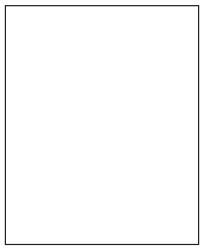
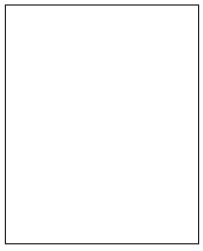
Configuration Management with CCC/Harvest

By Rich Hug and Helen Ernst

This is the second part in a two-part series. Part one introduced Configuration Management (CM) and outlined its five main areas: change and version control, application management, package management, dependency management, and process management. It also highlighted the benefits to organizations of implementing a well-conceived CM procedure. This part describes CCC/Harvest®, a new client/server configuration management solution from Softool® Corporation.



Rich Hug



Helen Ernst

Developing software in a client/server environment presents unique configuration management challenges. Most existing CM solutions fail to address the needs of client/server computing adequately. There are several reasons for this:

- **Usability**: Traditional CM tools, usually targeted for a particular user population, follow one set of usability standards. Their lack of versatility makes them unappealing to users with different backgrounds—a common situation in networked environments.
- **Openness**: Many CM tools do not provide the open architecture and Application Programming Interface (API) required to integrate easily with other development and maintenance tools.
- **Breadth of support**: CM tools have traditionally focused on the needs of developers, without providing adequate support for other groups involved with the development and maintenance process.
- **Adaptability**: Many tools in current use are too rigid to adapt to the diverse procedures and processes in a distributed environment.

CCC/Harvest, the latest in a series of configuration management tools from Softool Corporation, was designed specifically to address the inadequacies of current CM solutions in client/server environments. This article explores how CCC/Harvest meets these needs.

Usability

The CCC/Harvest user interface was built using state-of-the-art Graphical User Interface (GUI) technology. Figure 1 shows a sample screen. Ease-of-use and minimized training time were primary design criteria. The interface uses multicolored icons and tool bars throughout. It also includes a full online context-sensitive help facility. There are keyboard equivalents for those who do not like using a mouse.

Most importantly, the "look and feel" of the GUI can be changed by users according to their preferences, since Motif, OpenLook®, and Windows™ are supported. Users also have complete control over fonts, font size, and color schemes. For those who prefer it, CCC/Harvest functions can be executed through a command-line interface, built on the product's API.

CCC/Harvest has an object-oriented design. Associated with objects are *methods*—actions that can be taken against the object. Objects are represented as icon buttons or selection lists. The object-oriented approach allows users to apply multiple unrelated methods to individual objects or groups of objects from a single location. It also means that the user never has to remember the name of an object—all are available through lists.

The CCC/Harvest GUI has three entries, each suited to a particular group of users. Administrators approach CCC/Harvest through its main window. Developers can use the Workbench, which

provides easy access to the most common development functions. Individuals involved with problem management and scheduling can enter CCC/Harvest through the File Cabinet, which maintains documents associated with change requests.

Open Architecture

CCC/Harvest is a client/server application. The client and server portions can both be executed on the same machine or be distributed across multiple platforms. The client portion of CCC/Harvest consists of the GUI, command-line interface, and API. The server portion contains most of the program logic, including the delta engine that allows CCC/Harvest to calculate and store the differences between one version of a file and another. The server portion also includes an SQL layer that communicates with a relational database.

CCC/Harvest's open architecture allows easy access to CM information. Rather than developing yet another database standard, CCC/Harvest stores control information within commercially available relational databases. It currently supports the Oracle® database and will soon support databases from Sybase® and Informix®. Users access database information from the GUI or API. A site can access the database directly to generate reports. Database information can be integrated with other development tools.

Figure 2 shows the various components of CCC/Harvest and how they work together.

Broad Support

CCC/Harvest allows configuration management to be part of the change process. CM is not "added on" at the end of a process; rather it must be built in. CCC/Harvest provides a unified framework that supports all the core functions in the software development and maintenance process. The framework includes problem reporting and tracking, inventory management, change and version management, application management, release and baseline control, support for parallel and concurrent development, and the distribution of updated software and related information.

All functional groups involved in software development and maintenance can benefit from CCC/Harvest. The program includes extensive management reporting capabilities and support for the auditing functions. Usability features, described earlier, make it easy for non-technical groups to use.

CCC/Harvest's integrated approach is especially significant for problem tracking. Users manage problems rather than passively track their progress. Users can control how problems are reported, how they are assigned and scheduled as changes, and how these changes are implemented. All the history associated with a problem that occurred when initiating a software change is immediately available in the development environment. The easily obtained history makes it simpler to determine which problems motivated the user to make a change request. It also simplifies the process of identifying related problems.

Adaptability

Networking has greatly expanded the number of computing platforms that can work together. It has also compounded the complexity and diversity of the software life cycles that take place on them.

Adaptability is one of CCC/Harvest's most powerful features. An organization can define

What's In and What's Out in Software Change and Configuration Management?

How are organizations keeping up with the 1994 trends in software change and configuration management? One quick way to find out is to check what Softool says about what's in and what's out.

Change and configuration management begins by providing for the management and prioritization of problems and enhancement requests. Then, as software change requests are authorized, CM provides for the orderly notification, initiation, review, testing, approval, migration, and integration of changes. CM also analyzes the impact of changes before and after they are implemented and supports complete audit trails.

To enhance change and configuration management operations, Softool's advice is to focus on successfully implementing the items in the "What's In" column.

What's In	What's Out
Client/server	Disconnected heterogeneous environment
Tools that adapt to the environment	Tools that enforce a specific process
Well-planned implementation	Forced CM to end users
Ease-of-use for end users	Complicated interfaces
Integration with problem management	Proprietary CM databases
Proper end-user training	Learning on your own
Software distribution management	Home-grown CM solutions
Life cycle management	Version control only
Tools with open interfaces	Self-contained tools

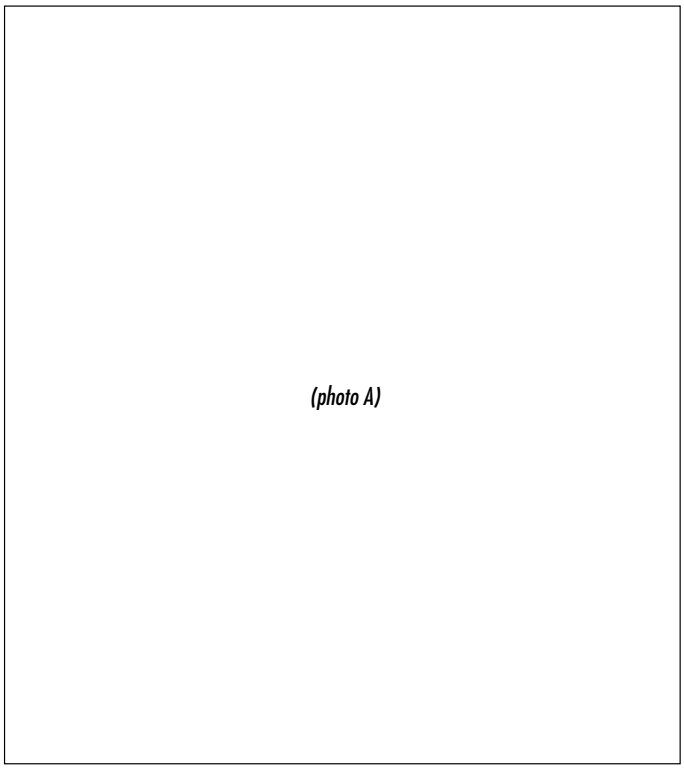


Figure 1. Sample CCC/Harvest user interface

CCC/Harvest Architecture

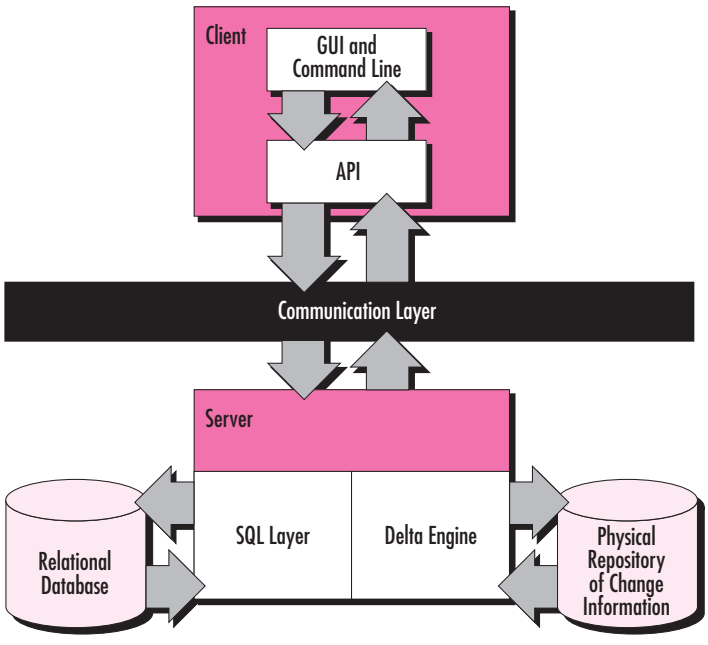


Figure 2. CCC/Harvest architecture

and follow a software life cycle according to the way it is accustomed to doing business. Furthermore, CCC/Harvest allows a site to synchronize development activities across all platforms involved in development, maintenance, and problem tracking activities. It offers true interoperability. Software can be developed on one machine, tested on another, and finally distributed to run on a third—all under one centralized control layer.

Development models are built from the objects provided with CCC/Harvest. These objects include environments, life cycles, views, processes, packages, package groups, and forms.

Building a development model is a setup activity that typically occurs only once and does not require any programming skills. CCC/Harvest has two default models that can be used "off the shelf." These models can also be used as starting points for creating models more suited to a particular site's needs and priorities. CCC/Harvest provides equal support for users building a model from scratch.

CCC/Harvest makes no assumptions about how development and maintenance activities occur. Users can define a software life cycle with many stages or with few. The user also defines processes that can occur during each stage. Each site establishes notifications, access control, and approval requirements.

CCC/Harvest solves one of software development's most long-standing battles: the terminology war. Each process can be named by the site using it. For example, the process of "check out" moves a stored file to a user's working context. If users do not like the term "check out," they can change it to "export," "extract," or "get." The chosen term will appear on the dialog screens associated with that function. This flexibility facilitates the seamless integration of CCC/Harvest with a site's existing procedures.

Building a Development Model

In CCC/Harvest, the environment object defines the control framework driving the development and maintenance process. Each environment has a master view, which delimits the items to be operated on. The items themselves are maintained in a physical repository, which can be shared by more than one environment. A master view can also include items from more than one physical repository, if needed, as with shared code.

Each environment also has a life cycle, consisting of a set of phases or states and the valid transitions between them. Examples of states are analysis, development, testing, unit testing, quality

The Release Environment Model

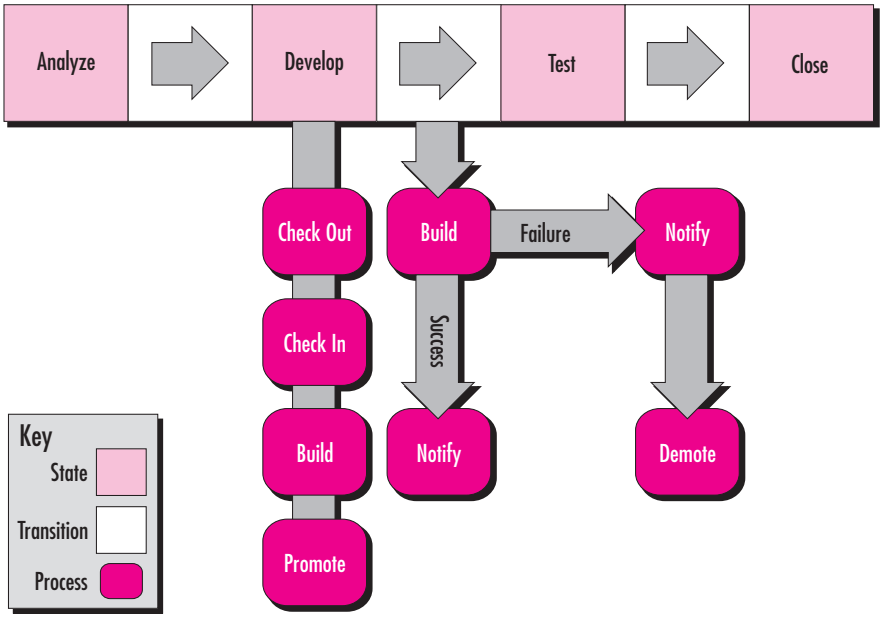


Figure 3. The Release environment model

assurance, and closure. A set of valid processes can be defined for each state and state transition in the life cycle. Although CCC/Harvest has a set of processes including check out, check in, merge, notify, approve, promote, and demote, users can also define their own processes and specify when they should be invoked.

Processes can be linked together dependently to perform complex tasks with little user input. Figure 3 shows the life cycle model of an environment named `Release`. This life cycle has been defined with several states, beginning with `Analyze` and ending with `Close`. The `CHECK OUT`, `CHECK IN`, `BUILD`, and `PROMOTE` processes can take place within the `Develop` state. Since these processes are optional, they can be invoked any time during the `Develop` phase of the `Release` life cycle. The `PROMOTE` process causes a state transition to occur.

Processes are also assigned to the transition between the `Develop` and `Test` state. The program invokes these processes automatically in a dependent fashion. The `NOTIFY` process occurs if the `BUILD` is successful. If the `BUILD` process fails, the `NOTIFY` and `DEMOTE` processes are invoked and the state is returned to `Develop`.

The unit of change within the life cycle is called a *package* and generally represents a change request or software trouble report. Several packages can be operated on as a unit by placing them in a group. A form for managing package information is associated with a package group. This form provides the link between the problem-tracking and change-tracking functions in CCC/Harvest.

Some information is supplied by users on a package form and the rest is supplied and updated by CCC/Harvest automatically as the package moves through its life cycle. For example, information about the current state of a package and its changes is automatically placed in the form. For more information about CCC/Harvest or Softool's other products and services, call (805) 683-5777.

***Rich Hug**, Softool Corporation, 340 S. Kellogg, Goleta, CA 93117. Mr. Hug, vice president of research and development, has over 15 years experience in developing CM solutions for large and small corporations. He has a Bachelor of Science degree from the University of California at Santa Barbara.*

***Helen Ernst**, Softool Corporation, 340 S. Kellogg, Goleta, CA 93117. Ms. Ernst, a senior member of the technical staff, has 8 years experience performing configuration management activities in many types of environments. She has a Bachelor*