Software Configuration Management

1. *Software configuration management* (SCM) is an umbrella activity that is applied throughout the software process, because change can occur at any time.
2. SCM activities are developed to
   (1) Identify change,
   (2) Control change,
   (3) Ensure that change is being properly implemented, and
   (4) Report changes to others who may have an interest.
3. It is important to make a clear distinction between software support and software configuration management. Support is a set of software engineering activities that occur after software has been delivered to the customer and put into operation. Software configuration management is a set of tracking and control activities that begin when a software engineering project begins and terminate only when the software is taken out of operation.
4. A primary goal of software engineering is to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes must be made.
5. The output of the software process is information that may be divided into three broad categories:
   (1) Computer programs (both source level and executable forms);
   (2) Documents that describe the computer programs (targeted at both technical practitioners and users), and
   (3) Data (contained within the program or external to it).
   The items that comprise all information produced as part of the software process are collectively called a *software configuration.*
6. As the software process progresses, the number of *software configuration items* (SCIs) grows rapidly. A *System Specification* spawns a *Software Project Plan* and *Software Requirements Specification* (as well as hardware related documents). These in turn spawn other documents to create a hierarchy of information. If each SCI simply spawned other SCIs, little confusion would result. Unfortunately, another variable enters the process—*change.* Change may occur at any time, for any reason. In fact, the First Law of System Engineering states: "No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle."
7. There are four fundamental sources of change:
   a) New business or market conditions dictate changes in product requirements or business rules.
   b) New customer needs demand modification of data produced by information systems, functionality delivered by products, or services delivered by a computer-based system.
   c) Reorganization or business growth/downsizing causes changes in project priorities or software engineering team structure.
   d) Budgetary or scheduling constraints cause a redefinition of the system or product.
8. Software configuration management is a set of activities that have been developed to manage change throughout the life cycle of computer software. SCM can be viewed as a software quality assurance activity that is applied throughout the software process.

Baselines

A *baseline* is a software configuration management concept that helps us to control change without seriously impeding justifiable change. The IEEE defines a baseline as: "A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

One way to describe a baseline is through analogy: Consider the doors to the kitchen in a large restaurant. One door is marked OUT and the other is marked IN. The doors have stops that allow them to be opened only in the appropriate direction. If a waiter picks up an order in the kitchen, places it on a tray and then realizes he has selected the wrong dish, he may change to the correct dish quickly and informally before he leaves the kitchen. If, however, he leaves the kitchen, gives the customer the dish and then is informed of his error, he must follow a set procedure:

(1) Look at the check to determine if an error has occurred,

(2) Apologize profusely,

(3) Return to the kitchen through the IN door,

(4) Explain the problem, and so forth.

A baseline is analogous to the kitchen doors in the restaurant. Before a software configuration item becomes a baseline, change may be made quickly and informally. However, once a baseline is established, we figuratively pass through a swinging oneway door. Changes can be made, but a specific, formal procedure must be applied to evaluate and verify each change.

In the context of software engineering, a baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and the approval of these SCIs that is obtained through a formal technical review. For example, the elements of a *Design Specification* have been documented and reviewed. Errors are found and corrected. Once all parts of the specification have been reviewed, corrected and then approved, the *Design Specification* becomes a baseline. Further changes to the program architecture (documented in the *Design Specification*) can be made only after each has been evaluated and approved. Although baselines can be defined at any level of detail, the most common software baselines are shown in Figure below.

The progression of events that lead to a baseline is also illustrated in Figure below. Software engineering tasks produce one or more SCIs. After SCIs are reviewed and approved, they are placed in a *project database* (also called a *project library* or *software repository*). When a member of a software engineering team wants to make a modification to a baselined SCI, it is copied from the project database into the engineer's private work space. However, this extracted SCI can be modified only if SCM controls are followed. The arrows in Figure above illustrate the modification path for a baselined SCI.

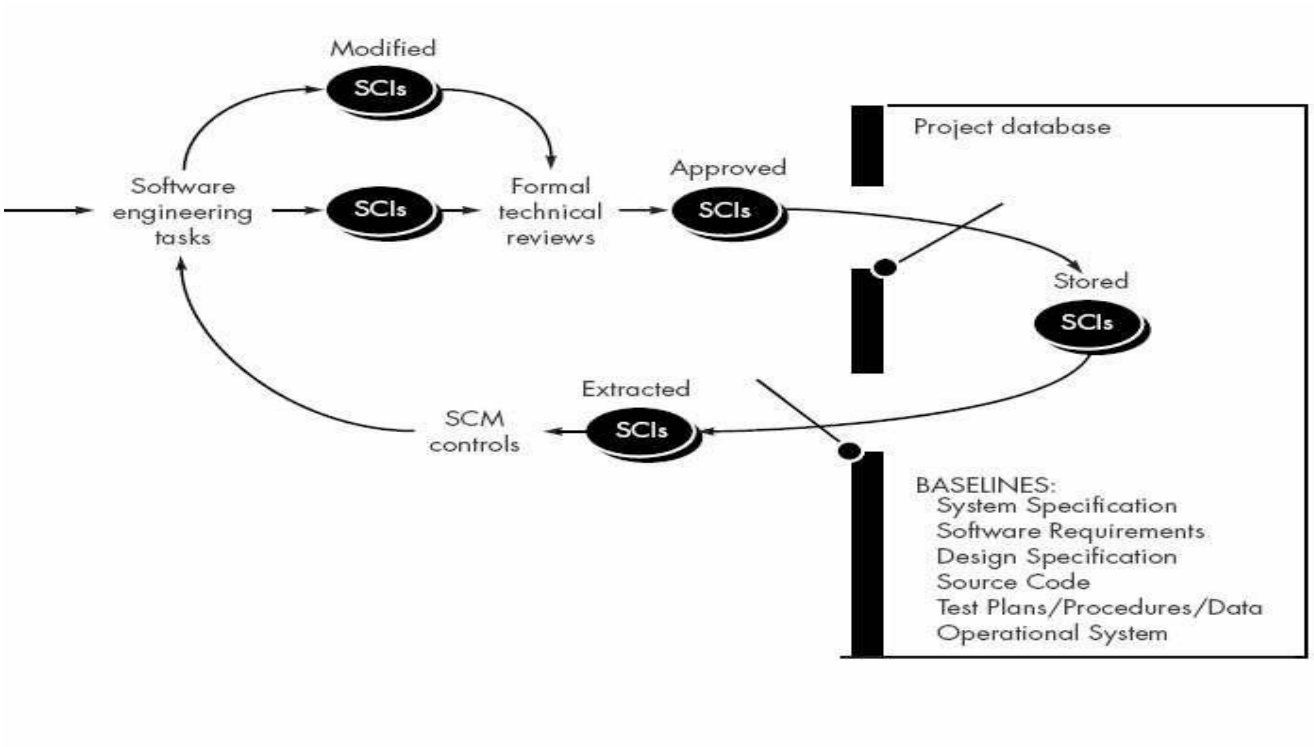

Fig Baselined SCIs and the project database

Software Configuration Items

1. A SCI could be considered to be a single section of a large specification or one test case in a large suite of tests.
2. An SCI is a document, an entire suite of test cases, or a named program component (e.g., a C++ function or an Ada package).
3. In addition to the SCIs that are derived from software work products, many software engineering organizations also place software tools under configuration control. That is, specific versions of editors, compilers, and other CASE tools are "frozen" as part of the software configuration. Because these tools were used to produce documentation, source code, and data, they must be available when changes to the software configuration are to be made.
4. Although problems are rare, it is possible that a new version of a tool (e.g., a compiler) might produce different results than the original version. For this reason, tools, like the software that they help to produce, can be baselined as part of a comprehensive configuration management process. In reality, SCIs are organized to form *configuration objects* that may be cataloged in the project database with a single name.
5. A configuration object has a name, attributes, and is "connected" to other objects by relationships. Referring to Figure below, the configuration objects, Design Specification, data model, component N, source code and Test Specification are each defined separately. However, each of the objects is related to the others as shown by the arrows. A curved arrow indicates a *compositional relation.* That is, data model and component N are part of the object Design Specification. A double-headed straight arrow indicates an interrelationship.
6. If a change were made to the source code object, the interrelationships enable a software engineer to determine

what other objects (and SCIs) might be affected.

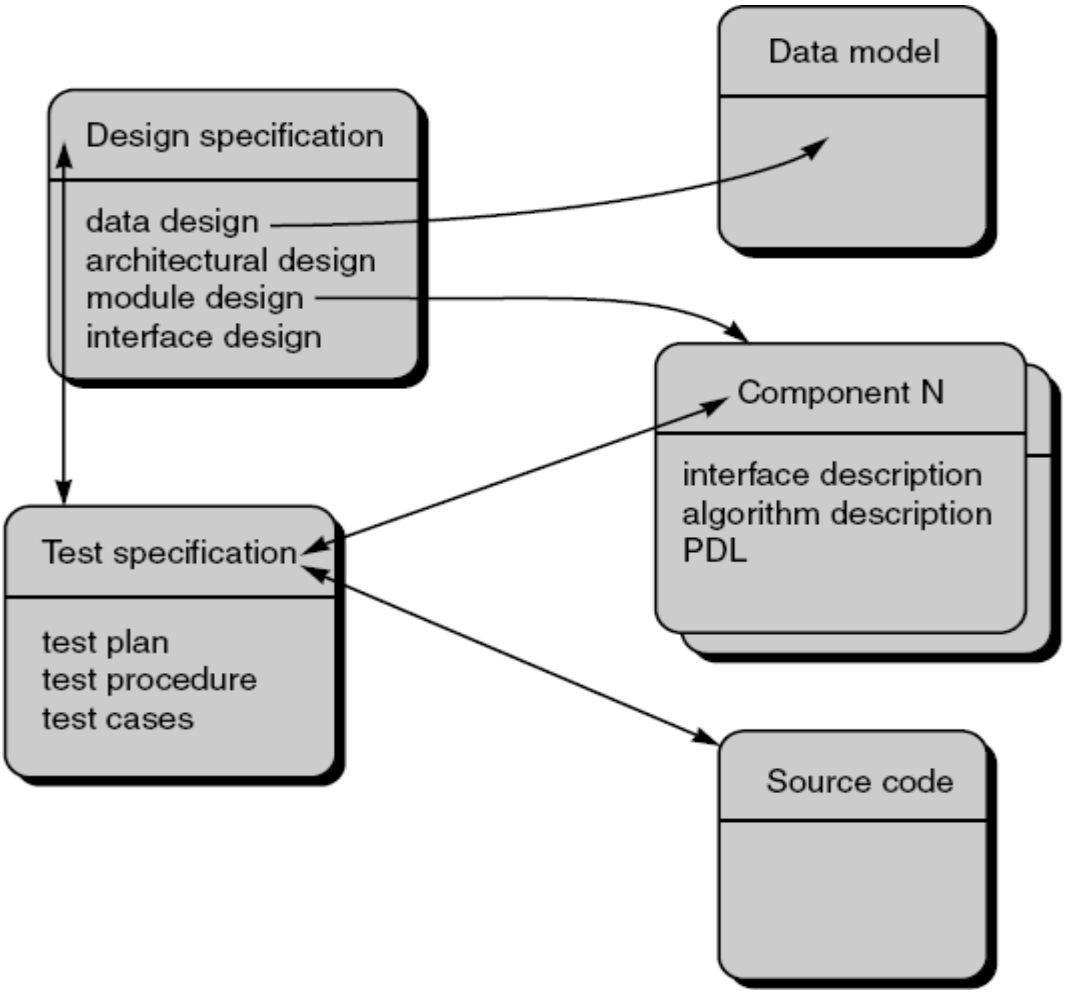

Fig Configuration objects

SCM Process

The five basic SCM tasks are

i) Identification of objects,
ii) Version control,
iii) Change control,
iv) Configuration auditing and
v) Reporting.

1) Identification of objects – To control and manage SCIs, each of them must be separately named and then organized using object- oriented approach. The SCIs can be classified in to two kinds of objects –
    a) Basic objects and
    b) Aggregate objects.

    a) Basic object – It's a "unit of text" created by a software engineer during analysis, design, coding or testing phases. For example, a basic object might be a section of a requirements specification,

a source listing for a component, or a suite of test cases that are used to exercise the code.

b) Aggregate object – It's a collection of basic objects and other aggregate objects. For example, Design Specification is an aggregate object.

Each object has a set of unique features for distinct identification. Every object should have –

i. A unique name,
ii. A description of the object including its SCI type eg. Document, program or data, a project identifier and version information,
iii. A list of resources, which are entities that are processed, provided or required by the object.

2) Version control – It combines various tools and procedures to manage and control different versions of SCI objects (as a result of change) that are created during the software engineering process. Clemm describes version control in the context of SCM:

"Configuration management allows a user to specify alternative configurations of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes".

These "attributes" mentioned can be as simple as a specific version number that is attached to each object or as complex as a string of Boolean variables (switches) that indicate specific types of functional changes that have been applied to the system.

3) Change control – For a large system development project, uncontrolled change rapidly leads to confusion and inconsistencies. Change control includes human procedures and automated tools to control the reasons for change. The following processes take place when a situation of change occurs:

i. Change request – A change request is first submitted and then it is evaluated to asses its –
    i) Technical merit,
    ii) Potential side effects, subsystems and the cost for implementing the change.
ii. Change report – After the evaluation is done, a change report is created that is submitted to the Change Control Authority/Board (CCA or CCB). CCA or CCB is a group who are responsible for evaluating the change report and makes the final decision on the status and priority of the change. This group generates an Engineering Change Order (ECO) for each approved change.
iii. ECO (Engineering Change Order) – It consists of
    i) The description of the change to be made,
    ii) Constraints that has to be taken care of and
    iii) Criteria for review and audit.
iv. Check out & check in – The object to be changed is "checked out" of the project database, the decided changes are made and appropriate SQA (Software Quality Assurance) activities are

performed. The object is then "checked in" the project database and appropriate version control mechanisms are used to create the next version of the software.

4) Formal technical reviews (FTR) & Configuration audit – These two activities are required to ensure that the change made to the software is properly implemented.

   a) Formal technical reviews (FTR) – It's a part of Software Quality Assurance (SQA) procedures. The objectives of FTR are

      i) To uncover errors in functions, logic or implementation,

      ii) To verify that the software under review should meet its requirement,

      iii) To ensure that the representation of the software is according to the standards,

      iv) To make the project more manageable

      v) It consists of walkthroughs, inspections and round-robin reviews.

   b) Software configuration audit – It consists of the following auditing procedures

      i) To check whether the changes specified in the ECO (Engineering Change Order) has been properly made and to check if any additional modules are added,

      ii) To check whether formal technical reviews are conducted for the assessment of technical correctness,

      iii) To check whether SE standards are properly followed,

      iv) To check whether the change is highlighted in the SCI (Software Configuration Items) documentation and also the attributes of the configuration object should reflect the change,

      v) To check whether SCM (Software Configuration Management) procedures like noting change, recording it and reporting it has been properly done,

      vi) To check whether all the related SCIs are properly updated.

5) Configuration Status reporting (CSR) – Its an SCM task which summarizes the activities done so far which includes the following

      a) The report of all the activities done,

      b) The report on the persons involved in the above reported activities,

      c) The report on the time and date when the reported activities were accomplished,

      d) The report on various other factors or objects that will be affected due to the reported activity.

   Following are the situations where the CSR needs updating

      a) Each time when a SCI is assigned a new or updated identification,

      b) Each time ECO is issued i.e a when a change is approved by the CCA/CCB,

      c) Each time a configuration audit is conducted.

The CSR is made available online and is updated occasionally in order to keep the management and concerned individuals updated on the changes. It improves the communication among all the people involved.