows presents when it has a major malfunction. Yet in a corporate network, there must be a way to ensure that systems will function 100 percent of the time because even in distributed systems, downtime is unacceptable.

1.1.2 Reducing Diversity

Most organizations from small to large require some form of automated software installation. Though there are thousands of programs on the market for Windows operating systems, medium to large organizations only use from 200 to 500 within their corporate network. Nevertheless, managing software installations for several hundred products can be complex, especially if the avoidance of conflicts is an issue.

Fortunately, there are methods that can be used to reduce diversity and thus reduce the possibility of conflict. For one thing, few organizations will need to install 200 to 500 products on the same computer. The first step relies on the use of a model for system design. One such model, the Point of Access for Secure Services, or PASS model , is presented in Figure 1-1.

Software Rationalization

Organizations that have not achieved standardization within their corporate networks may have more than 500 software products in use. It is important for organizations today to perform a software rationalization exercise – an undertaking that aims to reduce the number of software products in use by limiting the number of versions or the number of products performing the same function in the network. If this rationalization exercise has been performed, then medium to large organizations can expect to have on average about 300 or less products in use.

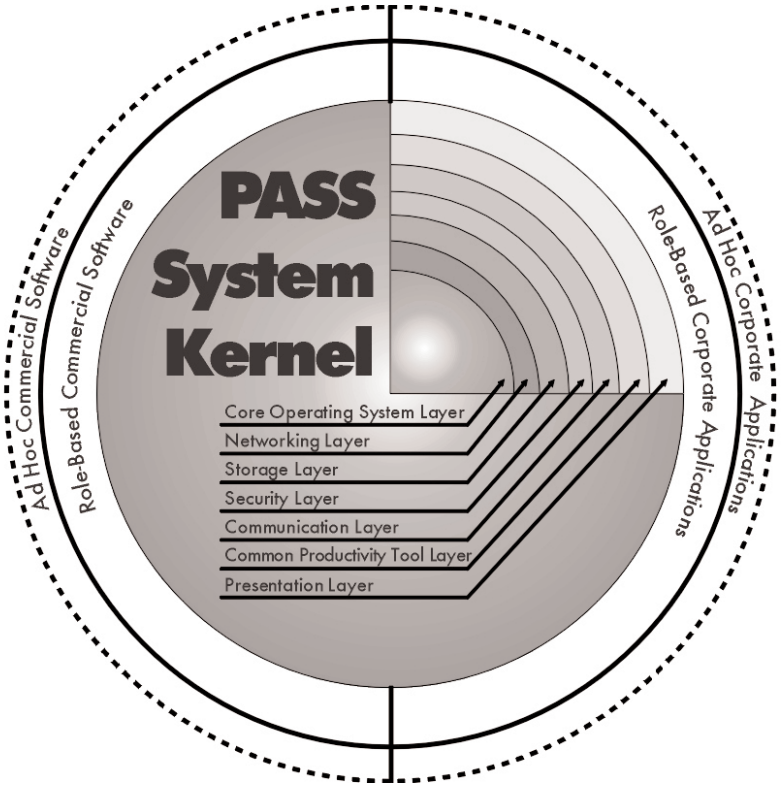


Figure 1.1: The PASS model

This model is based on the construction of a computer system that responds to corporate needs in three ways:

- The **PASS system "kernel"** is designed to meet the needs of the average corporate user. It contains all of the software components required to perform basic office automation and collaboration tasks. In addition, it is divided into a series of layers similar to the OSI Networking Model. In fact like the OSI Model, it uses seven layers to provide core corporate services. Because its functionalities are required by all personnel, this kernel is installed on all computer systems.

- **Role-based applications and software** are added on top of the kernel to meet the requirements of special IT roles everyone plays within the organization. Applications are in-house programs that meet mission-critical or mission-support functions. Software is identified as commercial software products that are not required by all personnel, but only for specific roles or tasks.

- Finally, an **ad-hoc layer** responds to highly specialized IT requirements that are often expressed on an individual basis. This ad-hoc layer can be applied to either software or applications.

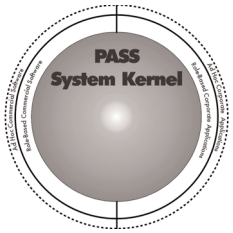
Constructing systems based on a model such as the PASS model greatly reduces system management efforts because it reduces the number of programs that must coexist on any system. First, a good portion of systems, sometimes up to 50 percent, will only require the system kernel. Remember that the kernel should contain every single program that is either royalty-free and required by the entire organization (for example, Adobe® System's Acrobat® Reader® or the Microsoft Reader) or every program for which the organization obtains an enterprise-wide license (for example, many organizations obtain an enterprise license of Microsoft Office).

[1]The PASS Model was first introduced as the SPA Object Model in "Preparing for .NET Enterprise Technologies" by Ruest and Ruest.

Second, by grouping programs into role-based configurations, organizations are able to reduce the number of programs that must coexist on a system. Role-based configurations include every program that is required by every member of the IT role grouping. For example, Web editors would require a Web editing tool, a graphics tool, a Web-based animation tool, and other Web-specific utilities. This group of tools can be packaged separately, but should be delivered as a single unit on all systems belonging to the IT role. Role-based configurations often include no more than 10 to 30 individual programs depending on the role. Only these groupings need to be verified with each other and against the contents of the system kernel. There is no requirement to verify or test the cohabitation of programs contained in different configurations because they are not likely to coexist on the same system.

Third, ad-hoc programs reduce system management efforts even further because they are only required by very few users in the organization. They are still packaged to enable centralized distribution and automated installation, but once again, they only require testing against both the kernel and the configurations they will coexist with but, because of their ad hoc nature, they may coexist with all possible configurations.

The PASS model is designed to support both the conception of user PCs and corporate servers. Each should include a core system kernel, role-based applications and software and ad-hoc components. The difference lies in that for PCs, role-based configurations are focused on user roles, whereas for servers, these configurations are focused on the server role within the corporate network. This model is quite useful, especially given the massive Windows migrations and system deployments most organizations must undertake on a regular basis due to the evolution of Windows operating systems.



The PASS model
In the PASS model, the kernel is considered as a closed component that is reproduced on all systems. Layers that are located beyond the kernel are considered optional on all systems

1.2 The Software Management Lifecycle

Another important aspect of software and application management is lifecycle management. Each software product has a lifecycle of its own that is independent of its location within the system construction model. The difference lies in the rate of application of lifecycle activities. All the components of the kernel will have an accelerated lifecycle rate (because they are located on all systems, they tend to evolve at a faster pace than other components because they are supported by corporate-wide funding) while products within the outer layers of the model will have slower lifecycle rates.

Software lifecycle management is one of the most challenging processes in the enterprise. Few organizations today know offhand what software can be found in their network. Fewer still can guarantee that there are no unused software products on their users' PCs. The issue stems from the very nature of distributed systems, but it can be circumvented with the application of a system construction model like the one mentioned above and the introduction of a software lifecycle management process.

This lifecycle is based on four major phases and can be applied to both commercial software products and corporate applications, though there are slight variations in the initial phases. The four phases include:

- **Software Evaluation or Application Preparation** — This involves the identification of the requirement followed by the selection of a commercial software product and/or the design of a corporate application.
- **Software Implementation** — This phase focuses on software packaging, quality assurance testing and deployment.
- **Maintenance** — This phase focuses on ongoing support activities for the product. It will involve the preparation, testing and distribution of scheduled updates.
- **Retirement** — The final phase is focused on removal of the product from the corporate network due to obsolescence. This removal may be followed by a replacement and thus initiates the lifecycle process once again.

All software has a lifecycle. It begins the moment the software development project is initiated by a manufacturer until the moment the software is retired from the marketplace. For user organizations, the lifecycle focuses more on when it is acquired, when it is deployed, how it is maintained and supported and when it is retired from the network. In the case

of corporate applications, it begins the moment corporate developers begin to work on the project until the product is retired from use. Figure 1-2 illustrates the Corporate Software Management Lifecycle.

During its lifecycle, most software will require corrections. Manufacturers often call these "service packs" or "service releases" while corporations call them "updates." If an organization adopts a software product before these corrections are released, it will have to deploy them in addition to the deployment of the original product during that product's lifecycle.

Thus organizations must arm themselves with comprehensive software maintenance processes and toolkits. Once again, Enterprise Software Packaging plays a key role within the software maintenance phase of the lifecycle because it includes the ability to apply patches or corrections to software products or corporate applications. But probably its most useful aspect is the full support of software retirement.

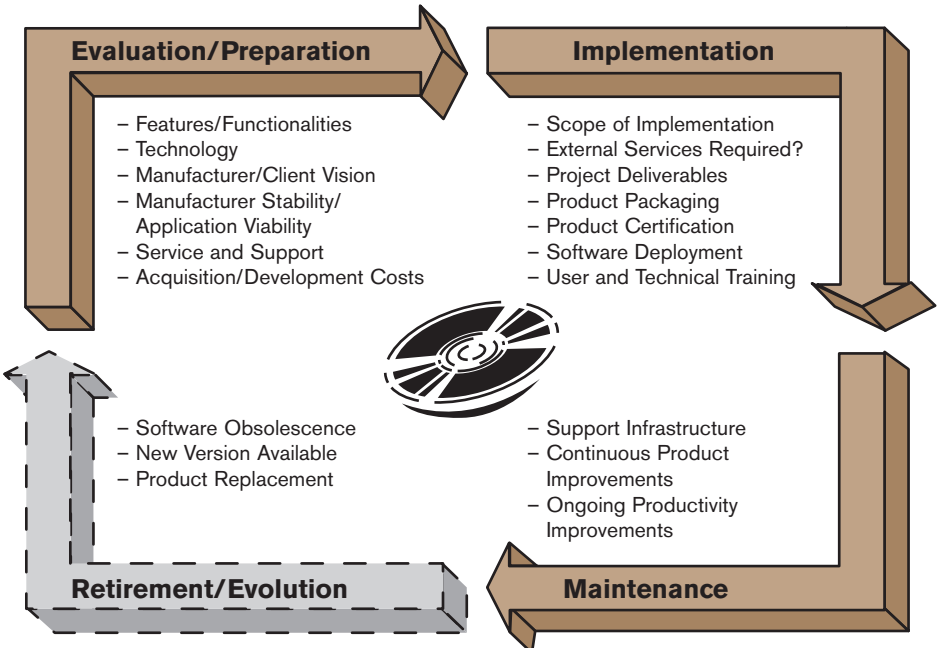


Figure 1.2 — The Corporate Software Management Lifecycle

1.2.1 Managing Software Licenses

One of the most important aspects of the software lifecycle management process is the management of software licenses. Every organization must be legally compliant in terms of the software licenses it has in use within the corporate network. But in actual fact, very few are completely compliant. Why? Because it is difficult to track software attributions and few take the time to implement complete software tracking systems. It is also because few organizations take advantage of the software removal features of the software installation process.

Software removal is probably the most unused portion of the entire software lifecycle, yet it is the most important if organizations want to maintain a legal and compliant network. The issue is simple. When users move from position A to position B, they change their role within the organization. With new roles come new tasks, IT groups are quite adept at making sure users' PCs are updated with the software required to meet their new requirements because if they don't users will be quick to log a support call. However, they are not quite so adept when it comes to removing unused software from the same PC.

Software removal is complex and cumbersome. The problem with software removal is that it is seldom effective. Modern software is made up of a series of private and shared components. Removing any of these components can affect the stability of the system. Thus IT makes the decision to opt for stability at the expense of legal compliance. After all, systems undergo regular hardware maintenance at which time they are reinstalled anyway. If the system isn't compliant, it is only for a short period of time.

Software removal either requires an expert in the field or a method that will guarantee proper and non-damaging removal. So, to be safe and guarantee the PC will remain stable, many organizations simply don't remove unused software products. This is not a valid strategy in the enterprise.
This is another aspect of Enterprise Software Packaging. Proper packaging methods will involve complete package testing and quality assurance including proper and non-damaging removal of packaged applications.

1.2.2 Maintaining Corporate Inventories

Another key aspect of the software management lifecycle process is the maintenance and upkeep of corporate inventories. This is an area where a model for system construction like the PASS model can also play a role because of the way it is designed to work. Maintaining an inventory need really only focus on identifying role groupings for role-based configurations because the kernel is available on all systems and ad-hoc products are only located on a few machines. These "vocational" groupings become the mainstay of the inventory system, but they are not its only focus. Corporate inventories for software product management should include all of the elements illustrated in Figure 1-3.

These elements include:

- **Package Repository** — A central deposit for all authorized software for the corporate network. This repository which is actually constructed by the Enterprise Software Packaging process is the source for all software product distribution within the organization.
- **System Kernel Inventory** — The system kernel must be completely inventoried. This inventory will be linked to the Package Repository because many of its original components will be stored in packaged format to facilitate automated system construction and conflict resolution.
- **Role-based Configuration Deposit** — This deposit identifies each of the packages found within each configuration. It is tied to the Package Repository to support system construction and vocational changes.
- **Vocational Groupings** — This deposit regroups all of the users or servers belonging to a given IT role. It is tied to the Role-based Configuration Deposit because it identifies which configurations systems require in addition to the kernel. In many cases these groupings can be found within an organization's Enterprise Directory such as Active Directory since it is mainly constructed of user and machine groups.
- **Core Inventory Database** — The core inventory database brings it all together. It includes inventories of all computer systems in the network, all users, all software components and much more. It is used to validate that systems contain only authorized components. It is also used to assign ad hoc product installations as these are mostly performed on a case-by-case basis. This database is maintained by the organization's Enterprise Systems Management tool.
- **Web-based Report System** — Another function of the Enterprise Systems Management tool is to provide detailed information on the inventories that have been collected as well as provide consistency reports on all systems.

Locked Systems

For processes based on system construction and inventory management to work, organizations must lock computer systems and ensure that software installations originate only from centralized and authorized sources.

Building the System Kernel

For easier system reproduction, the System Kernel is often installed on PCs and servers as a single disk image. But the original source or the staging computer for the construction of this disk image must be built component by component.

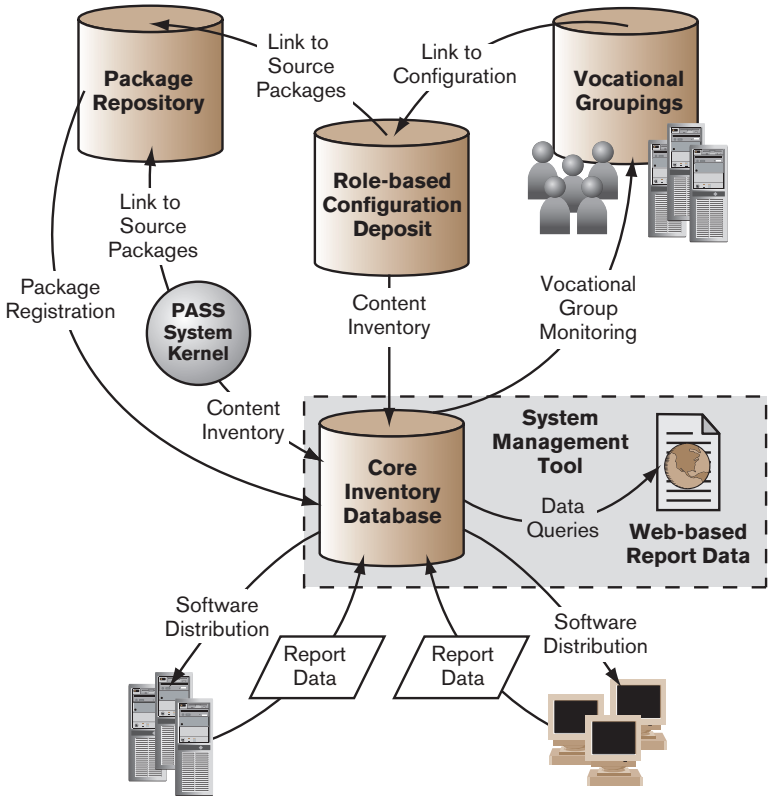


Figure 1.3: A Corporate Inventory System

No software distribution tool will be able to solve the all-too-common problem of desktop heterogeneity… This is done prior to deploying the application (or update) in a test lab. Without this critical step, software distribution tools will simply be an automation of a bad process.

– Gartner Group

System staging and software distribution can be performed on the basis of these inventories. In addition, when systems change vocations (i.e., a user or server changes IT roles), the inventories are used to remove software products that are no longer required and add software products required by the new role-based configuration. This means that there must be a very tight relationship between the Inventory Process and the Software Distribution Tools and Mechanisms.

1.2.3 Software Distribution Tools and Mechanisms

One of the core features of an Enterprise System Management tool is the ability to deliver software to all systems in the corporate network. Once again, this aspect of systems management is highly related to Enterprise Software Packaging because the very first step in any software product delivery process is the creation of a standard, customized software installation that will provide consistent and repeatable installations on all target systems. In fact, the software delivery process should include the following steps:

1. New software packages are prepared and once ready are integrated into the Package Repository. This repository is the single listing of all authorized software. Ideally, all files within the repository are in the same installation format.
2. The software package is assigned to a group. This can be either users or computers. Most often, software will be assigned to computers because it is simpler to manage. Assigning software to users, especially in environments where users move from PC to PC, will constantly cause software installations and removals.
3. The package is assigned to a central package distribution share point.
4. The delivery schedule is set.
5. Software deployment is initiated.
6. The source of the installation is distributed to all installation share points.
7. The installation code is cached locally on the target system.
8. The software product is installed on the system from the cached version.

9. The software installation updates a local installation record file for Help Desk and software tracking purposes.
10. The software installation returns a completion code to the central software delivery system to validate that it has installed successfully.

This process is illustrated in Figure 1-4.

For ongoing software distribution management, especially according to the software management lifecycle, a link must be maintained between vocational groupings and the system distribution tool. Vocational groupings are best kept within enterprise directories because that is where all accounts are maintained. Since most systems management tools have the ability to collect this group information from directories such as Active Directory, a link between the two should be established and maintained.

This link can provide the contents of these global security groups and insert them into the systems management database. In most cases, these discovery methods are recurring; i.e., any changes to global groups performed in Active Directory will be reflected within the systems management database. Thus a dynamic link can be set between the vocational groups and the software distribution system. To manage software and maintain legal compliance, organizations need only move users, PCs or servers from one group to another.

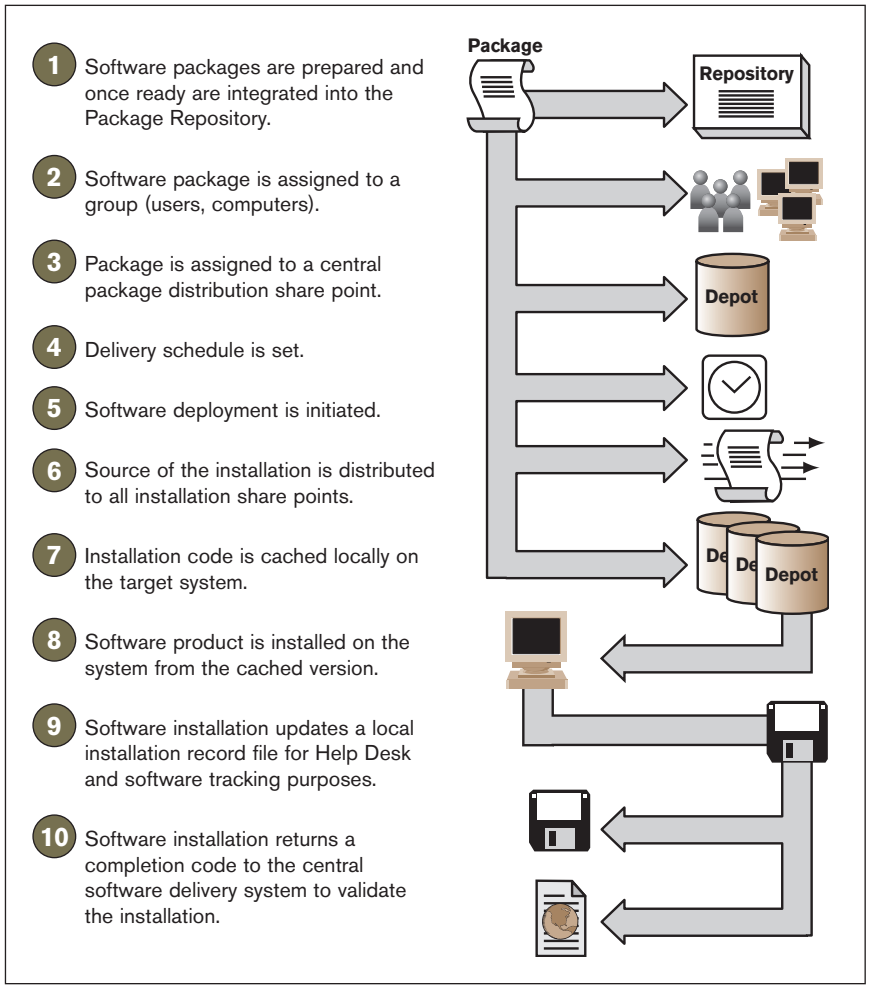


Figure 1.4: The Software Delivery Process

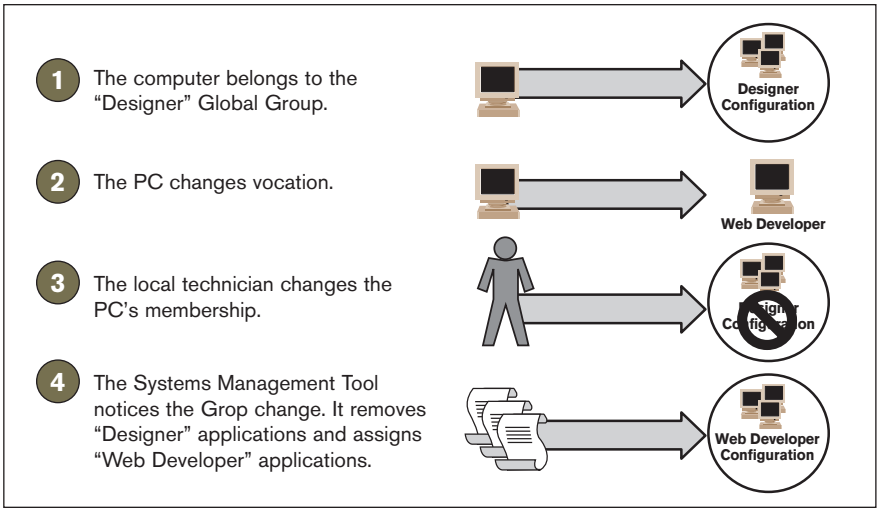


Figure 1.5: The Vocational Change Process

When a change is detected, the software distribution tool will automatically remove obsolete software from the target system and replace it with the new role-based configuration.

Two key elements are required for this process to work: the inclusion of **uninstallation instructions** within the software distribution tool and the use of software delivery packages that fully support **automated software removal**. Once again, Enterprise Software Packaging provides key support to this process. The vocational change process is illustrated in Figure 1-5.

1.3 Recent Microsoft Developments – The Windows Installer Service

All of the processes outlined above rely heavily on the Enterprise Software Packaging process. But for this process to work, a single standard installation and uninstallation structure must be provided to support the ESP strategy. Fortunately, Microsoft has taken great strides in this area in the past few years with the introduction of the Windows Installer service. This service is now native to all Windows operating systems since Windows 2000 and can be added to all previous Windows operating systems since Windows 95. It provides a single, Microsoft-supported standard for how all Windows applications should be installed.

The Windows Installer Service is designed to fully support software lifecycle management principles. In fact, it supports all aspects of this process from the implementation phase through maintenance and retirement as is illustrated in Figure 1-6. This service offers several enhancements over traditional software installation mechanisms. One of its greatest advantages is self-healing. Every software product that is installed through this service automatically can become self-healing. This means that should there be a corruption of one of its components, it will automatically repair itself the next time the product is launched.

During installation, the Windows Installer service copies every detail of an installation into a consistency database located on the destination computer. When a program is launched, the Windows Installer service double-checks program components, registry keys, configuration parameters and more against this database and if anything is amiss, automatically launches a repair cycle. Once the program is repaired, it continues its normal launch cycle and opens. The only thing the user sees is the Windows Installer service dialog box for a few seconds just before the program starts.

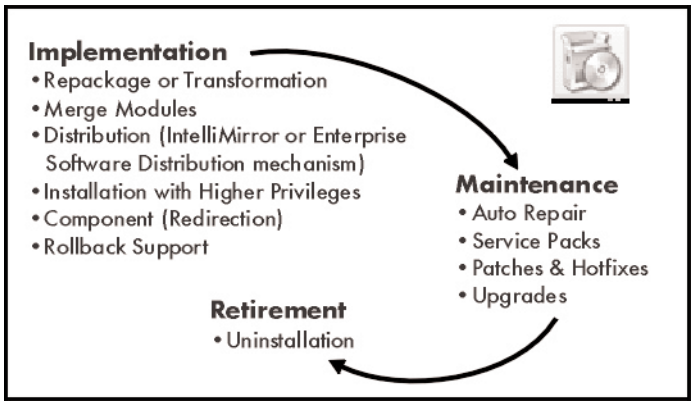


Figure 1.6: The Windows Installer Software Lifecycle

This consistency database also offers full support for software removal. Because Windows Installer always knows precisely which components belong to a program, it can safely remove them and restore a computer to a working state. And, because it is a service, Windows Installer can support software installations in locked environments, allowing normal users to profit from complete and problem-free corporate installations.

Windows Installer offers a series of additional advantages: installation rollback in case of problems, special support for program coexistence, fully unattended installations, integration of user parameters when new users activate a program, on-demand installation of program features, program patching and program transformation. Through this service, IT can ensure that every application or commercial software product they deploy will provide stable installation and operation.

Software approved to display the Windows 2000 or Windows XP logo uses the Windows Installer service by default. In addition, some new programs released for these operating systems are also integrated with this service. This can, in fact, be a criterion for program selection. Unfortunately, this doesn't include every software application on the market. But IT can repackage programs to integrate the native installation with this service, converting the programs into Microsoft Installation (MSI) format. Several repackaging tools support this function. They basically allow IT organizations to capture the installation image of a given program and repackage it into Windows Installer format. Once this is done, the program will automatically benefit from several of Windows Installer's features such as auto repair and elevated user credentials for installation in locked environments.

In fact, the Windows Installer service is Microsoft's vision of the future for all installations within Windows networks. Organizations aiming to implement an Enterprise Software Packaging strategy would be well advised to aim for complete integration of all installation packages with this service. Windows Installer is at the core of the ESP strategy because the ESP relies on the principles inherent in the software lifecycle management process.

2. Software Packaging Requirements

The Enterprise Software Packaging process includes several procedures, all of which must be supported by core standards. As mentioned above, the first of these standards is the packaging target: Windows Installer. All packages should aim to support this service.

The second should be the categorization of software within the organization. Software should be categorized according to Windows Installer compliance. As mentioned earlier, this mostly includes software programs with the "Designed for Windows 2000 or XP" logo. For successful ESP, all other program installations must be migrated to this standard.

Most corporations will not be able to achieve this through upgrades for several reasons. First, some programs, especially internally developed programs may not be so easily upgraded. Second, the average corporation (more than 1000 users) has about 300 different software programs and applications within its network. Upgrading all of these products would be cost-prohibitive and often unnecessary. Third, some software products simply do not offer upgrades. Fourth, some manufacturers, unfortunately, still do not integrate their software products to the Windows Installer service.

In most cases, organizations will have to consider repackaging software installations in order to take advantage of the many features of the Windows Installer service. Several tools are available on the market for this repackaging process. One of the prerequisites for an enterprise solution is a tool that will provide the same functionality for both repackaging commercial software and packaging corporate applications that you develop in-house.

When categorizing corporate software for ESP, software assets fall into four basic categories:

- **Native Windows Installer Software** — This software includes any product that bears the Designed for Windows Logo. Part of the requirement for the logo program is integration with the Windows Installer service. Organizations should consider upgrading a portion of their network's software to this level. This should include the most popular software on the network such as the software found within the system kernel.
- **MSI-integrated Corporate Applications** — New or upgraded versions of corporate applications should be integrated to and designed to work with the Windows Installer service.
- **Repackaged Commercial Software** — All commercial software products that are not upgraded should be repackaged to integrate their installation with Windows Installer. In most organizations undertaking this repackaging process, 99% of software has been repackaged to take advantage of Windows Installer. Only products such as device drivers, programs which install device drivers or programs which make low-level changes to the operating system will resist Windows Installer integration.
- **Repackaged Corporate Applications** — Corporate applications that do not require recoding, upgrades or cannot be reprogrammed should be repackaged to be integrated with Windows Installer. This will most likely include corporate, regional and departmental applications.

Categorizing all of these programs takes time, but it is a necessary step of the ESP strategy. This step is related to the Corporate Inventory Process and results of this categorization should be documented and included in the inventory database.

2.1 Packaging Considerations

A third important standard is "customized consistency" — customizing or repackaging an installation so that it behaves in a manner consistent with internal corporate standards. Customized consistency also means adapting the packaging process to each of the four software categories mentioned above.

For example, programs that are designed for Windows are already integrated to Windows Installer and already include an MSI-format installation file. What is required for these programs is customization rather than repackaging. Programs that do not meet the Designed for Windows standard will, on the other hand, require repackaging to integrate

Designed for Windows Software Catalog

Microsoft maintains a list of all products that have achieved the Designed for Windows standard on a special Web site at www.microsoft.com/windows/catalog

Wise Repackaging Products

Wise Solutions Inc. offers a repackaging tool for Windows Installer: Wise Package Studio. Full support for the ESP Process is found in the Enterprise edition of Package Studio.

their installation to the Windows Installer service. And corporate applications will need to be developed to meet the requirements of an application integrated to the Windows Installer service.

Special considerations need to be taken into account when preparing each package because the packaging process is slightly different for each software category.

2.1.1 "Logo-compliant" Commercial Software

Windows Installer supports several different file formats. The MSI format discussed previously is the format used for the installation database. It can take two forms:

- A single integrated file that includes both installation instructions and all of the program components making up the software installation in compressed format
- A smaller instruction file calling upon a selection of separate program components

In many cases, the first format is the simplest to work with even though the file size is sometimes excessive because there is only a single file to manage. The second format is more convenient for very large programs such as Microsoft Office. The Microsoft Office XP installation components are 450 MB in size. Creating a single installation file for this product would be most cumbersome. On the other hand, a utility such as Second Copy by Centered Systems includes a single installation file of less than 1 MB.

Original MSI files should not be modified during the customization process even though some packaging tools allow the modification of MSI files. Instead, organizations should make use of the second Windows Installer file format, the MST file. MST files are transform files. They are separate Windows Installer files that can be applied to an MSI file to transform or customize the automated installation. For example, corporations that acquire Microsoft Office XP will most likely use the Microsoft Office XP Professional with FrontPage® installation CD. This CD includes all of the Microsoft Office products: Word, Excel, Outlook®, PowerPoint®, Access and FrontPage. Most organizations will not want to install all of these products on each of their computers for several reasons.

For example, most organizations will want to transform the Office installation because, as is illustrated in Figure 2-1, installing Office interactively requires a certain amount of installation expertise. Fortunately, with the right tools, transform files are relatively easy to produce. Seven transform files required for Office:

- One for the four core products: Word, Excel, PowerPoint and Outlook
- One for the core products plus Access
- One for the four core products plus FrontPage
- One for the four core products plus Access and FrontPage
- One for the addition of Access to an existing core Office installation
- One for the addition of FrontPage only
- One for the addition of Access and FrontPage

This can be reduced to four, the first plus Access only, FrontPage only and Access plus FrontPage, but it means running the installation of Office three times on a system to install all of the Office products. The ESP strategy organizations adopt will need to document which approach to use in situations such as this.

In addition, special transform files will also be required to apply regional settings. Microsoft Word, for example, requires the assignment of special settings during installation. These include the location of corporate templates. If these templates are stored on a central server that differs from region to region, the installation package will need to be modified on a regional basis.

The final Windows Installer file format is the MSP file or the patching file. MSP files are special files that can be used to apply program fixes and/or minor updates to an already installed product. The application of MSP files to products is part of the maintenance aspect of the software lifecycle.

MSI and Active Directory

In addition to providing full support for the software lifecycle, programs integrated to Windows Installer can also be more easily integrated to the Active Directory software distribution mechanism.

Applying Transforms

Transform or MST files are applied during an installation process. The installation still proceeds with the original MSI file, but the transform file tells Windows Installer which components to install and how to configure them.

Windows Installer Resource Kit

Wise Solutions offers comprehensive reference documentation for both system administrators and developers on Windows Installer through the Windows Installer Resource Kit at http://www.wise.com/rescenter.asp .

Windows Installer is used to support the installation of the traditional program; i.e., the program that is delivered and installed on a target system. But Windows Installer also supports two other types of program installations: Web Services and Terminal Services.

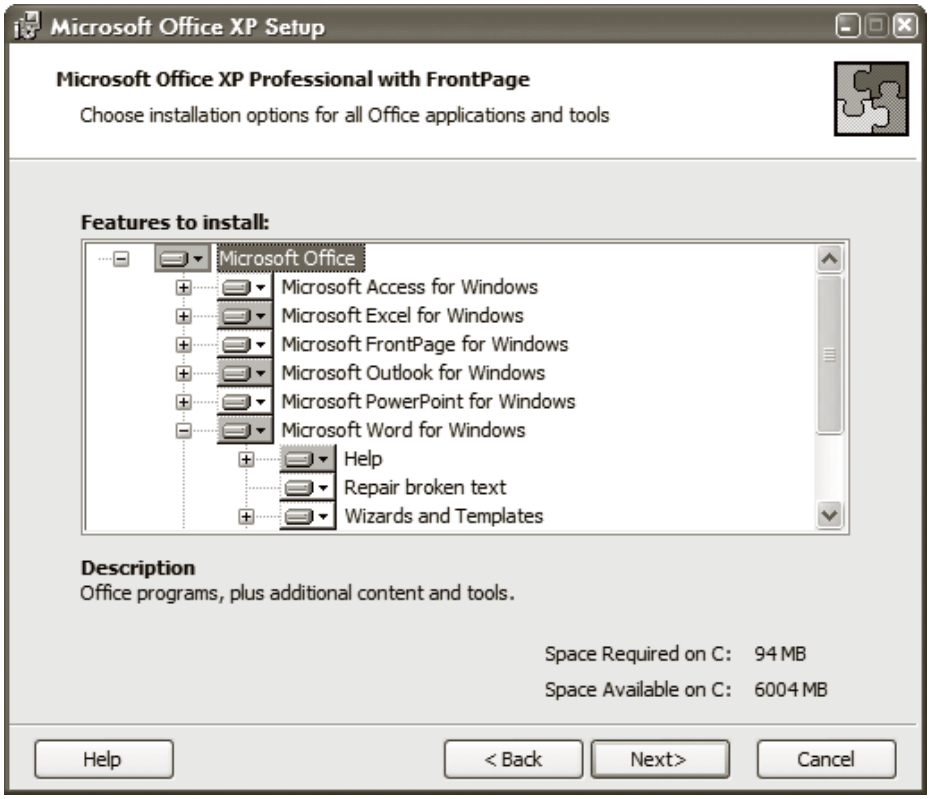


Figure 2.1: The Microsoft Office XP Interactive Installation

The Microsoft .NET Framework

Microsoft has created the common language runtime (CLR), which is included onto each computer system. .NET applications are designed to run with this runtime. Thus, system components will no longer need to be installed as part of an application installation, because the system files are already installed as part of the CLR.

Web Services and the .NET Framework

The .NET Framework is a new computing platform whose design goals deal with a number of different features but whose basic principle is to take advantage of the highly distributed nature of the Internet. This means that it is designed to call upon Web or Internet services from widely distributed and disparate sources. It is also designed to take advantage of distributed computing power — the power of both PCs and servers, even mainframes, that are connected together through the Internet.

The foundation of the .NET Framework is the common language runtime (CLR); it provides an environment that can manage code execution and provide core services such as memory management, thread management and remote execution. This core engine is included on every system that uses Web Services. For the Enterprise Software Packaging strategy, this means that Web Service installations are not as complex as traditional software installations.

Web Service installations are "light" installations because they do not require traditional registration on a computer system and only make use of an existing runtime. In order to maintain ESP standards, they should be integrated with the Windows Installer service. This integration will provide better support for the installation process. First, it can include the CLR if it is not present on the system. Second, it can be automated and customized through transforms. Third, it can provide a single compressed installation file. And fourth, if the service uses shared components, it can call upon Windows Installer's shared component management features to ensure that the service is always functional.

All of these functions will greatly simplify the installation process. Because Web Service installations are performed over the Web as clients request them, using Windows Installer-aware installations will greatly reduce service support issues. Once again, transforms can be used to customize the installation to either regional or corporate requirements.

Terminal Service Installations

Since Windows NT, Microsoft has included the ability to run software programs on a server providing a complete Windows computing environment to the user through a network terminal. With Windows 2000, Microsoft renamed this feature Terminal Services and embedded it into every Windows operating system . This service supports the publication of applications to remote computers giving them full access to programs running on Windows server. The greatest advantage is in deployment. Since the application operates on the terminal server, it is the only place it needs to be installed, updated and maintained. Also, since the application runs from the server, the only deployment requirement to end users is the deployment of a shortcut to launch the application. And this shortcut doesn't change even though the application may be upgraded or otherwise modified.

Nevertheless, the software program must be installed on the server. Because it will be a shared program, special considerations are required when using Windows Installer to prepare the installation. In a normal installation, many of the components that are specific to users are not installed. They are invoked when a new user launches the program for the first time. Windows Installer then uses its auto repair feature to install and configure these components. On a terminal server, users cannot invoke the Windows Installer service because of the potential impact on other users working with the same system. Therefore, installations that target a terminal server must be more complete and must include user configurations as well as complete local component installation. The preparation of the transform file must take these requirements into consideration.

2.1.2 Legacy Commercial Software

All of the commercial software and corporate applications that cannot be upgraded for a variety of reasons should be repackaged to take advantage of Windows Installer features. This process involves the "capture" of a typical installation of a program and the packaging of this installation in a format that is compatible with Windows Installer. Basically, a technician uses a cleanly installed machine, takes a "snapshot" of this clean machine state and performs an interactive installation of the software product or application before taking a post-installation snapshot. The repackaging tool then captures all of the differences between the pre-installation and the post-installation state. The advantage of this procedure is that transform files are rarely required because technicians can perform all customizations and configuration of the program before taking the post-installation snapshot.

The repackaging process is part of the customized consistency that is at the core of the ESP strategy. Here rigorous testing and precise practices are the norm. Technicians should take the time to become completely comfortable with the installation of a product before taking the final snapshot. In addition, they should prepare a detailed checklist to ensure that the package contains only the components required by the program and nothing else. They must keep in mind that any unnecessary file included in the capture will be found on all systems once the program is deployed in the corporate network.

In fact, a good repackaging tool will use exclusion lists — lists of files that are known to be either unnecessary for a package or that are known to cause support issues. This exclusion list should be well documented and should include reasoning behind the exclusion of each file listed. Finally, this exclusion list should be editable so that organizations can add additional elements to exclude from the packages they create.

Capturing Installations

It is extremely important to be very thorough during the repackaging process. This is why it is a good practice to ensure that technicians are very familiar with the product's installation before performing the actual repackaging operation. They should perform the interactive installation a number of times to be familiar with it, identify all of the customization steps and identify the order in which they must be performed. Finally, they should document all of this information. Then and only then, can they proceed to the repackaging operation.

[2]*Terminal Services is even embedded in Windows XP, though not as a server application. It is called the Remote Desktop and allows users to access their computer remotely over a network connection.*

Installation Snapshots

There are other technologies in addition to the snapshot approach, such as monitoring the installation or performing the installation in a virtual machine environment. The latter especially means that the capture process is much faster since machine re-imaging is not required between every capture.

Security Considerations for Legacy Software

In addition, organizations using the latest Windows operating systems (Windows 2000 on) with networks that still contain legacy software — software that was designed for previous Windows operating systems — will need to add security modifications to their legacy packages.

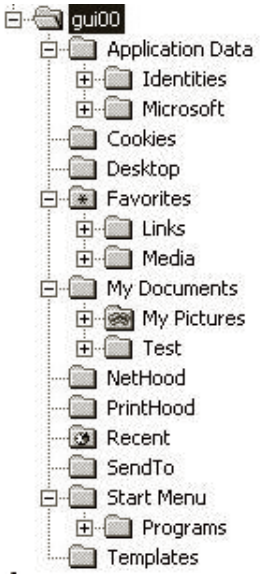
Windows 2000/XP operating systems use a 32-bit file system called NTFS. The advantage of this system over its predecessors is that every object stored in the system includes attributes. These attributes can contain security features — security features that are different for users, power users and administrators. The greatest limitations are applied to users. Since users only operate the system, they only need read and execution permissions for every system component. Thus NTFS "protects" system and program files by restricting access to these files.

But in pre-Windows 2000 operating systems, users were given much more leeway. This is because software integration was not controlled effectively. Many software products would install into and require constant read and write usage of the system or program directories. Giving users these rights would open the system to potential damage.

With Windows 2000, Microsoft changed the nature of the NTFS system lock down. They added further restrictions to users and changed the way applications work with the operating system. As a comparison, users in Windows NT have the same rights that power users do in Windows 2000. Today, users have significant restrictions within the operating system and program directories.

Software that is designed for Windows 2000/XP limits the installation of components in the system directories. Most of its components go into its own program directory. In addition, every component that is modifiable by a user (configuration settings, user preferences, etc) is stored within the directories containing the user profile. Here users rule and can read and write to their hearts' content. This is a good strategy because critical system and application files are protected for all users. If users damage something within their own profile, it simply needs to be erased and recreated .

The User Profile



In Windows 2000/XP, the user profile includes every element that is modifiable required between every capture.

Pre-Windows 2000 software works on Windows 2000/XP. But, because of the changes in NTFS security and because these applications have a tendency to store components within system and program directories, there is a requirement to "loosen" system security for the program to work with basic user rights. This is another feature that the repackaging tool should offer. It should allow the monitoring of all files that require modification rights during the operation of a program. Then, it should create an editable list of these files. This list can then become the source for a security script that will be added to the installation package. The purpose of the script is to change the default security permissions on user-modifiable program components contained in both system and program directories. Otherwise, users will not have the ability to execute the applications, because in Windows 2000/XP, they do not have write access to these directories.

This repackaging process should be performed for both legacy commercial software and corporate applications.

2.1.3 New Corporate Applications

All new corporate applications should also be integrated not only with the Windows Installer service, but should be designed to meet the requirements of the Microsoft Windows Logo program when practical. The Microsoft Logo program outlines the requirements for development of products that will operate on all new Windows platforms. It is designed to ensure that software will be stable and reliable and will work effectively with the resources of the operating system.

For example, one of the guidelines it outlines is the OnNow requirement. This requirement ensures that applications are properly terminated and reactivated when a Windows system enters Standby or Hibernation mode. Another is the requirement to function properly on Windows DataCenter Server, a 32-way multiprocessing "mainframe-like" Windows machine. This requirement ensures that programs can be redirected to processor groups in order to ensure proper resource assignment within the DataCenter server.

[3]*Care must be taken during this operation because the profile also stores user preferences. These must be saved and recovered after the profile has been recreated.*

Of course, most organizations will not need corporate applications that meet all of the Windows Logo requirements (if there is no DataCenter in the network, why invest to meet DataCenter specifications?), but it is a good idea to review the Logo specifications and select those that apply to the corporation's environment. Thus, a fourth core standard for the ESP strategy is the identification of a core set of development standards for corporate applications operating in the Windows network.

This set of standards will also need to identify when applications are designed to meet traditional "rich" Windows or the new "light" Web Service and .NET Framework environment requirements. Both will support the use of MSI installation files, MST transforms and especially, MSP patches for minor upgrades and repairs.

Designed for Windows Requirements

The details of this specification can be found at http://www.microsoft.com/winlogo/software/default.asp

2.2 Software Packaging Activities

The advantage of the four ESP standards outlined so far is that software lifecycle management will be simplified within the network, as almost all software products will be in Windows Installer format. But these are not the only standards required by an Enterprise Software Packaging strategy. Additional standards are required both at the packaging and testing level as well as at the enterprise level. Packaging standards begin with software component conflict management. Enterprise considerations are covered further below.

2.2.1 Conflict Management

Dynamic Link Libraries (DLLs) are essential Windows elements. A DLL contains the routines that can be called upon by a Windows program during its execution. In other words, a DLL is a library of functions with which a program can create dynamic links. In fact, DLLs are reusable components that are at the very core of the Windows programming concept. They can be used in different contexts. For example, Windows itself is a system that is composed of DLLs. The Windows DLLs offer system-wide services to other applications running on top of Windows. This includes how to talk to a printer, how to talk over the network, how to store information on a disk, how to display information on a screen, and so on. Windows programmers never need to worry about coding these particular functions; they only need to call the routines when they create their application.

A Windows program is like an orchestra conductor whose responsibility is to call upon the proper orchestra member at the proper time. In this case, orchestra members are DLLs. This approach offers great advantages. Since the program is a conductor, it does not load all of the components into memory when it is launched. Microsoft Word, for example, only launches its operating environment when a user double clicks on the icon. Then as the user applies different functions, Word calls the function from disk and loads it into memory, often unloading other functions to do so.

This approach greatly simplifies programming by dividing it into smaller components. Thus, the programmer only needs to concentrate on a given function while coding it. This programming method fully supports the distributed programming model, making large software development undertakings easier to manage and break down. But it does have its drawbacks.

The biggest problem with the DLL model is the model itself and the people who use it. Let's take Word again as an example. When Microsoft prepares a new version of Word, they do not need to concern themselves with all the Windows functions that Word will call on. Thus, the Word program or conductor will use both private DLLs — DLLs that are particular to Word and that are delivered with the product — and public DLLs — those that are already included in Windows. Figure 2-2 illustrates this concept.

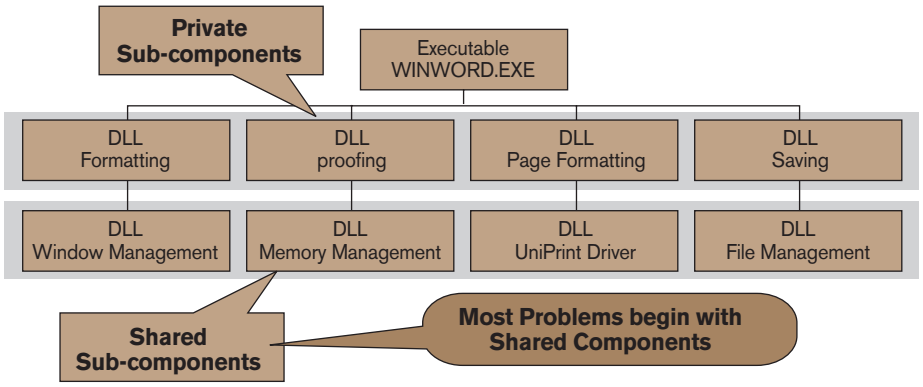


Figure 2.2: The DLL Concept in Windows

Legacy Software

While conflict management should be a lesser concern for programs that carry the Designed for Windows 2000/XP logo certification, it is absolutely required for the legacy software and applications found within corporate networks today.

Problems occur when, during the preparation of the software for market delivery, Windows software manufacturers decide to include public DLLs within their software installations. It's understandable. If a manufacturer has tested and retested their new application with a precise DLL kit (often amounting to a particular version of Windows), they will want to reproduce this version on destination systems by including all components in the system build.

So, when the software is installed on the destination computer, it will automatically install system components that are most likely already on the system, just to make sure that the software manufacturer receives a few less support calls. The logic is sound from the manufacturer's point of view: "We know that our product works with such and such a version of a DLL, so we have to ship it with the system."

This logic is a lot less sound from the client's point of view, especially if they are experienced Windows users and always keep their system up to date by downloading and installing service packs and hot fixes from Microsoft. If this is the case, it is likely that the version of the component on the destination system is newer than the version that comes with the software just purchased. When this software is installed, the original version of the component is destroyed, often without any warning to the user.

That's "DLL Hell" — a working system is destabilized because software manufacturers constantly deliver software which includes public DLLs. The result: the famous Windows blue screen of death!

Microsoft has come a long way towards solving this problem. Windows 2000 and Windows XP both include several features that aim to correct this problem. The Designed for Windows Logo specifications ensure that all new products will not behave in this manner. In addition, Windows Installer has its own built-in features for reactive conflict management. Thus all new products that completely adhere to the Windows Logo specifications should not produce this problem, but this does nothing for all of the legacy software products and applications that exist in corporate networks today.

This lingering problem is the basis of the fifth ESP standard: the repackaging process should include some form of conflict detection. This process is based on identifying all of the components programs install on network systems and repairing damaged applications. If a program includes a DLL that is older than the one stored on the system, three actions can be taken:

- Upgrade the DLL during the installation of the product. This only works if the DLL is backward compatible; i.e., it supports all previous features.
- Perform application isolation, part of which involves moving conflicting DLLs to another directory so that they will not destroy existing DLLs. This takes advantage of Windows' side-by-side DLL operational capabilities.
- Ignore the conflict.

Other conflicts can arise, but their impact is often not critical. They can be repaired on an as-needed basis. This DLL management process is outlined in Figure 2-3. This is another function of the packaging tool. It should fully support the conflict management and especially, the conflict resolution process.

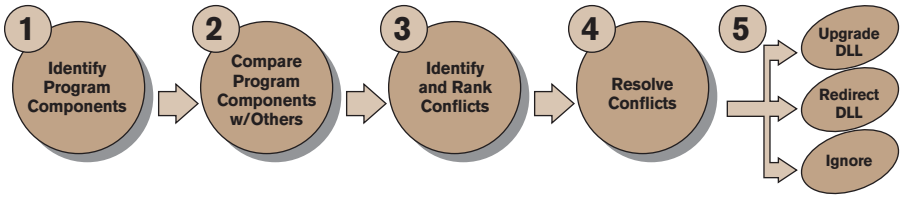


Figure 2.3: The DLL Conflict Management Process.

Conflict Management
Wise Solutions Inc. was the first to formalize this process.

It is during repackaging that conflict detection should occur. The packaging tool must inventory all of the packaged components and store them in a database containing all of a system's components. The database is then checked for conflicts. If conflicts are detected, then action can be taken to repair damaging components.

To properly perform this operation, the packaging tool must also be able to inventory the operating system even though it is not "packaged" in the same way a software program is. This must in fact be the original basis of the conflict database inventory.

DLL conflicts are not the only conflict type organizations encounter when creating installation packages. There are others such as registry conflicts, font conflicts, shortcut conflicts and so on. The conflict management process must take all conflicts into account.

Using both repackaging for the Windows Installer service and repackaging conflict detection techniques allows corporations to be proactive when designing new Windows networks. It proactively resolves problems in applications before they are deployed in the network.

2.2.2 Considerations for Mobile Personnel

The final consideration for Windows Installer packages deals primarily with mobile personnel and regional distribution within the corporate network. Remember that this service supports self-healing for all programs integrated with it during installation. Once a program is installed on a system, Windows Installer will perform a program consistency check every time the software program is launched. If there are inconsistencies between the actual program state and the contents of the installation database, Windows Installer will automatically launch a software repair phase.

During this repair phase, Windows Installer will connect to the original installation source of the software program and reinstall missing or damaged components. This means that if self-healing is to work, installation source files must be available on a permanent basis. This is a significant change from traditional approaches that focused on deploying software and then removing installation source files once deployment was complete. The sixth ESP standard is: organizations who want to use the self-healing capabilities of Windows Installer must maintain permanent software installation depots.

This has an impact on several processes, but two in particular are important to consider. The first is regional distribution of the network. Like initial installations, self-repair should never occur over the WAN; it should always be localized on the LAN. This means that software installation depots must be available in each regional site that includes at least one server. This also means that for a single package to work throughout the entire corporate network it must be possible to either use a single installation share name or it must be possible to use a variable to identify the installation source in the package. The first is possible within Windows 2000/XP networks through Microsoft's Distributed File System (DFS).

Local Source Files

Localizing source files on portables should not be an issue. Hard disk sizes today are substantial and new Windows operating systems fully support the creation of partitions that are hidden and inaccessible to the common user.

This system uses a single alias to regroup a number of distributed file shares. Thus the package need only reference the alias as the installation source.

But in actual fact, the second process is better because it also treats the mobile computer issue. Mobile computers are, by their very nature, often disconnected from the network. For self-healing to work, portables must carry the installation source files with them. Ideally, the portable would include a local, hidden partition on the hard disk where the installation source can be copied before installation. Then when self-healing is invoked for any reason, the portable will support it because original installation files will be available at all times.

This means that it is better to use variable installation sources within a package than to depend on the network because DFS cannot create an alias for portables. Ideally, both techniques can be used to complement one another — DFS on the network and variable installation sources in the package. This way the package will be able to support all installation instances: deployment in the central network, in a region, on a portable and even in the testing lab. This is another requirement for the packaging tool: it must be able to support the inclusion of variable installation sources within packages. It is also the basis for the seventh ESP standard: *all packages should include variable installation sources to support all installation scenarios.*

2.2.3 The Software Packaging Process

Now that most considerations have been taken into account, it is possible to outline the software packaging process. Preparing a package is 80% testing and 20% implementation. This means that the software packaging process must be based on a structured testing strategy which can include several different stages. Each focuses on a specific type of test. These tests are integrated into a single overall packaging process, an example of which is illustrated in Figure 2-4.

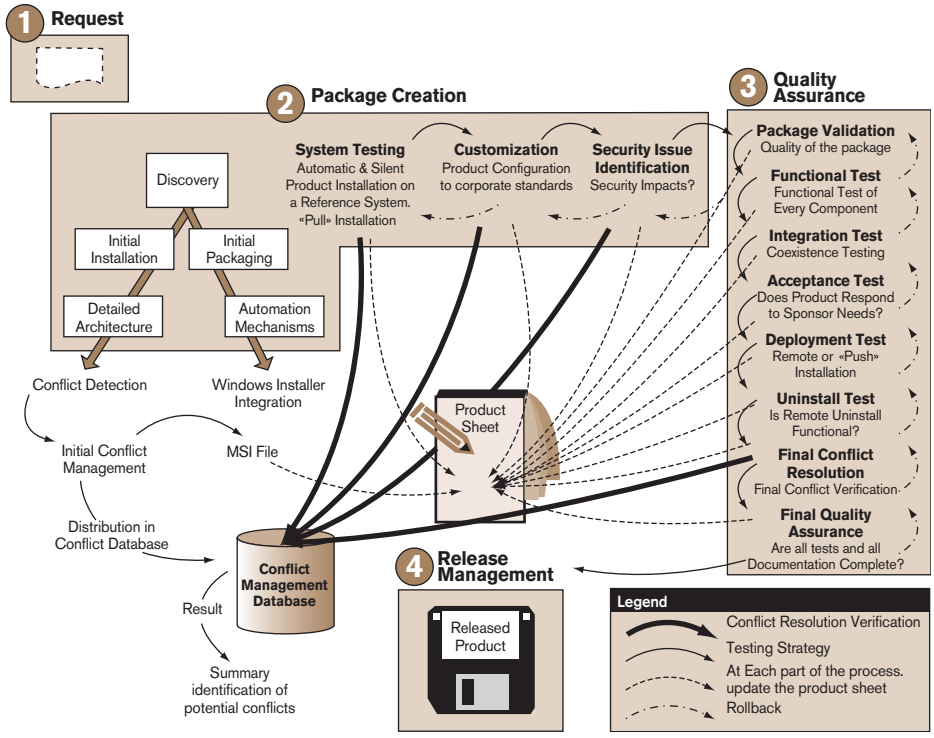


Figure 2.4: The Software Packaging Process

The software packaging process includes:

1. **Request** — A request for a new package is initiated as part of the overall software lifecycle for a product within the network. At any point in the process, the sponsor may want to know the status of the request.

2. **Integration** — This is the first technical stage. It involves the initial creation and testing of the package. It includes the following activities:

    a. **Discovery** — The first test is always an interactive discovery and analysis of a new product. This phase serves to identify the elements of the technical architecture for the product and its customization and configuration requirements. It serves to create a detailed interactive installation checklist to be used in the next stage.

    b. **Package Creation** — Once the first stages of discovery have been performed, it is time to automate the installation or create the initial package. This test focuses on the evaluation of the automated procedure by itself. Depending on the software product, this stage will include either a snapshot capture or an import of the MSI file within the packaging product.

    c. **Customization** — The package must be configured to corporate specifications. This is mainly focused on editing the MSI file to identify exclusions or creating a transform file for MSI-compliant software.

    d. **Security Issue Identification** — Are there any security issues with the product as installed during System Tests? Are there modifications required to operate it with user rights?

    e. **Initial Conflict Management** — The components of the package are integrated into the conflict management inventory database and initial conflict resolution is performed. It is also important to identify the product category at this stage. Will it be part of the kernel? Is it part of a role-based configuration? Is it an ad-hoc product? This will identify the conflict testing strategy and indicate against which products conflict testing is required.

3. **Quality Assurance** — Once the initial package has been prepared, it can move to quality assurance testing. During this phase, if the package is modified at any time, it should be rechecked for conflicts. The phase includes:

    a. **Package Validation** — Is the package valid? Is it well constructed? Does it conform to Windows Logo guidelines?

    b. **Functional Testing** — Does the product operate as expected once the installation is complete?

    c. **Integration Testing** — How does the product behave when merged with other products it must coexist with?

    d. **Acceptance Test** — Will the product sponsor (final client or user) approve the product as configured and installed?

    e. **Deployment Test** — Is remote distribution of this product required? If so, a deployment test must be performed to ensure that it behaves as expected during remote installation.

    f. **Uninstall Test** — If uninstallation is required, it should be tested both interactively and remotely.

    g. **Final Conflict Resolution** — Conflict resolution is an iterative process. A final check must be performed at this stage.

    h. **Final Quality Assurance** — Once all tests have been performed, a final quality assurance test should be performed. Is all documentation correct and complete? Have all testing procedures been followed correctly? These are some of the questions that must be answered during this phase before final release of the product to the corporate network.

4. **Release Management** — Once all testing is complete, the final package is released for distribution. It is inserted into the package repository and released to the software deployment process.

Each testing phase is important. If, for any reason, a product fails at any testing stage, it must be rolled back to the previous stage or quite possibly earlier stages and corrections must be applied.

Testing Systems

Testing systems should always be clean machines. Thus it is important in the ESP strategy to outline a specific process for the creation and recreation of clean machines. Disk imaging or virtual machine technologies are good support tools for this process.

2.2.4 Supporting Documentation

Documentation is important at every stage of the software packaging process. It is the basis of all quality assurance during packaging. But package documentation is not all that is required. ESP standards must also be documented and every packaging technician should be completely familiar with them. The packaging environment should also be fully documented. This environment, often a packaging lab, should represent every aspect of the corporate network. It should be managed by a Packaging Coordinator and packaging activities should be scheduled to avoid conflicts.

Standard packaging activities and practices should also be part of the documentation. Both general and specific packaging processes should be included. Ideally, these processes will be automated. In addition, the ESP tool should be able to generate automated documentation during the package creation process.

Following strict guidelines and rigorous testing procedures will make final packages more productive and will avoid most support issues. This is one of the major reasons for implementing an ESP strategy.

3. Enterprise Challenges

There are specific packaging challenges for medium to large organizations, especially if they are distributed geographically. Organizations of this size also tend to use several different financial models that can include centralized IT budgets or decentralized departmental IT budgets. These and other factors require careful consideration when designing the Enterprise Software Packaging strategy.

3.1 Establishing Packaging Standards

So far, seven core ESP standards have been identified:

1. All packages should aim to support the Windows Installer service.
2. All software within the organization should be categorized.
3. All software packages should use customized consistency — customizing or repackaging an installation so that it behaves in a manner consistent with internal corporate standards.
4. A core set of development standards for corporate applications operating in the Windows network should be set.
5. The repackaging process should include some form of conflict detection.
6. Organizations who want to use the self-healing capabilities of Windows Installer should maintain permanent software installation depots.
7. All software packages should include variable installation sources to support all installation scenarios.

But many more are required. Standards should cover everything from orientations affecting the ESP strategy itself to the actual details of package configurations. For example, packaged commercial software should not include any other shortcut than the ones required to launch the application. Start menus should be structured in a single, standard manner with a single, standard content. Kernel programs and popular corporate application shortcuts should be found in the Taskbar's Quick Launch Area. In fact, the presentation of all computer systems should be standardized to facilitate program access and enhance productivity.

In addition, ESP standards should include structured procedures for the packaging process itself. These should identify how to build packages for the system kernel, for role-based configurations and ad-hoc components. This includes selecting a "clean machine" to use, how to reset machines to "clean" environments, and how to create these machines. Detailed procedures should also outline how to use the various databases involved in the ESP process. Finally, they should include when and how to document packages.

Base Machines

In some cases, it will be necessary to use a base machine instead of a clean machine. A base machine is a clean installation of the components of the PASS model kernel.

The packaging tool selected to support the ESP strategy should help identify the required standards before beginning to package programs.

Organizations will also want to develop a technique for the identification of cost/benefit analyses for program packaging. The purpose of this technique is to identify when packaging costs more than the benefits it brings. Most evaluation techniques are based on the number of potential users for a package. Kernel products should all be packaged because they will be used by the entire user population. Programs that are part of role-based configurations should also be packaged because a package supports the automated assignment and removal of programs. Ad-hoc programs may or may not be packaged. Usually organizations will not package programs intended for very small user populations, for example, under 10 users. Figure 3-1 illustrates the packaging decision process.

Program Packaging

Some organizations choose to package 100% of their programs because of the gains in standardization and the ability to distribute all software from central locations.

Programs that are not packaged should still be standardized. A thorough and detailed installation procedure should be documented and provided to installation technicians.

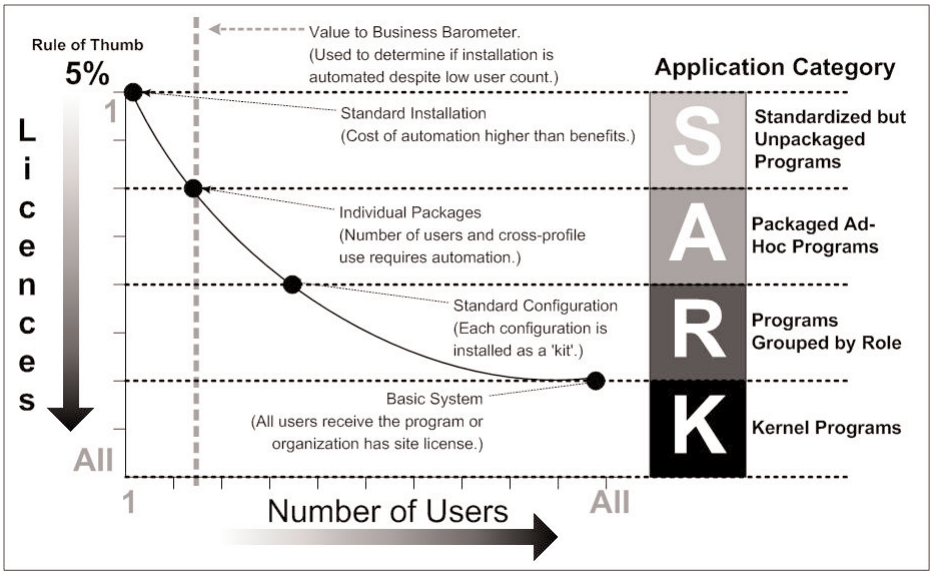


Figure 3.1: The Packaging Decision Process

Distributed ESP Team Models

There are two distributed ESP team models:

- *Central-based packaging with local transformations*
- *Fully distributed model with centralized standard support*

3.2 Distributed Packaging Teams

Organizations that are distributed regionally or that have distributed IT budgets will often want to use distributed packaging teams. These teams face special challenges because they must determine their working relationship. These teams are inevitable in certain situations. For example Company A has five main operational sites distributed in several different countries. Each site has a minimum of 2,000 users. Each site also has its own specialized manufacturing processes that are somewhat similar, but not identical to the others. Each site has its own IT budget.

This company wants to standardize their desktop and use a model similar to the PASS model. It is faced with a challenge because the head office can recommend, but it cannot direct because it does not hold the budget strings. Company A needs to set up a collaborative Enterprise Software Packaging process. Members of the head office IT group act as advisors in the process and members of each departmental IT group are part of the distributed packaging team. Standards are outlined by the team as a whole. Central MSI-based packages as well as the associated basic transform files are created at headquarters. These transform files are modified at each regional site to adapt the package to departmental requirements.

In such a scenario, the organization may even decide to delegate some of the core packaging activities to the different sites, especially if it is implementing ESP strategies for the first time and all products need to be packaged. The different departments can then share the initial packaging burden.

But such an ESP strategy must still uphold the standards it has set. To do so, the packaging product used must support the use of distributed databases in two instances: for conflict management and product inventory as well as for the package repository. This latter database should also host all of the documentation for all of the packages.

3.2.1 Distributed Package Repositories

The package repositories are relatively simple to distribute because they are made of structured data and are usually based on technology such as Microsoft SQL Server. Once again, the packaging product should support the inventory and deposit of information into an enterprise-level relational database. Ideally, it will also include an interface for replication design.

This interface will allow the ESP team to configure replication of all package information from a central database to departmental databases. It should also support multi-master modifications especially if the distributed teams are also creating new packages.

Figure 3-2 illustrates the interactions between these databases. The best topology is a hub and spoke topology. Data is uploaded from spoke servers (regions) to the hub server (head office) and then distributed to the other spoke servers from the hub.

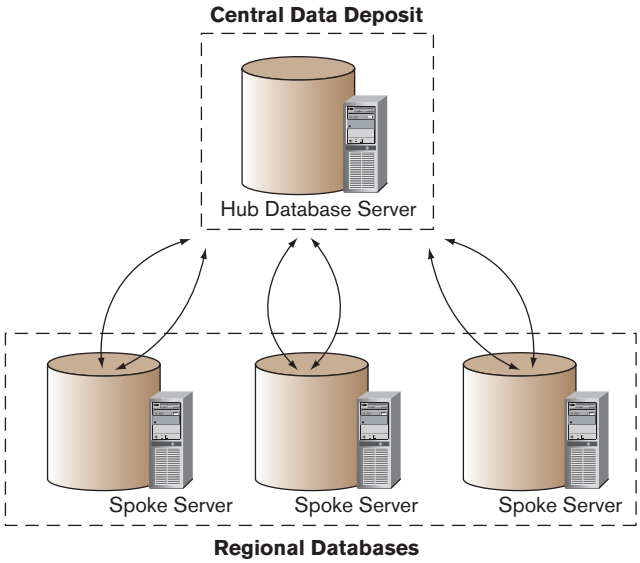


Figure 3.2: A Hub & Spoke Package Repository

3.2.2 Distributed Package Share Points

The same strategy must be applied to the packages themselves. The difference here is that the package share point is not a database, but rather a file share. This means that central and regional file shares must be synchronized and replicated throughout the enterprise network so that all packages can be available to all teams at all times.

This is an excellent opportunity for the use of technology such as the Microsoft Distributed File System. A single file sharing alias can be created with file shares located in each site. The Microsoft File Replication Service can then be configured to automatically replicate the deposits from site to site. In such a scenario, Active Directory must be used to publish the domain DFS root that will be required to synchronize these distributed file shares.

For this system to work, the ESP strategy must put in place stringent naming standards. All packages must use standard names and must be versioned; i.e., they must all indicate which version of the package the deposit refers to. In addition, three directory structures are required in the file shares:

- A working directory structure where all teams can store packages that are in the development process.
- A regional directory for packages that require regionally adapted transforms before final release to a region.
- A final directory where all released packages are stored when they are ready for distribution.

3.2.3 Distributed Packaging Documentation

The final technical challenge distributed packaging teams face is the distribution of documentation. Documentation is usually in unstructured form because it is stored in documents such as text files, spreadsheets or presentations. First, the ESP strategy must determine which of these formats will be used for package documentation. In many cases, the HTML format is the best because it is the format of the Internet.

If other document formats are used (for example, Microsoft Word), the distributed ESP team will require some means of replication for the documentation. In this case, it is simplest to store the documentation folders within the package repository and have it replicate along with the packages.

If documentation is in HTML format, then it might be better to use a content management system. These systems fully support the implementation of standards and approval workflows as well as ensure that all documentation is in the same format, and because they are based on a Web format, they are easy to replicate through the use of content replication systems.

3.2.4 International Packaging Particularities

A final challenge distributed and world-wide ESP teams face is package localization. There are often legal considerations to take into account when creating packages for world-wide distribution. For example, it is still illegal to distribute packages containing 128-bit encryption in some countries. For this reason, each localized team coordinator should be aware of such issues and should be responsible for flagging them to team members in other sites.

The best way to support this process is to provide a new package project inscription service within the package documentation strategy. Coordinators should receive new project listings on a regular basis and should raise flags when they are aware of international issues related to these projects.

In addition, a new category of Windows software has emerged recently. It is a program type that can be in a core language such as English and support the application of a Multilingual User Interface (MUI) that transforms the user interface language to that of the user's choice. Microsoft has included MUI capabilities in Windows since the 2000 edition as well as in Office. This means that international organizations can create a core English package and localized teams can add the language pack to this core installation. Thus the localized ESP team does not need to create the package for the core product. They may have to modify a transform for the core package but they will have to either create the MUI package or identify through an additional transform which language it is to apply during installation.

4. Conclusion

As can be seen, Enterprise Software Packaging is a core IT process, one that is not without challenges. But the application of a structured global approach to system construction, software distribution and especially software lifecycle management, will greatly simplify the integration of ESP strategies within a corporation.

Much relies on the selection of an enterprise-level packaging tool that provides both support for the implementation of standards and support for distributed ESP teams. Appendix B lists a summary of the requirements outlined in this paper for such a product. In addition, such a product should embed and support the best practices outlined here for software packaging. Finally, it should provide an appropriate return on investment. This is mostly tied to the packaging product's fee structure.

4.1 Summary of Best Practices for Software Packaging

Organizations should use these best practices when designing an ESP strategy.

1. Use a system construction model such as the PASS model.
2. Use a software lifecycle management process.
3. Integrate the ESP strategy with software distribution processes.
4. All packages should be designed to support the Windows Installer service. There will be exceptions to this rule, but they should include less than 2% of packages.
5. Categorize all corporate software programs in order to begin the packaging process; which are already in MSI format, which require repackaging and which will be developed to the MSI standard.
6. MSI files should not be modified from their original format. They should rather be customized through the use of "transform" or MST files.
7. Use Windows Installer packages for the installation and delivery of Web Services instead of simple file copies to reduce support issues.
8. Use transforms that will install all components locally when preparing an installation for use with a Windows terminal server.
9. Create detailed interactive installation checklists when repackaging software or applications.
10. Ensure that all program testing is performed with network accounts that have only user rights.
11. Pre-Windows 2000 program packages should include security scripts to ensure they will operate with only user rights.
12. All packages should be integrated into the package repository to ensure complete understanding of the systems within the corporate network.
13. All packages must use standard names and must be versioned.
14. Base the ESP strategy on standards at all levels — processes, procedures, documentation, distribution and packaging coordination.
15. Ensure that distributed ESP teams have comprehensive communications programs to support them.

4.2 The ESP Return on Investment

Implementing an Enterprise Software Packaging strategy is a comprehensive process for organizations that do not yet have it in place. One of the best occasions for the implementation of this strategy is during migration projects when the organization is migrating the operating system of its computers on the network. Because the migration will involve the staging of all systems and because the staging must be automated as much as possible, an ESP strategy becomes an absolute must in this situation. The ESP undertaking involves considerable effort, but it is one of the migration processes that provides the best and most immediate return on investment because it stabilizes the network.

Enterprises … desiring more than 99% desktop availability should embark on a desktop lock-down program that includes implementing software configuration tools and processes.
– Gartner Group

Organizations that implement ESP strategies find that they must invest more on proactive network management practices — in fact, invest more on preparing the network. But they also find that they will have considerable savings in Help Desk and support activities because few issues arise from software products that do not work once deployed. The ESP strategy ensures that all software products are completely functional before they are introduced into the network.

ESP and Support Issues
In some cases, organizations have reduced the number of support calls relating to software malfunction by 85% after implementing ESP practices.

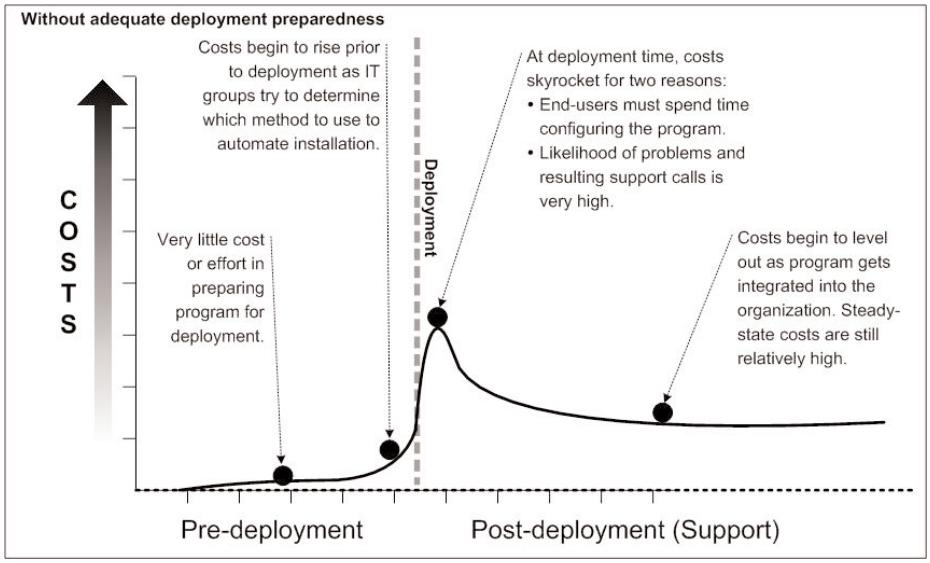


Figure 4.1: The Cost of Software Distribution without an ESP

The difference in costs between the proactive and the reactive approaches to software packaging is illustrated in Figures 4-1 and 4-2. It is clear that pre-deployment costs are greater with the ESP, but long-term savings are also quite evident. Thus, the ESP is an essential part of a total cost of ownership (TCO) strategy.

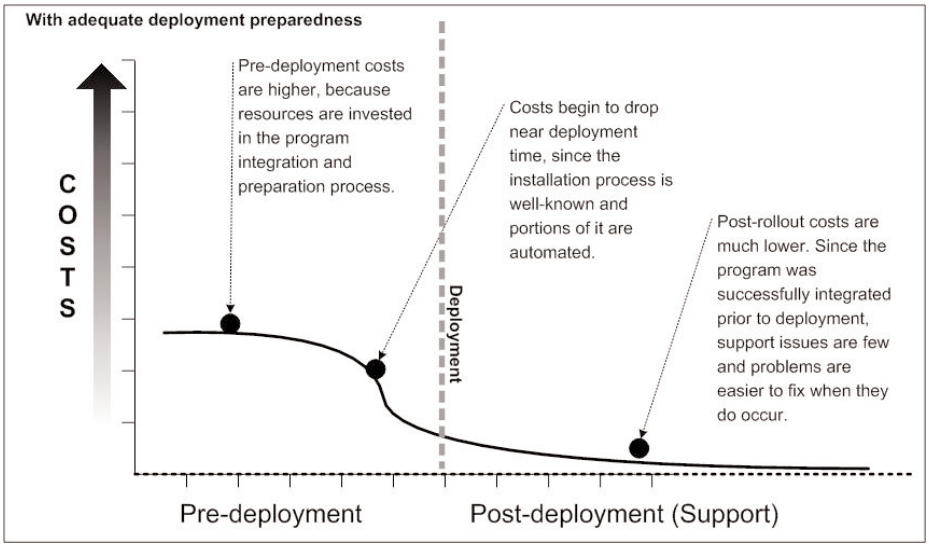


Figure 4.2: The Cost of Software Distribution with an ESP

4.3 Repackaging Tool Selection

Finally, the ESP strategy will only work if the packaging tool supports it. When selecting a repackaging tool, organizations should consider tools that meet the following criteria:

1. Support the entire ESP process, including all phases from request, application integration, testing, and through to release management.
2. Be based on industry best practices.
3. Provide flexibility to customize best practice processes based on organizational standards.
4. Provide built-in integration to the organization's distribution system. This helps automate the release management process and reduce the possibility of errors through manual processing.
5. Provide an open database structure that can be integrated with the organization's IT infrastructure.
6. Have a strong focus on accurate capturing. The capturing process is at the heart of repackaging legacy applications. When evaluating tools organizations should spend considerable time understanding and testing the capturing technology. All tools offer a basic snapshot approach, but few tools offer additional technologies that make captures faster and more accurate. In most cases, snapshot technology is adequate, but in the small percentage of cases where it's not, these additional technologies can save countless hours of effort.
7. Provide tools to facilitate teamwork and communications, features that are especially critical for decentralized packaging teams.
8. Provide tools that help document and streamline the packaging process.

Appendix A – Glossary of terms

Base machine — A PC with the minimum corporate standard software installed on it. If the organization uses the PASS model, a base machine includes the system kernel.

Clean machine — A PC with only the operating system and virus protection installed on it. No software applications are installed.

Conflict management — The process of inventorying all program components and verifying them against the system kernel and role-based configurations.

Corporate application — a program that is developed in-house to meet mission-critical or mission support activities within an organization. It also runs on Windows operating systems.

Package — An automated installation that is created during the ESP process. It is usually in MSI format.

Packaging lab — An environment that represents the production corporate network and that is designed to support packaging activities.

PASS model — A conceptual model used for machine construction. It is based on a layered functionality system and supports the construction of either PCs or servers.

Program — a software component that is either a commercial software product or a corporate application.

Reference machine — A base build machine that is used for package testing.

Repackaging — A process used to capture an interactive installation with the purpose of automating it and integrating it to the Windows Installer service.

Software product — a commercially-available product that is designed to run on Windows operating systems.

Sponsor — The individual who represents users during the packaging process. This individual must be familiar with how the program should work because of his role in the acceptance testing phase.

System kernel — A core software regrouping that delivers the productivity and collaboration functionalities required by every system in the corporate network.

Appendix B – Packaging Tool Requirements

Use these guidelines to select a software packaging tool to support the ESP strategy. This list is not exhaustive but at the minimum the tool should provide:

1. The same functionality for both repackaging commercial software and packaging corporate applications that you develop in-house.
2. Support for the .NET Framework installations.
3. Support for Terminal Services installations.
4. Repackaging with integration to the Windows Installer Service.
5. Editing of MSI files.
6. Creation of transform (MST) files.
7. Support for editable exclusion lists with explanatory text for the exclusion.
8. Support for the inclusion of scripts to the MSI file to include activities such as security modifications for repackaged legacy software.
9. Support for the Windows Logo Guidelines in regards to Windows Installer integration.
10. The ability to inventory all of the packaged components.
11. The ability to store package inventories in a database containing all of a system's components in order to detect conflicts.
12. Support for the resolution of conflicts.
13. Support for multiple package repositories such as one for the system kernel and one for each role-based configuration.
14. Support for the inventory of the operating system even though it is not "packaged" in the same way a software program is.
15. Support for the inclusion of variable installation sources within packages.
16. Support for installation snapshots to repackage software.
17. Support for the inclusion of the required ESP standards before beginning to package programs.
18. Support for the use of distributed databases in two instances: for conflict management and product repository as well as for the package share point.
19. Support for the inventory and deposit of information into an enterprise-level relational database.
20. Support for distributed packaging.

References

Aligning the Right Resources is Key to Software Distribution. R.Colville. Gartner Decision Framework, DF-18-1378. 21 October 2002.

Automated Software Distribution Does Not Reduce Staff. R. Colville. Gartner Advisory. 30 November 2001.

Creating Installations for Microsoft's .NET Framework using Wise for Windows Installer. Wise Solutions Inc.

Desktop Software Configuration: Key to Desktop Availability. R. Colville & D. Scott. Gartner Advisory. 28 June 2001.

Preparing for .NET Enterprise Technologies. Nelson Ruest & Danielle Ruest. Addison-Wesley. ISBN: 0-201-73487-7.

Repackaging Basics. Wise Solutions Inc.