CM Live!

Configuration and Change Management: Taming Change and Complexity with Telelogic Synergy

Release 6.6a

Before using this information, be sure to read the general information under Appendix C:, "Notices" on page 83.

This edition applies to VERSION 6.6a, Telelogic Synergy (product number 5724V66) and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

1 Introduction

This document guides you through the steps necessary to get your project teams up and running on Telelogic® Synergy™ as fast as possible.

Telelogic Synergy is used by hundreds of project teams around the world to implement configuration management (CM). Experience has shown that when new sites follow specific key steps, the time to adopt Telelogic Synergy is short and well spent.

This document contains the critical information necessary for you to perform the implementation yourself, but it is not intended to be comprehensive.

Scope

The primary focus of this document is the implementation of configuration management with Telelogic Synergy. Telelogic® Change™, our change tracking product, is referenced where relevant because change requests can be stored in the same database as Telelogic Synergy data.

In addition to Telelogic Synergy for configuration management, the following suite of development tools is offered:

- Telelogic Change for change request management
- Telelogic® DOORS® for requirements management
- Telelogic® Rhapsody for embedded systems and software engineering
- Telelogic® TAU for analyzing, specifying, modeling, and testing advanced systems
- Telelogic® Focal Point™ for product decision making
- Telelogic® System Architect® for tools necessary for development of successful enterprise systems
- Telelogic® Logiscope™ for ensuring quality assurance in software development projects
- Telelogic® DocExpress® for organizing data in numerous formats that can be tailored for your specific archival and reporting requirements

Implementation success

CM Live! tells you and your CM implementation team how to successfully adopt and use the Telelogic Synergy solution.

Use a staged approach to implementing Telelogic Synergy; that is, start with the fundamentals and work towards the application's more sophisticated capabilities through iterative extension.

Wherever possible, *CM Live!* describes the simplest approach to ensure clarity and enable CM implementation teams to benefit from the basic capabilities of Telelogic Synergy.

CM Live! lays out a comprehensive process for implementing Telelogic Synergy and follows principles relevant to any technology adoption effort. The content represents the collective experience of consultants, application engineers, and support team, and it includes the feedback of our customers and partners. Therefore, the process represents the best practices developed over years of implementing CM solutions.

Terms and concepts

Before proceeding, learn the following Telelogic Synergy terms and concepts so that you understand the information in this document.

Term	Definition
Application	An *application* is the software to be loaded into Telelogic Synergy. An application typically consists of a collection of source code files, graphic images, executables, libraries, test cases, help files, and other files included in the software system that you deliver.
Baseline	A *baseline* is a snapshot of a set of projects usually created at a milestone. A set of projects for a particular release and purpose bases its members on a baseline. This means that each project looks at the baseline to find its own baseline project. If a project uses update templates, the update template identifies which baseline will be used. A baseline must include the full set of projects to represent the full product (at least within one database).
Database	A Telelogic Synergy *database* is a repository for the objects that make up an application. An application may be stored in multiple Telelogic Synergy databases or a single Telelogic Synergy database. A single Telelogic Synergy database may contain multiple applications.

Term	Definition
Database topology	*Database topology* refers to the mapping of applications to Telelogic Synergy databases.
Migration	*Migration* is the process you use to load existing directories and files from your file system (or another source control product, such as subversion) into Telelogic Synergy for the first time.
Product	A *product* is a file produced by a build process; for example, `.class` file, `.jar` file, or `.exe` file. Products are sometimes known as derived objects in other CM tools.
Project	A *project* is an arbitrary file and directory hierarchy that represents some piece of the application. An application can be stored in a single project or in many projects. A project often contains a component, such as the code for a single library or executable. *Project topology* refers to the mapping of an application to a Telelogic Synergy project.
Release	A *release* is a specific version of a software application or a particular branch of code development. For example, your organization just delivered version 1.0 of your application. Now, development is progressing in parallel for both releases 1.1 and 2.0 simultaneously. In this situation, there are two parallel releases under development.

User interfaces

Telelogic Synergy provides the following user interfaces:

- Telelogic Synergy—a graphical interface that provides more advanced CM capabilities for developers. Build management capabilities are also included in this interface. This interface is available on Windows and UNIX.
- Telelogic Synergy Classic— is an advanced interface for build managers and CM administrators. This interface is available on Windows and UNIX.

  Telelogic Synergy Classic is no longer being enhanced. In a future release, when Telelogic Synergy has sufficient functionality to replace Telelogic Synergy Classic features, Telelogic Synergy Classic will not be provided.

- A Telelogic Synergy terminal-based command line interface (CLI) is available on Windows and UNIX. The CLI is an option for any user who prefers the command line or wants to automate Telelogic Synergy operations.

Developers normally use Telelogic Synergy. Team leads and build managers use Telelogic Synergy. Users responsible for administration, migration, or distributed change management (DCM), use Telelogic Synergy Classic for these operations.

This document contains references to specific dialog boxes and commands. As some features required by the implementation process are available only in the Telelogic Synergy Classic interface, this document mainly references Telelogic Synergy Classic dialog box names. Some terminology changes were implemented in the Telelogic Synergy interface, so these differences are noted as necessary.

Adoption road map

The following items link to the different operations you will need to prepare for and implement the CM methodology for your team.

Planning

- "Implementation strategy" on page 9
- "Select a database topology" on page 11
- "Allocate system resources" on page 17

Deployment

- "Set up the database servers" on page 23
- "Install Telelogic Synergy" on page 25
- "About license data" on page 24
- "Configurations" on page 25
- "Establish the project topology" on page 26
- "External and installation projects" on page 44
- "Configure Telelogic Synergy" on page 48
- "Migrate the application software" on page 55
- "Build products" on page 63
- "Verify and test the application" on page 63
- "Set up releases" on page 65
- "Perform load testing" on page 70

Usage

- "Timing the training" on page 73
- "Go live!" on page 76

Conventions

The following describes the conventions used in this document.

Typeface	Description
Italic	Used for book titles and terminology. Also designates names of roles (*developer*), states (*working*), groups (*ccm_root*), and users (*mary*).
Bold	Used for items that you can select, such as buttons, icons, etc., and menu paths. Also used for the names of dialog boxes, dialog box options, toolbars, folders, baselines, databases, releases, properties, and types. Also used for emphasis.
`Courier`	Used for commands, filenames, and directory paths. Represents command syntax to be entered verbatim. Signifies computer output that displays on-screen.
`Courier Italic`	Represents values in a command string that you supply. For example, (*`drive`*:*`username`**`commands`*).

This document also uses the following conventions:

Note A note contains important information that should not be overlooked.

Caution A caution indicates potential danger to the database, the database server, or some other integral piece of the Telelogic Synergy software or your system.

Command line interface

Prompt

This document uses the `$` (dollar) character as the shell prompt for UNIX examples.

Option delimiter

Both UNIX and WIndows clients support the dash (-) option delimiter. This document shows UNIX client examples.

Standards

Instructions for editing text files are given using Notepad (Windows clients) or `vi` (UNIX clients) commands. Notepad (Windows clients) and `vi` (UNIX clients) are the Telelogic Synergy default text editors. If you use a different text editor, substitute the appropriate commands.

2 Planning

This section provides information to help you plan for a successful Telelogic Synergy implementation. If you carefully plan the implementation, you will be able to install and configure Telelogic Synergy quickly and make it ready for successful production use by your team.

The remainder of this document describes the specific planning tasks that are essential for preparing a project team for production use. Review the full set of steps, using the "Adoption road map" on page 4, and schedule an appropriate amount of time for your team to complete each step.

The need for planning

Sometimes an organization purchases a product such as Telelogic Synergy and is tempted to rush through installation to use it immediately. Planning can make a tremendous difference in your satisfaction with the end result. While it's true that after you have implemented the product, you can change your database or project topology, or reallocate which processes run on which hardware, sound planning from the beginning usually obviates the need for later changes. Everyone benefits when a tool performs optimally from the start.

To plan an effective roll-out, key stakeholders must understand the technology they are about to adopt. If stakeholders have not been a part of product demos or training, it might be difficult for them to commit their time and effort to implementation planning. Be sure to include all key stakeholders in training courses so they see what they're committing to, have the opportunity to ask questions, and are ready to help with the planning. Similarly, the implementation effort depends upon your environment, which has to be well understood before beginning implementation.

Manage the scope

The planning effort required for implementing Telelogic Synergy is proportionate to the scope of the anticipated roll-out. Depending upon whether the roll-out is a single project or an entire enterprise, different management structures need to be followed.

CM Live! addresses the issues that every CM implementation team must consider as it adopts Telelogic Synergy. Larger teams or enterprise roll-outs might need to create an implementation program consisting of different projects that each follow the guidelines in this document.

Customers may want to start with a modest implementation that adapts Telelogic Synergy's out-of-the-box capabilities. The customer can develop an understanding of the product, and identify and fix problems in their early stages. Telelogic Synergy then can be extended through the organization by the iterative application of the implementation process.

Product knowledge

The people responsible for planning and executing the implementation project must be familiar with the aspects of Telelogic Synergy that influence the plan.

The CM implementation team that is responsible for the roll-out needs to be trained in the basic concepts of Telelogic Synergy, its use, and its administration. The training is an integral part of the planning process.

Planners do not need to become experts in using the product to successfully implement it. This document provides sufficient information to enable you to recognize the issues to address during the implementation.

Gather information

Customers use different programming languages, development tools, processes and architectures; they use their own unique environments. Local conditions affect the implementation of Telelogic Synergy, so you must understand your local environment and its effects in the planning process.

The "Implementation questionnaire" on page 79 identifies the information needed from each project team that will use Telelogic Synergy. You can use the questionnaire as a tool to help gather the necessary information that can help determine the resources and effort required for adopting Telelogic Synergy. The questionnaire also helps in making decisions about the appropriate configuration of the requirements for each team.

For example, when planning the implementation for a team, decide which Telelogic Synergy user interface is most appropriate for each team member to use. This decision helps you plan the team's training requirements, and because different clients have different memory requirements, the decision might affect your hardware requirements as well.

Time the cut-over

Pay particular attention to the cut-over point. This is when you train your end users, especially developers and testers, and switch into the production environment. The timing is important to how developers perceive the transition

to Telelogic Synergy. If the transition occurs during a product release, where developers suddenly need to use a new tool with a different methodology to complete work under pressure of deadlines, they will likely be resentful of this intrusion. Ideally, the conversion should happen at the start of a new project cycle. If this is not possible, try converting immediately after a project milestone.

Implementation strategy

The following guidelines contribute to the successful implementation when rolling out Telelogic Synergy across a large organization, or even across several product teams.

- Phase the implementation across teams and products.

  Do not try to roll out a new technology to all teams at once, or even in a short time frame. The most effective technique is to start with a single team and schedule other teams at intervals after that. As a team goes live, many day-to-day questions will arise that will require your attention. By scheduling implementations at intervals, you're not overloaded with questions from multiple teams. In addition, as you roll out Telelogic Synergy to the first team, you learn valuable lessons that help subsequent implementations be more efficient and effective.

- Start with a team that has a high probability of success. For example, start with a team that has some of the following characteristics.

  - The team recognizes the benefits of CM. For example, the team should not be at CMM Level 1 maturity.
  - The team has simple requirements, such as a simple product or process. For example, work with a small team, a team working on a product with limited platforms, or a team working on a product with limited dependencies on other teams.
  - Team members understand how their software application is configured and built, and they are willing to provide information and assistance.

This strategy might appear to delay resolving key issues your organization had for choosing an advanced CM solution. However, a successful first implementation benefits the organization in many ways, both tangible and intangible. The implementation and roll-out team gains valuable experience that helps to make other teams successful, even if they have more complex issues. The organization gains confidence that the decision was good and the solution works for them.

The role of management support: the sponsor

Management support is critical to the successful implementation of any CM system. The fact that management approved the purchase of a configuration management system is already a strong indicator of management support for improved tools and processes. Nevertheless, the purchase of the product is only a part of the investment required. Additionally, there may be a need to invest in server hardware, time to set up the system and load legacy software, train users, and perform ongoing administration.

The role of the CM implementation team

The personnel assigned to manage and perform the implementation are the CM implementation team. They should include representatives such as managers, developers, and CM experts who have a firm grasp of the company's software, current CM procedures, development environment, and CM improvement goals. Often this knowledge is found in software project leaders or senior developers. When these people are not the ones doing the implementation, it can be very important that access to them is available.

The role of the build manager: the configuration authority

Every project team needs someone to play the role of configuration authority, known as the build manager. The CM health of the project team can be measured by the number of successful builds conducted per agreed unit of time. A healthy team is able to have frequent, successful, repeatable builds. If the build frequency is the "pulse" of the team, then the build manager is the "heart" of the team. Successful software teams nearly always have build managers who are highly technical.

The build manager need not be a person dedicated only to that task; it is often a role that one or more of the project team members play. Whether or not the role is full-time, ensure that the person is trained in this role and has sufficient time allocated to perform the daily build management tasks that are so critical to an effectively functioning project team.

The role of developers: the end-users

Most software teams include people in many roles, although the majority of the team members are usually software developers. All potential end-users need to have some awareness and training on the importance of sound CM practices, so that they understand the decision to adopt a sophisticated team solution.

Software developers enjoy the benefits of many features of a configuration management system. However, developers might need to use only a small subset

of the product's features to do their jobs. While developers will recognize the full richness of the commands and operations available to them, they might mistakenly believe that they must learn all of these operations, when actually most of the operations are for release management and advanced use. Training helps developers focus on the Telelogic Synergy set of operations they'll need to complete their work.

The role of system administration

As part of the Telelogic Synergy installation and set-up, the CM implementation team provides access to a variety of machines, through a variety of networks. Security considerations and firewalls might need to be accommodated. Remote developers might need to access the system through a Virtual Private Network (VPN).

All of these issues involve your System Administrators in planning for the installation, configuration, and maintenance of the product. They also need to be present or provide the necessary access to undertake key aspects of the implementation.

Select a database topology

A database topology is the mapping of each application to its controlled objects in the database. The database topology that you choose for an application impacts performance and usability, especially for large teams.

Perhaps the most important decision you must make regarding implementation is what database topology to use. However, defining a database topology is not a precise science; a number of key factors have to be weighed to determine the most appropriate topology for a site.

Usually, a team has several valid and viable topologies available, and the preferred topology is a function of the team size, the organization, and distribution, the processes it uses, and the size and structure of the application. Strict compliance with the information in this section is critical because a poorly chosen topology eventually leads to problems.

A Telelogic Synergy database stores project data, such as source files, products, tests, documentation, change requests, tasks, and so on. A Telelogic Synergy database includes archive and delta information for the project files under control, a file cache that contains files currently being accessed, and a relational database containing metadata, such as owner and creation time. The physical layout of a Telelogic Synergy database is described later; this section discusses how many Telelogic Synergy databases you'll need to create, and what should go into each database.

The factors in the following sections are important to consider in determining an appropriate database topology. The topics are presented in no particular order of importance; your team will establish which are the most important to consider.

Team organization structure

Many topology designs start naturally, as a mirror of the current project team structures. Teams commonly organize around the project data. For example, the supplier of a client-server financial system may have a GUI team, a DBMS-server team, a business rules team, and a team responsible for shared and reusable code. Each team is probably responsible for a different area of the code, and often some code and people overlap a little. Each team would have a project in the database, and the build manager would integrate and build those projects into one application.

Therefore, one way to select a database topology is based on the natural structuring that is present in a team's organization or an application's organization. Keep in mind, however, that even if the team structure is a clear and obvious match for the database topology, there are several other factors that may cause this structure to be modified.

Team and database size

Hardware configurations have practical limits to the number of people who can access a repository. These limits are usually more of a concern with lower-end hardware, because higher-end hardware can support hundreds of users simultaneously on a single Telelogic Synergy database. Except for very large teams, the number of simultaneous users on a Telelogic Synergy database is usually not a determining factor in the database topology. For specific hardware requirements, see the *Telelogic Synergy Installation Guide* for the platform you are using.

If you use a single large database with several hundred developers working on it, the database size becomes a factor when it reaches 5 GB or larger; consider topologies that allow the data and users to be spread over multiple repositories.

The primary issue with very large databases is backup and administration. Telelogic Synergy's default backup utility creates a portable archive file that contains all data in the database, but large databases will reach a point where it is no longer practical to back them up in this manner; you can use alternative backup strategies. For example, you can use the underlying RDBMS's backup in combination with a file system backup.

Additionally, with so many users in one database, the actions of one user or team can adversely affect another team. For example, if one team accidentally deleted some data and wanted to restore it from the previous night's backup, all data in the database would need to be restored, causing other teams' data to be rolled back as well. Splitting data into multiple databases results in databases that are more manageable, in terms of administration.

You may want to divide databases according to work group and product architecture. In most cases, development groups are organized into teams with specific responsibilities for different parts of an application. One way to organize the data is for each development team to have its own components developed by that team. Teams may need components produced by other teams within their development group. If so, they can use DCM to transfer components from one team to another.

One example of an application that requires a large development effort is an operating system. Some system vendors have several hundred developers working on a single release of an operating system. In such cases, it is important to devise a topology that allows the database size to be manageable and the number of simultaneous users to be within bounds of the available hardware resources.

Note Worksheets for determining the maximum number of simultaneous users for a specific hardware configuration and

the optimal hardware configuration for a specific number of simultaneous users are available in the *Telelogic Synergy Installation Guide* for the platform you are using.

Team location

Many development teams include developers, testers, and writers who are geographically dispersed. For most geographically dispersed teams, each physical site normally works on different areas of the software. A database topology may fall out naturally based on the different groups at the different sites.

Increasingly, teams are dispersed geographically where people in different countries are working on the same data. In this case, there are 3 ways to ensure productivity.

- Remote developers can use a remote desktop product such as Citrix® software to access a centrally located database.
- Remote developers can work in a VPN with a common Telelogic Synergy database. This technique provides the least administration overhead; however, its effectiveness varies depending on distance and network latency.
- Remote sites can have their own local Telelogic Synergy database and use distributed configuration management (DCM) to transfer the changes between physical locations.

When distributed teams have to be accommodated, you must consider performance issues and the affect of network topology, bandwidth, and latency. Telelogic Synergy needs low latency between its user-interface and Telelogic Synergy server or engine and prefers having the engine on the same machine as the database server.

Security concerns are another factor. If one of the remote groups is an outsourcing firm, you might want to restrict its access to some or all of the application code.

Sharing requirements

The CM implementation team must determine the degree of sharing required between the various databases in the database topology. You can run a number of sessions on a number of databases. However, within a single session, you have access to the data in that database. If a user must work closely on the source for a library and an executable that links with that library, for example, the source for both should be in the same database.

There is almost always some sharing between databases. It's common to have one database that holds reusable libraries, and different databases that contain the

source to the applications that link with the reusable libraries. In this case, the database with the reusable libraries exports the libraries and associated header files to the various consumer databases. The developer of the executable in the consumer database has everything he needs from the reuse database, without having the actual source to the library in the same database. The key is that although data can be shared between databases, the degree of sharing required is often an important factor in deciding on an optimal database topology.

Number of objects

A Telelogic Synergy database normally holds a large number of objects. The constraining factor is usually the hardware available for the database server. However, even low- to mid-range hardware can usually support over 50,000 objects. The number of simultaneous users is usually a factor before the number of objects is. In applications with a large number of objects relative to the size of the team, you may want to break down the application into a database topology that distributes the total number of objects, which is commonly called restructuring.

See "Allocate system resources" on page 17 for a more detailed description of performance and space considerations.

Component-based software architecture

Component-based development is the creation of applications from reusable parts.

Telelogic Synergy works well with component-based application architectures and supports component development and integration that are required to manage product architectures and families. This is achieved using the flexibility Telelogic Synergy offers in database topologies. It relies on projects and subprojects as the organizing mechanisms. Mapping component-based applications to team responsibilities enables Telelogic Synergy to support separate component and product releases.

Process and workflow requirements

The following information can be configured in a Telelogic Synergy database:

- Data types and behaviors
- Releases
- One or more development processes spanning development, integration, and system testing

This data is stored on each Telelogic Synergy database. Therefore, if it is important for different parts of the project team to have a behavior model, be sure to design it into the database topology.

Security and access control requirements

Group security enables users within a single database to control who can modify what kind of data. Additionally, group security provides a means to assign different privileges to different projects or objects on an individual or group basis. You can also use group security to control the read access of source files.

Recovery

The design of the Telelogic Synergy database allows files to be accessed even when the database server machine goes down. If the file server machine crashes, or if there is a media failure on one of the disks that store the data files, the files are inaccessible until the system is repaired. RAID disks minimize this risk. Project teams with a large amount of data may choose to distribute the data across databases to minimize the risk.

Administration

Administering the Telelogic Synergy databases is another factor to consider when designing a database topology. Typical database administration activities include adding users, performing backups, checking the data for consistency, deleting unneeded data, maintaining types and releases, and upgrading to a new Telelogic Synergy release.

More administration activities are required when more databases are used. Conversely, smaller databases are faster to backup and contain room for growth. In general, the administration of a Telelogic Synergy database is simple enough that the cost of administration is rarely a determining factor in the topology design.

Selection

Finally, the process of selecting a database topology involves considering each of the factors described above and balancing their various aspects to arrive at an initial topology. As a starting guide, consider giving each application its own Telelogic Synergy database. Then, you can combine one or more applications into a single database if, for example, particularly heavy code sharing is required among applications. Conversely, you can split an application into multiple databases if you have compelling reasons to do so, such as very large numbers of

users or very different security needs for different teams working on the application.

There is no single solution that applies to all situations, and you can have several viable database topologies.

Allocate system resources

This section describes how to allocate system resources to support Telelogic Synergy. It provides the background information necessary for selecting appropriate hardware configurations to run the Telelogic Synergy product.

Telelogic Synergy is implemented in a three-tiered, client-server architecture designed to use the resources available on the network in today's distributed computing environments.

Telelogic Synergy process architecture

A Telelogic Synergy session comprises two or more processes: the client (or interface) processes and the server (or engine) process. The client processes manage user interaction and work areas. The server handles database interaction and access. A database server also processes service requests from all of the server processes.

- Telelogic Synergy clients

  Telelogic Synergy has several interface clients. "User interfaces" on page 3 describes them. When the user starts an interface, Telelogic Synergy automatically starts one or more server processes to service the client interface.

- Telelogic Synergy server

  Each user session has one or more server or engine processes. As described below, the engine process connects to the database server.

  The engine can run on either the database server system or the user's desktop system, or it can be distributed to a different system on the network. Running the engine on the database server system usually results in the fastest performance for users — unless the resources of the database server system are insufficient for hosting engine processes for all users.

- Database server

  The database server process is the commercial Relational Database Management System (RDBMS). For each remote session that is started, a small RDBMS process runs on the machine (the database server system) where the database resides.

- Daemons or services

  Certain processes must be running before users can start sessions. On UNIX, the processes are called daemons, and on Windows they are called services. On the UNIX network, the `ccm_start_daemons` command starts the `ccm_router`, `ccm_objreg`, and `ccm_esd` daemons. On Windows, the services are configured to start automatically when you start the system.

  Router (`ccm_router`): The router process sends all communication traffic to the correct recipients. This process can run on any supported system on the network.

  Object Registrar (`ccm_objreg`): The object registrar provides the data-driven trigger capability that keeps users' views synchronized. By default, the process must run on each computer that is running a database server.

  Engine Startup Daemon (`esd`): The UNIX engine startup daemon enables users to start Telelogic Synergy engines on a remote server without using the **rsh** (remote shell) or **rexec** (remote execution) utilities.

  OR

  Engine Startup Service (`ccm_ess`): The Windows engine startup service enables users to have local login permission so that it can use engine processes. You must run one `ccm_ess` service process per engine machine. The `ccm_ess` service starts automatically at installation and startup. It registers with the router. When you start a Telelogic Synergy session, the interface process requests that `ccm_ess` start the engine.

  Help Server (`ccm_helpsrv`): Telelogic Synergy uses a help service to serve help requests from users' sessions. The help service runs on the same system as the router. The `ccm_start_daemons` command also starts the help service.

  Inter-process communications: For Telelogic Synergy's different processes to communicate with each other, several services and conditions at the operating system (OS) level must be available on the customer's network.

  The site system administrator must ensure that the general trust requirements required between systems running clients and engines are set up. Also, on UNIX, the user-owned client process needs to start as the *ccm_root*-owned engine process, requiring **setgid** and **setuid** permissions on the engine executables.

These environmental requirements need to be understood, agreed upon, and set up early to keep the implementation moving. For further details refer to the *Telelogic Synergy Administration Guide*.

The Telelogic Synergy data architecture

The following describes how data is distributed in Telelogic Synergy.

- RDBMS and project files

  A Telelogic Synergy database consists of two parts: the RDBMS that stores the metadata, which is information about the project files, and the file system storage, which is the file archive and the file cache.

  The metadata is approximately 10 to 15 percent of the size of the file system storage. If your application takes 100 MB for the source and binary, the RDBMS requires between 10 and 15 MB. The ratio can vary according to usage and development model customizations, but the ration is a useful guideline for planning disk space. If your database also contains Telelogic Synergy data, it usually contains a higher percentage of metadata.

- Spreading data across the network

  The RDBMS must be physically located on the database server. However, on UNIX, the file system storage can be anywhere on the network file system. By default, the UNIX file system storage is on the same system as the RDBMS, but you can use symbolic links to move all or part of the file system storage to a different location on the network.

  Telelogic Synergy's architecture is more sensitive to network latency than bandwidth between interface and engine, so performance is better when the engine is on the same machine as the database server.

  UNIX machines running Telelogic Synergy engine processes must have the Telelogic Synergy database's file system area visible. Therefore, the files either must be local or be mounted across the network. The next section, "Standard hardware configurations", describes the possible UNIX hardware configurations.

  You can either hard-mount or use the automounter, as long as the data files are visible from each system that runs an engine. However, the database's RDBMS does not need to be visible to the engine process.

  The files must be visible through the same logical path on all engine machines. For machines running Telelogic Synergy clients, the Telelogic Synergy file system area does not need to be visible.

Standard hardware configurations

You can run each database in a UNIX-only, Windows-only, or heterogeneous system environment.

The process configuration that performs best for UNIX teams is where each user's client runs on a personal workstation, and all the server processes run on a single dedicated machine. This configuration requires a database server with adequate resources.

An alternative configuration occurs when each user runs the Telelogic Synergy client and server on a personal workstation. The process requirements for the database server in this configuration are very low. A mid-range workstation or server normally handles the load of most teams. However, the performance is not as good as the previous technique of running engines on the database server.

Multi-vendor platform support

Telelogic Synergy supports heterogeneous combinations of hardware running processes on the same Telelogic Synergy database.

Telelogic Synergy engines, interfaces, and database servers can run on different hardware platforms. The only requirement is that support and technical assistance must be available for each different hardware platform.

Memory and swap space requirements

The efficiency of virtual memory varies in machines from different vendors. Therefore, RAM requirements sometimes depend on the machine.

Consider the virtual memory efficiency and vendors when you calculate RAM requirements. For example, Sun machines page well and require less than average RAM per session, while Hewlett Packard machines require a higher than average real-to-virtual memory ratio.

Refer to the *Telelogic Synergy Installation Guide* for memory and swap-space requirements.

Distributed builds

By default, Telelogic Synergy performs compilations on the computer the client process is running on. However, by editing the configuration file, you can cause the build tool to use a set of machines available on the network.

Many sites dedicate extra or seldom-used workstations as compile servers; other sites send compiles to the central server systems with excess capacity.

Assessing hardware and network capacity

The following common indicators show that available system resources are potentially insufficient:

- All developers are running on a single server using either an X11 server emulation package or low-end PCs, and the server is low on memory or already heavily loaded.
- Developers have their own workstations but they complain about network performance or that they have low-end workstations or PCs.
- The server is low on available disk space.
- Developers report slow remote or shared file access times.

If one of more of these symptoms are present at your site, determine whether sufficient hardware and network capacity is available.

3 Deployment

The previous chapter on system resource planning described the factors involved in deciding where to run which processes and where to locate data. Now you are ready to install the product on the various systems.

Telelogic Synergy might already be installed on one or more of your computers for either demonstration or evaluation purposes. However, now that you are preparing to use Telelogic Synergy in production, you will need to reinstall it, or install it on additional systems.

Consider running a full load on the system so that you can test the performance, identify any bottlenecks, and adjust the configuration accordingly.

Set up the database servers

The database topology largely determines the number and location of the database servers. Consider the following additional factors to prepare your installation for production use.

Raw versus cooked database servers

Informix, which is the default underlying RDBMS for Telelogic Synergy, supports two forms of installation on UNIX. The database server on Windows supports only cooked chunk files. A simple Informix installation of cooked files is easy for demonstrations and for small project teams. However, for most production use, you may want to run with a raw file system. Raw file system Informix servers provide the highest performance and outstanding recoverability in the event of media or power failures.

The *Telelogic Synergy Administration Guide for UNIX* describes how to set up Informix for use with raw file system partitions. If you're using an Oracle database, see *Telelogic Synergy Administration Guide for UNIX (on Oracle)*.

Number of databases per database server

Informix allows many Telelogic Synergy databases for each Informix database server. However, for large production databases, each Telelogic Synergy database may be stored in its own Informix server because online backup and recovery is provided by Informix on a database server basis. Therefore, if you need to restore, as in the event of a disk failure, you have to restore all databases in the server, which is probably more than is necessary. Consider online backup and transaction logging for medium and large project teams.

You can create additional database servers easily. Instructions are provided in the *Telelogic Synergy Installation Guide* using Telelogic Synergy tools for UNIX. Only one database server can be created per database host.

Location for running engine processes

When you run the Telelogic Synergy engine process on the same machine as the Informix server, they communicate through shared memory. When you run the engine process on a separate machine, they communicate through TCP/IP. Shared memory is much faster; many queries run up to ten times faster when the engine is on the database server machine.

However, if you run too many engine processes on a machine without enough CPU power or memory, system performance is degraded.

Consider running your engines on the database server, and purchasing a server with enough memory and CPU to accommodate the expected number of simultaneous sessions. If this is not possible, for example, when the expected number of simultaneous sessions is more than you can run efficiently on the server that you can afford, consider setting up one or more separate Telelogic Synergy engine server machines.

When running UNIX engines on machines other than the database server, ensure that `ccm_root` can log in from the engine machine to the database server without providing a password.

Install the license server

Before installing Telelogic Synergy, you must obtain and install a new license server. The license server is a FLEXnet-based license server that you use to provide licenses to products. The installation of the license server is a separate procedure from the installation of the Telelogic Synergy product. For complete information about license information, read the *Telelogic Lifecycle Solutions Licensing Agreement*.

About license data

During your evaluation, you might have received a license valid for a limited time to run the application on your evaluation machine. For your production environment, ensure that you have obtained a full production license data file that is valid for the machine on which you intend to run your Telelogic Synergy license manager.

Install Telelogic Synergy

Before installing Telelogic Synergy, read the *Telelogic Synergy Readme* and the *Telelogic Synergy Installation Guide* for the platform you are using.

Telelogic Synergy must be installed on the Telelogic Synergy server and client machines, or on a shared file system that other systems can use. On Windows, you can install just the required components on the various machines.

On UNIX, you need *root* access to install Telelogic Synergy. On Windows, you need domain administrator privileges to create the users *ccm_root* and *csuser*, and then you need local administrator privileges to install Telelogic Synergy on each machine.

Heterogeneous installations

If you have a variety of hardware platforms that can run Telelogic Synergy, you probably want to set up a heterogeneous installation. In this configuration, Telelogic Synergy users can have a mix of workstations and servers covering a range of hardware and operating systems. Although arbitrary mixes of hardware are supported, the installation must be set up so that the various installations reference common configuration information. The *Installation Guide for UNIX* describes installing for heterogeneous environments.

Install updates

If you have updates or patches, for example, to enable Telelogic Synergy to run on a new operating system upgrade, install these updates now.

On Windows platforms, a feature is available to automatically notify users if they need to update their client installations. Telelogic Synergy checks to see if all users are running on the same release as the server installation, and if they are not, notifies them that they need to update their client installations. Users can then launch the installation and complete the update. For more information on this feature, see the *Installation Guide for Windows*.

Configurations

The following section describes what to create and configure for your deployment effort.

Database servers

Create and configure a database server for each Telelogic Synergy database. The *Telelogic Synergy Installation Guide* describes how to create database servers.

Client workstations

Each machine assigned to run either a Windows-based or UNIX-based Telelogic Synergy client must have the Telelogic Synergy software installed locally that is capable of accessing a network installation. UNIX workstations can share the server installation. Although the database cache does not have to be visible from the client workstation, visibility improves performance. This visibility is mandatory in UNIX link-based work areas and is normally provided through NFS. Windows-based clients use either a local installation or a shared installation from a PC file server. The *Telelogic Synergy Installation Guide* describes the different client installations.

Build servers

If you perform remote or distributed builds, configure the build servers so that they also have Telelogic Synergy installed properly. For UNIX link-based work areas, the Telelogic Synergy database file cache must be mounted and visible from each build server.

Also, ensure that build managers can log in from the client systems to the build servers without providing their passwords. Refer to the *Telelogic Synergy Installation Guide* for details. You can distribute builds to platforms that do not support Telelogic Synergy, and in those cases Telelogic Synergy does not need to be installed on those platforms.

DCM and DCS

If you intend to use distributed configuration management features, you should read *Telelogic Synergy Distributed*. This book provides a methodological description of the various ways geographically-dispersed sites can use DCM and Distributed Telelogic Change (DCS) to distribute code and change requests. Additionally, the book gives step-by-step instructions for configuring installations to use DCM and DCS.

Establish the project topology

This section describes how to establish the optimal project structure before migrating your software applications into Telelogic Synergy.

Project topology refers to the organization and structure of projects that represents an application and its associated files and directories. A good project structure is especially important when you migrate very large applications. Planning your Telelogic Synergy project structure in advance of performing the migration has both short- and long-term benefits.

Note Large applications comprise over 5,000 files or one million lines of code.

This section assumes that you have read *Introduction to Telelogic Synergy*.

Applications

Many large software systems stored under Telelogic Synergy do not, strictly speaking, fit the term "application." For example, operating systems and other systems-level software, large quantities of documentation, or test data are not generally thought of as applications. However, you can migrate these types of applications into Telelogic Synergy successfully. For example, all Telelogic Synergy documentation is controlled in Telelogic Synergy databases.

Projects

An application is represented by one or more Telelogic Synergy projects.

A project is a user-defined group of related files and directories. A project can also contain other projects, known as subprojects. A project normally represents a logical grouping of files, such as a library or an executable. For example, the software that implements an editor application can be stored in a project named **editor**.

Projects are versioned like any other object in Telelogic Synergy. Different versions of the same project can contain different versions of the members, and even different members, as the software changes over time. For example, different versions of the **editor** project may be represented as **editor-1.0**, and follow-on versions of **editor-1.1**, **editor-1.2**, and **editor-2.0**. The **editor** project for **editor-2.0** can contain new files that did not exist in **editor-1.0**, as well as newer versions of many of the same files. Some files in **editor-1.0** might not be part of the **editor-2.0**.

A single file or object version can be a member of multiple projects. Although the same object version appears in different projects, it exists only once in the Telelogic Synergy database.

Many versions can exist for a single project. The following are some examples:

- Each developer who is developing the project has his own version, used to develop and unit test his own changes. These are called development projects.
- Build managers have versions for preparing the software for integration testing, system testing, and release. These are called build management projects.

- Released projects are versions of the software that have been released or have reached a milestone.
- A baseline is a set of projects at a point in time.
- A baseline project is a project that is a member of a base line.

Work areas

A work area is a location in the file system where Telelogic Synergy writes the project data for a specific project version. A project's directory tree structure is identical to the project's tree structure in the Telelogic Synergy database. The work area is a location where users can work directly with the files using Integrated Development Environments (IDE) and build tools, such as *make*.

The actual files reside in the Telelogic Synergy database, but Telelogic Synergy keeps the work area synchronized with the database. On UNIX systems, the files in a work area often are linked to the database files. On Windows systems, the files in the work area are copies of the database files.

When you create or change a project, Telelogic Synergy updates your work area automatically and transparently. When you add members to a project, Telelogic Synergy updates the work area with the new files; when you remove members from a project, Telelogic Synergy removes the corresponding files from your work area.

You can set up a subproject's work area to be either dependent on the parent project's work area (relative) or in a separate location (absolute). If you have tools or processes that rely on a certain directory structure, keep this in mind when setting up your project work areas.

Factors influencing project topology

There are several viable project topologies for any application. Coming up with a good project topology involves considering several factors and prioritizing them based on the needs of the particular project team.

The primary goal is to create a project topology that meets the following objectives.

- Provide developers with an efficient and compact work area for development and unit testing.
- Match the natural composition of an application into smaller components.
- Delineate team responsibilities.
- Match the way the application and its components are released.

- Provide satisfactory performance.

Another important factor is the build dependencies within the application. Before designing a project topology, you must analyze the application's directory structure, build dependencies, and build process. If you are not an expert on the application, consider enlisting the aid of team members who are expert in the application to help design the project topology.

Project size

How large can a Telelogic Synergy project be? It can be much larger than you probably want it to be. A single project can hold thousands of files and millions of lines of code. You also can have hundreds or even thousands of projects in a Telelogic Synergy database, multiple Telelogic Synergy databases in an Informix server, and even multiple Informix servers on a single system or spread over a network of systems. But working with and manipulating projects this large can be cumbersome and cause your project to lose flexibility.

For these and other reasons described below, an ideal number of objects to include in individual subprojects is several hundred objects or less.

When you preview a large migration, the **Migrate** dialog issues a warning if the project you asked to load contains more than 2,500 objects. For reasonable performance, Each project should have no more than 2,000 members and each directory no more than 500 members. Smaller projects of approximately 500 members and directories of between 100 and 200 members are even faster.

If your application contains over 1,000 files, consider representing the application as a collection of projects for better performance.

Building projects

When designing the project topology, consider the units of software required for building the main product deliverables.

To build with any other *make* tool, the necessary files first must exist in a work area. (Recall that each project has a work area in the file system.)

If many developers are working on a large application, and the files are separated so that a developer doesn't need everyone else's files to build his part of the application, the application can be structured into multiple Telelogic Synergy subprojects to reduce the number of objects that each developer must manage.

Such elegant design of the code structure is the consequence of using component-based development architecture. Dividing the application into subprojects that reflect different components minimizes the time and space required to build the application.

Variant projects

Normally, one version of the Telelogic Synergy project is defined for each platform to which the application must be ported. The project object is the primary location for platform-specific build macros and environment properties.

Large sections of the application, such as documentation, test data, designs, and scripts, are often platform-independent. If these areas are separated into their own Telelogic Synergy subprojects, variant projects can all share the same, common subprojects. For Windows-based users, the project must have an absolute work area path that can be shared.

Reusable components

Configuration and change management can play important roles in enabling you to reuse software objects. The Telelogic Synergy project object provides a self-contained, relocatable unit of software. The project object is a flexible mechanism for managing collections of multiple numbers or types of other objects, including other projects. Thus, a hierarchy of project objects can allow any application architecture to be represented as a structure of its constituent subsystems and components.

Mechanisms are provided so that if you structure your application into smaller subsystem or component projects, Telelogic Synergy facilitates sharing and reusing these projects as appropriate. Windows-based users must ensure that each low-level project has an absolute work area path that is visible to the high-level projects that will reference it.

Representing applications as projects

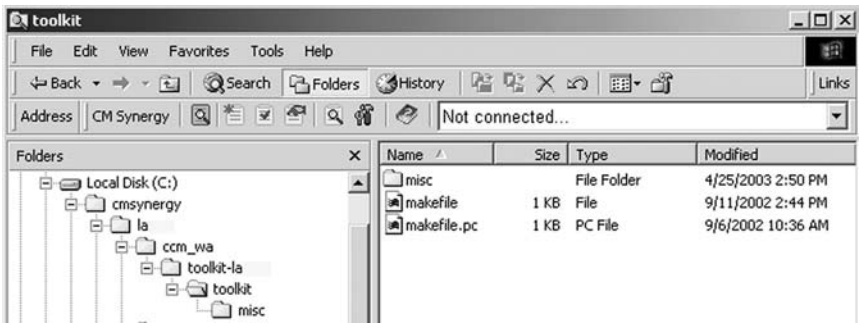
When selecting an initial project topology, consider the factors previously described and weigh them against the requirements of your application and your project team. This section describes several approaches to a project topology, and explains the applicability of each.

On UNIX systems, where the clients run on the same network as the database, projects usually have *absolute* work areas, which means any project can contain any other project as a subproject. On Windows systems and remote UNIX clients, a subproject's work area can be either *absolute* or *relative*. A writable project with a relative work area can only be used as a subproject within a single parent project.

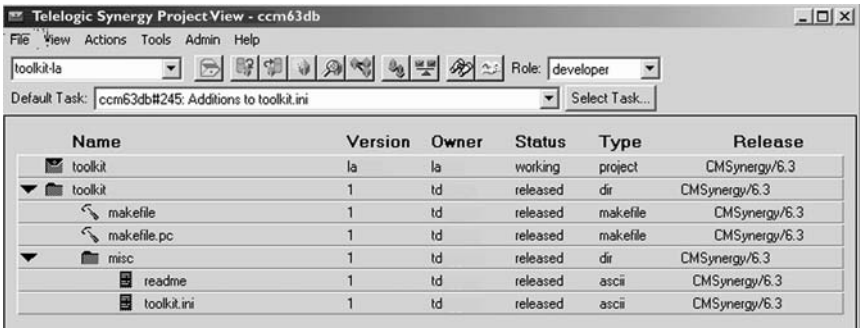
The following examples show that an application's build process is affected by whether a project's work areas are absolute or relative. Relative work areas usually require minimal changes to the application's build process, but absolute work areas result in a much higher degree of componentization and reusability.

Single-project representation

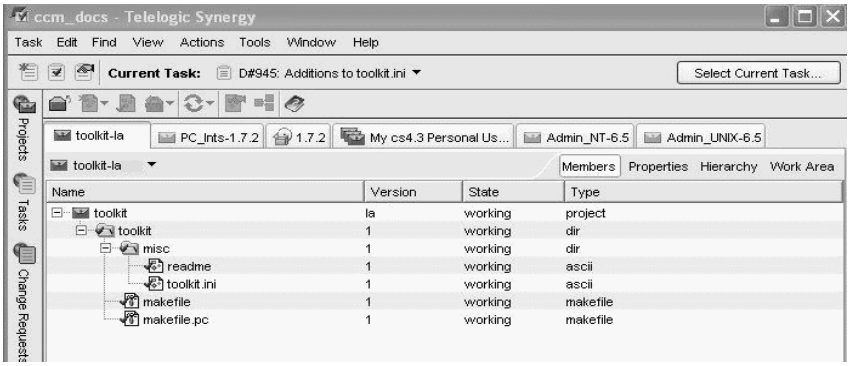
One way to structure an application's source code in Telelogic Synergy is to load it all into a single project, as shown below.



This shows a single project, **toolkit-la**, as you would see it in the Windows file system



This shows the same single project, **toolkit-la**, as you would see it in Telelogic Synergy Classic.



This shows the same single project, **toolkit-la**, as you would see it in Telelogic Synergy.

This approach has the advantage of being quick and easy to set up and maintain. However, it also has some limitations that prevent it from being scaled to large applications.

First, because the entire application is located within a single project and developers operate in Telelogic Synergy at the granularity of projects, each

developer must view the entire application rather than one or more logical components of the application. On very large applications, performance is negatively impacted and requires a larger amount of disk space for each developer.

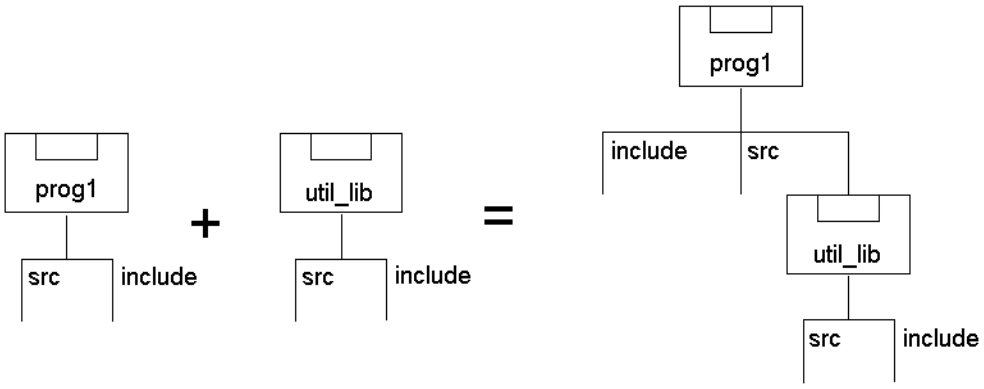
Second, because developers have a view of the entire application, they are responsible for building all parts of the application, even those they are not currently modifying. Other structuring techniques provide ways for developers to work with and build only the parts of the source tree in which they are interested. These structuring techniques are described later in this chapter.

Size is only one of the factors to consider in determining a good project topology. Other factors are addressed later.

Multiple-project representation

Representing all but small applications as hierarchies of projects has several advantages:

- You can organize your files into logical groupings. Product teams can have projects specifically for the areas for which they are responsible.
- You can distribute the work areas for different projects to different locations instead of having one very big work area in the same location.
- Developers need working versions of only the projects containing the files they change, which speeds up operations such as update.
- It allows you to have parallel versions of only the projects that must be built for parallel platforms, instead of duplicating the entire project for every platform on which you build, as shown below.



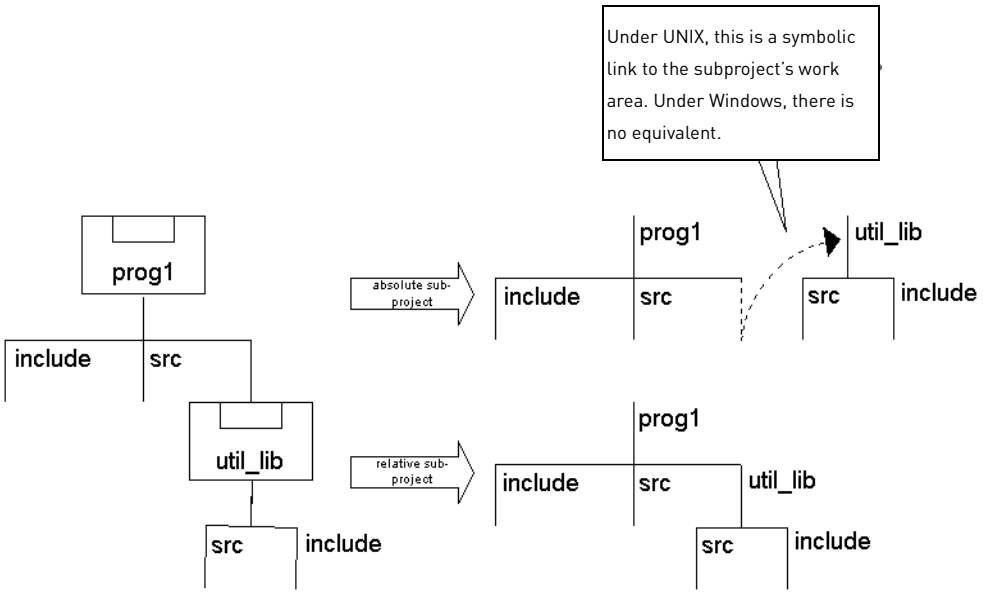
When individual projects are assembled together in this manner, the lower-level project is a *subproject*. The higher-level project is the *parent project*. Calling a project

a subproject or a parent project only indicates its position in the relationship between the two projects. It does not mean that either project is somehow better than the other. In reality, each is treated as an independent entity within Telelogic Synergy.

The ability to combine projects in this manner can be understood if you remember that a project represents a *version* of a source tree. By selecting a particular version of a project to put within another project, you are selecting the entire source tree currently associated with that project. In the example above, if the **prog1** project is owned by a developer, the developer can choose whether he wants to view his own version of the **util_lib** source including the correct version of the **util_lib** project.

Relative versus absolute work areas

When you combine projects into a hierarchy, the look of the resulting work area depends on whether the subproject has an absolute or relative work area. Every project has a work area, which is the place in the file system that reflects the contents of the project. When a project is used as a subproject, a property of the project's work area (set through the **Work Area Options** dialog), defines the location of the work area relative to that of its parent.



The figure above shows that a relative work area is contained within the work area of its parent project, which creates a seamless directory tree. That is, the location of the relative work area is *relative* the work area of the parent project.

On the other hand, an absolute work area is located at an absolute file system location. This location, the *work area path*, is specified on the project and can be set through the **Work Area Options** dialog. On UNIX, a symbolic link is placed at the location of the subproject in the parent project's work area. This link points to the location of the subproject's work area. Because the Windows environment has no equivalent to UNIX symbolic links, the link is only present on UNIX. However, the subprojects appear in the GUI on both Windows and UNIX as members of the parent project.

The different representations afforded by absolute versus relative work areas have the following implications:

- Because each writable project can have only a single work area, and a relative work area is contained within the parent project's work area, a writable project with a relative work area can be a subproject to exactly one project.

  On the other hand, projects with absolute work areas have their own absolute location and can be shared with multiple parent projects. An example of sharing is a library that is used by multiple applications and modified in each one.

- If you want the work area location to be visible to others, set the work area location for your absolute work areas to a shared file system.

- Because absolute work areas are not represented as true directory trees within their parent projects' work areas, a makefile or script in an absolute work area cannot use relative paths to access objects in its parent project's work area.

  For example, a makefile contained in the `util_lib` project (shown in the figure above) cannot use an include path directive set to `-I../../include` to access **includes** in the **prog1** project because traversing up from the project's directory tree does not take you into the parent's work area.

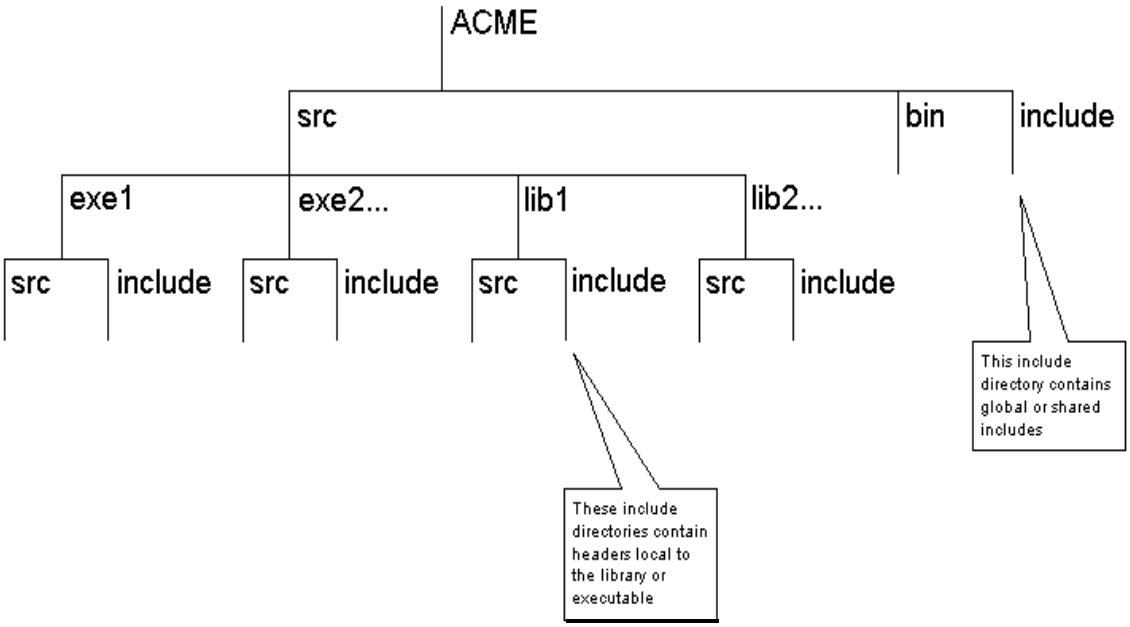
- If you need a true directory tree structure, you might need to give up the ability to share absolute work areas and use relative work areas instead. Makefiles, development tools, and scripts that depend heavily on a true directory tree structure must use relative work areas.

Note The decision between using absolute or relative work areas is primarily one that impacts Windows users. On UNIX, absolute work areas are usually the better choice.

Ultimately, understanding the differences between absolute and relative projects and choosing the correct work area type for the situation is necessary for establishing a good project topology for your application.

Multiple-project representation using relative work areas

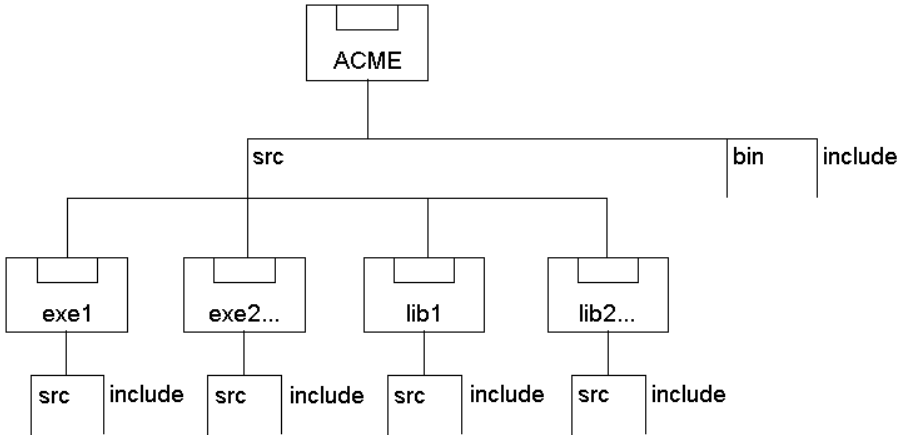
For all but very small applications, the ideal representation of your application is usually a segmented set of subtrees, each with its own project. The figure below shows a source tree for a medium-sized application.



The ACME source tree contains a subdirectory tree for each executable and library created by the application. Because development work is often targeted towards specific executables or libraries, these natural divisions form a reasonable place to subdivide the source tree into projects.

Subdividing the application into multiple projects has not affected its build characteristics because the subprojects have relative work areas; that is, executables and libraries can still reference the global *includes* through relative

paths, and several benefits have been obtained. The figure below shows an application structured using subprojects with relative work areas.

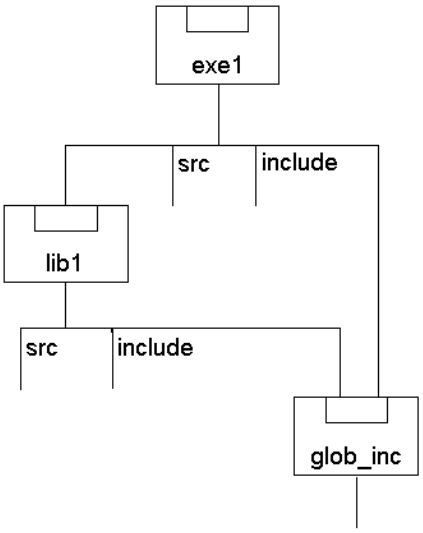


Dividing the application into multiple projects is beneficial because operations that can simultaneously work on all objects within a project, such as check in, update, and query, now can be limited in scope to just the desired executables or libraries. If all objects are contained in a single project, this is not possible. Other benefits include reusability and improved performance.

Project sharing

Another technique relies on projects with absolute work areas to create a self-contained development environment for each executable in the application. Each project contains only the objects necessary to build that executable. The following is an example of how the development environment might look for

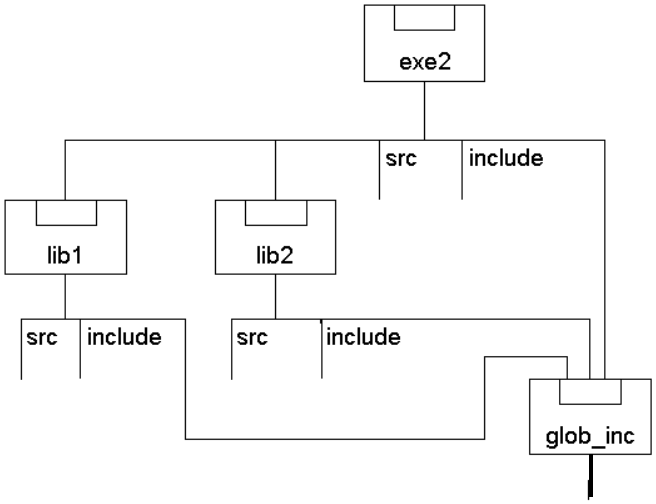
exe1. This is a self-contained environment created using projects with absolute work areas.



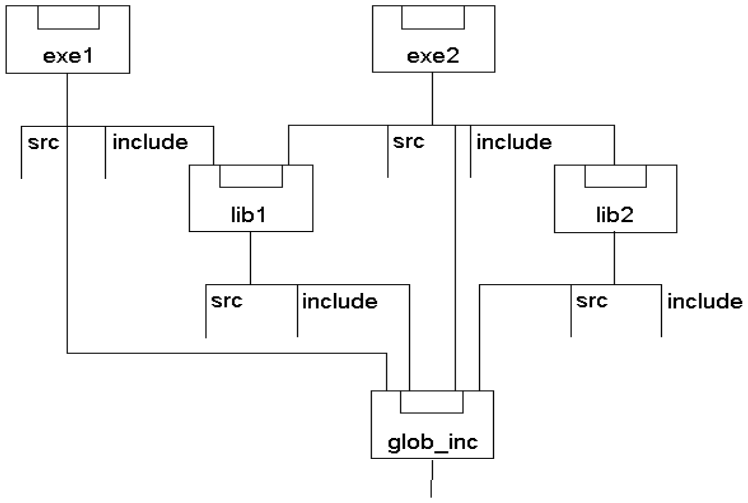
Both **lib1** and **exe1** share the **glob_inc** project. The project contains the include files found in the application's global `include` directory. Because **lib1** and **exe1** have absolute work areas, a makefile within them cannot access those *includes* at a higher level in the tree. However, by putting the *includes* into their own project and including that project in the areas where those includes are needed, they can be accessed relatively through the link pointing to the **glob_inc** project (UNIX).

Note Because both the `exe1` and `lib1` projects share the **glob_inc** project, and the **glob_inc** project is still changing, it must have an absolute work area.

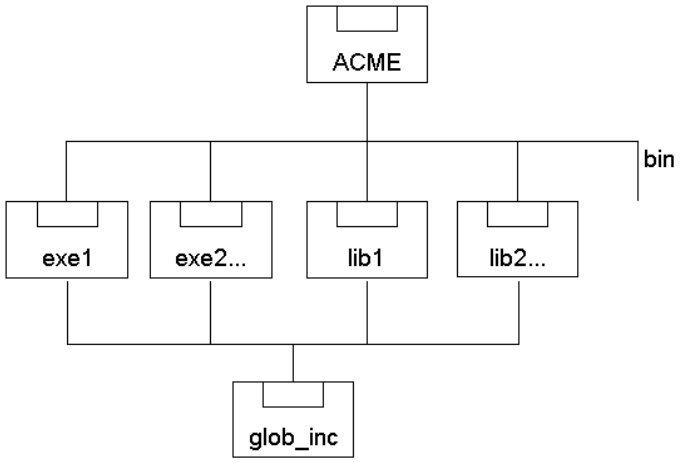
The figure below shows a similar environment for **exe2**. Although **exe1** uses only **lib1**, **exe2** relies on both **lib1** and **lib2**. This is a self-contained environment created using projects with absolute work areas (**exe2**).



The figure below shows an example with projects for two executables that share the same libraries and include files. By using absolute work areas for the subprojects, the project hierarchy will contain the minimum set of files in its work areas.



The following figure shows how the entire application hierarchy looks.

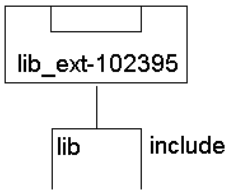


Product sharing

When you use the techniques just described, the resulting hierarchies may still be larger than you want, even if the application tree is separated into multiple projects. Additionally, each developer is responsible for building all the libraries needed to link his executables, including those that he is not changing.

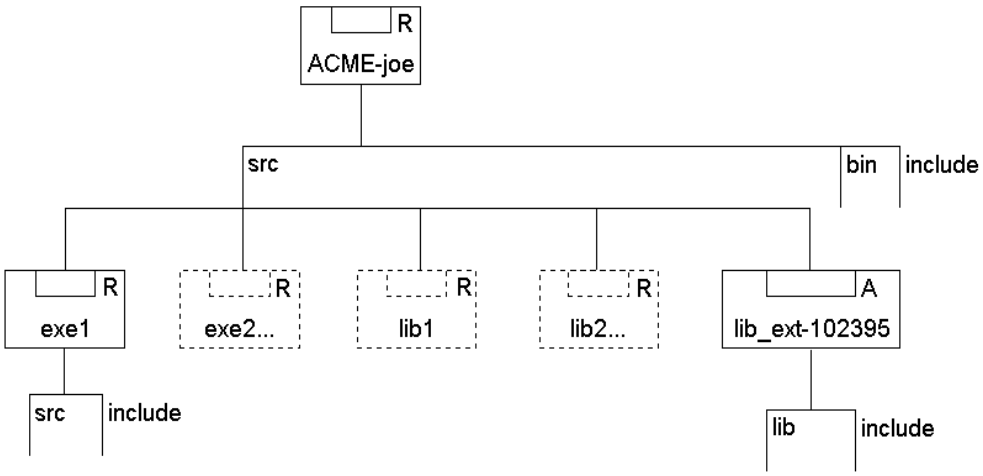
These issues can be addressed using a technique where developers share prebuilt versions of the libraries that they are not modifying. The advantage of this technique is that developers no longer require the source for these libraries in their development environment, provided they are not making any changes to those libraries.

Access to the prebuilt versions of the libraries is accomplished through logical projects created by the build manager.

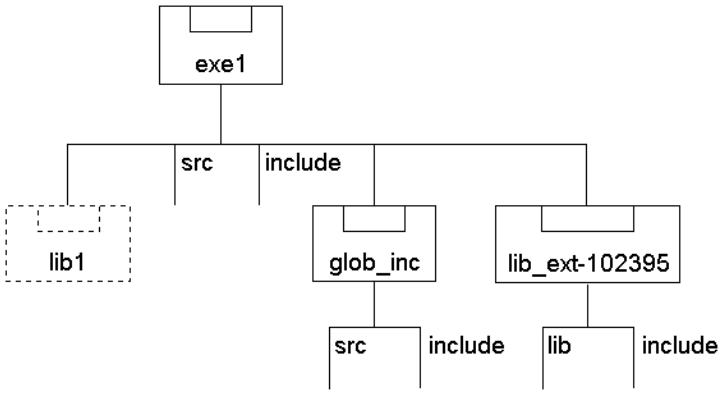


Projects like the one shown above are called *external* projects because they contain built versions of the libraries and the headers needed to access them, but

not the libraries' sources. External projects are created on a periodic basis, usually nightly, from a build of the integration area, which enables developers to share current versions of the libraries.



The figure below shows the external project inserted into a self-contained environment created using absolute projects.



The example shows that the makefile for `exe1` must be modified to pick up includes and libraries from the external project. Use the `$(CCMSUBPROJ)` macro to modify the makefile file.

When a developer needs to modify the library, the include and library paths must reference the library source projects before the external project so that the

developer's modifications take precedence over those contained in the external project.

Project topology example

This example shows how you might select the topology for the **payroll** project shown below.

Note Although the following is a Windows example, the selection process applies to a UNIX project, too.

```
C:\apps\payroll\              src\
                                          Makefile
                                          checks.c
                                          display.c
                                          main.c
                                          taxes.c
                                          payup.exe
                              incl\
                                          payroll.h
                                          taxcode.h
                              lib\
                                          Makefile
                                          compute.c
                                          print.c
                                          util.c
                                          util.h
                                          libutil.dll
                              doc\
                                          payroll.doc
                                          payroll.ps
                              test\
                                          suite1.data
                                          suite1.out
                                          suite2.data
                                          suite2.out
```

Large applications usually comprise a suite of modules, each similar to the hierarchy shown above. For example, a full financial management system might have **payroll**, **budgeting**, **payables**, and **receivables** modules.

Another example of a large application is an operating system comprising 10 million lines of code, including a kernel, commands, networking, and device driver modules. Each of these modules normally has its own substructure.

These application structures can be migrated into Telelogic Synergy in several ways. At one extreme, the entire structure can migrate as a single project, and at the other extreme, each subdirectory can be its own subproject. An optimal

structure is probably somewhere in between. In the **payroll** example above, the `src` and `incl` directories can be member directories in one subproject, and individual subprojects can be created for the `lib`, `doc`, and `test` directories.

The determining factors for the **payroll** project were the following:

- Project Size

  The **payroll** example is small enough that all files can be included in a single project. However, to allow for additional code that significantly increases the size of the application, consider creating subprojects for related files.

- Building Projects

  All files in the example are used for building, documenting, or testing the application. Whether or not you use subprojects in your configuration, include all of the files in your hierarchy so they are available in your work area.

  In a scenario where the application development, library utilities development, documentation, and testing teams are working on an application, you can create subprojects for each of these teams, so the teams can modify and build their portions of the application separately. Example subproject names are **paycode**, **util**, **paydoc**, and **paytest**.

- Variant Projects

  If your libraries and some of your code are platform-dependent, you can have variants for multiple platforms such as Windows XP and Solaris. You can have one version of each platform-specific project for each platform you build. You can assign meaningful versions to indicate the platform for which the code and library subprojects are built. For example, the Windows projects can be named **paycode-win32** and **util-win32**, respectively. You can then version the parent project with the platform-dependent version and add the variant subprojects to it.

  If you use your variant configuration for release 1 of your application, you can add this information to the version for each subproject in your project. In the current example, your parent project is **payroll-win32_rel1** and the subprojects are **paycode-win32_rel1**, **util-win32_rel1**, **paydoc-rel1**, and **paytest-rel1**.

  Refer to the *Build Manager's Guide* for more information on creating and controlling variant configurations.

For these descriptions of the four subprojects, the top-level project version for the **payroll** application is shown below.

```
payroll-win32_rel1\
payroll\
                        paycode         (> paycode-win32_rel1)
                        util            (> util-win32_rel1)
                        paydoc          (> paydoc-rel1)
                        paytest         (> paytest-rel1)
```

In Windows-based file systems, if the subprojects are not relative the file systems do not contain an entry for the individual subproject's root directories, but the project displays as shown above.

An example of a Solaris version of the **payroll** application is shown below:

```
payroll-sol_rel1/
                    payroll/
                                    paycode         (> paycode)
                                    util            (> util)
                                    paydoc          (> paydoc)
                                    paytest         (> paytest)
```

- Reusable Objects

  Converting your file system directories into subprojects enables you to add the entire subproject to another project. For example, the **util-win32_rel1** subproject can be added to another WIN32-dependent project that needs those libraries.

  For Windows-based users: the **util-win32_rel1** has an absolute work area so that it can be shared.

  The **paydoc** and **paytest** platform-independent subprojects were reused, Only the platform-specific subprojects needed to be labeled with a variant version.

- Updating Makefiles

  To simplify your makefiles, create project macros for the paths to the files in your subprojects. In the current example, you can define the following platform-dependent macros on your project and reference them in your generic makefile:

```
PROJ_ROOT = C:\apps\payroll\payroll-sun_rel1\payroll
OBJ_PATH  = ${PROJ_ROOT}\paycode\src
INCL_PATH = ${PROJ_ROOT}\paycode\incl
LIB_PATH  = ${PROJ_ROOT}\util
TEST_PATH = ${PROJ_ROOT}\paytest
DOCS_PATH = ${PROJ_ROOT}\paydoc
```

You can reuse the makefile in another project because the platform- and release-dependent information is associated with the project instead of being defined in the makefile itself.

The complete project hierarchy for the first release of the WIN32 variant of the **payroll** application is shown below. The commands used to create the hierarchy are at the end of this chapter.

```
payroll-win32_rel1
                          payroll
                                     paycode\
payroll-win32_rel1
                                     src\
                                                Makefile
                                                checks.c
                                                display.c
                                                main.c
                                                taxes.c
                                                payup
                                                Makefile
                                                checks.c
                                     incl\
                                                payroll.h
                                                taxcode.h
                                     util\      util-
                                                win32_rel1
                                                Makefile
                                                compute.c
                                                print.c
                                                util.c
                                                util.h
                                                util.dll
                                     paydoc\    paydoc-rel1
                                     payroll.doc
                                     payroll.ps
                                     paytest\   paytest-rel1
                                     suite1.data
                                     suite1.out
                                     suite2.data
                                     suite2.out
```

The above hierarchy assumes relative subprojects. If the subprojects are not relative, the subproject root directories are just pointers to the subproject.

External and installation projects

The Telelogic Synergy database is made up of a versioned object pool that contains all of the versions of all of the objects, including source files, directories,

documents, libraries, and executables. Project objects represent collections of these objects. Most of the projects that have been described so far represent collections of the objects required to build the application.

An object version exists only once in the Telelogic Synergy database, yet many projects can use that object version. For example, an include file can be in the project used to build a library and in every project that calls a function in that library.

With a versioned object pool, a project in Telelogic Synergy is not limited to representing the physical configurations used to build a product; you can also create arbitrary logical configurations that group related objects.

External projects

External projects are logical projects that contain the products of builds that were performed in other build projects; they are used for product sharing. An external project makes a component available to consumers without the overhead of the source files.

In this example, the build manager might create versioned external projects for the **util** project. You then have access to different versions of external projects, such as **util-ext_int08**, **util-ext_int09**, and **util-ext_int10** for shared products from successful integration builds.

Developers have access to *shared* products by adding a version of the external project to their project and can obtain updates to the products by using a later version of the external project.

Because the external project contains no source objects, adding it as a subproject to your project reduces the time required for creating and updating your project. You also can reduce the time to build your own products because the shared products satisfy many of your dependencies. Finally, disk space is conserved by not creating intermediate object files in your work area. An example of an external project is shown below.

```
                                        payroll.h
                                        taxcode.h
                              util\     util-win32_rel1
                                        Makefile
                                        compute.c
util-ext_int08
```

```
                    util
                              lib\
                                                util.dll
                              incl\
                                                util.h
```

Using external projects

Supplementing a build project with products from other projects is often useful. For example, developer John might want to have all of the source for the projects that he is using, but he might want to use previously built libraries from several other projects. A possible configuration for John's project is shown below.

```
billing-rel3\
                    billing\
                              Makefile
                              src\
                                        main.c
                                        billproc.c
                              includes\
                                        billing.h
                                        util.h (from util project)
                              lib\
                                        util.dll (from util
                                        project)
```

In the example above, the **billing** application links with the utility library from the original **payroll** project. Only the library and header files for the **util** library are included, and not the source objects from which the library and header files were built.

If John had included the entire **util** project, he would have received all of the **util** source, which is not what he wants to see or build; he wants to use only the products generated from the **util** project.

Another approach to this problem is useful for larger projects: the **util** build manager can create a shared, run-time version of the **util** project, similar to the one shown below.

```
util_ext-int\
                                        util_ext\
                                                  util.dll
                                                  util.h
```

This external project comprises the related objects required to use the **util** library. Using the `util_ext-int` subproject shown above, John's project can be configured as shown below.

```
billing-rel3\
                billing\
                           Makefile
                           src\
                                       main.c
                                       billproc.c
                           includes\
                                       billing.h
                           util_ext    (> util_ext-int)
```

John is reusing the build manager's integration testing project in this example. When a developer shares a build manager's prep project, there are two concerns:

- The objects contained in the shared product could change without warning when the build manager updates them, so John is not insulated from ongoing changes.
- If John has the *build_mgr* role, he could accidentally modify the prep project.

  In this case, John should copy his own version of this project so that it is updated only when he updates his own project. Alternatively, the build manager can baseline the external project on a regular basis.

  When projected into the file system, John's modified project appears as shown below.

```
billing-rel3\
                billing\
                           Makefile
                                     src\
                                                 main.c
                                                 bill_proc.c
```

```
                                                  includes
                                                  \
                                                            billing.h
                                                  util_ext
                                                  \
                                                            util.dll
                                                            util.h
```

Product sharing techniques are very important. Use them to share and use libraries without using the full source to those libraries.

By default, developers receive the latest checked in versions of the libraries and header files or the external project version. If a developer wants a different version, he invokes the **use** command to select a specific version. For more information, refer to "Product Sharing" in the *Build Manager's Guide*.

Installation projects

When your products are ready to release, create the media for your customers. An installation project is structured like your product's installation image; its work area can be used to create delivery or installation areas for the executables, libraries, scripts, and other files from which you create an archive or self-extracting executable, or save the image to CD or DVD. Below is an example of an installation project for the **payroll** application:

```
C:\releases\payroll-rel5.0\payroll

    bin\
                            payup.exe
    lib\
                            util.dll
    help\
                            payroll.ps
```

Configure Telelogic Synergy

This section describes how to configure Telelogic Synergy's behavior to meet your team's needs. It provides information on commonly configured settings, such as adding users and setting their roles, creating types, creating releases, and configuring a team's workflow and development process.

Information also includes strategies for configuring Telelogic Synergy that allow you to save and reuse your configuration. It also describes the upgrade implications of configuring Telelogic Synergy.

You can configure Telelogic Synergy in advance, or you can adjust the behavior while a project is underway. Most companies do a combination of both.

Add users and set roles

Use the `ccm users` command to add new users and define the roles for each user. This user list is your most basic security mechanism in Telelogic Synergy, and the easiest way to control what operations are available to each user. Only a user in the *ccm_admin* role can add users and roles.

See Telelogic Synergy Help for more information.

Define releases

A release is a stream of development. For example, one team can be working on the **quickedit/2.0** release of the QuickEdit program while a separate team works on a **quickedit/1.1** bug fix release of the same program. Releases are the basis for the development process in Telelogic Synergy; when you start using Telelogic Synergy, you should define all the releases your teams are currently working on, as well as those that were completed recently. A user in the *build_mgr* role adds and maintains releases.

Use the **Create Release** dialog or `ccm release` command to add new releases or define the properties of a release. See Telelogic Synergy Help for more information.

Configure the workflow

Telelogic Synergy's default workflow and methodology supports many best practices for software development and is useful for many teams. However, Telelogic Synergy enables each team to configure the workflow to fit its own needs. You can configure the development and testing process in the following ways:

- Easily add or remove test stages. By default, Telelogic Synergy provides integration and system test stages. You can easily add stages to support test cycles, such as performance testing, regression testing, and other types of testing.
- Configure the level of insulation for developers. On some teams, developers want to be insulated from other developers' changes until the tasks have passed testing, but on other teams, developers may want to get other developers' completed tasks, or even get each others' changes before the tasks are completed.

- Use a different process for each release. For example, a team working on a new-feature release may want to have only one test phase, integration testing, and the developers on this team may want to share each others' changes as soon as they are complete. Meanwhile, another team that is working on a maintenance release of the same software application may want a very insulated environment where developers do not see each others' changes until the changes have passed several test cycles.
- Automatically change the team's process during the course of a release. The build manager can add or remove test phases and configure the level of insulation for developers during a release and automatically apply these process changes to the entire team.

Some workflow settings can be configured by the build manager, but others require assistance from the CM administrator.

For an overview of how to configure the workflow, refer to the *Introduction to Telelogic Synergy*. Refer to Telelogic Synergy Classic Help and the *Build Manager's Guide* for information on purposes and update templates.

Create and configure types

Every object has a type, which dictates the class of data contained in the object and the object's behavior. Telelogic Synergy is shipped with the definitions of many different types of objects; however, you can extend the set of types so that Telelogic Synergy can understand and track objects for types not originally supplied.

You can create new types by using the **Type Definition** dialogs in Telelogic Synergy Classic. You can also configure many aspects of a type's behavior.

Types inherit their behaviors from their "super types." You can configure the behavior of an entire tree of types by changing the super type. The root of the source type tree is **misc**; the primary subtypes of **misc** are binary and ASCII. Configure the ASCII or binary type so that your changes apply to all of its subtypes.

Some of the type behaviors you can configure include the following:

- Require a task

  You can indicate whether a task is required to check in or create objects of this type. To indicate that objects must be associated with a task for check in to occur, set the **Require Task At** field to *integrate*; to indicate that a task must be selected before an object will be created, set the **Require Task At** field to *working*.
- Require a comment

You can indicate whether a comment is required before objects of this type are checked in.

- Add properties

  You can specify which properties are created automatically to new objects of this type. For example, you might add a property named **reviewer_comments** to new objects.

- Parallel development

  You can control whether or not users are allowed to check out or check in parallel versions by type. For example, because bitmap files cannot be merged, you can disallow checking them out in parallel.

- Source file initialization

  You can define the initial value for different types of source files.

Users must be in the *type_developer* role to create types. For detailed information regarding type configuration, read Telelogic Synergy Classic Help available from the **Type Definition** dialog. (You can read about the `ccm typedef` command as well.)

Forms of configuration

So far, we have primarily described settings you can configure using a point-and-click interface or a command. But, Telelogic Synergy has many other options that can be configured in other ways:

- Database options that are set by editing database properties
- User options that are set in a configuration file
- Data file configuration

A description of each of these forms of configuration follows.

Database options

Some options are configured by setting parameters in the Telelogic Synergy database. These options affect all users of that database.

Configuring database options usually involves editing a property, either on the model or on a type. Each database contains objects that represent a model and types. In some cases, you can use the `ccm attr -modify` command to edit the property's value. In other cases you must first use the `ccm attr -create` command to create the property before setting its value.

To change database options, you must be in the *ccm_admin* role.

Examples of database options you may want to configure include the following.

- The **parallel_exclude_rules** setting defines whether parallel notifications are received for different types of objects. For example, rejected objects are excluded from parallel notification by default.
- The **conflict_parameters** setting defines which conflicts users will see when they show conflicts for a project. For example, if your team typically associates a file with several tasks, you might want to turn off the multiple tasks conflict so it does not distract you from more important conflicts.

The **allow_prep** setting enables developers to include prep subprojects when updating their project. This setting is not normally used for CM best practices; however, some customers have a methodology that requires the sharing of build management projects.

For a complete list of database options, refer to "Default Settings" in Telelogic Synergy Classic Help.

User options

Some options can be configured in files such as the `ccm.ini` or `ccm.properties` file. The `ccm.ini` file contains options for configuring Telelogic Synergy Classic and business logic, and the `ccm.properties` file contains options for configuring Telelogic Synergy.

To set options for each individual user, edit his personal copy of the `ccm.ini` or `ccm.properties` file. You can set the default for all users by configuring the initialization or properties file in the installation directory; if a user does not set the option in his personal file, he inherits the default from the installation directory. A user who wants to override any of these settings can set the option in his own file.

The following are examples of some user options that you can set:

- **save_to_wastebasket**

  This option moves any uncontrolled file in your work area that needs to be removed to a wastebasket directory.

- **migrate_default_state**

  When you migrate data, the files are checked into the *integrate* state by default. You can choose to set the state to *working* if you want to modify or delete the files after migration.

- **reconfigure_parallel_check**

When this option is on, Telelogic Synergy displays parallel messages in the log file for parallel versions found during update (reconfigure). Although not all team members need this option turned on, build managers should consider turning this option on.

- **range_for_keyword_expand**

  By default, Telelogic Synergy expands keywords in the first 2048 characters of your file. You can change the value, according to the normal size of the section of your source file header that contains key words.

For a complete list of user options, see "Default Settings" in Telelogic Synergy Classic Help. For options specific to Telelogic Synergy, see "Using the Options Dialog Box" in Telelogic Synergy Help.

Environment variables

Some user options can be configured using environment variables. Users can set their own environment variables.

A commonly used environment variable includes setting the `CCM_ENGLOG` and `CCM_UILOG` environment variables to the path where you want the Telelogic Synergy session logs written. Use this technique to store your logs in a common location so the CM administrator has access to them.

For more information about environment variables, see "Default Settings" in Telelogic Synergy Classic Help.

Data file configuration

Some options are configured by editing data files. Here are some examples:

- A host file specifies the list of platforms that appear in the **Properties** dialog. Refer to Telelogic Synergy Classic Help for more information.
- The `attrange.dft` file in the `pt` directory under the database path defines the lists of priorities and subsystems available in the **Create Task** dialog and other task dialogs. See Telelogic Synergy Classic Help for more information.
- To change the layout of many of the Telelogic Synergy Classic dialogs, edit the Graphical User Interface Language Definition (GUILD) files. GUILD files are stored in the `guild` directory under the database path. You can edit GUILD files to change field layout and length, but you cannot add or remove fields without modifying the source code.

Strategies for configuration

You can apply most configuration settings directly to a production database (a Telelogic Synergy database that is being used for development).

However, if you have many production databases, configuring each database separately can be time-consuming and error-prone. Several techniques enable you to ensure that your configuration is saved and consistently propagated, and if necessary you can roll back a database to a previous configuration.

Save an empty pack file

Telelogic Synergy provides a set of pack files, which are database archives that you unpack to create a new database. The `base.cpk` pack file is an archive for an empty database with Telelogic Synergy's generic development model. When you start using Telelogic Synergy, identify the configuration that applies to all of your databases and create your own version of the `base.cpk` file with that configuration. The following is an example:

1. Unpack the `base.cpk` pack file to a database named `acme` (or your company name).
2. Configure the `acme` database with all changes that apply to all databases. For example, add users, add types, configure the workflow, and set the work area path template.
3. Pack the `acme` database to a new pack file called `acme.cpk`. Save it in the `packfiles` directory of your Telelogic Synergy installation area.
4. When you are ready to create a new production database, unpack from the `acme.cpk` pack file.
5. Retain a list of the changes you have made so that if you decide to upgrade to a new version of Telelogic Synergy, you can create a new pack file using the updated `base.cpk`.

See the *Telelogic Synergy Administration Guide* for detailed instructions on each procedure above.

Save changes in the model database

The model database is a special Telelogic Synergy database that stores the development model, including the types, data files, and property settings. From this database, new versions of the development model can be installed into production databases.

You can use the model database to version your changes and roll back to an earlier version if necessary. The model database is useful when you upgrade to a

new version of Telelogic Synergy because you can use it to merge your custom changes with the changes in the new version. The model database is not required for basic configuration of Telelogic Synergy; however, it is required for source code customization.

Customization

The standard out-of-the-box development model is very flexible, and you can configure it in several ways. However, if the available configuration options do not meet your requirements, you can customize the Telelogic Synergy model source code.

Although model source code customizations are infinitely flexible, the user-training costs increase with the modified features and when you upgrade to a new version of Telelogic Synergy because you must merge and test your source code changes internally before upgrading. For customers with extensive customizations, the merge and upgrade process takes significant effort.

The types of customizations customers request and are monitored and often incorporated as modifications in future releases of the standard Telelogic Synergy product. Over time, many customers discard their customizations when they determine that the standard product fulfills their needs.

Migrate the application software

At this point, Telelogic Synergy is properly installed and configured, and the project topology has been identified, so you are ready to load your data into the database.

The key to an easy migration is the project topology. If the project topology has been identified in advance, and you know how you want to represent your application in the Telelogic Synergy database, the **Migrate** dialog can help you load your application into Telelogic Synergy.

This section guides you through some of the issues to consider when migrating your application and outlines some best practices that might be overlooked in the rush to go live.

If your team is starting an application from scratch, you can bypass the migration step because you have no software or history to load. However, if you have any source for your new application, such as a prototype, you may want to migrate the source for historical reasons.

However, most project teams have large amounts of existing code to load, and the remainder of this section describes how to do this.

Prepare for the migration

Migrating an existing code base into Telelogic Synergy is one of the first steps in realizing the benefits of a comprehensive CM solution. However, the apparent simplicity of this first step highlights some key principles and tests them against the reality of what your team has been performing to date.

Work the plan

The implementation plan identifies the number and type of applications to be migrated into Telelogic Synergy. The various application stakeholders are engaged at this point in preparing for and assisting in the migration and test activities.

Lessons learned from the past

During the migration process, teams often discover issues with the quality of their code base and the repeatability of past releases.

These issues come as no surprise because the decision to adopt Telelogic Synergy was probably based on the recognition that the existing CM practices were inadequate. Do not be concerned if this part of the Telelogic Synergy implementation forces your team to confront past problems.

Teams are forced to address a number of issues and recognize deficiencies to ensure that the application is migrated. Teams and their members must not be made to feel inadequate and defensive because of their past omissions. Instead they should be supported and motivated to adopt Telelogic Synergy's sophisticated CM support so they can apply industry best practices.

The ultimate goal is to ensure that the code is loaded into Telelogic Synergy so that these kinds of issues become a thing of the past.

Engage the application's stakeholders

For each application to be migrated into Telelogic Synergy, the implementation's CM implementation team must ensure that the build manager or configuration authority is identified and consulted. This person must be able to answer the following questions about the application:

- What are the characteristics of the application?
- Where is the application source? Is the application team using a rudimentary version control system that is the repository for all the information or is it in the file system? What are the identifiers used to distinguish the different releases?

- When is the migration best performed? Is a current release being completed that is a good candidate, or must an incremental migration be performed later to get the latest changes?
- How is the application built? How are the releases identified? Which releases are still supported?

This information is collected as a part of the implementation planning, so there should be no surprises and ideally, the configuration authority is well informed of the planned migration.

The cooperation and assistance of the configuration authority and potential key development personnel are necessary to ensure that the right configuration is migrated with the correct identification.

Identify the correct configuration

Before migration commences in earnest, the application development team prepares by identifying the elements of the releases to be migrated and ideally includes design specifications, user manuals, and other documentation.

The easy solution might seem to be to migrate everything into Telelogic Synergy. However, migrating all version histories into Telelogic Synergy might not provide an accurate historical record. For example, identifying past releases or using Telelogic Synergy to reproduce them might be difficult unless that's the requirement and additional work is done to ensure it.

The best approach is to view the migration into Telelogic Synergy as a fresh start and an opportunity to clean up the configuration and consolidate efforts around the releases that are critical to the business.

Migration strategy

Determining the approach, order, and scope of migration is a part of the planning activities discussion. The time and effort required for the migration depends upon the adopted migration strategy.

Typically, the migration should be kept separate from the related but independent agendas identified below. Benefits to undertaking these activities must be made visible and become a part of the plan. They are not simply to be undertaken opportunistically because unexpected issues often arise when preparation is inadequate.

Restructure the code base

Some teams decide to totally restructure the code base prior to migration. This can cause the migration to be confusing because it will throw another unknown element into a team's use of Telelogic Synergy.

Avoid extreme measures. With a monolithic existing code base, you can obtain greater performance results by carefully restructuring. When the application can be represented as a component-based structure, you'll gain significant advantages in maintaining the projects. This can be important for applications that will evolve and grow over time.

A compromise position is to delay code restructuring until after the initial migration. This opens the possibility of using Telelogic Synergy to assist in code restructuring in parallel with mainstream development activities.

Regardless of the decision on restructuring the code base, the configuration authority must be a key part of the decision-making team and must be prepared to provide the resources to ensure that application integrity is retained.

Focus on business supported releases

Instead of migrating all history or every release, consider focusing the migration strategy only on the current releases that the customer's business supports. Telelogic Synergy is then the starting point from which teams develop the application with no baggage from older releases. Experience shows that when the migration effort includes the history, teams are delayed in adopting Telelogic Synergy and don't reap the benefits of CM best practices.

By initially migrating an application's supported releases, you can begin development by using Telelogic Synergy against the supported baselines. Avoid migrating recent releases that might impact existing schedule commitments.

The migration of past releases can be done in the background, so that it catches up with current development activities. This ensures that the migration of less critical and less volatile historical information does not delay the team's use of Telelogic Synergy.

The migrate dialog

The **Migrate** dialog in Telelogic Synergy Classic supports the migration of medium size applications. The facility allows the user to mark directories in the target source as Telelogic Synergy projects and migrate the whole structure as a single action. In other words, the **Migrate** dialog sees each source tree as a single project, but you can mark a directory to be created as a subproject.

Consider migrating all applications into Telelogic Synergy in the *working* state, so the configuration can be tested and validated before it is marked as released.

Manage and evolve the project structure

The following information sources describe how to migrate, set up, and manage the project structures. Because your application grows and changes, your project structure grows and changes to meet your evolving requirements.

- For general product usage information, refer to Telelogic Synergy Classic Help. The following bullets show where to find specific information about Telelogic Synergy functionality.
  - Migrating data

    Read Telelogic Synergy Classic Help available from the **Migrate** dialog. (You can read about the `ccm migrate` command as well.)
  - Create a project

    Read Telelogic Synergy Classic Help available from the **Create Project** dialog. (You can read about the `ccm create` command with the **-p** option as well.)
  - Add an existing project to another project

    Read Telelogic Synergy Classic Help available from the **Add Object** dialog. (You can read about the `ccm use` command as well.)
  - Set work area properties

    Read Telelogic Synergy Classic Help available from the **Work Area Properties** dialog. (You can read about the `ccm work_area` command as well.)
  - Convert an existing directory to a subproject

    Read Telelogic Synergy Classic Help available from the **Create** dialog. (You can read about the `ccm create` command as well.)
  - Copy a property to subprojects

    Read Telelogic Synergy Classic Help available from the **Attribute Copy** dialog. (You can read about the `ccm attr -copy` command as well.)
  - Create platform-specific variants

    Read Telelogic Synergy Classic Help available from the **Create Project** dialog. (You can read about the `ccm create` command as well.)

Additionally, refer to the *Build Manager's Guide* for information about managing multiplatform project hierarchies.

- Set up external projects, installation projects, and platform-specific variants

  Refer to the *Build Manager's Guide*.

About larger migrations

For larger applications, the time taken to register the numerous objects and structures in Telelogic Synergy can significantly slow the migration. Memory usage can be high, which increases the risk of failure due to inadequate computer resources. In this case you should split the migration into multiple sessions. For example, migrate a project but ignore one of its large directories that will be brought in as a subproject. In a separate session, migrate the large directory as a project, and then either add it from the GUI or use it from the CLI as a subproject of the first project.

When you migrate a large code base, plan your project layout so you can specify which projects are subprojects. (This is called a "bottom up" way of migrating.) If needed, split the migration into several sessions. Consider migrating the individual subprojects first, then assembling the higher-level projects by combining these subprojects.

Preview the migration

The **Migrate** dialog works either in a batch or interactive mode. The interactive mode allows you to preview the migration and ensure that all of the necessary types are present, the directory structure is the way you want it, and files that you want to ignore are identified. Use the interactive mode be used and study the preview carefully. The extra time needed to ensure that the data is migrated correctly is much less than the time required to rearrange the data after it is migrated.

Review the information carefully before the migration because correcting some information after the migration can be difficult, and making corrections after new versions have been checked out can be even more complicated.

Migrate applications and version histories

When you migrate applications that already have controlled versions created by another CM product, you must decide whether to migrate each object's entire history or only selected versions in each object's history.

If your project team already has version history data stored in either SCCS, RCS, or PVCS, you must decide how much of the version history to load into Telelogic Synergy.

There are advantages to exporting or extracting the files from the version control tool into the file system and performing the migration from these files. This technique allows the team to identify the precise configuration they want to migrate into Telelogic Synergy and use the same build tools that were used to create the release originally.

The three common approaches to loading in an existing version history are described below.

- Load latest version only

  The fastest migration is to load only the latest version. The disadvantage is that you do not have the history data in Telelogic Synergy. You are not able to perform operations, such as comparing new versions with old versions of the same object. From Telelogic Synergy's perspective, the project is starting with all new file versions.

- Load all or selected versions

  By default, the migration rules are defined to load all version history. The advantages of this method are its simplicity and ability to import the full history. The disadvantage is that it might import more versions than you want it to.

  Most basic version control systems do not distinguish between intermediate work-in-progress versions and versions that represent significant milestones. Therefore, migrating entire histories does not automatically generate previous releases. The set of object versions in a release might still need to be assembled.

  Because version control tools do not keep track of directory versions, a full migrate of the entire version history results in a directory structure that is a superset of all files that ever existed in the project. File deletions, moves, and renames are not reflected in the resulting Telelogic Synergy database.

  The duration of the migration is proportional to the amount of history that is loaded. Migration can be excessively time consuming when loading a large history.

- Load only versions that were for particular release milestones

  This option loads only versions corresponding to particular releases. To load release-specific versions, point the **Migrate** dialog to the first milestone

release. After the first milestone release is loaded, point the **Migrate** dialog to each milestone release in succession and run an incremental migration.

An incremental migration allows the current build tools to set up a previous release baseline that is ready for migration into Telelogic Synergy. Using your current build tools ensures that the extraction of specific versions for a particular release is validated against your company's current best practice.

If the scripts correctly retrieved and reconstructed each milestone release, which contained only the files in that release, you will not experience the directory version problem described above.

Convert keywords

If your current source code uses SCCS, RCS, or PVCS keywords, consider running a script that converts those keywords to Telelogic Synergy keywords. Keywords can be expanded when a check-out or check-in is performed, and they let you embed properties in the source automatically; therefore, you can easily propagate the properties to products such as libraries and executables.

Migrate vendor code

Many project teams build on code obtained from outside vendors. Vendor-supplied code can be either in the form of source code that gets built locally or libraries and header files that are linked with your code at build time.

Consider managing vendor code along with the local code, especially if the vendor code changes frequently and new versions are used. Updates from the vendor should be handled using an incremental migrate approach from the previous vendor release.

Code management might need to be a scripted process repeated at frequent intervals. Consider the potential advantages of holding your supplier responsible for the migration where you accept the code in a controlled environment, possibly with DCM.

Note This is more suitable for customized code rather than standard packages.

Migrate products

Marking controlled products is one of the keys to the task-based methodology. Decide which products to migrate and which to ignore. Typically, teams ignore intermediate products, such as `.o` or `.obj` files, and migrate libraries, DLLs, and executables. Implementers mark items as products with the **is_product** property.

Must mark the migrated versions as controlled products. To mark them, right-click over the product objects and open the **Properties** dialog. Select the **Is Product** option. (If you did not migrate the existing version of the products, you must do this the first time you build the products.)

The following is an example of marking an item as a product from the CLI:

```
$ ccm query -t executable "is_member_of(`top-toolkit/1.0')"
$ ccm attr -c is_product -type boolean -value TRUE @
```

Build products

One of the key decisions project teams must make is whether to use a third-party build tool approach to build their applications or continue to use the method they were using before converting to Telelogic Synergy.

Build with a third-party build tool

If you choose the third-party build tool approach, you must manage your existing makefiles, insert steps into your build scripts to ensure that the products are checked out and can be modified before the build, and ensure that the products are checked in after they are updated.

Problems reported by your *make* tool are probably due to the new location of the files. Search for absolute paths in your makefiles to determine whether you must add include search directories so that header files can be found.

For copy-based work areas, set the work area options **Use New Timestamps** and **Make Copies Modifiable** when using third-party build tools. Also, reconcile the project or directory after the build if you are controlling any products.

Verify and test the application

The verification and test step ensures that the products built from within the Telelogic Synergy environment are the same as the ones that were built previously. They may not be built identically, but the end product must be identical.

There are several reasons for this step:

- The development team must have confidence that they are building from a solid foundation. Developers want to know that problems introduced during the construction of release **toolkit/2.0** result from work on the **toolkit/2.0** release, and not because the **toolkit/1.0** baseline was partially or incorrectly migrated.

- The quality assurance team wants to ensure that the baseline from which future patches are made has been validated; then they can focus on identifying regressions and testing a specific patch instead of testing the full application each time.
- Project management wants assurance that the full application is building correctly before making it available to the users.

System validation

To ensure that you have a good baseline for your application, validate both your project structure and your build products.

Validate project structure

In this stage, members of both the development team and quality team review the project structure, the build process, and the resulting products. It is an excellent time to begin educating key members of the team about the CM system and incorporate their feedback. Developers especially live in the project structure, so they must understand the structure and believe that it is a usable and effective representation of their applications.

Validate build products

Validation is necessary for confirming that the Telelogic Synergy implementation is behaving as expected. Validation and test tasks often identify some missing files, determine that an executable was linked with incorrect options, or find a library that was not migrated. Sometimes the problems are quite subtle. For example, you might discover that you had been building in your previous system with different versions than you thought, or that developers were building with personal versions of makefiles rather than with the controlled versions.

If mistakes in the original file system structure or build process are discovered, you must echo these errors in the initial version of the product in Telelogic Synergy so that you can reproduce the same results as in your original baseline. Correct these errors in a new version of the product.

The situation might require you to go through a full software validation cycle. If you are loading legacy applications that are still undergoing maintenance, you probably want a full validation cycle.

The time required for validation is a function of the stage the project is in when Telelogic Synergy is deployed. Ideally, the team validates that products built with Telelogic Synergy compare to a previously built version of the product. However, when software is migrated into Telelogic Synergy after a code freeze, but before a release milestone, there is no release to compare and validate against.

Rebuild and retest after any changes to the structure, project data, or build process/makefiles. Iteration during this phase is very common. The next step is to turn your project into a frozen baseline upon which to build future work. By the end of this phase, ensure that the application represents and builds the way it was designed.

Create an initial baseline

After the application structure is finalized, the project data has been verified as successfully migrated, and the products are building properly, you are ready to baseline the initial application project hierarchy.

The *Build Manager's Guide* describes the steps for creating a baseline. You must release, or check in to the *released* state, your full project hierarchy of source objects, products, and projects.

Note Now is an ideal time to take a snapshot of Telelogic Synergy databases that contain the baseline and to validate your backup strategy. Refer to the description of backup strategies and commands in the *Telelogic Synergy Administration Guide.* Test your backup procedures and ensure you can revert to this version of your database.

Set up releases

When a working baseline is available in Telelogic Synergy, set up the release preparation areas for the various releases that the project team will produce. Set this up beforehand so everything is ready and running smoothly when you go live.

The amount of work depends on how many variants you produce for each release. For example, the number of platforms you support and how many releases your team is producing in parallel will impact how many release areas you'll need to create. A team producing Windows software for a new product release might have a single variant and a single release, but a team building ten different platforms with three simultaneous releases has a *prep* project or *prep* project hierarchy for each combination.

Create the release preparation projects in the order described in this section. Although you do not have create them in this order, it makes the process easier.

Normally, the integration build management projects store the latest checked in software, which has the latest completed tasks.

Refer to the *Build Manager's Guide* for detailed, step-by-step instructions.

Support different platforms

For each platform you support, create a version of the project hierarchy for that variant, for example, Windows or Solaris. Only projects that are variant-specific need their own *project version*. That is, projects that contain platform-independent data, and have absolute subprojects can be shared. You do not need different versions for each platform.

The exception occurs when you have work areas for both Windows and UNIX; platform-independent projects can usually share a single work area for all UNIX platforms across NFS, but you usually need a separate Windows version of the platform-independent projects just to get a Windows work area.

Use the **Check Out Project** dialog in Telelogic Synergy Classic, **Copy Project** dialog in Telelogic Synergy, or `ccm copy_project` command to copy the variant project hierarchy in one step. Set the **release** and **platform** values, and set the version to a meaningful value. For more information on checking out integration testing projects, refer to the *Build Manager's Guide*.

Create parallel release projects

For each project variant, create a version of the project hierarchy for each parallel release. You defined the release earlier, when you configured the development workflow. For each release, create a version of the project hierarchy. For the Windows platform variant, you might have releases **billing/2.1** and **billing/3.0** under construction for the **billing** project. If so, you'll have two projects: **billing-int_win_2.1** and **billing-int_win_3.0**.

Use the **Check Out Project** dialog in Telelogic Synergy Classic, **Copy Project** dialog in Telelogic Synergy, or `ccm copy_project` command to copy the variant project hierarchy in one step. Set the release property on the project version and the project purpose to *integration testing*.

For information on creating the build management projects for each parallel release, refer to the *Build Manager's Guide*.

Create system test build management projects

System test build management projects are used for system testing. They are created much the same way as integration build management projects. In the **Check Out Project** dialog in Telelogic Synergy Classic or **Copy Project** dialog in Telelogic Synergy, select **System Testing** as the purpose and set the release, platform, and version. As with integration build management projects, create one system testing project hierarchy for each platform and release. System test build management projects contain tasks that are ready for system testing. Start from a

baseline created from an integration testing hierarchy, the set of tasks that passed integration testing, or from a specific list of tasks.

Automate build management

Much of the work for which the build manager is responsible can be fully or partially automated. You may want to automate activities to be completed during the night when computing resources are more available. The results are then available to the development team first thing in the morning.

One of the build manager's main responsibilities is producing the integration test area and the system test area.

Note Your development process may call for more, fewer, or different test areas, but the concept remains the same.

Quality assurance engineers normally use the integration and system test areas for testing, but developers can also use one or both test areas for unit testing.

Automation means writing scripts or batch files that call Telelogic Synergy commands and other commands, such as file system commands, to search log files for error messages.

Note If you use both Windows and UNIX development platforms, consider using a multi-platform language, such as Perl, for your scripts.

The following list contains just a few of automation's many advantages:

- The process is reproducible.
- The process is not subject to manual errors.
- You can schedule the process to run at convenient times, such as at night. You can use `cron` on UNIX or `at` on Windows to run the process at a scheduled time, such as every night at midnight.
- In the absence of the build manager, others can run the process.

Integration testing

The integration test area is usually built nightly. The goal of the integration testing cycle is to find problems as early as possible. The normal integration testing process involves the following actions:

1. Update your project to get the latest completed tasks from the development team.
2. Build the controlled products.

This might include packaging; for example, creating a CD image. If you created an installation project, update it after building the products, and then use it to package the products.

- If the build is successful, run a base level of tests.
- Review the results of the build and tests, and if they are good, proceed to the next steps.

3. Make the tasks that passed testing available for use by the development team.
   - Create a baseline from an integration testing hierarchy. Refer to the *Build Manager's Guide* for more information.
   - As an option, create an installation with the latest integration test software that is available for the team's use. Developers and testers can run on the previous night's build, and developers can update their personal work areas to include the submissions from their team members.

Refer to the *Build Manager's Guide* for detailed information about the integration test process.

System testing

The goal of the system testing cycle is to obtain a version of the software that meets specific quality criteria, which is often used to prepare for a release or some other milestone. Unlike the integration test area, the system test area is normally built on demand. Also, rather than including everything developers submit, as the integration build does, the system test area is more selectively updated. The build manager needs to specify which tasks to include, thereby providing the critical level of insulation for fixing defects without risking the introduction of changes beyond those required to solve specific problems.

The normal process for the system test area includes the following actions:

1. Identify which tasks to test and prepare the system test area with those tasks. Set the baseline to a particular integration testing baseline. Refer to the *Build Manager's Guide* for more information.
2. Update the test area to include specified tasks.
3. Build the controlled products and create an installation image if needed.
   - If the build is successful, test the resulting product to determine the quality level of the software and to verify that the functionality expected for this version of the software is included.
   - If the tests identify critical problems, create and assign tasks or change requests, and identify which are required for the release or milestone.

When the required tasks are completed, add them to the system test task folder and repeat the update, build, and test process.

After the system test area has all the features and fixes needed for the release, and it has passed testing, the release is ready and the system test build management projects can be released or baselined.

Refer to the *Build Manager's Guide* for detailed information about the system test process.

Automated systems administration

Only a few systems administration activities are required for the ongoing maintenance of Telelogic Synergy. These activities are normally automated using nightly scripts or batch files that run by `cron` jobs or another job scheduler. The *Telelogic Synergy Administration Guide* describes all of the necessary systems administration processes, but the following are the two most important jobs.

Nightly backup

All production Telelogic Synergy databases must be backed up regularly. Excellent recovery capabilities are available, but successful recovery requires consistent backups. At some point, most development organizations experience a media failure, such as a hard disk crash. With the backup capabilities in Telelogic Synergy, you can recover from a media failure, including the last transaction made before the failure.

Routinely monitor for database integrity. Use the `ccmdb check` command as part of the nightly backup and the `ccm fscheck` command at least once a month.

Refer to the *Telelogic Synergy Administration Guide* for a description of backup strategies and tools. Ensure that these procedures are in place and validated by this point in the Telelogic Synergy implementation.

Database cleanup

Although the Telelogic Synergy product has powerful capabilities for sharing products, single-instance source pools, and delta and compression mechanisms, Telelogic Synergy databases can grow very quickly under some circumstances. In particular, when many developers work on multiple parallel releases and the rate of change is fast, many new versions, especially of controlled products, are created each day. Decide on a strategy for identifying obsolete products and source, and then purge older versions to reduce database sizes.

Telelogic Synergy provides administration commands to "garbage collect" unused versions. This function is normally performed at night. Several options

enable you to control which objects to remove based on their age. For more information, see the **Save Offline and Delete** dialog's **All unused products for a specified release** scope in Telelogic Synergy Classic Help.

For example, the following commands query for and remove controlled product versions that are not members of any project in the database:

```
$ ccm query "is_product=TRUE and not is_bound()"
$ ccm collapse @
```

Perform load testing

By now, most project teams can see that they are almost ready to go live, and users are ready to learn how to use the system and move into production. Consider taking some preparatory actions to ensure that the roll-out goes smoothly.

Only a few users have probably been on the system, and unless computing resources are extremely scarce, the users have performed well. However, during the next stage, the number of users and overall system activity will probably increase dramatically, so the system load must be tested before end-user training or live use.

Insufficient planning and preparation can result in a Telelogic Synergy implementation that does not scale. The failure to scale can easily give end users a negative first impression if they encounter trivial problems caused by a poor or inappropriate setup.

System resources

Sometimes a site discovers resource issues, such as insufficient memory or swap space on some machines or servers not configured for enough processes. Ideally, these problems are predicted and corrected early in the system resource planning stage; however, the system is not subjected to a typical load until sometime in this stage in the implementation.

Sometime before end-user training and going live, the CM implementation team should stage a test in which the maximum expected number of Telelogic Synergy sessions are started on the actual systems that are running a typical load. For example, if your project team includes 120 developers and testers, and you expect that 75 is the maximum number of simultaneous sessions, the test team should start 75 sessions on the same network with the same servers to be used when running live and conducting training.

Individual workstation test

Verify that each end-user system is tested before the user starts training or goes live. If the system is a PC with a Windows operating system, the client is installed on that system. Network access is possible, but for performance reasons, install Telelogic Synergy locally when required.

Ensure that the test includes starting the user's common development tools, such as ™, Visual Studio®, or JBuilder for developers and WinRunner or TestDirector for testers. If the user normally runs desktop applications such as e-mail, MS Word, or an editor, confirm that these programs are also running.

Any number of potential problems can exist: specific system components, such as networking, have not been installed; the system is not configured for the appropriate domains; or not enough RAM or disk space is available for the system load the user has on the machine.

The key is to ensure that each user's systems perform acceptably during peak load periods before you proceed to the next step.

4 Usage

At this stage, your CM implementation team, and especially the build and release manager, should be fairly comfortable using the product and explaining its use to other personnel. All the project data is loaded, and you are now ready to train the developers and other users.

Average developers need to know only a few Telelogic Synergy commands to perform their daily development tasks. However, developer training is still important because the developer is operating in the context of an overall team, and a primary focus is team support and overall team productivity. Developers must understand the workflow of software through the team members, and how their changes eventually get to the quality team, then to the release manager, and finally to the customer.

Interfaces installed in Telelogic Synergy 6.6a

Telelogic Synergy offers several different graphical user interfaces, as well as a command line interface. The following graphical user interfaces are included in the Telelogic Synergy 6.6a release:

- Telelogic Synergy

  This interface is for users in the *developer* or *build_manager* role. It contains functionality for daily development and build management activities.

- Telelogic Synergy Classic

  This interface contains functionality for build managers and CM administrators. This interface is no longer being enhanced.

Timing the training

The CM implementation team receives its training before the deployment stage. Timing the developer training is equally important. Developers should learn the system immediately before going live. If developers are trained too early, they might forget much of what they learned by the time they are ready to use the system with their projects.

Training data

Ideally, the development team is trained on site with actual project data that replicates the production environment. This training is usually done with a copy

of the production database so that student developers feel free to try anything without fear of doing permanent damage. In fact, making mistakes benefits student developers because they can see the consequences and how changes can be undone.

However, customers often find it convenient to use the training material provided as part of the formal courses offered.

Developer training strategies

Customers must recognize the risk associated with the inadequate or delayed adoption of Telelogic Synergy and determine the appropriate training strategy.

This section describes three schools of thought on developer training:

- Provide a thorough, formalized, developer training course.
- Provide a three-hour, localized, training course followed by hands-on use.
- Give the developers a manual and point them to the *Tutorial*.

The following sections describe each of these approaches. The approach that is best for your project team is a function of several factors, including the experience level of the developers, the culture of your organization, the level of awareness of CM concepts and benefits, and the time available for training.

Consider providing a combination of the three training types. After receiving the formal training, users often benefit from individual study using the tutorial to cover any areas they might need to revisit. Finally, a follow-up "chalk talk" discussion, based on the local configuration and usage of Telelogic Synergy at the customer site, prepares them to work effectively.

Note Users should train on the types of interface they will use. For example, most developers are trained using Telelogic Synergy, and build managers are trained using Telelogic Synergy and Telelogic Synergy Classic.

Localized developer training

For many teams, the ideal level of training consists of a morning session with approximately three hours of presentations, demonstrations, and white-board discussions using actual project data. This is followed by an afternoon session when developers use the system at their desks, while an experienced user roves from office to office answering questions and providing tips and shortcuts.

This activity is a part of user-mentoring and can be conducted by a skilled member of the CM implementation team or by a consultant.

Self-guided developer training

The *Telelogic Synergy Tutorial* provides a self-guided introduction to the most common developer and build manager activities. The guide includes modules of typical operations that developers use most in Telelogic Synergy. Important terms and concepts are described in the *Introduction to* Telelogic Synergy. Although tool-savvy developers can usually get by with just the guide, most organizations use the guide with either formal or localized developer training. Check the Support site for the most current version of the *Telelogic Synergy Tutorial*.

Developer training essentials

Regardless of the form of training you choose, cover the following important topics. Consider supplementing these topics with additional topics that are important to your project teams.

Terms and concepts

During training, ensure that you define the following terms and concepts:

- Object types
- Projects and work areas
- Personal and release prep work areas
- The task-based workflow
- Roles and security — that is, who can do what

Tool operation

Discuss the following tool operations during your training session:

- The graphical user interface to be used
- The command line interface, if needed

Developer actions

During your training session, discuss the following developer actions:

- Creating and completing tasks
- Making a change
- Adding a new file
- Incorporating teammates' changes
- Managing work areas

- Building and testing

Go live!

The team is now ready to go live and begin using Telelogic Synergy in a production environment. This last step is often carried out in parallel with the user training.

The following sections take you from implementation to live usage.

Migrate incremental changes

The development team has probably made additional changes after the baseline was migrated into Telelogic Synergy. Before Telelogic Synergy is put into production, changes must be loaded into Telelogic Synergy so that the developers go live on the most current data.

The **Migrate** dialog has an incremental migrate option designed for this purpose. The option examines your code base and applies only those versions that are newer. This is useful when you do not have an easy way to know what files have been changed. If Telelogic Synergy is replacing an older generation tool, the easiest approach is to extract the source and perform an incremental migration from the file system. You can perform an incremental migration by using the **Reconcile Work Area** dialog in Telelogic Synergy Classic, **Sync Work Area** dialog in Telelogic Synergy, or the `ccm reconcile` command, if the version history is not needed.

If you already know the set of changes that were made during this period, you can develop a simple script to load the new versions.

Alternatively, some teams have developers reapply their changes as part of the training process. Because they already know the changes they made, they can concentrate on the CM process involved in creating work areas, checking out files, applying the changes, and checking in the modified files. If you take this approach, verify that all the changes were applied.

Activate users

You are now ready to open the gates to the production database. To do this, edit the users list to add developers, testers, writers, project managers, and other users. If the database is protected, you must unprotect it.

Even new users who have undergone training will need some support when they first start using Telelogic Synergy. The more competent members of the CM implementation team should be on hand to help with the transition to Telelogic Synergy.

Create personal projects

Developers need at least one dedicated area in which to work. A developer creates this area by copying the desired project from the integration testing project for the desired release and variant. For example, when user Ann wants to work on the WIN32 version of the **billing/3.0** release of the **billing** project, she copies out the project **billing-ann_win_3.0** from the **billing-int_win_3.0** project.

Creating personal projects is often the first action in the developer training. Personal projects can be created in advance by the build manager, but he must ensure that the projects are created in a session that is running as the user so that the project and work area permissions are correct, so this is normally done either *with* the user or *by* the user.

The CM process evolution

Continuous process improvement is important. Your CM needs will change over time as your company grows, you expand your platform coverage, you support more releases, your code base grows, your team spreads to multiple locations, you out source, and myriad other reasons.

Telelogic Synergy was designed to grow and evolve along with you. Be sure to review your CM process after each major release milestone and ask yourself if your CM process needs adjusting to reflect changes in the way you produce or want to produce software. Adjusting Telelogic Synergy to reflect these changes is usually easy. The improved level of automation that results from continuous improvement is significant.

Appendix A: Implementation questionnaire

This section contains a list of information needed from each project team to perform the implementation tasks described in the preceding chapters. You can use the questionnaire to help gather the necessary information.

Source code

These questions pertain to the source code you anticipate bringing into Telelogic Synergy. The answers help determine the amount and type of code to migrate into a Telelogic Synergy database.

1. Approximately how many source files make up a release of your application?

   This number should include test files, documentation files, and other files except object files, executables, libraries, or other generated files.

2. Approximately how many lines of code in total make up a release of your application?
3. How many executables are built for a release?
4. How many libraries are built for a release?
5. How much space, in kilobytes, does one release of your application use? Include all files of all types used to create the software application, such as source, objects, documents, tests, and others.
6. List the types of files you use in your application, for example, Java or C++ source, C++ headers, graphic images, XML, Perl scripts, database SQL, and UML tool data.
7. How many different languages are used in the application, such as C, C++, Java, and others?
8. List any third-party code you use in your application that you may want under Telelogic Synergy control, for example, third-party libraries.
9. Do you currently use any version control technology, for example, PVCS, SourceSafe, SCCS, RCS, CVS, or custom? If so, what?
10. If you currently use a version control system, what are your current releases? Which past releases are currently supported?

Build environment

The following questions pertain to the way you currently build your application.

1. What is your current build mechanism, for example, ANT, Make or shell scripts?
2. If you use *make*, what version, for example, Native platform, Sun make, or GNU make?
3. Do you use any source code generators or special compilers, such as UML tools or a GUI builder? If so, which ones?
4. Do you deploy your code to an application or web server? If so, which ones?

Project environment

The following questions pertain to some additional factors in your development environment.

1. How many developers are on your project?
2. How is the project team organized?
3. Where do developers work? One or more locations? Across campus, town, country, international? How do they currently share their work? What is the frequency with which they need to synchronize?
4. How many releases are you developing in parallel?
5. What development platforms do you use?
6. What platforms do you use for builds?
7. What delivery platforms do you use?
8. What other projects does your project depend upon, for example, do you use source or libraries from other project teams?
9. Is a development team responsible for one or more component? How many components are there? How is the system organized?

Appendix B: Technical support

Contacting IBM Rational Software Support

Support and information for Telelogic products is currently being transitioned from the Telelogic Support site to the IBM Rational Software Support site. During this transition phase, your product support location depends on your customer history.

Product support

- If you are a heritage customer, meaning you were a Telelogic customer prior to November 1, 2008, please visit the Synergy Support Web site.

  Telelogic customers will be redirected automatically to the IBM Rational Software Support site after the product information has been migrated.

- If you are a new Rational customer, meaning you did not have Telelogic-licensed products prior to November 1, 2008, please visit the IBM Rational Software Support site.

Before you contact Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, or messages that are related to the problem?
- Can you reproduce the problem? If so, what steps do you take to reproduce it?
- Is there a workaround for the problem? If so, be prepared to describe the workaround.

Other information

For Rational software product news, events, and other information, visit the IBM Rational Software Web site.

Appendix C: Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of

performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, Telelogic, Telelogic Synergy, Telelogic Change, Telelogic DOORS, Telelogic Tau, Telelogic DocExpress, Telelogic Rhapsody, and Telelogic System Architect are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html.

Citrix and other Citrix product names referenced herein are trademarks of Citrix Systems, Inc. and/or one of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

is a trademark or registered trademark of the Foundation.

Informix is a trademark or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

FLEXnet is a registered trademark or trademark of Macrovision Corporation.

Microsoft, Windows, Windows XP, Visual Studio, and/or other Microsoft products referenced herein are either trademarks or registered trademarks of Microsoft Corporation.

Sun, Sun Microsystems, Solaris, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product or service names may be trademarks or service marks of others.

Index

E

F

H

I

M

O

P

R

S

T

U

V

W