Part I

What Is Configuration Management?

Configuration, "to form from or after," derives from the Latin *com-*, meaning "with" or "together," and *figurare*, "to form." It also means "a relative arrangement of parts or elements." Configuration management therefore refers to managing a relative arrangement of parts or elements. It's as simple as that.

Configuration management, as we know it today, started in the late 1960s. In the 1970s, the American government developed a number of military standards, which included configuration management. Later, especially in the 1990s, many other standards and publications discussing configuration management have emerged.

In the last few years, the growing understanding of software development as a collection of interrelated processes has influenced work on configuration management. This means that configuration management is now also considered from a process point of view.

Chapter 1

Definition of Configuration Management Used in This Book

There are many definitions of configuration management and many opinions on what it really is. This chapter describes the definition on which this book is based. In short:

> Configuration management is unique identification, controlled storage, change control, and status reporting of selected intermediate work products, product components, and products during the life of a system.

Figure 1–1 shows the activity areas included in the definition of configuration management used in this book. It also shows their relations to each other, to common data, and to elements outside the configuration management process area.

When you work professionally with configuration management (as with anything else) it's important to have the fundamental concepts in place. If all else fails, you can go back and seek a solution there.

Definitions of configuration management used in various standards are covered in Chapter 3, and definitions of configuration management used in various maturity models can be found in Chapter 2. The definitions in these standards and maturity models are similar to a large extent. However, they're expressed in slightly different words and with different divisions between the detailed activities that constitute configuration management. It's perfectly okay for a company to use its own definition of configuration management, but it's a good idea to investigate how that definition maps to the definition used in this book and other relevant definitions, to make sure no activity has been left out.

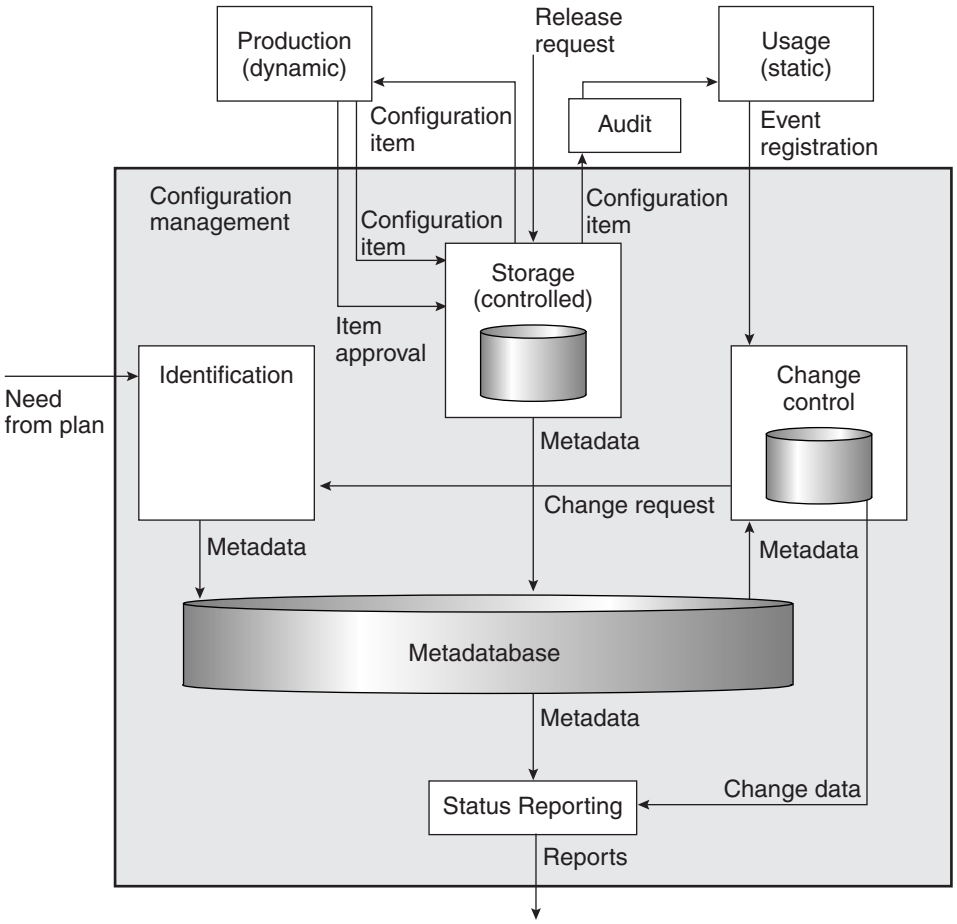

Figure 1–1 *Overview of Configuration Management Activities*

1.1 CONFIGURATION MANAGEMENT ACTIVITIES

The view on configuration management in this book is process oriented. Therefore, the definition includes activity areas, which can be described in terms of process descriptions. The activity areas described in detail in the following paragraphs are identification, storage, change control, and status reporting.

Configuration management has many interactions with other development and support processes. Figure 1–1 illustrates the production and usage activity areas via their respective libraries.

Metadata

All the activity areas in configuration management share metadata for items placed under configuration management. Metadata is a database concept that means data about the data stored in the database. So metadata in this context describes the configuration items. Metadata for a configuration item may include its name, the name of the person who produced the item, the production date, and references to other related configuration items. Figure 1–1 shows a logical separation of metadata, even though this data is often stored physically at the same location (in the same database) as the items in controlled storage.

Change control uses metadata—for example, the trace information for a configuration item for which a change is suggested. Change control does not in itself contribute to metadata, because information produced during change control will be present only if a configuration item is affected by a suggested change. A configuration item can exist without change control information, but it can't exist without metadata.

Configuration Management Is Cyclic—or Is It?

In everyday language, "configuration item" is often used to refer to an item, which is then said to be produced in several versions. This is not strictly correct, but it's acceptable as long as the reference is clearly understood by all involved. In fact, each new version of a configuration item is a new configuration item in its own right.

This can be illustrated by an analogy to an object-oriented approach. The "configuration item" may be seen as a class and the versions as instantiations of the class, as shown in Figure 1–2. Version chains of configuration items—that is, versions 1, 2, 3, and so on—may be formed by indicating which configuration item a given configuration item is derived from or based on.

Configuration management activities may be viewed as cyclic for each item class placed under configuration management. This means that a configuration item class continuously goes "through the mill." The first cycle is initiated by a (planned) need for a configuration item, and later the driving force is a change request (and only this!). This is illustrated in Figure 1–3.

In each cycle, a configuration item will be identified, produced, stored, and released for usage. Event registrations will occur as a consequence of experience gained during usage. These will lead to change control and the creation of change requests, which will lead to identification, and so on, of a new version of the original

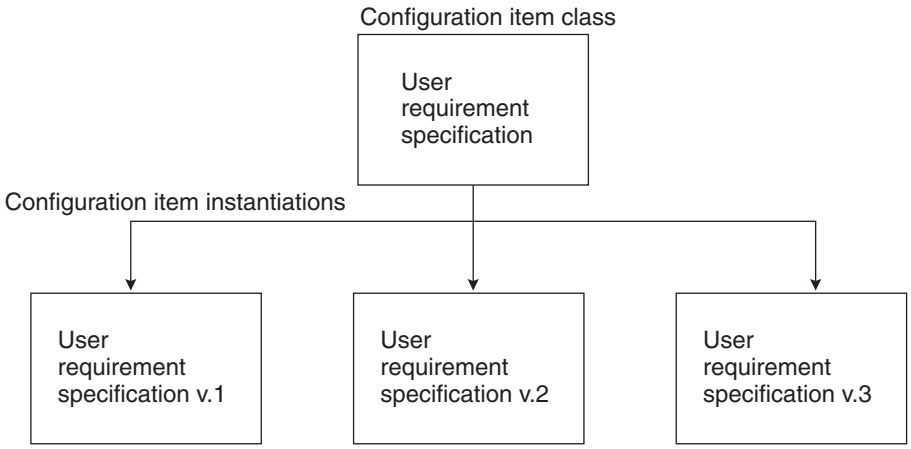

Figure 1–2 *Configuration Item Class and Instantiations*

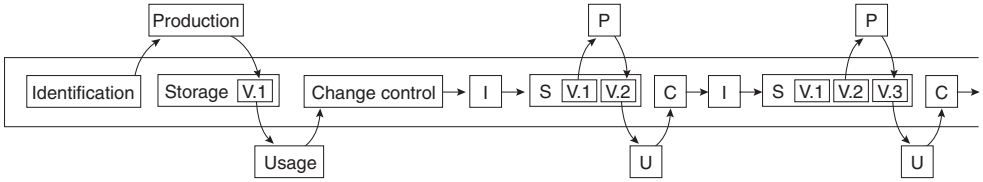

Figure 1–3 *The Life of a Configuration Item Class*

configuration item. Items placed under configuration management must never be changed, but new versions may be created.

Configuration items that are different versions of the same original item are obviously strongly related, but each one is an individual item, which will be identified and may be extracted and used independently. This is one of the main points of configuration management: to be able to revert to an earlier version of an item class.

Quality Assurance Process

Configuration management interacts with quality assurance, as illustrated by the item approval process that accompanies a configuration item from production to storage. The item approval, which may be a written quality record or verbal, is a

product of quality assurance. Some see it as a product of configuration management, but it's actually the gateway from production to configuration management, provided by quality assurance.

Auditing

Auditing is included in some definitions of configuration management. An audit ensures that the product—the configuration item released for use—fulfills the requirements and is complete as delivered. This includes configuration management information, so that everything required is delivered in the expected versions and that the history of each item can be thoroughly accounted for. This activity area is not considered part of configuration management in this book. It's viewed as an activity area under general quality assurance, which partly concerns the products and partly the processes, rather than a configuration management activity area.

This may be a controversial point of view, but the idea of audits is a legacy from the Department of Defense origin of configuration management. Today there is a much broader understanding in the software industry of the importance of quality assurance and, therefore, also of configuration management.

Auditing uses configuration information extensively in the form of status reports, but it also uses quality assurance techniques and methods, such as reviewing and test. In practice, people involved in configuration management also carry out the audit. Consequently, the audit will be referred to elsewhere in this book.

1.2 IDENTIFICATION

The purpose of the identification activity area is to determine the metadata for a configuration item—to uniquely identify it and specify its relations to the outside world and other configuration items.

Identification is one of the cornerstones of configuration management, as it's impossible to control something whose identity you don't know. If the tables in a restaurant have no numbers to which orders can be traced, the waiter will have difficulty matching the dishes to the proper guests. Perhaps this will come off all right if you have three or four tables, but definitely not if you have more than 20 and the guests keep coming and going all evening and the waiter who serves the food is not the one who received the order. To be sure everything is under control, the waiters and orders need to be uniquely identified as well. Figure 1–4 shows how identification is influenced by its surroundings.

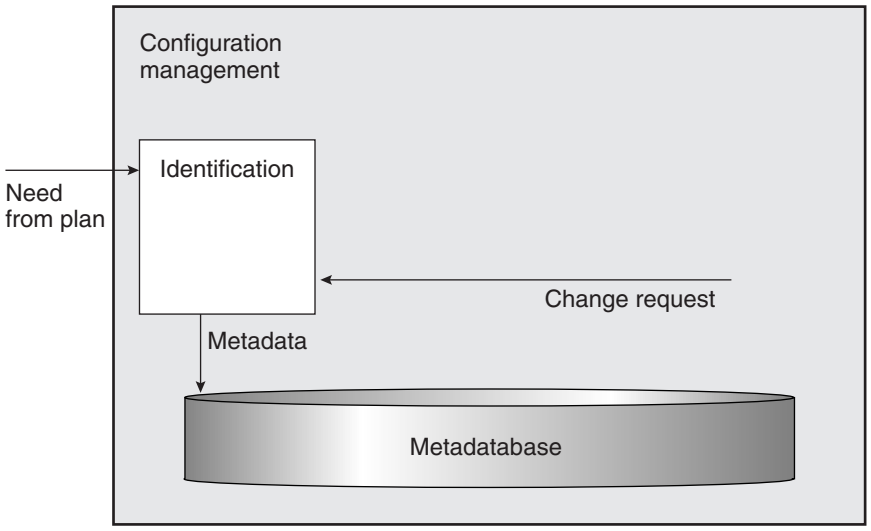

Figure 1–4 *Identification in Context*

Inputs

Two incidents may initiate the identification process:

- The first time an item is to be placed under configuration management, identification starts with a need defined in the plans. From the project plan, you know an item has to be produced (for instance, a design or a code module), and from the configuration management plan, you know this item must be placed under configuration management.
- When you have to change a configuration item, identification will start with a change request. In this case, Change Control has decided that a new version of the configuration item must be produced, (for example, the requirement specification document must be changed in light of a review), and subsequently this new version must be placed under configuration management.

Outputs

Identification results in the registration of metadata for a configuration item.

Process Descriptions

Methods, conventions, and procedures necessary for the activities in identification may be

- Procedure(s) for registration of metadata, including procedure(s) for tracing
- Procedure(s) for inheritance of metadata
- Convention(s) for unique identification
- Convention(s) for authorization, including restrictions on distribution, if any
- Convention(s) for identification of components in a delivery

Unique Identification

Each organization must define its own conventions for unique identification. One general convention most often is not enough; typically, a number of conventions are necessary for various classes of configuration items, such as documents and code.

These conventions may be difficult to define. You have to consider what purpose the unique identification serves and how to implement it in the easiest way. It will connect the registration and the physical configuration item in such a way that it's always possible to find the configuration item. This should be considered when defining conventions for unique identification. Furthermore, the procedures are tool dependent and often highly predefined in the tools available.

The file name under which the item is saved may constitute the unique identification. It should also be possible to write the unique identification on the configuration item's medium—for instance, on a diskette label. The unique identification also registers the relations between configuration items, such as variations or parts of each other. It may be a good idea to consider making identification a key in a database registration.

It's important to define the conventions for unique identification so that the formation of new configuration items will not make it necessary to change or delete existing configuration items. For instance, the number of digits in a number must be sufficient to cover the total number of configuration items; it's not appropriate to have only two digits if more than a hundred items may be produced.

Examples

Figures 1–5 and 1–6 illustrate unique identification for two types of configuration items.

The document identifier in Figure 1–5 contains

Project and year: SC.91
Document number: 009
Author and affiliation: OA.ect
Activity identifier: T2.3.1
Document type: RP (Report)
Version: 02

Project name	:	SCOPE
Document title	:	SCI Description Format
Document identifier	:	SC.91/009/OA.etc/T2.3.1/RP/02
Distribution	:	Project
		Manager R&D Department
		Manager Technical Support
Date	:	May 2nd, 1991
Author	:	Ole Andersen, Elektronikcentralen
Approved by		
Project Manager	:	_____ Date: _____
QA	:	_____ Date: _____

Figure 1–5 *Document Front Page*

The test cases in Figure 1–6 are included in a test specification document, and the identification contains (for the first test case)

The current section number in the document: 10.3.1.6
Running unique number: 80
Version of test case: 1.A

10.3.1.6 (80)	Test for correct bank identity number	1.A
10.3.1.7 (93)	Test for correct bank account number	1.B

Figure 1–6 *Test Cases*

Authorization

Authorization information may be derived from a quality assurance plan and a development plan. It's an advantage to have general directions for authorization in such plans, such as descriptions of the kinds of people who are responsible for acceptance and production of various types of configuration items.

Perhaps documents placed under configuration management in the form of a draft could be approved by the author himself, while documents placed under configuration management in a version to be used by everybody should be approved by a group consisting of the project manager, the person responsible for quality assurance, and a customer representative.

Roles

Often the person who produces an item is also in charge of its identification. Identification is based partly on conventions for individual data elements and partly on information available from various plans (for instance, general procedures determining who is in charge of the approval of a particular kind of document). It may be necessary to draw on a library tool, for example, if numbers that are part of the unique identification are generated automatically and/or reserved in a storage tool.

Connection with Other Activities

An item cannot be declared as being under full configuration management until it has been placed in controlled storage. It is not sufficient that the item be identified. Identification, production, and placement in storage often overlap or are carried out interchangeably.

Sometimes the identification, or parts of it, is carried out before the item is produced, such as when the decision is made about its production. In other cases, it may be carried out in connection with placing the item in storage (for instance, when a tool increases a counter).

1.3 STORAGE

The purpose of storage is to ensure that a configuration item will not disappear or be damaged, that it can be found at any time and delivered in the condition in which you expect to find it, and that a record is kept to indicate who has been given the item or a copy of it.

Storage is something physical. Items that are stored are physically present at a specific place. In the following discussion, we shall call this place a library, but physically, it may be a directory structure on a workstation, a looseleaf binder, a shelf, or something else. Figure 1–7 shows how storage influences and is influenced by its surroundings.

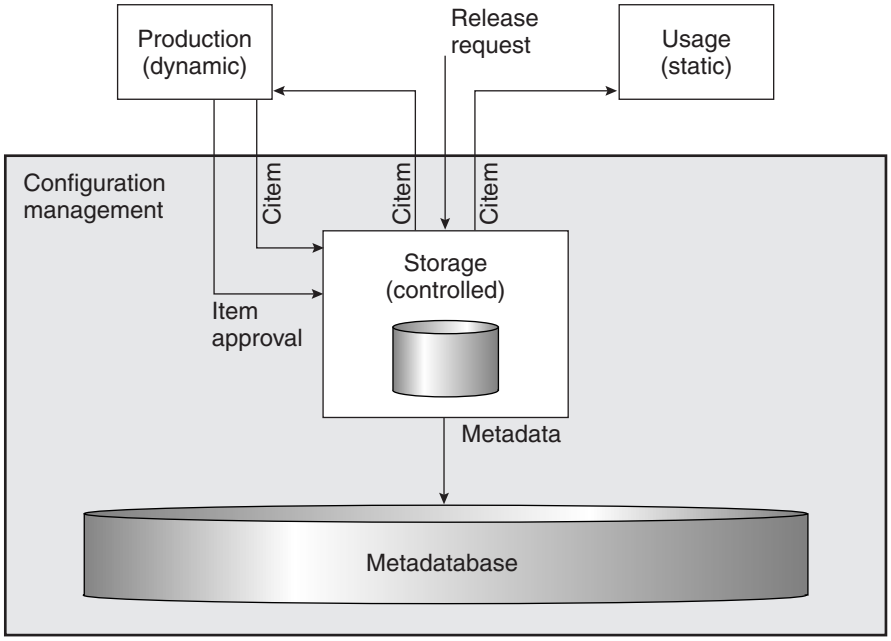

Figure 1–7 *Storage in Context*

Library

Storage takes place in libraries. Traditionally, three types of libraries are mentioned in connection with software development: controlled, dynamic, and static libraries.

The Controlled Library

The controlled or configuration management library is where configuration items are stored. It may be divided into a number of physical libraries, especially where the configuration items are of different types: documents, source code, hardware, and so on. Naturally, hardware cannot be placed in a database on a computer but on a shelf or directly on the site of application.

The Dynamic Library

The dynamic or development library is where items are kept while they are being produced. Typically, this will be in the producer's own area. Therefore, in reality, this library will consist of a considerable number of independent libraries. In this library, it's possible to work with an item without bringing it into contact with other items or exposing it to the influence of other items.

This library is not part of the configuration management system! It's under the responsibility of the development manager, even though many modern configuration management tools also encompass the dynamic library. In some cases the dynamic library and configuration management library may overlap. This should be carefully considered and preferably avoided, unless a deployed tool manages both libraries and the distinction between them is clear.

As an example, many companies use the same tool (such as Visual SourceSafe) during production as well as for configuration management of source code, and practically in the same way. This means that the same library is used as both dynamic and configuration management libraries. During production, a file is often checked in and out several times before it's actually finished. The configuration management library is hence also used as a backup medium for the producers' own intermediate results.

These intermediate results are neither approved nor under change control and consequently not under configuration management, but it may be difficult to distinguish them from versions of the same source code item under configuration management. This is especially true if you're not careful with status codes or if you can't extract configuration items on the basis of different status codes.

Consequently, if libraries are mixed, procedures and conventions should be employed to

- Make clear when the changed version of the source code is approved and actually placed under configuration management and in the configuration management library (by the use of status codes)
- Ensure that if you want to extract an item, you don't just pick up the last version of the item (the one the producer is working with) but a version that has actually been approved and is under configuration management

Another consideration when mixing libraries is space. Space considerations often cause tools to save only the difference between two versions—so-called delta storage. But if the library is used for intermediate results that are later removed, it's not possible for all tools to keep only the differences between configuration-managed versions. The whole new version will have to be kept each time.

The Static Library

The static or user library is where items are used. Usage is all imaginable applications of configuration items, not just by final users. It may be a review, if a document is placed under configuration management in the form of a draft and subsequently has to be reviewed. It may be testing parts of the system, integrating a subcomponent into a larger component, or proper operation or sale of a finished system.

While being used in the static library, items must under no circumstances be changed. The static library may consist of many different physical repositories or storage media—it need not be a library in the classic sense.

When usage involves source code to be included in a module test of another item, the static library may be identical to the dynamic library of the developer. This means that the source code to be used is copied from the controlled library to a read-only copy in the developer's own environment. When usage is review of a document, the static library may be the reviewer's physical workspace, where a copy of the document is placed. When usage is the running in production of a system, the static library may be situated on a production machine or the user's workstation.

This library is not part of the configuration management system! The responsibility for this library may lie in different places, depending on the context of the use—e.g. with the developer, test manager, or customer.

Main Processes

Storage involves three main processes:

- Placement in storage from production
- Release from storage for usage
- Release from storage for production

Placement in Storage from Production

The event initiating placement in storage is that the item reaches a state where it's ready to be placed under configuration management—when it's approved according to its type. It is up to each company to determine when an item can be approved. Approval criteria should be described in a quality assurance plan or the like. A source code item might be considered ready when it compiles, or perhaps when it has passed a module test with certain coverage or with less than a certain number of failures of a given type, such as a given severity. Placing an item in storage should be accompanied by item approval, so it can be documented that the criteria for finishing the item have been fulfilled.

For safety-critical products and/or in very formal configuration management systems, it's sometimes considered useful to deliver the items to a person (for instance, a configuration management librarian) rather than merely enter an item into a tool. This process has a significant positive effect on the quality of the products, as it's too easy to evade quality demands when facing just a tool. However, it's both slower and more expensive to handle placement in storage manually rather than automatically, so the benefits and costs must be considered carefully.

The result of this process is that the item can be reached only through the configuration management library, and metadata for the item is properly up-to-date. Of course, it's necessary to ensure that items already in storage cannot be destroyed—for instance, by being overwritten by new configuration items. An item in version 1.3 must not overwrite version 1.2 of the item having the same name.

Release from Storage for Usage

The event initiating the release of a configuration item from storage for usage is the perception of a need for the item. Release for usage ought to be accompanied by a release request, so it can be documented that the release is permitted and has taken place. To prevent this process from becoming too costly or cumbersome, the release request could be automated, in the form of a permission scheme associated with items under configuration management and automatic control of the permission and log-

ging. Here, too, it may be useful in some cases to have the release take place via a person (a configuration management librarian) rather than via a tool alone.

The result of this process is that the configuration item—or, for documents and software, typically a copy of the configuration item—is delivered to a static library, and an entry in the metadata registers to whom it was delivered and when.

Release from Storage for Production

The event initiating the release of a configuration item from storage for production is the need for production of a new item on the basis of one already produced. This need is typically expressed in the form of a change request. The result is that a copy of the configuration item is delivered to a dynamic library (and always a copy—never the configuration item itself if only one copy exists).

Many configuration management systems and tools can indicate that the configuration item of which they have received a copy is locked or the like. This should be unnecessary if the configuration management library is controlled so that new configuration items cannot destroy existing ones and if planning and implementation of the change request is performed correctly.

If a problem arises from two or more people working with "the same item" at the same time, it reflects a poor configuration management system—lack of control over who is permitted to work with what and lack of proper identification. If several people have to work on the same branch simultaneously, this process must be handled as parallel development.

Process Descriptions

The methods, conventions, and procedures necessary for activities in storage may be

- Procedure(s) for placing items in storage and related updating of metadata
- Procedure(s) for release for usage
- Procedure(s) for release for production
- Template(s) for item approval
- Template(s) for release request

Roles

The librarian in charge of the configuration management library plays a decisive role in establishing and maintaining the configuration management library, so the configuration management library doesn't fall into decay. This is also true for automated

configuration management systems. If others establish and maintain the configuration management library, who is responsible for what must be clearly defined.

Connection with Other Activities

Storage may overlap with identification, if, for instance, a tool adds to a counter when an item is placed in storage.

Example

The development environment for a project, especially a large project, is much more than just a developer's directory, an editor, a compiler, and a linker. This example illustrates the controlled library and two dynamic libraries in the form of the directory structure for a project called the Meteorological Archiving and Retrieval Facility (MARF) carried out for the European Meteorological Satellite Organisation (EUMETSAT).

The directory structure, shown in Figure 1–8, includes

- A controlled library, structured according to the architectural design, offering separate subdirectories for data, object code (compiled sources), executables, sources, test scripts, and so on
- Dynamic libraries for all developers, reflecting the structure of the controlled library
- Substructures for version control of sources—where "Ingestion" is one of the architectural modules and "Other-globals" indicate others

The controlled library for the MARF project is /home/MARF/marfdev. Subdirectories under the Sources directory existed according to subsystems, for example:

/home/MARF/marfdev/Sources/Ingestion

/home/MARF/marfdev/Sources/Other-globals

Each subdirectory has a repository, called AtFS (Attributed Files System), where versions of the sources are kept in a controlled manner using a simple configuration management tool. Object files are collected in object libraries, to ease linking and reduce the size of executables. There is one object library for each source subdirectory.

Each developer has a development area (dynamic library) under /home/MARF/marf/*username*, with a structure identical to the one just described. A link is set up from the AtFS master repository to each user area. Developers use writable copies of

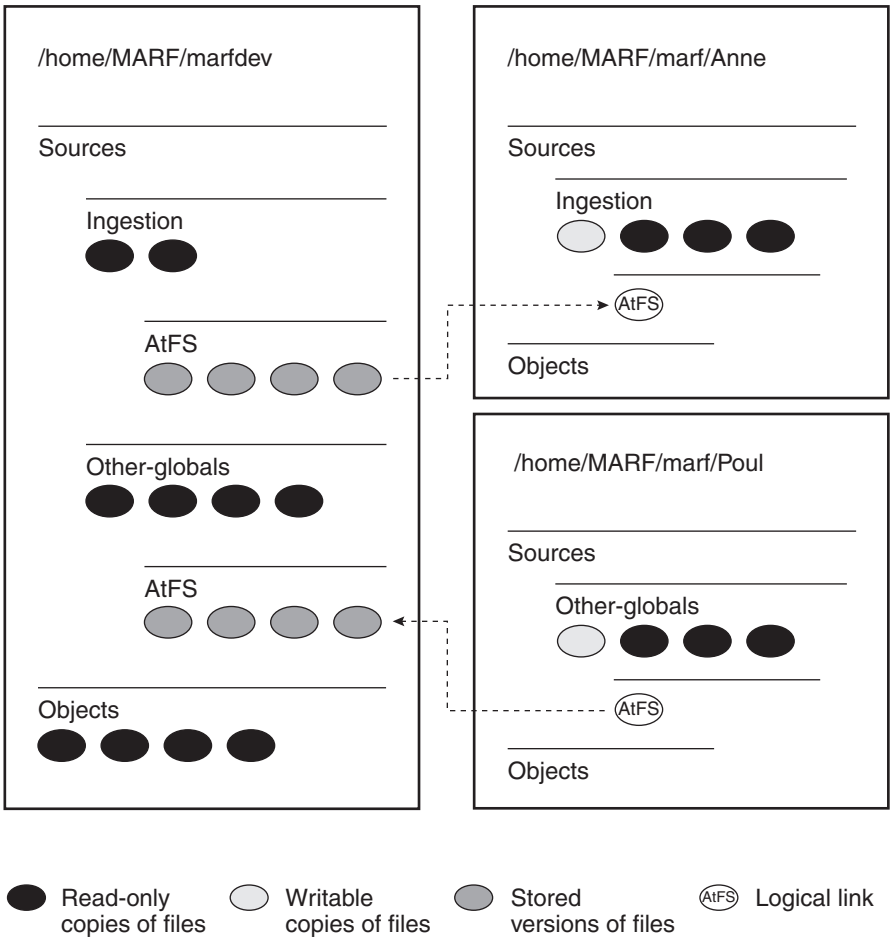

Figure 1–8 *Example Library Structure*

stored files in their own directory structure for development. Read-only copies of dependent files are also placed in the developer's own directory structure, to ease compilation and linking during development and test.

Files are kept in saved versions in the central AtFS areas, and read-only copies are made available there as well. This setup ensures shareability with a minimum of disturbance.

1.4 CHANGE CONTROL

When developing and maintaining a product, changes are inevitable. People make mistakes, customers need changes, and the environment in which the product operates evolves. In addition, people constantly develop their knowledge of the problem and their ability to solve it. In software development, it's generally said that the solution of a problem will create new problems. In other words, we get wiser all the time.

The purpose of change control is to be fully in control of all change requests for a product and of all implemented changes. For any configuration item, it must be possible to identify changes in it relative to its predecessor. Any change should be traceable to the item where the change was implemented. Figure 1–9 shows how change control affects and is affected by its environment.

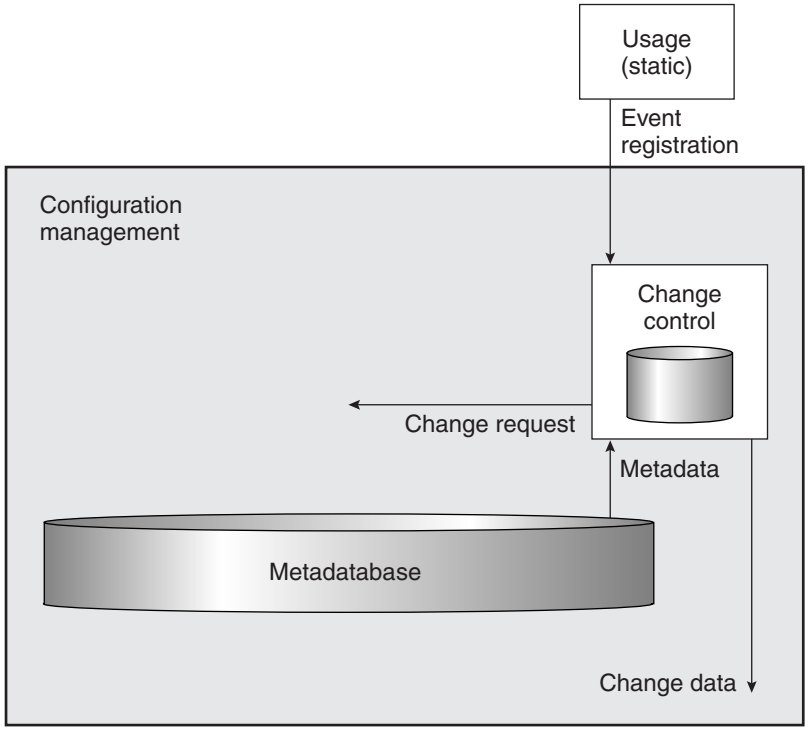

Figure 1–9 *Change Control in Context*

Inputs

Change control is initiated by an event. An event may also be called a wish for modification but need not be expressed as a clearly formulated wish. In this context, an event is any observation of something surprising, unexpected, inconvenient, or directly wrong during usage of the configuration item. It may, for instance, be

- A wrong formulation, caught during the review of a document
- A coding mistake found during a walk-through of a piece of source code
- An enhancement request arising from a new idea from the customer during work on the project
- A mistake found in the integration test
- A wish to expand or enhance the finished product, arising once the product is in operation
- An inquiry to a helpdesk about a problem in connection with usage of a system
- A change required in the code because of an upgrade to a new version of the middleware supporting the system, which may not be backward compatible

An event should be documented in an event registration, which is the input to the change control activity. Some changes, such as those due to a review, can be foreseen and planned, while those due to, for instance, a new customer request cannot.

Outputs

The result of change control is documented events and change requests derived from these events. Both should be securely maintained, as in a database, so that relationships between change requests and configuration items can be reliably maintained. Event registration and change requests may be put under configuration management, but this happens rarely, except where configuration management has to be very formal.

Change Control Activities

A change process is a miniature development project in itself. An event registration should have a written and controlled life cycle, consisting roughly of the phases described in Table 1–1. Each phase should be described in detail, stating the responsibility and specific actions in the company. It may be necessary for a company to describe different kinds of life cycles, depending on the types of events to be handled.

Table 1–1 *Overview of Change Control Phases*

Phase	Description
Creation of the event registration.	The event registration is created, and the event is described.
Analysis of the event registration.	Configuration item(s) affected by possible changes are determined, and the extensiveness of these changes is estimated.
Rejection or acceptance of the event registration.	If the event registration is accepted, a change request is created for each configuration item affected.
The change request initiates a new configuration item.	A new configuration item is identified and created, and the change is implemented. In the course of accepting the new item and placing it in storage, feedback is given to the configuration control board.
Closing of the change request.	The change request can be closed when the change has been implemented and accepted.
Closing of the event registration.	The event registration can be closed when all corresponding change requests are closed.

Quite often the change request is joined with the event registration, so no independent change requests are created. This is not a very good idea, unless it remains possible to extract statistics and status information on individual change requests as well as on the event. This is especially true if an event causes changes in several configuration items, which is often the case.

Usage of Metadata

When performing the change process, metadata is used for analytical purposes. This may be in the form of reports or a direct search in the database or the databases where metadata is maintained. Trace information is often used—for instance, to determine in which configuration item changes are required due to an event. Also information about variants or branches belonging to a configuration item is used to determine if a change has effects in several places.

Finally metadata may be used to determine if a configuration item has other outstanding event registrations, such as whether other changes are in the process of being implemented or are awaiting a decision about implementation.

Consequence Analysis

When analyzing an event, you must consider the cost of implementing changes. This is not always a simple matter. The following checklists, adapted from a list by Karl Wiegers, may help in analyzing the effects of a proposed change. The lists are not exhaustive and are meant only as inspiration.

Identify

- All requirements affected by or in conflict with the proposed change
- The consequences of not introducing the proposed change
- Possible adverse effects and other risks connected with implementation
- How much of what has already been invested in the product will be lost if the proposed change is implemented—or if it is not

Check if the proposed change

- Has an effect on nonfunctional requirements, such as performance requirements (ISO 9126, a standard for quality characteristics, defines six characteristics: functional, performance, availability, usability, maintainability, and portability. The latter five are typically referred to as nonfunctional.)
- May be introduced with known technology and available resources
- Will cause unacceptable resource requirements in development or test
- Will entail a higher unit price
- Will affect marketing, production, services, or support

Follow-on effects may be additions, changes, or removals in

- User interfaces or reports, internal or external interfaces, or data storage
- Designed objects, source code, build scripts, include files
- Test plans and test specifications
- Help texts, user manuals, training material, or other user documentation
- Project plan, quality plan, configuration management plan, and other plans
- Other systems, applications, libraries, or hardware components

Roles

The configuration (or change) control board (CCB) is responsible for change control. A configuration control board may consist of a single person, such as the author or developer when a document or a piece of code is first written, or an agile team working in close contact with users and sponsors, if work can be performed in an informal

way without bureaucracy and heaps of paper. It may also—and will typically, for most important configuration items—consist of a number of people, such as the project manager, a customer representative, and the person responsible for quality assurance.

Process Descriptions

The methods, conventions, and procedures necessary for carrying out the activities in change control may be

- Description of the change control process structure
- Procedures in the life cycles of events and changes
- Convention(s) for forming different types of configuration control boards
- Definition of responsibility for each type of configuration control board
- Template(s) for event registration
- Template(s) for change request

Connection with Other Activities

Change control is clearly delimited from other activities in configuration management, though all activities may be implemented in the same tool in an automated system. Whether change control is considered a configuration management activity may differ from company to company. Certainly it is tightly coupled with project management, product management, and quality assurance, and in some cases is considered part of quality assurance or test activities. Still, when defining and distributing responsibilities, it's important to keep the boundaries clear, so change control is part of configuration management and nothing else.

Example

Figure 1–10 shows an example of a process diagram for change control. A number of processes are depicted in the diagram as boxes with input and output sections (e.g., "Evaluation of event registration"). All these processes must be defined and, preferably, described.

1.5 STATUS REPORTING

Status reporting makes available, in a useful and readable way, the information necessary to effectively manage a product's development and maintenance. Other activity areas in configuration management deliver the data foundation for status reporting,

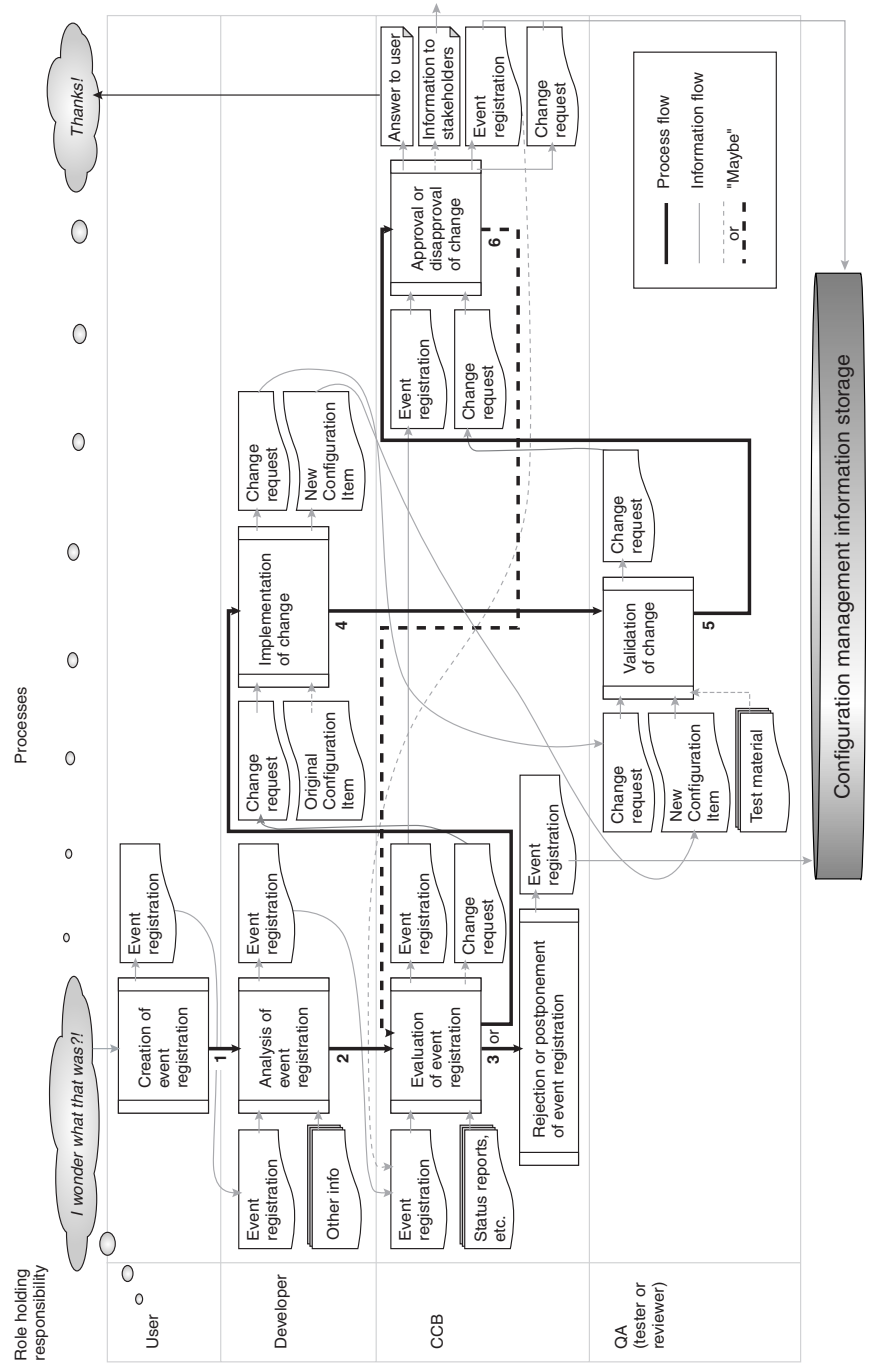

Figure 1–10 *Change Control Process Diagram*

in the form of metadata and change control data. Status reporting entails extraction, arrangement, and formation of these data according to demand. Figure 1–11 shows how status reporting is influenced by its surroundings.

Inputs

Status reporting can take place at any time.

Outputs

The result of status reporting is the generation of status report(s). Each company must define the reports it should be possible to produce. This may be a release note, an item list (by status, history, or composition), or a trace matrix. It should also be possible to extract ad hoc information on the basis of a search in the available data.

Process Descriptions

The methods, conventions, and procedures necessary for the activities in status reporting may be

- Procedure(s) for the production of available status reports
- Procedure(s) for ad hoc extraction of information
- Templates for status reports that the configuration management system should be able to produce

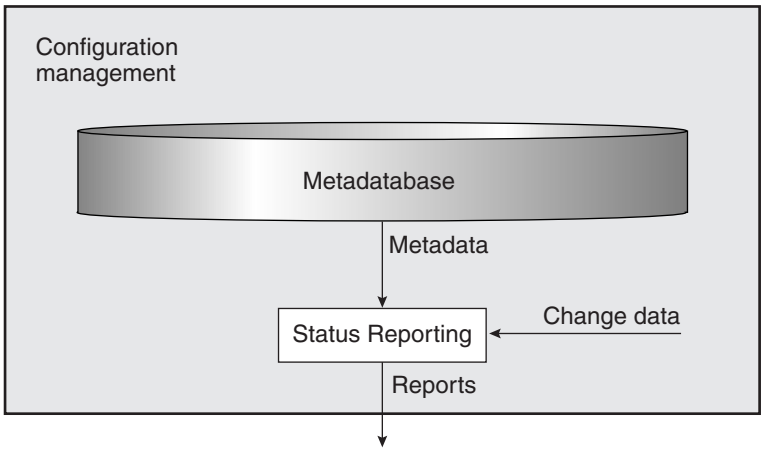

Figure 1–11 *Status Reporting in Context*

Roles

The librarian is responsible for ensuring that data for and information in status reports are correct, even when reporting is fully automated. Users themselves should be able to extract as many status reports as possible. Still, it may be necessary to involve a librarian, especially if metadata and change data are spread over different media.

Connection with Other Activities

Status reporting depends on correct and sufficient data from other activity areas in configuration management. It's important to understand what information should be available in status reports, so it can be specified early on. It may be too late to get information in a status report if the information was requested late in the project and wasn't collected. Status reports from the configuration management system can be used within almost all process areas in a company. They may be an excellent source of metrics for other process areas, such as helping to identify which items have had most changes made to them, so these items can be the target of further testing or redesign.

1.6 FALSE FRIENDS: VERSION CONTROL AND BASELINES

The expression "false friends" is used in the world of languages. When learning a new language, you may falsely think you know the meaning of a specific word, because you know the meaning of a similar word in your own or a third language. For example, the expression *faire exprès* in French means "to do something on purpose," and not, as you might expect, "to do something fast." There are numerous examples of "false friends"—some may cause embarrassment, but most "just" cause confusion.

This section discusses the concepts of "version control" and "baseline." These terms are frequently used when talking about configuration management, but there is no common and universal agreement on their meaning. They may, therefore, easily become "false friends" if people in a company use them with different meanings. The danger is even greater between a company and a subcontractor or customer, where the possibility of cultural differences is greater than within a single company. It is hoped that this section will help reduce misunderstandings.

Version Control

"Version control" can have any of the following meanings:

- Configuration management as such
- Configuration management of individual items, as opposed to configuration management of deliveries
- Control of versions of an item (identification and storage of items) without the associated change control (which is a part of configuration management)
- Storage of intermediate results (backup of work carried out over a period of time for the sole benefit of the producer)

It's common but inadvisable to use the terms "configuration management" and "version control" indiscriminately. A company must make up its mind as to which meaning it will attach to "version control" and define the term relative to the meaning of configuration management. The term "version control" is not used in this book unless its meaning is clear from the context. Nor does the concept exist in IEEE standards referred to in this book, which use "version" in the sense of "edition."

Baseline

"Baseline" can have any of the following meanings:

- An item approved and placed in storage in a controlled library
- A delivery (a collection of items released for usage)
- A configuration item, usually a delivery, connected to a specific milestone in a project

"Configuration item" as used in this book is similar to the first meaning of "baseline" in the previous list. "Delivery" is used in this book in the sense of a collection of configuration items (in itself a configuration item), whether or not such a delivery is associated with a milestone or some other specific event—similar to either the second or third meaning in the list, depending on circumstances.

The term "baseline" is not used in this book at all, since misconceptions could result from the many senses in which it's used. Of course, nothing prevents a company from using the term "baseline," as long as the sense is clear to everyone involved.