Chapter

11

Planning the Repackaging Project

Good planning is essential in many of life's endeavors, and application repackaging is no exception. This chapter discusses the various parts of planning a repackaging project. This information should help you in planning your own repackaging project. In particular this chapter focuses on those items that you need to consider when you are preparing your project procedure.

A good analogy to planning, for people who have ever flown an airplane, is the proper setting up of your approach to landing is critical to having a good landing. Good planning of your repackaging project can result in not having to redo parts of it thus saving both money and time.

The Planning Process

To create a good plan for repackaging a group of legacy applications, it is important to have a good grasp of the material in the first three parts of this book. Repackaging lends itself to the creation of a step process. A step process is where you can break the complete process into discrete sets of actions. You can then monitor the passage of each legacy application through each step and thus keep track of progress. Before discussing the creation of a step process for repackaging applications, consider the big picture first, as shown in Figure 11-1.

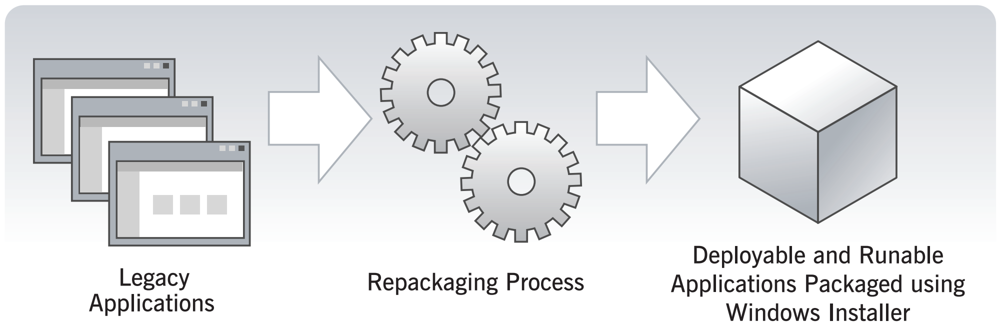
Legacy Applications
Repackaging Process
Deployable and Runable Applications Packaged using Windows Installer

Figure 11-1: *The repackaging big picture.*

In Figure 11-1, you see a simple diagram that is good to keep in mind when creating your plan. You input to the process is the group of legacy applications that you want to repackage and your output is to have these legacy applications repackaged using the Windows Installer. These repackaged applications have to be both deployable and runable in the end user's actual environment.

Before getting into the details of defining the steps in your process, you need to be aware of the benefits and the downside of repackaging. This is the subject of the next section.

The Pros and Cons of Repackaging

There are several good reasons to make the decision to repackage all your legacy applications to use Windows Installer. There are also a few disadvantages for repackaging legacy applications instead of creating native Windows Installer packages. The basic reasons for repackaging your legacy applications are listed below:

- Using a Windows Installer package allows for distribution of software to a locked-down system can dramatically reduce the total cost of ownership (TCO). This is because it is not required to have the administrator visit each machine in order to install the software using administrative privileges. Repackaging legacy applications instead of requiring native Windows Installer from software vendors allows for a much faster introduction of Windows Installer benefits into an organization.

- Windows Installer provides a minimal level of self-repair for applications when a file is accidentally deleted. A full capability for applications to self-repair themselves must be programmed in the application itself using APIs exposed by Windows Installer. Sometimes, this ability for an application to self-repair is termed application resiliency.

- Source resiliency is another feature of Windows Installer. It is possible to define multiple locations where the source files for an application are located. The purpose of being able to do this is so that an application that needs to be repaired or updated does not have to depend on one location that might be unavailable for some reason. This capability of Windows Installer allows the user of a desktop to be fully productive instead of being idle because the application they need cannot run correctly.

- Windows Installer provides a much more robust capability for preventing the uninstallation of one application from disabling another application. However, to ensure this functionality works as advertised, you need to follow the rules for components as described in Chapter 7.

There are also disadvantages in repackaging instead of creating native Windows Installer packages, many of which are the same as the above advantages if you do not perform the repackaging correctly.

- Typically, a repackaged legacy application consists of only one feature with all components under this one feature. In this situation, when the key file of one of these components is deleted, the complete application is reinstalled when the user tries to launch it from the shortcut. For a large application this may not be a good thing because of the delay that can occur while this reinstall occurs. Also, if you do not create a Windows Installer shortcut for your repackaged application, this self-repair will not occur at all. Also, if you do not use a Windows Installer shortcut for your repackaged application, you will not be able to assign this application using Group Policy. You will only be able to publish it.

- The incorrect reference counting of components in multiple applications that install the same file to the same location can cause major conflicts. This is especially true where the uninstallation of one of these applications can disable all the other applications that require the same common file. This can occur because different component codes are used for each of the components that install this same file. To avoid this situation, it is imperative for you to perform a conflict resolution between all the applications that are being repackaged.

- When you have target systems that consist of both Windows 9x machines and Windows NT-based machines, you are required to create two different packages, one for each type of system. One of the reasons for this is that there are often ANSI specific libraries required on a Windows 9x machine and Unicode libraries required on Windows NT-based machines. Also, the directory structure is different for such things as for profiles and the system folder on a Windows 9x machine versus a Windows NT-based system. A native Windows Installer package for an application can handle both types of systems in the same package.

- A native Windows Installer package allows run-time customization using transforms. The process of repackaging creates an installation package that is not amenable to run-time customization since the customization is essentially done when installing the legacy application during the capture of the changes being made to the target system.

Hopefully, the above list of pros and cons of repackaging can help in having realistic expectations from any repackaging project that you undertake. Having realistic expectations can go along way toward creating a valid project plan.

The Basics of Creating a Step Process

Breaking a process into a series of steps has some important advantages. These advantages are described in the following list:

- First and foremost, the discipline of breaking a process into its constituent steps forces you to think through how the work is going to actually be accomplished. This exercise can uncover unanticipated problems with your initial vision of how the job is to get done. You then have the opportunity to solve these issues before they become problems after the job has already started.

- Having a process broken into its constituent steps makes it easier for personnel to understand what to do, and it makes it easier to train the people to correctly implement the project.

- Once you have worked through the effort of breaking a process into its steps, you are now in a position to make a much more accurate estimate of the resources and schedule required to complete the project.

- During the implementation of the repackaging project, you can now easily track the actual performance against your original estimate. If necessary, you can go back and create a new estimate for the project based on actual results gained during the first experience of working the process.

The mental effort of creating a good step process can be substantial, but there are only a few rules that you need to use when defining a step:

- Steps must have a clear definition of when work in the step starts and ends.

- It needs to be possible to clearly estimate the resources required to complete the step and it also needs to be defined in such a manner so that it can easily be estimated how long the work in a step should take for any particular legacy application.

- Each step cannot have more than one entity responsible for completing the work defined by the step. This is necessary so that the responsibility for completing the step is clear and accountability can be assigned to either the success or failure of completing the step in an acceptable fashion.

The overriding rule that you need to follow when managing a step process is to work within the process and not break it. Often, the first impulse when a project runs into trouble is to take the expedient route and forget the process. This usually results in losing control over what is happening, leaving you unaware of the project progress and output quality. When your process is not working, you need to examine the original assumptions you used in creating the process. You can then you modify the process to correct the problem and retrain everyone in the changes that have been made.

The Project Procedure

It is very important that you create a project procedure before you begin a repackaging project. Preparing a project procedure has the major benefit of forcing you to think through the entire process before you actually begin work. Putting down in writing how you think you want to proceed allows not only you, but also reviewers, to identify possible problems that may occur.

A project procedure consists of a number of elements. These elements are described in the following list:

Statement of Purpose: Your statement of purpose needs to be clear on what you hope to achieve by performing the repackaging effort. Subjecting this statement of purpose to review is a good way to clear up misunderstandings about what is possible and what is not. The statement of purpose can also serve as a good executive summary for management so they do not have any expectations that are not possible to fulfill.

Definitions: It is important to define all the terms used in the project procedure. This makes it easier for people to read and understand the procedure. Understanding the procedure is especially important for the personnel who are assigned to carry it out.

Review & Approval: When a project procedure is created, it should be put through a review and approval process. It is a management responsibility to identify the preparer, reviewer, and approver of the project procedure. The review and approval process requires a procedure for the reviewer to transmit the comments to the person who prepared the project procedure. This same procedure needs to define how comments are to be resolved between the preparer and reviewer. The project procedure needs to be signed by the preparer, reviewer and approver. The main purpose of the approval signature on the project procedure is to verify that the review and comment process has taken place and that all comments have been resolved satisfactorily.

Change Management: It is very common occurrence to need to make changes in a process based on experience. All project procedures need to specify how to handle changes to a repackaging process and to make all affected personnel aware of the changes. Normally, a procedure will have a location where revisions are recorded along with the preparer, reviewer, and approver signatures.

Scope Definition: The description of the scope of a repackaging project needs to include, at minimum, the following:

- Names of all the legacy applications to be repackaged.
- The instructions on how to install each of the legacy applications.
- A description of the target environment to which the repackaged applications will be installed.
- A categorization of the difficulty level of each of the legacy applications.
- The method of deployment that is to be used for distributing the repackaged software to the organization.

The information in the scope description is used to generate a schedule and the number of chargeable hours that will be required to complete the project.

Repackaging Environment Specification: Based on the scope description of the repackaging project, you need to define the makeup of the lab that needs to be used to perform the work. You also need to define the base set of software installed on all target machines in the organization. The lab will need to have the hardware, operating systems, and base software available to adequately perform the project.

Process Design: Describing the process to be used to repackage the legacy applications detailed in the scope statement consists of the steps through which each legacy application will pass in order to turn it into a Windows Installer package. It is very likely that on a large project there will be more than one step-process based on the categorization of applications defined in the scope definition. For example, there may be one set of applications that target Windows NT and another set that targets Windows 98. An important part of the process design is to specify the type of testing that needs to occur before a legacy application is considered ready for deployment. The description of the process should always have a flowchart showing the steps in the process and their relationship to each other.

Training: To have a project that proceeds with a minimum of problems, it is necessary to provide proper training for all participants. This training not only needs to include training on the tools to be used to perform the repackaging, but also training on the process itself.

References: The project procedure needs to provide a list of pertinent references that participants in the project can go to obtain further information on a subject. Providing references can cut down on the length of the project procedure in that this information does not have to be included in the procedure itself.

The above list is, in a way, a checklist of the elements that make up a good project procedure. The focus of this chapter is on three of these elements: the scope definition, the repackaging environment specification, and the process design.

The Scope Definition

The scope definition is the most general of the technical sections of your project procedure. Each subsequent section in the project procedure increases in its technical detail. The information in this section deals with defining the legacy applications that are to be repackaged during the project. The scope definition also needs to set the general requirements for implementing the project such as the type and frequency of status reporting that is necessary. The information provided in the scope definition can be thought of as a prerequisite for creating the repackaging environment specification and the process design. The scope definition provides all the information required to turn a legacy application into a Windows Installer package. It does not address the hardware requirements for performing the repackaging effort.

In the following subsections the various elements that make up the scope definition are detailed.

Legacy Application Description

Every legacy application that is to be repackaged needs to be listed. For each application in the list the following information needs to be supplied:

- Identify the purpose and use for each legacy application and who the users of the application will be. For example, some applications are to be deployed to everyone in the organization and other applications are only for particular organizational units.
- Document the installation instructions that need to be followed for each legacy installation. This documentation should include screen shots of each dialog in the installation user interface showing each option that is to be selected.
- Document all necessary pre-installation and post-installation steps required for the proper installation of the legacy application.
- Identify any passwords or serial numbers required to install the legacy applications.
- Define all pre-conditions required for the installation of each legacy application. This includes identifying any other applications, such as Internet Explorer, that need to be installed prior to installing the legacy application.
- Identify any special technologies that are being installed by the legacy application, such as ODBC, kernel drivers, file system drivers, virtual printer drivers, etc. This information is necessary to estimate the cost and schedule for the project. It also helps in categorizing the applications for possible different processes.

- Where the properties that appear in Add/Remove Programs are specific to each application, the value of these properties needs to be specified for each legacy application. Examples of such properties are the name of the application, the name of the application vendor, and the support URL for the vendor.

- Identify the specific entries for the Summary Information Stream that are to be used for each application. This information normally includes the name of the application, the software vendor, and the supported languages.

- Provide instructions on how to test the functionality of the application after it is installed. This is necessary to properly test the repackaged application.

There can be many generic properties that are used across all the repackaged applications. These generic properties can be specified in a separate section so as not to have to repeat them for each application in the list.

Based on the above information supplied for each application, you assign the applications into categories. The type of categories that you define is up to you, but a common practice is to create categories for low, medium, and high difficulty. This then permits you to make accurate estimates of time and cost for performing the repackaging effort.

The Generic Application Configuration Specification

In this section of your scope definition you define any generic parameters that apply to all the legacy applications that will be repackaged. The typical generic configuration items consist of the following:

- A definition of the build format in which the Windows Installer package is to be produced. There three possible configurations: all source files in an uncompressed state, all source files compressed into one or more cabinet files, and a combination of some source files in an uncompressed format and some compressed in cabinet files. It is recommended that all application source files be compressed into a cabinet file unless there is some overriding requirement to do otherwise.

- Define the target hardware and the operating systems that are in use in the organization. When specifying the operating systems, you need to include, at a minimum, the service packs required and the version of Internet Explorer that should be used. It is entirely possible that some applications will only be installed on a single operating system. Some other applications may need to install on the full range of operating systems that are in use in the organization.

- Define the deployment method(s) that will be used. This definition needs to specify any command-line arguments that might have to be used.

- Set the value of Windows Installer properties so Add/Remove Programs is configured the same for all repackaged applications. You can define the name of the registered user and registered company to be constant across all repackaged applications. This information will appear in the repair dialog in Add/Remove Programs. You may also want to define an internal help desk phone number where users can get help. You can also configure Add/Remove Programs so that one or all of the Remove, Change, or Repair buttons are disabled.

- Define the installation user interface sequence that is to be used whenever an application is installed with a full user interface. The definition of the user interface can be defined to include all dialogs, or it can be configured so that it will only run with a progress bar that shows the status of the installation.

- Define whether each application is to be installed for all users of the machine or just for specific users. This same setting is something that may not apply across the board for all applications, but you will only have an application installed for the current user or for all users.

- Define the standard customizations that are to be applied to the Windows Installer package. These standard customizations can consist of such items as adding custom actions, adding logging functionality, component isolation, and the handling of desktop shortcuts.

- Define the creation and use of merge modules. Normally it is recommended to only use the merge modules created by Microsoft, InstallShield, or some other third party. The experience of the InstallShield Consulting Department has shown that creating merge modules during a repackaging project can lead to problems.

- Define the makeup of the exclusion list to be used for repackaging all applications. The name of the default exclusion list that comes with AdminStudio is isrepackager.ini.

The generic configuration rules defined in this section do not need to cover all the legacy applications. It is possible to have certain generic configuration rules that apply to a subset of the total scope of applications being repackaged.

The Testing Requirements

After a legacy application is repackaged, it needs to be extensively tested. The following items should be considered for any testing requirements that you create.

- When you build the Windows Installer package using Developer, it will display any build warnings and errors that occurred during the build. The first thing that you should do is to eliminate all build warnings and errors before you proceed to any other testing.

- The next thing you should do is run an internal consistency evaluation (ICE) check on a Windows Installer package once it builds without warnings or errors. There are three sets of internal consistency evaluator (.cub) files that are provided by Microsoft in the Windows Installer SDK. There is also a CUB file that is provided for evaluating merge modules, but this is only necessary if you are creating your own merge modules. If you have special testing requirements, you can create your own custom internal consistency evaluators and include them in your testing at this point. You should resolve all errors that arise from running the internal consistency evaluators, and you should also resolve all warnings except those created by ICE33 and ICE36.

- Perform an installation of the repackaged application on to a clean machine. If there are any initialization or run-time errors that occur, you need to completely resolve these.

- When the repackaged application installs without error on the clean test machine, you need to test the uninstallation of the application from the same machine. You need to resolve all initialization and run-time errors that occur during the uninstallation. Even if the application does not require a reboot of the system, you need to reboot in any case to verify the operating system integrity. You should also exercise each of the items in the Control Panel as a further check that the uninstallation did not remove operating system files.

- As soon as you have a package that installs and uninstalls without error on a clean machine, you need to test the functionality of the installed application. This will normally include a check of the functionality of all shortcuts, the functionality of all file associations, and the running of the entire set of main menu options to see if the application responds in an appropriate manner.

- Perform a per-machine advertised install and then see if launching the application from the advertised shortcut installs the application and then runs the application. You should also test to see if double-clicking on a registered file type will also launch the application and load the file. You then want to test the functionality of the application as discussed in the previous item.

- Perform a per-user advertised install and test the launching of the application as described in the previous item. Also perform the functionality test of the application after it has been installed and launched.

- Test that the uninstallation of a repackaged application completely removes all application related files from the machine. You can perform this test by running Repackager before you install the application and after you uninstall the application. Between the installation and the uninstallation, you should exercise every shortcut created by the installation. If you find that there are files created by running the installed application, you should modify the install package in order to have these files, folders, or registry keys removed during an uninstallation.

- Test that an installation that is performed for all users of the machine will allow different users of the machine to run the application. To test this, you need to install the application for all users and then sign on with at least two different user accounts and verify that the application is available for both users.

- If deployment is to use Group Policy, you need to test the application installation for both assignment and publishing to computers and users. The client machine should be a clean machine. For an assigned application, you need to verify that the application can be installed and run by activating the assigned shortcut, or by trying to load a registered file type. For both assigned and published applications, you need to verify that you can install the application from Add/Remove Programs in the Control Panel.

- Test the application in the actual environment that will be used in the organization. This may uncover areas where the application cannot run correctly due to a lack of the proper access rights to files, folders, or registry keys.

- Import your Windows Installer package into the ConflictSolver database and remedy any conflicts that are found. Depending on the conflicts that are found and fixed, you may need to run more tests on the package as described in the previous items. Depending on your special circumstances you may need to create your own custom application conflict evaluators (ACEs).

The above list of test scenarios provides you information on many of the types of tests that are possible to run. Depending on your own circumstances, you may not need to run all of these tests or you may need to run additional special test scenarios.

Reporting Requirements

The final item in your scope definition is to specify the reporting requirements necessary to track the progress of the repackaging project. You can make these requirements as detailed as you need to properly give you control over the project. The following minimum requirements are recommended:

- Based on the categorization of the difficulty and the priority assigned to each legacy application, you need to create an estimated schedule for each application. You will track actual performance against this estimated schedule.

- During the implementation of the project, you should track the time that each application enters and leaves each step of the process. You can then plot a curve of the quantities that are actually moving through each step against the planned or estimated number of applications that should be moving through each step.

- You should report any special difficulties that are arising with any particular application so that special attention can be applied to solving the problems.

- Based on actual experience, you should create a new estimate for the schedule if actual experience proves that it takes longer than originally estimated to complete the work in any step. You probably do not have to create a new estimate if actual performance is less than 10% greater than what was originally estimated.

If you have a means to capture the hours used in each step, you can track the hours actually spent against the hours that were originally estimated for performing the project. It also may be a good practice to repackage several legacy applications of differing difficulty in order to get a good estimate of the time and cost of performing the project.

The Repackaging Environment Specification

In this section of your project procedure, you will specify the hardware requirements and the hardware configuration for implementing the project. You will also define the software that is necessary to perform all the steps in the process.

Hardware Requirements

The actual target hardware is specified in the scope definition section of your project procedure. Here you want to define in detail the makeup of the lab that will be used to implement the repackaging activities. When specifying these requirements you want to consider the following items:

- The network configuration of the lab to include that servers required and the workstation setup for each desktop engineer that will be performing repackaging tasks. Typically, each desktop engineer will have a development machine and a test machine. In addition, there will also be one server functioning as a database server. If you will be testing Group Policy based deployment, you will need an additional server functioning as the domain controller.

- You need to specify the size of the test partition on the test machines in the lab. Normally you will want to have a test partition that is kept to a minimum so that recreating the partition can be done quickly.

- If there are any special hardware configurations that affect only a few of the repackaged applications, you need to identify how these special needs are to be met. Typically, you would set up one or more machines with these special hardware configurations, and use them for repackaging the applicable legacy applications.

- You need to specify the directory structure that will be used to hold the initial legacy applications and also the structure that is to hold the final repackaged applications. You may want to have a directory structure that provides the repackaged applications before they are tested as well as after they have passed all testing.

The extent of your lab setup for repackaging will be determined by how many desktop engineers will be working on the project. The number of desktop engineers is normally dictated by the schedule that must be met for the completion of the project.

Software Requirements

The hardware requirements are normally fairly simple, but the software requirements can be more complex. The software requirements span the tools that are required to implement the actual repackaging, test the repackaged application, and control and report on the status of the project itself. The following list shows many of the items that you need to consider when generating the software requirements section of your project procedure.

- You need to specify the version of AdminStudio that is to be used by all desktop engineers for the repackaging project. You also need to specify whether you are going to run the AdminStudio tools from a server or whether AdminStudio is to be installed on each development machine in the lab.

- Specify the location that is to be used for creating Developer projects, the location where merge modules are to be stored, and the location to be used for the output files that are copied during the capture of the install image of the legacy application.

- You need to determine which imaging software you are going to use to create a clean machine quickly and easily. Examples of this type of software are Ghost by Symantec Corporation, Drive Image by PowerQuest Corporation, and VMware Workstation, by VMware Inc.

- You might want to specify that the Orca database-editing tool be installed on each of the development workstations. This allows the desktop engineer to easily open up a Windows Installer database, and it also provides an alternative approach for performing an internal consistency evaluation of the database that is opened. The Orca tool is available from the Windows Installer SDK as described in Chapter 4.

- You need to specify whether the project is gong to use Microsoft Access as the database for maintaining workflows and for performing conflict analysis or whether you are going to use Microsoft SQL Server for this purpose. Generally, if you are going to have more than one person doing the repackaging, you will want to use Microsoft SQL Server as your database management system (DBMS). You will also need to specify the access rights that each involved person will have to the ConflictSolver database. You will normally only want one person to have the rights to import a Windows Installer package, but you will need to provide all desktop engineers the right to perform conflict resolution.

- You need to specify the configuration of the project template or templates that are to be used for creating Developer projects. As part of this specification, you need to define the location where these templates are to be stored for use by all desktop engineers.

- The default exclusion list to be used for repackaging all legacy applications needs to be specified.

- The configuration of the test machine on which you are to take the OS Snapshot needs to be specified. Normally, you should take the snapshot of the base operating system without any other software installed. It is possible to install on the test machine the base set of applications that all users in the organization will have before taking the OS Snapshot, but this is not recommended.

- If you are going to use any particular software for scheduling and controlling the repackaging project, you should specify that here along with the responsibilities for running this software.

This list provides you with some good guidelines for determining the software requirements of a standard repackaging project. You may, however, have different requirements based on your specific situation.

The Process Design

In this section, you need to define the steps that each of the legacy applications will pass through to turn them into deployable Windows Installer packages. The generic operations that have already been discussed such as creating the OS Snapshot, creating the Developer project templates, etc., should not appear in this section. This is because they are performed only once and not for each legacy application.

Figure 11-2 is a simple example of a process that shows the type of information that needs to be documented in this section of the project procedure. The steps in this process are described in the following list:

Step 1: This first step is to capture the changes that are made to a system by installing the legacy application on a clean system. This normally consists of taking a snapshot prior to installing the legacy application, installing the legacy application, and then taking another snapshot of the system. The difference between the first snapshot and the second snapshot is captured in a text file that has an .inc extension.

Step 2: This step is the start of the actions that are required to turn the system changes documented in the .inc file into a Windows Installer package. Developer is used to convert the information in an .inc file into a project file that has an .ism extension. When the project is created, it is necessary to review the contents of this file and remove any extraneous items such as files, folders, and registry entries. You also make application specific customizations in the project file such as the name of the application that will appear in Add/Remove Programs.

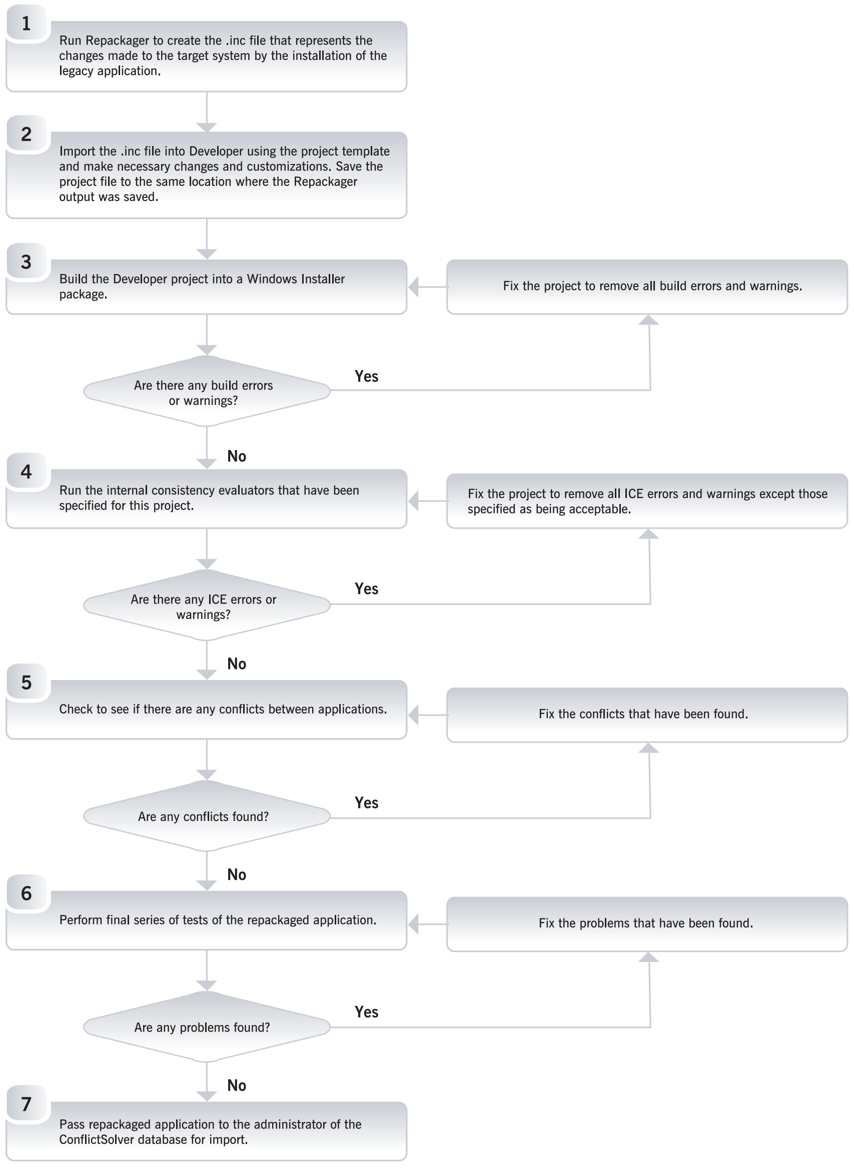
1
Run Repackager to create the .inc file that represents the changes made to the target system by the installation of the legacy application.
2
Import the .inc file into Developer using the project template and make necessary changes and customizations. Save the project file to the same location where the Repackager output was saved.
3
Build the Developer project into a Windows Installer package.
Fix the project to remove all build errors and warnings.
Are there any build errors or warnings?
Yes
No
4
Run the internal consistency evaluators that have been specified for this project.
Fix the project to remove all ICE errors and warnings except those specified as being acceptable.
Are there any ICE errors or warnings?
Yes
No
5
Check to see if there are any conflicts between applications.
Fix the conflicts that have been found.
Are any conflicts found?
Yes
No
6
Perform final series of tests of the repackaged application.
Fix the problems that have been found.
Are any problems found?
Yes
No
7
Pass repackaged application to the administrator of the ConflictSolver database for import.

Figure 11-2: *Diagram showing a simple repackaging process.*

Step 3: After you made all the necessary changes to the project file, you need to make the first build of the Windows Installer package. If there are any errors or warnings generated during the build, you need to correct these by making changes in the project file. This becomes an iterative process until you achieve an error and warning free build.

Step 4: The first test that you run on the Windows Installer package is a validation of the internal consistency of the database entries. Except for the specified internal consistency evaluators (ICEs), you need to fix all warnings and errors using an iterative process as shown in Figure 11-2.

Step 5: After all ICEs have been run successfully, you need to evaluate any conflicts that might exist between this Windows Installer package and the packages that have already been imported into the ConflictSolver database. If there are conflicts that need to be resolved, you need to start an iterative process to resolve these conflicts.

Step 6: Here you perform the final series of tests, which should include deployment and application functionality tests. The correction of any problems is also an iterative approach.

Step 7: The database administrator imports the Windows Installer package into the ConflictSolver database after the package has passed all tests successfully. The Windows Installer package is also made available for deployment in the organization.

In Figure 11-2, the iterative process of solving problems is simplified. In the actual performance of a repackaging project, you will be running back through all tests that occur prior to the existence of a problem. For example, if you have to make modifications to a project in order to fix an ICE error, you may end up with a build error that needs to be fixed.

Conclusion

The guidelines in this chapter are very general and are not meant to provide anything but a list of items that need to be considered when planning a repackaging project. The important thing to learn from this chapter is the importance of good planning so you have a successful project. It is important to be flexible and realize that any initial plan you make may not hold up under actual experience. In this case, you need to modify the project procedure to account for this real-world experience. Of course, it will be important to train everyone involved on any changes that you make in the project procedure. Planning is not an easy thing to do, but it is well worth the effort.