The Case for Quality Assurance in Enterprise Software Packaging

Nelson Ruest

A Report by Wise Solutions, Inc.

Abstract

This white paper provides an overview of quality assurance practices in enterprise software packaging. Its intended audience is IT executives, system administrators and packaging technicians who belong to organizations that want to improve their return on investment for software packaging in the enterprise. This paper presents a comprehensive look at the components of a quality assurance approach and details the different testing stages and practices that can help organizations make the most of their software packaging strategies.

In addition, this paper outlines a selection of best practices for quality assurance in software packaging and general package testing. While it is not product specific, this paper is focused on the management of Windows-based software packages, especially packages based on the Microsoft Windows Installer service. Its processes and some of its best practices can also be used in non-Windows environments. Software packaging in general is a complex process from which organizations can derive great benefits, but only if their quality assurance processes are comprehensive. After all, the goal of software packaging is to reduce support issues once the package is deployed and in use. To this end, this paper provides a general discussion on the topic with strong specific recommendations in terms of gaining control over the related quality assurance processes in corporate environments.

About the Author

Nelson Ruest is an IT professional specializing in systems management and design. He is coauthor of multiple books, notably "Preparing for .NET Enterprise Technologies", published by Addison Wesley, ISBN 0-201-73487-7, and two books published by McGraw-Hill Osborne, "Windows Server 2003: Best Practices for Enterprise Deployments", ISBN 0-07-222343-X and "Windows Server 2003 Pocket Administrator," ISBN 0-07-222977-2.

Table of Contents

1. The Case for ESP Quality Assurance

Software packaging is the process that controls software installations in corporate networks. It allows organizations to prepare automated, silent installations that conform to corporate standards and that behave according to preset guidelines. As such, it is one of the most important processes in any software management strategy.

Enterprise Software Packaging (ESP) 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 an environment using ESP, packages are prepared for both commercial off-the-shelf software as well as internally-developed corporate applications.

Many organizations often fall into the "MSI trap" – having the complacent belief that all is well just because their software packages are designed to use the Windows Installer MSI format.

Until recently, there were no standards for software packaging in a Windows environment. This state of affairs changed with the release of Windows 2000, when Microsoft introduced a standard installation and software management tool for Windows systems: the Windows Installer service. This service is available for Microsoft's own software as well as other commercial software products. It can also be used to automate the installation of corporate applications that are developed in-house. The Windows Installer service enables organizations to implement a standard, consistent installation methodology, which is at the heart of Enterprise Software Packaging.

The Windows Installer service makes it possible for organizations that have implemented ESP to avoid many of the pitfalls related to software installation, especially those related to installations that may damage the target computer.

While use of the Windows Installer service is an important part of the ESP process, organizations need to be careful not to fall into the "MSI trap." The mere fact that an organization uses software packages in the Windows Installer format is not sufficient on its own for the realization of the full benefits of ESP.

1.1 The Software Management Lifecycle

Each software product has a lifecycle of its own that is independent of the other software products found within a network. 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 and development of a corporate application.
- **Software Implementation** — This phase focuses on software packaging, quality assurance testing and deployment.
- **Production/Maintenance** — This phase focuses on ongoing support activities for the product. It also involves the preparation, testing and distribution of scheduled updates.
- **Retirement** — The final phase focuses on removal of the product from the corporate network due to obsolescence. This removal may be followed by a replacement and thus may initiate the lifecycle process once again.

The software lifecycle begins the moment the software development project is initiated by a manufacturer and lasts 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 and lasts until the product is retired from use. Figure 1-1 illustrates the Corporate Software Management Lifecycle.

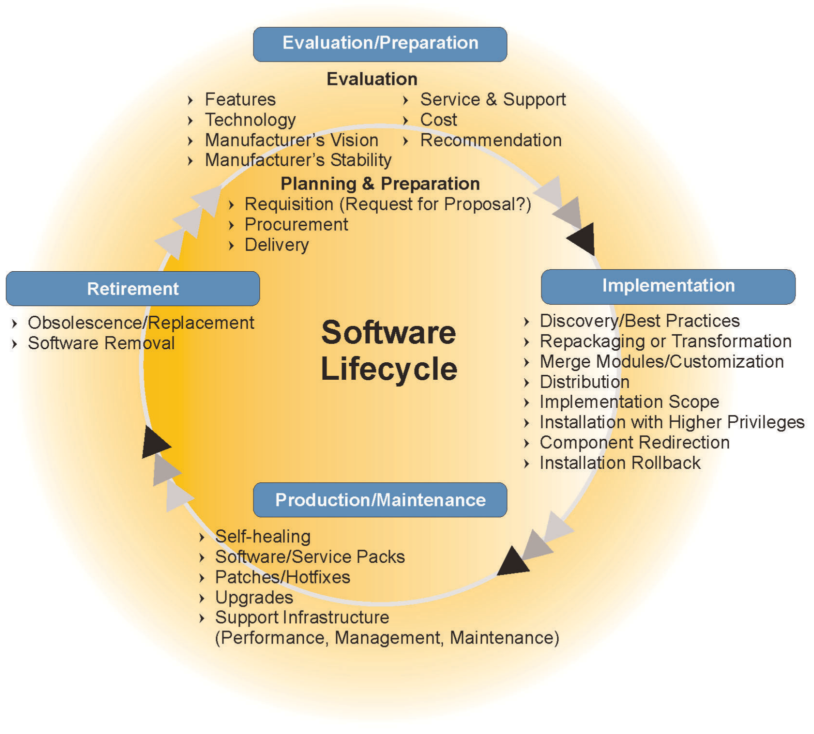


Figure 1.1: The Corporate Software Management Lifecycle

Enterprise Software Packaging processes focus on the implementation phase of the software lifecycle, but they are also designed to support the maintenance portion of the production/maintenance phase. The positive impact of ESP processes on production depend directly upon the amount of effort invested on the implementation phase for each given software product.

1.2 The ROI for Software Packaging

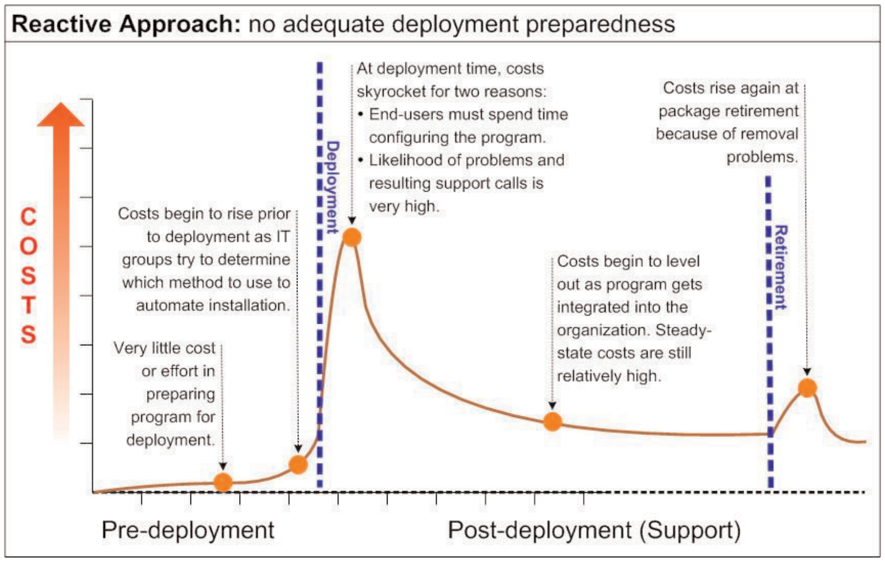


Figure 1.2: The cost of software distribution without an ESP strategy

Organizations that do not invest in an ESP strategy find that their software lifecycle management costs differ from organizations using such a strategy. If you are not using an ESP strategy, you will spend little time preparing software for introduction into your network. You will, however, find that your software management costs will focus on the production/maintenance phase since that is where problems and issues will arise (see Figure 1-2).

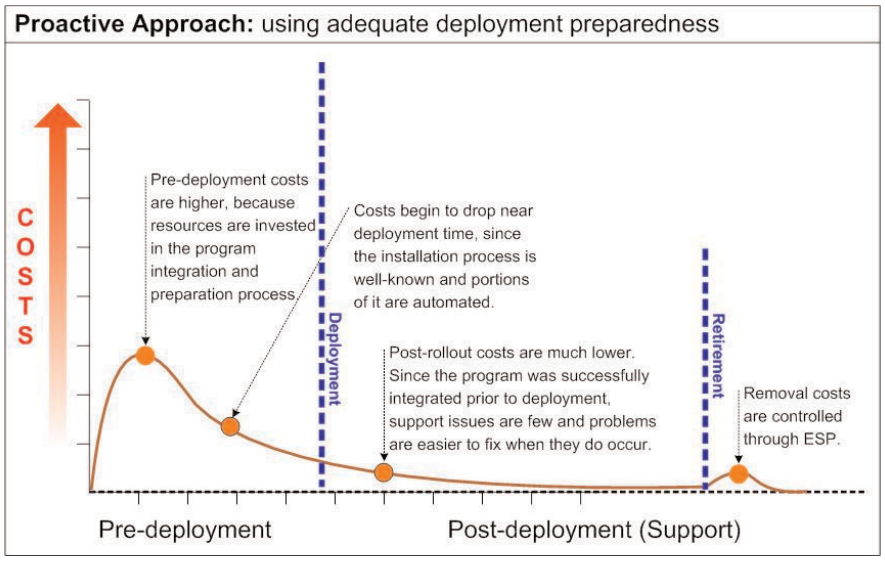


Figure 1.3: The cost of software distribution with an ESP strategy

"By using packaging tools first, enterprises can significantly reduce their rate of installation failure and make applications much more predictable."
– Tim Wilson, Enterprise Management Associates

Organizations that implement ESP strategies find that they must invest more on proactive network management practices — in fact, invest more on preparing products for introduction into the network. But they also find that they will have considerable savings in Help Desk and support activities, especially 3rd level support, because few issues arise from software products that work once deployed. The ESP strategy ensures that all software products are completely functional before they are introduced into the network (see Figure 1-3).

The difference in costs between the proactive and the reactive approaches to software packaging makes it clear that even though pre-deployment costs are greater with the ESP, long term savings are quite evident. This is the main reason for corporate implementations of ESP. But implementing ESP without proper processes can lead to uncontrolled pre-deployment costs and fewer post-deployment benefits. This is the MSI trap — putting too much trust into the Windows Installer service and too little effort in controlled quality assurance in the pre-deployment phase of package preparation. This is especially true when organizations begin to realize that the MSI format is not the answer to every deployment situation. For example, the Windows Installer service does not yet support the installation of hot fixes, service packs and device drivers. Therefore, it is important for organizations packaging applications for Windows Installer to perform some form of pre-deployment testing of these packages to ensure their components will not conflict with existing components already distributed within the network. This testing must be performed in a structured manner, using the same, standard testing stages for each package no matter which operator performs it. This is the essence of quality assurance.

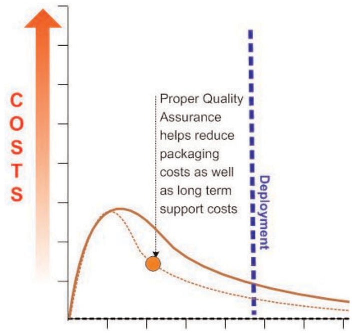


Figure 1.4: Reducing pre-deployment costs through proper QA practices

By using ordered and fully documented quality assurance processes, organizations using ESP can gain better control over package preparation costs and further reduce support costs during the production phase of a product's lifecycle (see Figure 1-4).

2. The Case for ESP Quality Assurance

The objective of quality assurance in software packaging is to supply a consistent result within fixed guidelines for a given operation. In fact, software packaging can be compared to the basic manufacturer model of assembly lines. Each time a package is prepared, operators perform the same basic activities, but in Enterprise Software Packaging, the actual product that goes through the assembly line changes each time the assembly line runs, even though the assembly process is always the same (see Figure 2-1).

Another element that is essential to ESP is the holistic view of the final product — the packaged software product does not operate in an isolated environment, but becomes a working component of the entire network. For this reason, it must be viewed as one cog in the system, not as a single package amongst many others. Quality assurance practices must reflect this holistic approach.

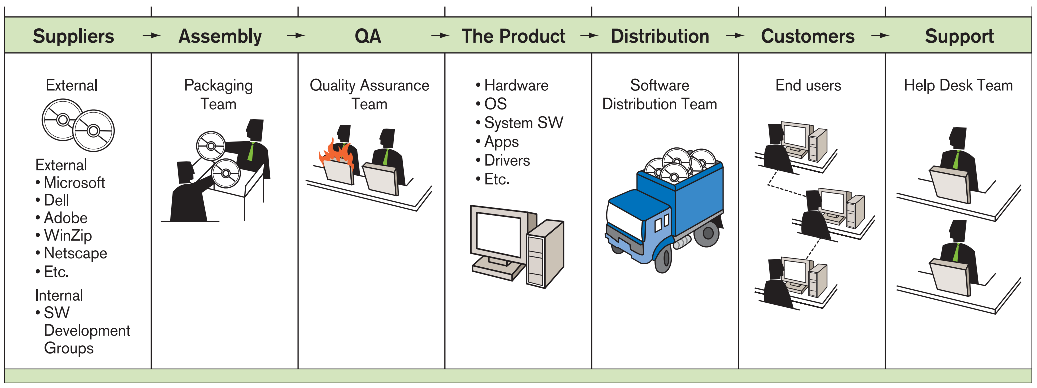


Figure 2.1: The assembly-line nature of software management

Wise Repackaging Products
Wise Solutions Inc. offers a repackaging tool in support of ESP: Wise Package Studio, Professional Edition. Full support for Quality Assurance processes is included in the Wise Package Studio Quality Assurance module.

There are actually four different types of software products that go through the ESP assembly line:

- **Native Commercial Windows Installer Software –** this software includes any product that bears the Designed for Windows logo as well as most newer commercial products destined for the Windows environment.
- **MSI-integrated Corporate Applications –** new or upgraded versions of corporate applications are often integrated to or designed to work with the Windows Installer service.
- **Repackaged Commercial Software –** with the ESP, all commercial software products that are not replaced with newer versions are repackaged to integrate their installation with Windows Installer.
- **Repackaged Corporate Applications –** corporate applications that will not undergo recoding or upgrades are repackaged to be integrated with Windows Installer.

Each uses slight variations of the same packaging process.

2.1 The Software Packaging Process

Preparing a package is 80 percent preparation or testing and 20 percent implementation or operation. This means that the software packaging process must be based on a structured testing strategy which can include several different stages, all depending on the complexity of the product to package. Each stage focuses on a specific type of test or assembly activity. These tests are integrated into a single overall packaging process, an example of which is illustrated in Figure 2-2. Once this process is complete, the package is ready for operation within the network.

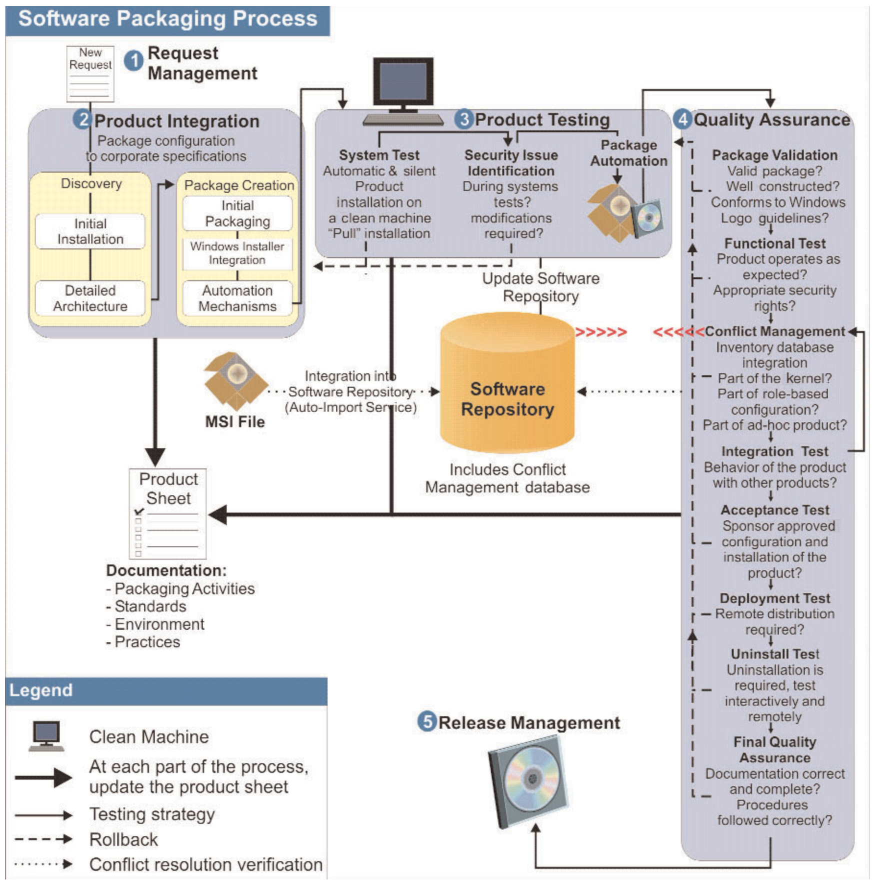

Software Packaging Process
New Request
1 Request Management
2 Product Integration
Package configuration to corporate specifications
Discovery
Initial Installation
Detailed Architecture
Package Creation
Initial Packaging
Windows Installer Integration
Automation Mechanisms
3 Product Testing
System Test
Automatic & silent Product installation on a clean machine "Pull" installation
Security Issue Identification
During systems tests? modifications required?
Package Automation
Update Software Repository
4 Quality Assurance
Package Validation
Valid package? Well constructed? Conforms to Windows Logo guidelines?
Functional Test
Product operates as expected? Appropriate security rights?
Conflict Management
Inventory database integration
Part of the kernel? Part of role-based configuration? Part of ad-hoc product?
Integration Test
Behavior of the product with other products?
Acceptance Test
Sponsor approved configuration and installation of the product?
Deployment Test
Remote distribution required?
Uninstall Test
Uninstallation is required, test interactively and remotely
Final Quality Assurance
Documentation correct and complete? Procedures followed correctly?
>>>>>
<<<<<
MSI File
Integration into Software Repository (Auto-Import Service)
Software Repository
Includes Conflict Management database
Product Sheet
Documentation:
- Packaging Activities
- Standards
- Environment
- Practices
5 Release Management
Legend
Clean Machine
At each part of the process, update the product sheet
Testing strategy
Rollback
Conflict resolution verification

Figure 2.2: 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 for the product—the person responsible for the evolution of the product within your network—may want to know the status of the request. In an ESP implementation, the tools used to support the ESP should support all aspects of the request phase: request, request validation and request tracking. In addition, they should support the integration of packaging requests into the project management aspects of software packaging.

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 (or Analysis)** – The first test is always an interactive discovery and analysis of a new product, especially its installation process. 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. This part of the process is mostly manual.

   b. **Initial Package Creation** – Once the first stages of discovery have been performed, it is time to automate the installation or create the initial package. Depending on the software product, this stage will include either a setup capture or an import of a Windows Installer file within the packaging tool and configure it according to corporate specifications. 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. The result is an automated installation file which is imported into the corporate software repository.

Ideally, the packaging tool used to support the ESP will include technologies which accelerate initial package creation. This should include systems that virtualize the capture and creation of the clean machine hosting the standard operating environment or system kernel your corporation deploys on each machine. This will help reduce the efforts required to create and test initial packages.

3. **Product Testing –** This phase focuses on testing the new automated installation. It covers three steps:

   a. System or Unit [nr2] **Test** – This test focuses on the evaluation of the automated or "silent" installation. It is also called unit testing because it tests the installation by itself. This test is performed on a clean machine through a "pull" installation; i.e., launching the installation from a server sharepoint containing the new package.

   b. Security Issue Identification – Following the system test, the operator identifies if there are any security issues with the product as installed. The operator must also identify if there are any modifications required to operate the product with user or non-administrative rights.

   c. Package Automation – Now that all of the package's components have been identified and potential security issues have been dealt with, it is time to finalize the package through the inclusion of scripts and other external components and then automate the package for deployment. This refined package will serve to support the next battery of tests.

4. **Quality Assurance –** Once the initial package has been prepared, it can move to quality assurance testing. This phase is critical to the proper viability of the package once deployed. It includes:

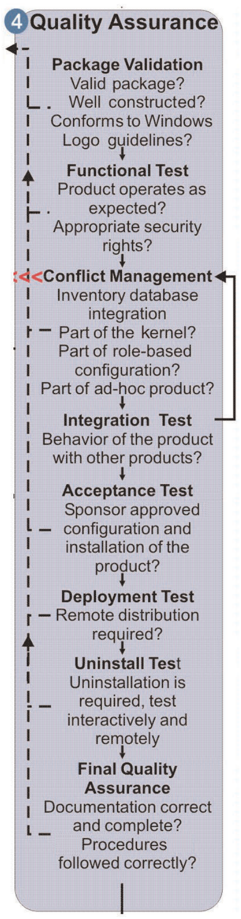
   a. Package Validation – Is the package valid? Is it well constructed? Does it conform to recommended Windows guidelines or even the Windows logo requirements? Ideally, the system supporting the ESP includes the capability to perform automated package validation, assessing the validity of the package against both standard industry and corporate-defined guidelines.

   b. Functional Testing – Does the product operate as expected once the installation is complete? Are configuration settings and security rights appropriate? Once again, the tool supporting the ESP should include capabilities that support functional testing, a sort of "testing expert" since it is unlikely that packaging operators have the knowledge required to test the operation of the functionalities of all of the products in your network. Having a tool that is able to generate a series of functional tests against a package will greatly reduce your testing times and ensure that all tests are standard. Ideally, this tool will help identify if the package is complete or not — pointing out if applications require shared or isolated versions of given files as well as spotting additional components that may need to be added to a package or even extraneous files that may be removed from the package.

   c. Conflict Management – Despite the capabilities of the Windows Installer service, conflict testing is an essential component of quality assurance because it will ensure that the package will be optimized for your network. In fact, conflict resolution is at the core of the package certification process. During this test, the components of the package are integrated into the conflict management inventory database and conflict resolution is performed. It is important to identify the product category at this stage: Will the package be deployed to all systems or will it target only specific systems? This will identify the conflict testing strategy and indicate against which products conflict testing is required. Thus the ESP tool must support various conflict testing strategies: against the system kernel (standard operating environment), against the products included in a given category and/or against individual products. Having automated conflict resolution techniques will also help reduce the overall testing workload.

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.


4 Quality Assurance
Package Validation
Valid package?
Well constructed?
Conforms to Windows Logo guidelines?
Functional Test
Product operates as expected?
Appropriate security rights?
Conflict Management
Inventory database integration
Part of the kernel?
Part of role-based configuration?
Part of ad-hoc product?
Integration Test
Behavior of the product with other products?
Acceptance Test
Sponsor approved configuration and installation of the product?
Deployment Test
Remote distribution required?
Uninstall Test
Uninstallation is required, test interactively and remotely
Final Quality Assurance
Documentation correct and complete?
Procedures followed correctly?

Proper processes within the QA phase ensure packages are fully tested and ready for deployment.

d. **Integration Test** – How does the product behave when merged with other products it must coexist with? Since this live test is performed after conflict resolution, it is usually a cursory examination of the operation of the product on its expected destination environment.

e. **Acceptance Testing** – Will the product sponsor (final client or user) approve the product as configured and installed? This greatly depends on the interaction with the sponsor throughout the package preparation process since it is the culmination of packaging and client request. Sound communications from the beginning to the end of the packaging project greatly facilitate this test.

f. **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. Ideally, this test can target a number of different machines, but most often, it only targets the machines available within the packaging environment.

g. **Uninstall Test** – Since all products require uninstallation as part of their overall lifecycle within your network, this test is also crucial. As such, uninstallation should be tested both interactively and remotely.

h. **Deployment Test** – 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. The ESP tool should support the tracking of all tests on a task-based activity list, checking off completed activities. This will reduce the time it takes to evaluate if all tests have been performed properly.

5. **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. Your ESP tool should support the release process and be able to integrate it to your software distribution and management tools.

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 phases and corrections must be applied. Rolling back from one test to another within a phase should not be of major concern. But package preparation costs rise significantly each time a package must slip from a packaging phase back to another. This is where your quality assurance process and the capabilities and feature set of your packaging tool come into play to help control rising costs within the packaging process.

In addition, the more your packaging strategy resembles the activities included in the ESP process illustrated in Figure 2-2, the more your packages will continue to reduce support costs once they are deployed in your network.

2.2 Using Documentation to Support QA

Documentation is important at every stage of the software packaging process, but it plays a special role in quality assurance during packaging. Each stage of the packaging process must be supported by proper documentation tools. Ideally, much of the packaging documentation will be generated by your ESP tool. But package documentation is not all that is required. Testing procedures must also be fully documented and every packaging technician, especially every QA tester, should be completely familiar with them. Operators should be given template documents that outline and support each testing phase they must run through.

Also, the testing coordinator should review documentation and testing results as often as possible, following the testing phases closely to be ready to help if pitfalls arise. This coordinator is often the overall laboratory coordinator. If this is the case, then this role should be filled by a reliable, disciplined person who is dedicated to the task and is given the authority to oversee it.

3. Improving QA Processes

As illustrated earlier in Figure 1-3, organizations having implemented a software packaging strategy already benefit from reduced post-deployment costs, especially if all packages are integrated to the Windows Installer service. But, if they have done so without assigning stringent quality assurance practices and without a complete process such as the software packaging process illustrated in Figure 2-2, they probably experience a package failure rate of about 15 percent. These organizations can benefit from improved package quality by reviewing current practices and implementing a proper process. This review should keep the following elements in mind:

- Ensure that proper communication flows occur during the breadth of the packaging project
- Use quality assurance practices at all stages of packaging

These two elements alone can help organizations to achieve the cost reductions illustrated in Figure 1-4, gain increase package stability and reduce failures to almost two percent or in some cases, even lower.

3.1 Communications, a QA Cost-saver

In all IT processes, communications plays a crucial role in the success or failure of the operation. Yet, few organizations give it enough attention. A good example is massive deployment projects. While organizations spend appropriate efforts to design and determine proper communications plans to reach end users, they often completely overlook intra-project communications — communications between the major actors of the project itself. This can have disastrous effects on the expected outcome of the project.

This is one reason why the software packaging process must be supplemented with appropriate communication flows between all the personnel involved. Special attention in the design of these flows should be paid to getting the product sponsor involved at the proper stages of the packaging project as well as making sure all the members of the packaging team understand communication flows within the team. Also, one good way to ensure proper communication takes place is to use QA checkpoints throughout each packaging project.

3.1.1 Getting the Product Sponsor Involved

In a structured network, product sponsors — subject matter experts assigned responsibility for both the operation and evolution of any given software product in your network — should normally be the initiators of a package request. Organizations cannot simply permit users to directly request new products, especially new products that initiate packaging projects, otherwise you will soon find yourself with an uncontrolled network. Organizations that want to gain mastery over their network, usually implement a software rationalization process to reduce the actual number of software products within the network. This process is based on the removal of products with duplicate functions (two different word processors, for example), products with several different versions in use, and products whose functions can be performed through the use of runtimes. Performing this rationalization is no small task, but it reaps vast cost reductions by reducing the number of software components to manage within the network.

For this reason, sponsors form a crucial component of the packaging process by using rationalization guidelines to address product requests at the very outset of the packaging process. This means that if users are allowed to request new products, the tool supporting the request process should be able to redirect them to the appropriate sponsor for approval or dismissal. This gets sponsors involved in the packaging project right from the outset and ensures better results in the end.

Organizations that choose to assign the product request phase to a dedicated resource should ensure that once a product has been approved for packaging, that it is first assigned to a sponsor and second that the sponsor is notified of the request before packaging begins.

If sponsors have been notified at the onset of a project, make sure they will continue to receive status information about their request so that they will be able to plan their efforts and be ready and available for acceptance testing when the time comes.

3.1.2 The Team Communication Process

Another important aspect of proper communications is the communication flow between all the personnel involved in the packaging project. This flow is based on the packaging process itself (see Figure 3-1). It involves the following personnel in your organization:

- **Users** — As the initial requesters of new packages.
- **Software Product Authority** — This is an optional position; it depends on whether the organization has decided to centralize or decentralize software product authorizations.
- **Product Sponsors** — As the people responsible for each product in the network. If software product authorization is centralized, they are contacted by the Project Manager. If it is decentralized, they are contacted directly when a new package request is performed. In both cases, they give their personal recommendations about the future package. They may even be involved in initial packaging phases to help improve processing.
- **Project Manager** — This is the packaging lab coordinator who also acts as the packaging authority. This person is responsible for the initiation of the packaging project and the scheduling of packaging activities within the lab. The package project gets the go ahead from product sponsors. This person also assigns the packaging personnel.
- **Packaging Team Leader** — This person is highly talented in packaging (it may be the same person as the Project Manager). The packaging expert is responsible for all packaging standards and packaging project template management. Ideally, this person uses built-in ESP tool functions to create templates and manage the security access to template modifications. This person reviews the packaging request and assigns the appropriate template.
- **Packagers** — These packaging operators will vary in skill sets. Junior operators can be used for the initial package captures since the process is based on templates. More senior operators can be used for the initial package testing phases. After each step of the process, these operators report to the packaging authority.
- **Quality Assurance Testers** — Different personnel are used for QA testing. This ensures that they do not have any preconceptions about the package. These individuals are responsible for package certification. Once certification is complete, they report back to the packaging authority who alerts the sponsor when acceptance testing is due. QA testers will then work with the sponsors to complete acceptance testing. Once the testing is complete, QA testers advise the packaging authority who authorizes the official release of the package to the deployment team. A complete report should be included with the package to facilitate deployment.

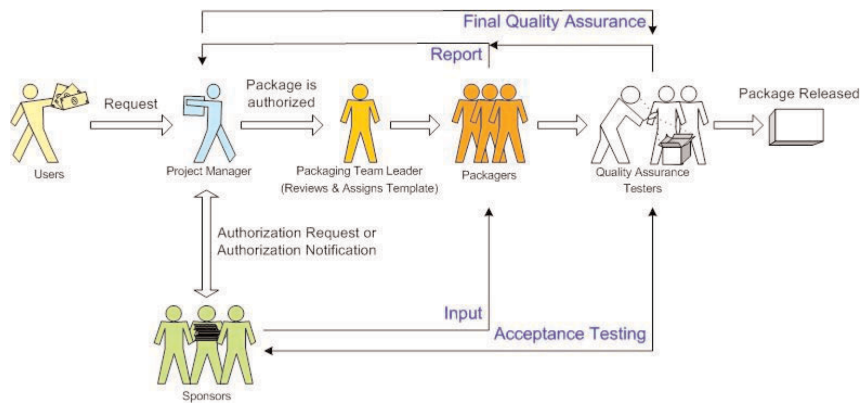


Figure 3.1: The packaging team communication flow

3.1.3 Using Standard Operating Procedures

Another key factor to QA success is the use of standard operating procedures (SOP). These procedures are step by step instructions [nr3] that document each phase of the packaging process. SOPs should be used in every aspect of network operations, but if this is not the case, it may be easiest to begin with packaging processes since they are repetitive, assembly-like tasks that can easily be documented.

The advantage of the SOP is that it makes it much easier to train new personnel when they arrive in the packaging team. More information on how to create and write SOPs can be found in Appendix B.

3.1.4 Using QA Checkpoints

Throughout the packaging process, it is important to make use of QA checkpoints at the end of each stage. When one of the major phases is complete, a complete report for the phase must be delivered to the packaging authority. The authority can also perform spot checks throughout the process to ensure additional quality control.

Ideally, the packaging tool tracks the complete packaging process and records project advancement. It should also include a link to the product request tool to report on progress to the requesters.

3.2 Pre-Deployment Testing

Though the role of the product packagers is complete after the package is released for deployment, the testing itself is not complete. Most organizations will perform pre-deployment testing of the package to analyze its behavior within the actual network. This may involve two types of tests: the proof of concept and/or the pilot project.

3.2.1 The Proof of Concept

Proof of concept testing is usually performed when a package will affect the entire network and will be deployed on all systems, either servers or workstations. The proof of concept population usually represents one percent of the user population and will focus primarily on the project team itself as well as product sponsors. This means that most of the personnel involved in this testing phase are highly skilled and will be able to identify any initial problems with the package.

This phase is optional, but it is a good idea to get project personnel to "eat their own dog food" before releasing the product to the entire population. Many initial issues can be identified in this testing phase. If this is the case, the project will have to review either the package itself (which is unlikely given the level of QA testing), the original assumptions of the project or the distribution strategy.

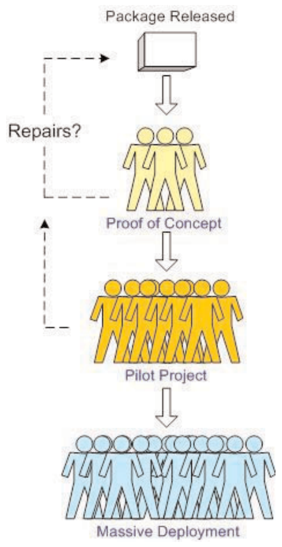


Figure 3.2: Pre-deployment testing processes

3.2.2 The Perfect Pilot Project

Pilot projects represent between five and 10 percent of the user population. This should involve a user population sample that is representative of the entire user population. One of the best ways to select this population is to review human resource records and select personnel that represent a sample from each job category within the organization. This selection may be slightly skewed since thee may be personnel to whom the project owes "favors" and therefore must be included in the pilot. There will also most certainly be an over representation of management personnel since they have the authority to request participation in the project. Unfortunately, this over-sampling can rarely be avoided.

In addition, if there are regional offices in your organization, your pilot project should include at least one representative site as well as personnel from your main headquarters.

Results from the pilot are used to review and repair any issues.

3.2.3 Going Beyond the Pilot Project

ESP tools have evolved to include facilities that go beyond simple packaging and can assist even in pre-deployment testing. These tools can now create a "zero-touch" package that can be sent to a sample population of production desktops that can even include the entire network. This empty package will simulate the installation without actually making any changes to the target machine, then will report back results to a central server for analysis.

This "pre-flight" testing capability goes well beyond the pilot project to identify additional issues with deployment. For example, it will immediately identify all machines that will not succeed in installing the package due to lack of disk space or machine capabilities. It will also identify issues in the deployment strategy such as improper target collections.

If your ESP tool supports this feature, you will find an additional decrease in support issues during the actual deployment phase because you will know exactly what to expect.

4. Conclusion

As can be seen, Enterprise Software Packaging is a core IT process that can greatly reduce operational costs for software products and applications within any organization. But the effectiveness of this process can be greatly enhanced through the implementation of strict quality assurance processes. Strict processes do not mean restrictive processes. In fact, QA provides the foundation for doing things in a right and proper way, not to stymie creativity. In any structured process, it is important to include capabilities for positive feedback. It is the structure itself that provides most benefits since it ensures that all operators perform the same tasks in predictable ways.

As with ESP itself, much relies on the selection of an enterprise-level packaging tool that provides both support for the implementation of standards and support for advanced QA processes. This paper lists a series of requirements for such a product, especially in terms of quality assurance. The selection of the appropriate tool will guarantee a rapid return on investment and vastly improve software operational issues in your network.

4.1 Best Practices for Packaging Quality Assurance

Organizations should use these best practices when implementing quality assurance in their ESP strategy.

1. Focus on quality assurance throughout your ESP strategy.
2. All packages should be designed to support the Windows Installer service. There will be exceptions to this rule, but they should include less than two percent of packages.
3. Use a lifecycle to manage all software products within your network.
4. Categorize all corporate software programs in order to perform the packaging process; which are already in MSI format, which require repackaging and which will be developed to the MSI standard.
5. Use a software packaging process that focuses 80 percent preparation or testing and 20 percent implementation or operation.
6. Assign product sponsors for each software product in your organization and make sure you reassign a sponsor to a product if an existing sponsor leaves the organization.
7. Involve product sponsors at the earliest stages of the packaging process.
8. Use rationalization to reduce the number of software products in your network.
9. Use conflict detection testing to certify all packages and avoid the "MSI trap."
10. Create detailed interactive installation checklists when repackaging software or applications.
11. Ensure that all program testing is performed with network accounts that have only user rights.
12. Pre-Windows 2000 program packages should include security scripts to ensure they will operate with only user rights.
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 ESP teams have comprehensive communication programs to support them.
16. Document all aspects of the software packaging process.
17. Use standard operating procedures as a guide to introduce packaging processes to new personnel.
18. Use separate personnel to prepare packages and perform QA testing to ensure no preconceptions are taken into account during testing.
19. Carefully select both proof of concept and pilot project target audiences to ensure proper representation of the user population.
20. Select an ESP product that supports "pre-flight testing" to reduce the risk of deployment issues before actually deploying an application to the entire user population.

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.

Package Certification — The process of applying full quality assurance testing for a package before it is released to the user population.

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. The difference between the reference machine and the base machine is that the reference machine includes the system kernel or standard operating environment, while the base machine includes only the base operating system.

Repackaging — A process used to capture an interactive installation with the purpose of automating it and converting it into a format that will enable it to be installed using 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.

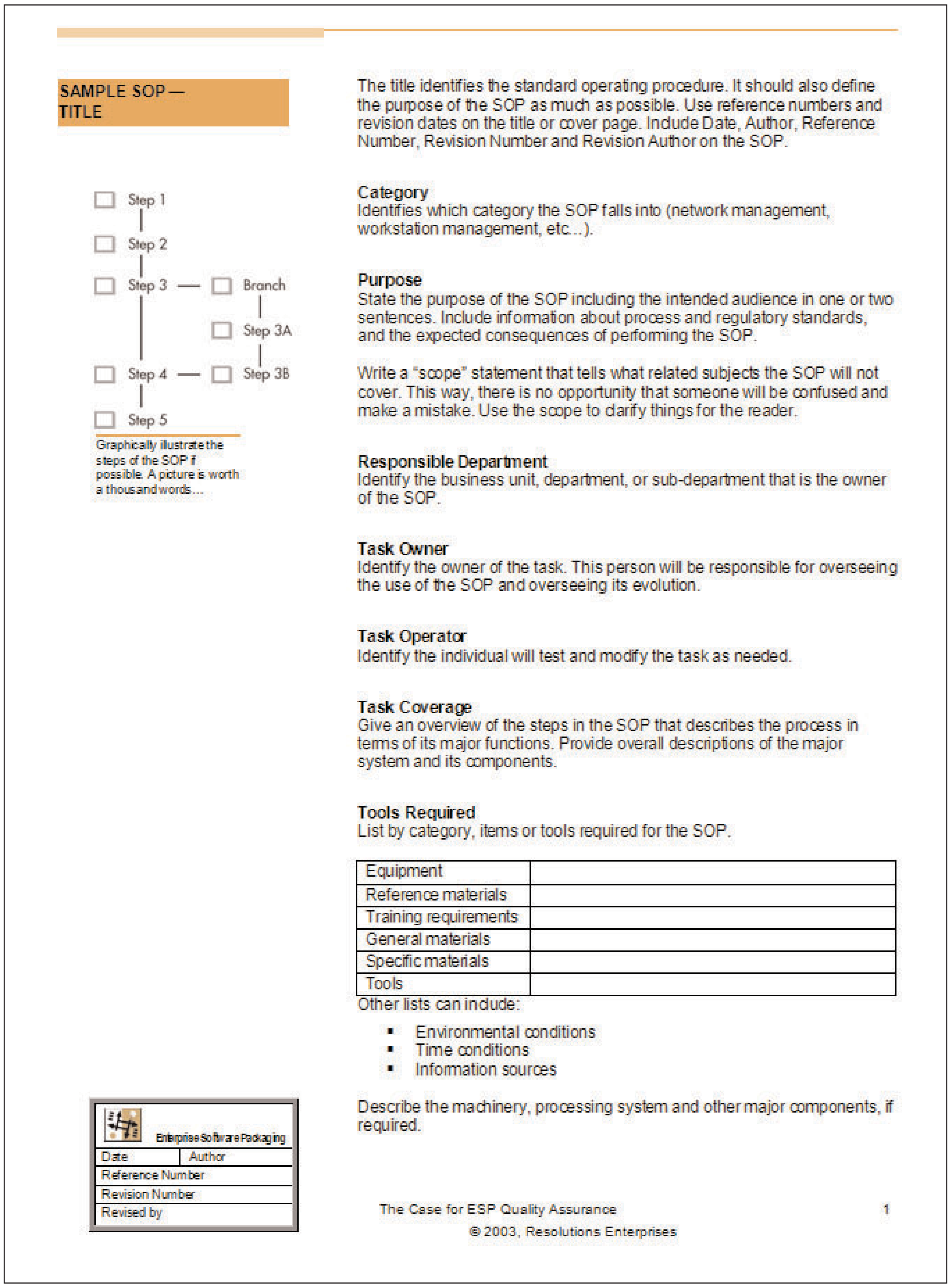
Standard operating environment — A core software grouping that delivers the productivity and collaboration functionalities required by every system in the corporate network. Also known as "system kernel."

Standard operating procedure — A structured, documented step by step approach for the execution of a given task.

System kernel — A core software grouping that delivers the productivity and collaboration functionalities required by every system in the corporate network. Also known as "standard operating environment."

Appendix B – Designing Standard Operating Procedures

Use the following tools to help design your own standard operating procedures for Quality Assurance in Enterprise Software Packaging.



Step 1
Step 2
Step 3
Branch
Step 3A
Step 4
Step 3B
Step 5

SAMPLE SOP — TITLE

Graphically illustrate the steps of the SOP if possible. A picture is worth a thousand words…

The title identifies the standard operating procedure. It should also define the purpose of the SOP as much as possible. Use reference numbers and revision dates on the title or cover page. Include Date, Author, Reference Number, Revision Number and Revision Author on the SOP.

Category
Identifies which category the SOP falls into (network management, workstation management, etc…).

Purpose
State the purpose of the SOP including the intended audience in one or two sentences. Include information about process and regulatory standards, and the expected consequences of performing the SOP.

Write a "scope" statement that tells what related subjects the SOP will not cover. This way, there is no opportunity that someone will be confused and make a mistake. Use the scope to clarify things for the reader.

Responsible Department
Identify the business unit, department, or sub-department that is the owner of the SOP.

Task Owner
Identify the owner of the task. This person will be responsible for overseeing the use of the SOP and overseeing its evolution.

Task Operator
Identify the individual will test and modify the task as needed.

Task Coverage
Give an overview of the steps in the SOP that describes the process in terms of its major functions. Provide overall descriptions of the major system and its components.

Tools Required
List by category, items or tools required for the SOP.

Equipment	
Reference materials	
Training requirements	
General materials	
Specific materials	
Tools	

Other lists can include:

- Environmental conditions
- Time conditions
- Information sources

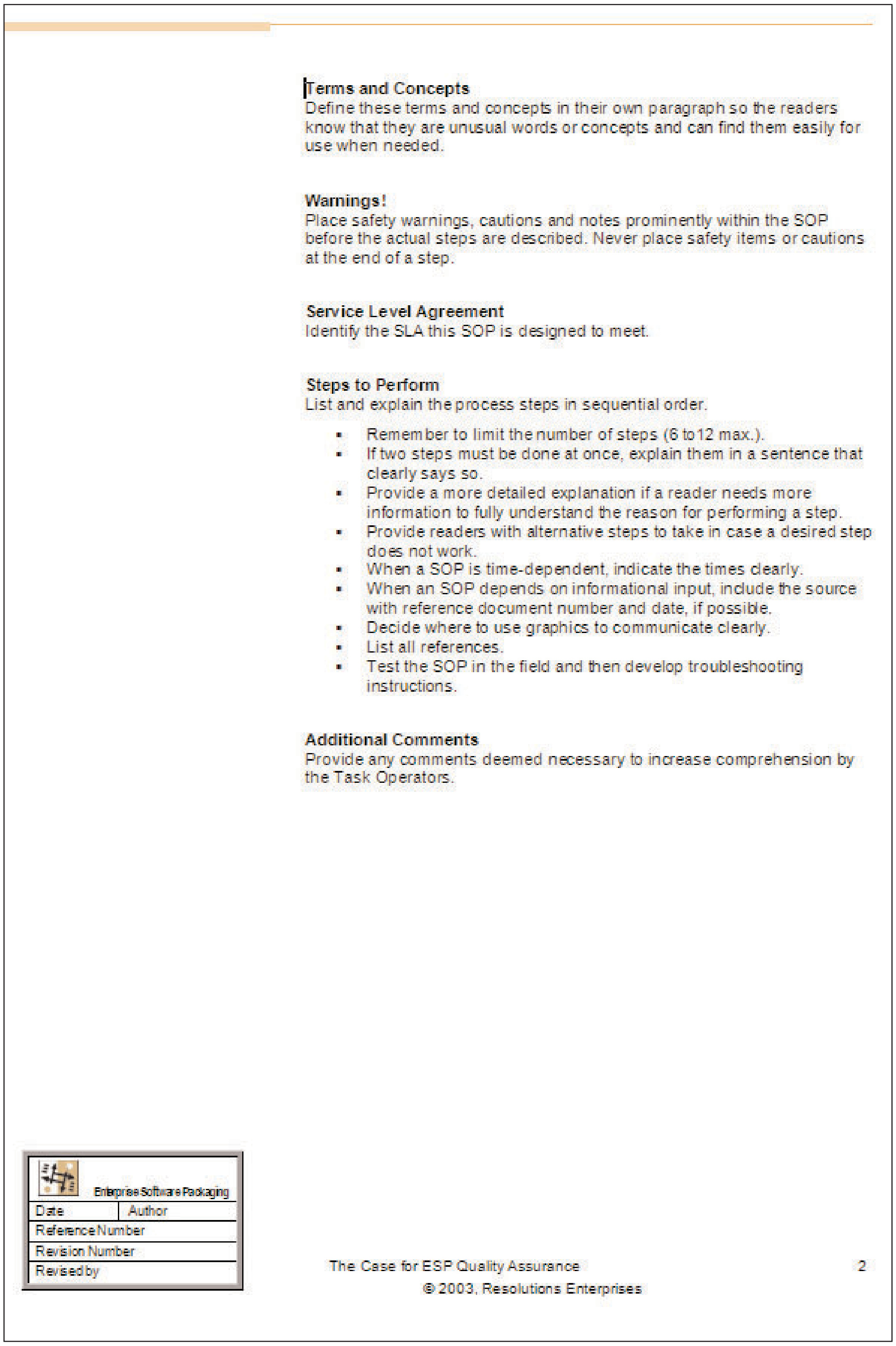
Describe the machinery, processing system and other major components, if required.

Enterprise Software Packaging

Date	Author
Reference Number	
Revision Number	
Revised by	

The Case for ESP Quality Assurance 1

© 2003, Resolutions Enterprises



Terms and Concepts
Define these terms and concepts in their own paragraph so the readers know that they are unusual words or concepts and can find them easily for use when needed.

Warnings!
Place safety warnings, cautions and notes prominently within the SOP before the actual steps are described. Never place safety items or cautions at the end of a step.

Service Level Agreement
Identify the SLA this SOP is designed to meet.

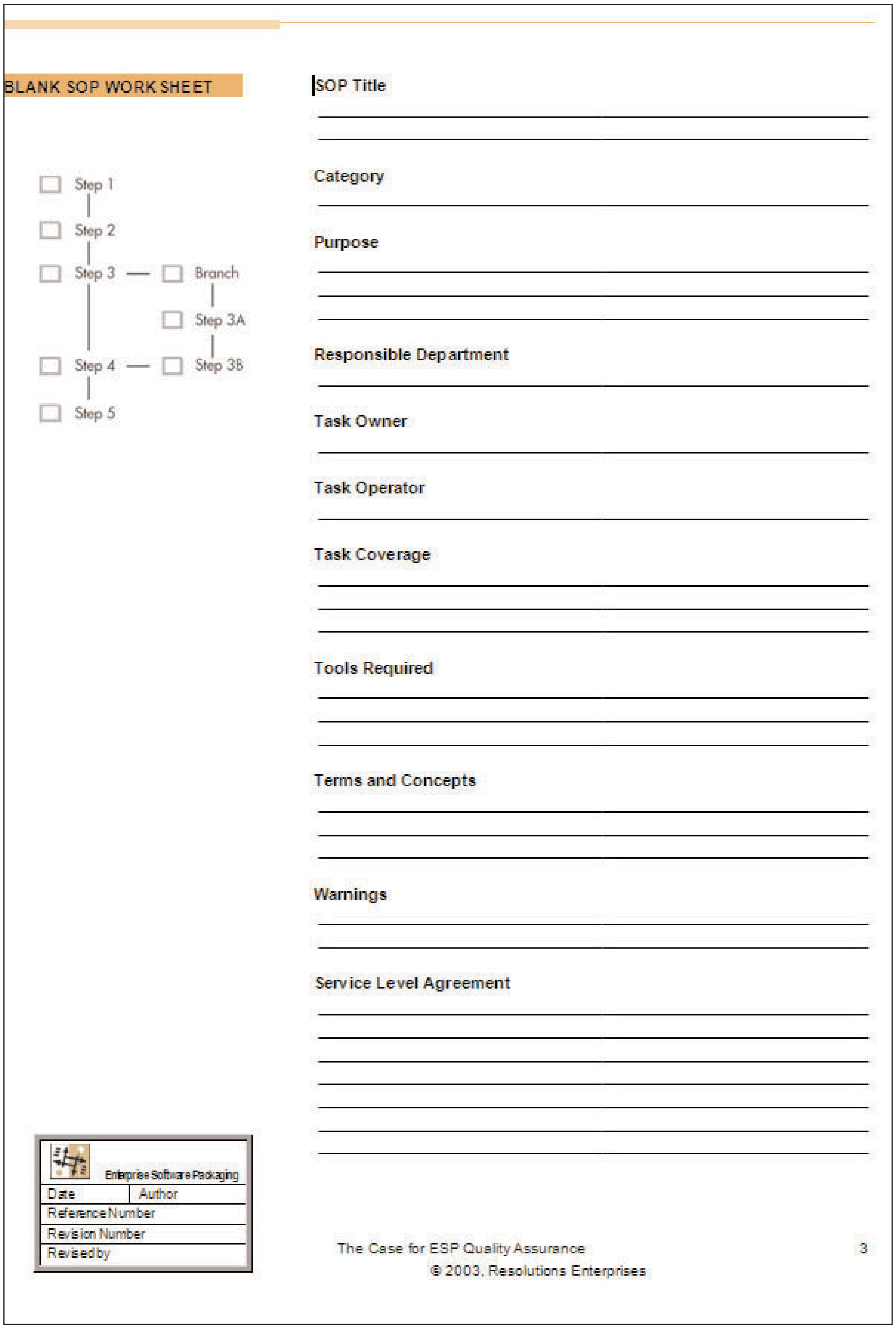
Steps to Perform
List and explain the process steps in sequential order.

- Remember to limit the number of steps (6 to 12 max.).
- If two steps must be done at once, explain them in a sentence that clearly says so.
- Provide a more detailed explanation if a reader needs more information to fully understand the reason for performing a step.
- Provide readers with alternative steps to take in case a desired step does not work.
- When a SOP is time-dependent, indicate the times clearly.
- When an SOP depends on informational input, include the source with reference document number and date, if possible.
- Decide where to use graphics to communicate clearly.
- List all references.
- Test the SOP in the field and then develop troubleshooting instructions.

Additional Comments
Provide any comments deemed necessary to increase comprehension by the Task Operators.

Enterprise Software Packaging	
Date	Author
Reference Number	
Revision Number	
Revised by	

The Case for ESP Quality Assurance 2
© 2003, Resolutions Enterprises



BLANK SOP WORK SHEET

☐ Step 1
☐ Step 2
☐ Step 3 — ☐ Branch
☐ Step 3A
☐ Step 4 — ☐ Step 3B
☐ Step 5

SOP Title

Category

Purpose

Responsible Department

Task Owner

Task Operator

Task Coverage

Tools Required

Terms and Concepts

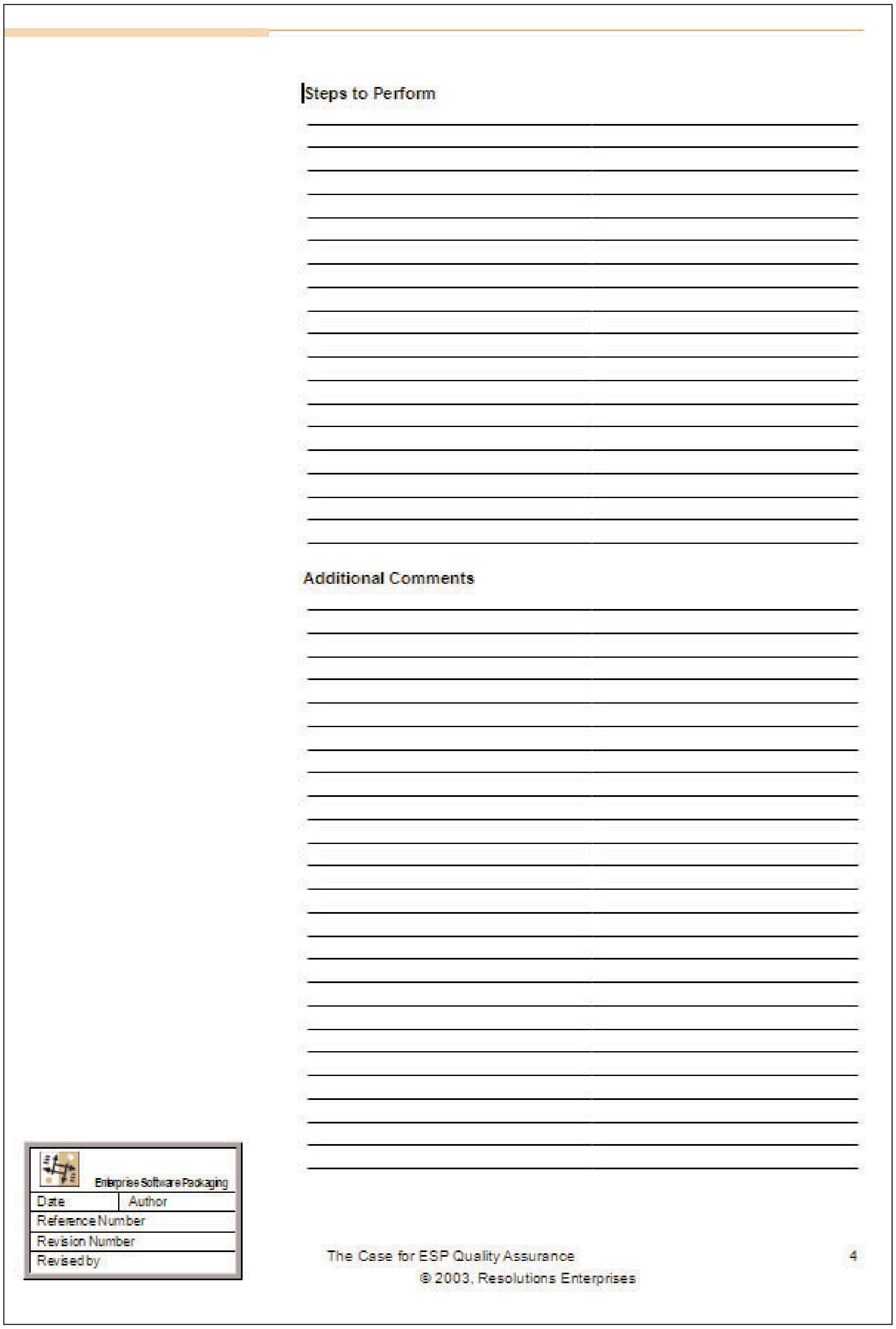
Warnings

Service Level Agreement

Enterprise Software Packaging	
Date	Author
Reference Number	
Revision Number	
Revised by	

The Case for ESP Quality Assurance 3

© 2003, Resolutions Enterprises



Steps to Perform

Additional Comments

Enterprise Software Packaging

Date	Author
Reference Number	
Revision Number	
Revised by	

The Case for ESP Quality Assurance 4

© 2003, Resolutions Enterprises

References

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

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

Windows Server 2003: Best Practices for Enterprise Deployments. Nelson Ruest & Danielle Ruest. McGraw-Hill Osborne. ISBN: 0-07-222343-X.

Repackaging Basics. Wise Solutions Inc.

Measuring the Returns of Software Packaging. Tim Wilson, Enterprise Management Associates, April 2003.

Desktop Management, Certifying Software Prior to Deployment. David Friedlander, Giga Information Group, September, 2003.