SOFTWARE ENGINEERING

Prof. Shashi Kelkar
Computer Science & Engineering
IIT Bombay

Lecture -37
Project Configuration Management

Today we are going to talk about configuration management. Now let us look at what are the minimum services that a project manager must provide to all the people working on his project. Let us take a simple example; suppose all of you are sitting in a class and you are given a ten minutes break so you just walk off from the class room leaving your things behind and when you come back after ten minutes it is expect, you expect that everything that you left behind will be where it was left behind.

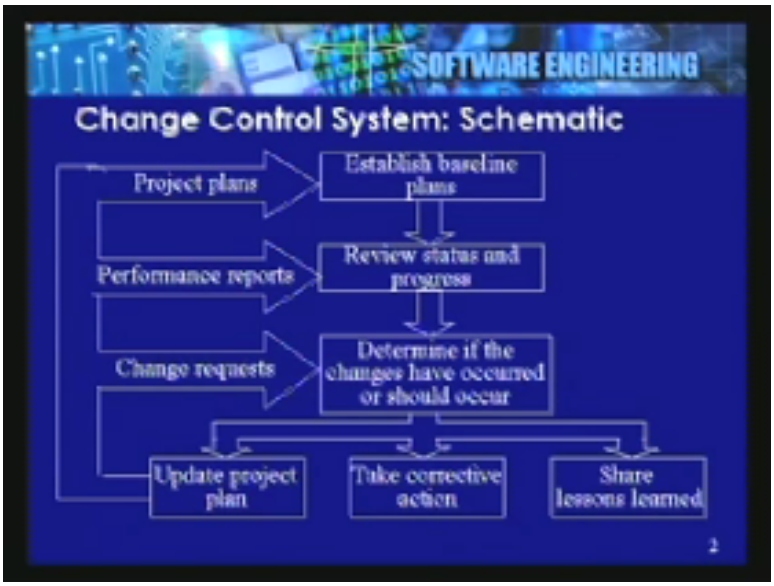
Then we have the next particular question; whose responsibility is it to make sure that the things that you left behind are where they are left behind and the answer obviously is the college management's responsibility to make sure that this particular assurance is provided to the participants. Now we are talking about the same thing; suppose you have many people working in a project and they have different artifacts that they are working on and simple example like somebody having a file name created by one name and another person creates another file by the same name and then you have a problem or somebody by mistake deletes a directory or moves a directory which has a profound impact on a built sequence of the product. So from that point of view what we need to make sure is in general there is a very strong change control system that is put in place for every project. And configuration management system basically is one part of that particular whole change management system. So we ask a question; that the project manager is supposed to provide the minimum service to all the software developers like project control, quality assurance, verification, and validation and configuration

management. So the changes are made in a controlled fashion and the integrity, availability, consistency, accuracy and all these particular things are guaranteed for every one.

In short, work of one person should not interfere with the work of other people. Then we ask a question; why this particular aspect is more important for software and the answer is very simple, changes to software are unavoidable, the changes takes place while the software is in the making process and the changes takes place even after the software is delivered. So the change is a constant kind of a factor in the product life cycle of a software product. Now these changes may be initiated by variety of people, for instance the customer may initiate requirement changes, a developer may initiate the bug detected changes and the product support people may initiate it platform or environment related changes and in all these particular cases you must have a good particular change control system. So before we talk of configuration management let us look at typical schematic change control system.

So if you look at the slide now what you see here is we need to have some kind of a base line plan for a particular project then we need to keep on reviewing the status then determine if the changes are there or not and if the changes are there then we need to make sure that we update the project plan, we take corrective action and we share the lessons that is basically our continuous learning kind of an *exposure*. Remember, whenever you find fault take corrective action and change the process.

(Refer Slide Time: 04:41)



Obviously this may have a consequential effect on several other aspects of the product and from that point of view the plan may have to be also changed. So this is general schematic change control system that continues to exist in any particular product. So a good change control system is essential when planning for changes. Planning in the sense that you do not like changes but if they do happen you want to bring them about in a controlled way. So, the change control system is a collection of formal documented procedure that defines as to how the changes will be assessed and met. So the entire paper work, tracking system, processes and different actions at different particular appropriate levels all this is a part of your change control system.

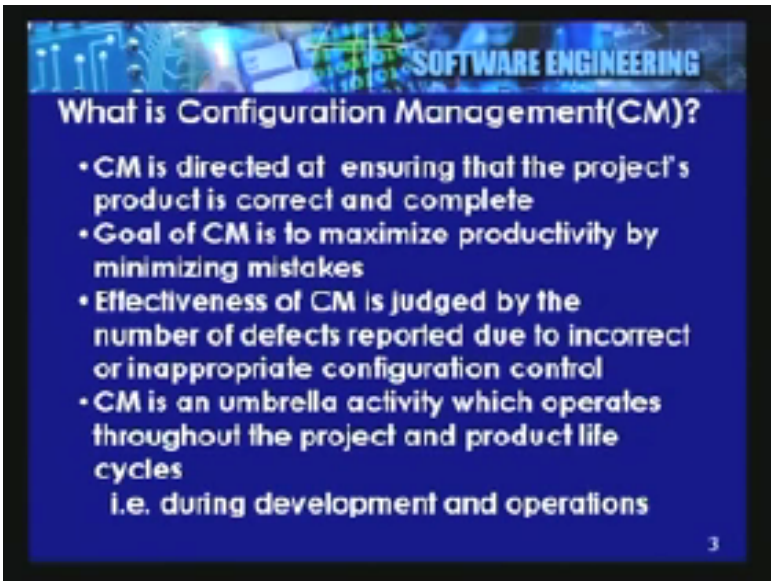
Now the change control system will normally include authority which authorizes changes, somebody who can which can also look after the configuration management. That is in a loose sense it is like a store keeper's kind of a thing, store keeper's job. Remember, in a factory, for instance, suppose you are manufacturing an Indica car and you need to change the housing of a particular tail light the decision whether to or not to make a change in the tail light, the change is undertaken by an engineer. But once that

change is accepted accepting those particular parts, taking care of the old parts and the issuing of the old and new parts all this is looked after by the store keeper.

So the storekeeper does not play a role in authorizing whether or not to change the part but plays a role in whether or not that correct part is being received and issued. Similarly we also need to intimate the change, in case we make a change the corresponding change will have to be intimated to all concerned. So this total aspect is basically referred to as a change control system. The main objective of change control system is what, first, influencing the factors that create changes to ensure that the changes are minimal and beneficial. This involves like making a trade off between the scope and cost and schedule and quality and so on and so forth.

Second, determine that the changes that we say need to happen actually do happen. That is to determine that the status of those particular items has changed. Third, we need to make sure that whenever the changes do occur those changes are communicated to all concerned. As a matter of fact the all concerned need to be somewhat affected by taking the decision but after making the change they need to be informed that the change has been made. And then we have managing those changes in a general sense. These are basically the broad objectives of a change control system. Now let us go one step further and see what configuration management is. If you look at the slides configuration management is directed at ensuring that the project's product is correct and complete at all times.

(Refer Slide Time: 08:33)

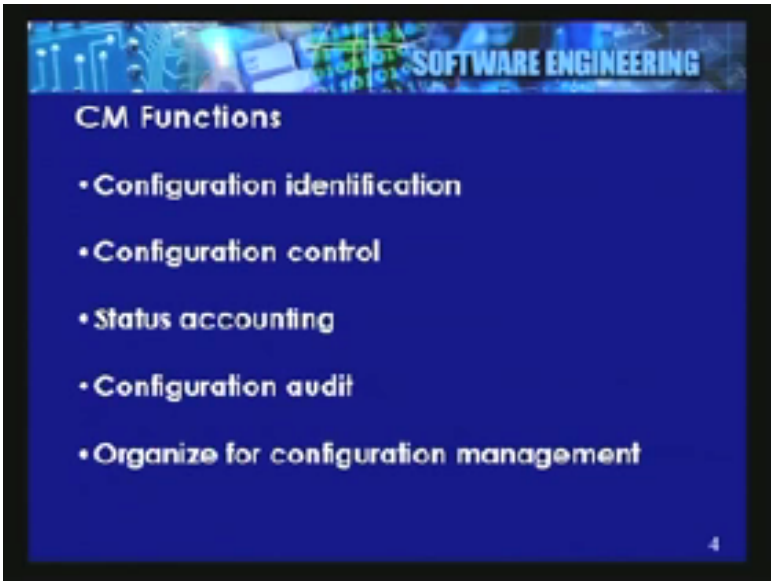


The next particular thing is the goals of CM is to maximize the productivity by minimizing mistakes. Here we are talking of mistakes related to configuration problems in the sense that somebody was to make a change to a software component and wrong component of the wrong version got changed or the changed component went to a wrong library then it is a configuration management problem, its not a sort of making the change kind of a problem in that particular sense. One of the categories that we have in fault analysis is whether the particular fault can be attributed to lack or poor configuration control. So the goal is to minimize the productivity, minimizing the mistakes and increasing the size.

The third is effectiveness of CM is judged by the number of defects reported due to incorrect or inappropriate configuration control. I already mentioned this particular thing to you. And last but not the least we say configuration management is an umbrella activity which operates throughout the project and the product life cycle. As a matter of fact the organization may have several norms and standards for how to undertake configuration management not for only the single project that we are talking about but for all projects in the organization. So remember CM operates during the development and during the operations.

The next question is, what are the main functions that are performed by CM. First let us enumerate them and once we have enumerated them then let us look at all these particular functions one by one. So the first and the foremost if you look at the slide is configuration identification. This looks like a simple issue but it is a very difficult issue.On a lighter side it is like giving a name to a child Bandhini or own household versus giving an ID to somebody on the internet so there is a difference in the way you go about doing these things.

(Refer Slide Time: 11:02)

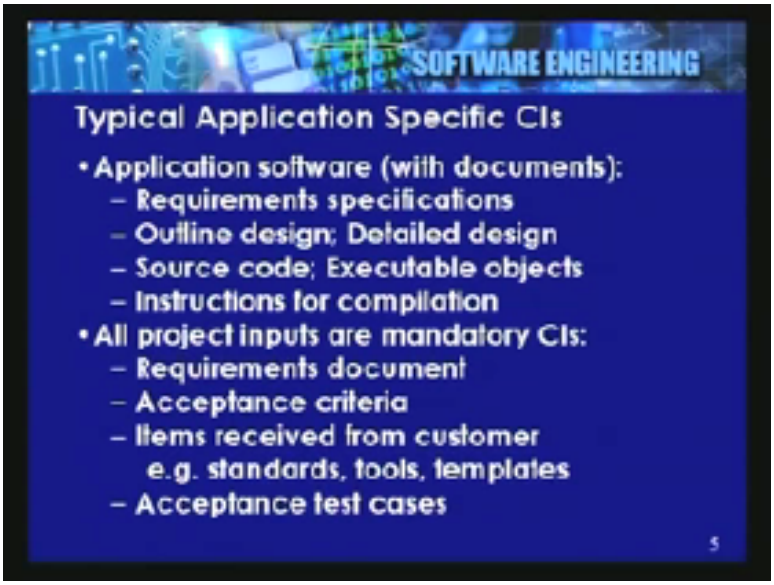


You also need to define that this particular configuration item belongs here or where and so on and so forth. The next function that we have is configuration control. Once we have put an item under configuration we need to make sure that its integrity and traceability is monitored, controlled like receiving, approving, monitoring, changing recode, defect reports all these particular things and of course releasing the corrected product at appropriate time fall under configuration control. The third thing is status accounting.

This configuration management reporting is a very powerful tool for the management to know whether the product making process is working as planned.

So, the status of the product and reporting of the product status is taken care of by this particular process. Next particular one is configuration audit. We already mentioned what is audit. Audit is an independent assessment to ensure that the things are happening the way they are supposed to happen, the processors are being followed you know the processor, the right processor is being used for the right purpose. Last but not the least like every other activity we need to organize for the configuration management activity. So when we look at these particular 5 functions. Now let us look at these particular functions one by one. So first of all we say what are typical configuration items? If we look at the slide it says application software with documents. The typical configuration items are requirement specifications. This is a very important item, outline of design, detailed design, source codes, executable codes, for instance corresponding test data file, the input test file, the output test files, the programs specification, the user manuals, instructions for compilation, executable object you name it all this happen to be configuration items. They need to be identified.

(Refer Slide Time: 12:41)



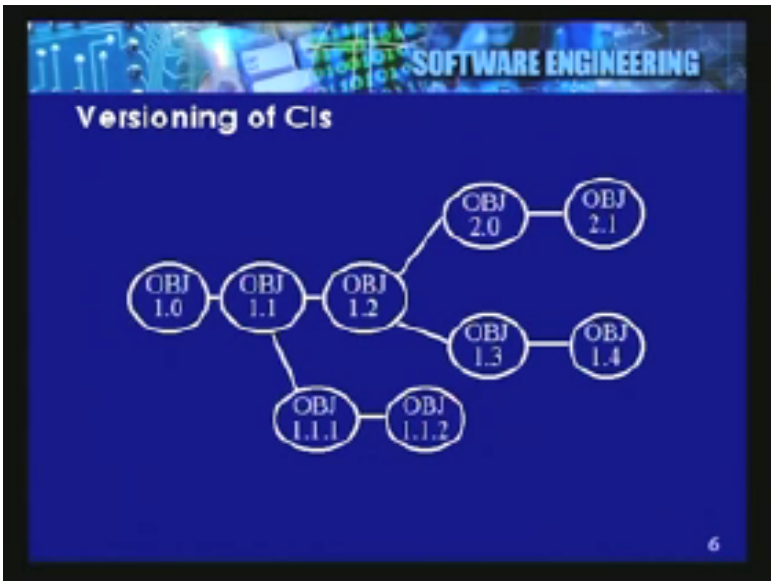
Remember why the trouble happens. Suppose we have a source code and we have a object code we can make modification to the source code without affecting the object. Somewhere down the line we have a problem, how we know that which particular object corresponds to which particular source code. And in case we have an integrated problem between the source code and its corresponding object code you are in deep trouble. So, the first thing we are worried about is application identification. Then we have to take a caution. First thing you must realize, all project inputs are mandatory CIs, they must belong to the configuration control category.

So whenever we have a requirements document what we have received from customers and all that then, acceptance criteria or any particular data that the customer may give you in terms of test data, items received from the customer like standards, tools, templates etc all these are part of your configuration items. Of course what you must realize is that there may be hardware items also which are under configuration control and there may be physical documents also under configuration control. A customer's contract for instance may be a hard copy signed by the customer in which particular case this hard copy itself becomes a configuration item.

Now we need to ask a question what is not a configuration item. Anything in between communication letters, status reports, time sheets, project specific forms or guidelines which are used they do not really form the configuration items, they do not need to be preserved in the sense of identifiability that we are talking about. Once we got the items identified that are only the beginning of the story. So if you look at the slide again now we talk of versions of CI.

Every time you check an item in to a configuration stores this particular configuration stores will give it a version number. Suppose we start from here then the object will be numbered object 1.0. Every time this item is issued out and received back it must receive another version number say 1.1. Many times you find that the manufacturers are working on several configuration versions.

(Refer Slide Time: 15:27)



Take a typical example; so suppose you are supporting a RDBMS product, Oracle or any other particular product that you may take, so what you will have is you will be supporting the current version, you will past say two versions of this particular product and you may be working on next two versions of the product. Now again you are going to ask me a question; why two versions and why not one version? The answer is very simple, completing a version and putting it into the market is a very different thing.

So one version will have been completed and it may have gone for production launch, distribution and that sort of a thing, it is going to take a finite time and a finite amount of energy to put this particular version into market and while that particular process is going on you are not going to sit on your back side and do nothing so what you are going to do is we are going to start continue to work on improving the product. So from that point of view you will always be working on two future versions; one will be a finished version which is up for marketing and the other one is a particular thing.

Now another very important aspect that you must realize is that the versions that you are having, okay, usually there is only one appropriate correct version that is in the market though you support the earlier one. But there is another aspect called a variant. Remember, multiple versions do not coexist; only one version will exist at a time whereas

you may have many variants of the same product in the market at the same time. What do you mean by a variant? Take a simple; suppose you had a banking product and it was sold to one particular bank and after that particular product you need to sell it to another particular bank and that bank may want some minor modifications and though you are a firm believer of using the products on as is kind of a condition you may sometimes want to do minor modifications to a product.

So this minorly modified productwould really be a variant. So, several variants of a product may coexist. Here is a very interesting example; sometime back Indian Airlines had put into use the reservation system and they wanted to do away with the concept of the wait list number, so they gave each particular passenger some pin, pan, kind of a number and at any point of time you had to only say whether that particular number you had in your position whether that particular number had come up.

But people found that this particular thing was not accepted years together they were used to the concept that you go to Indian Airlines and they tell you that your wait list number is 36 and if you check after two hours expect the wait list number to go down but not up and kind of a thing and when the call is made for the people who are waitlisted it is easier for them to conceive that "waitlist number 1 please come forward and check in" rather than saying that some p28396252 to come in and check in. So in this particular case they would have to make the different variants of this particular product if that company was supplying that sort of software. You know this may different problems may exist in different countries and that sort of thing. So we have versions and we have variants.

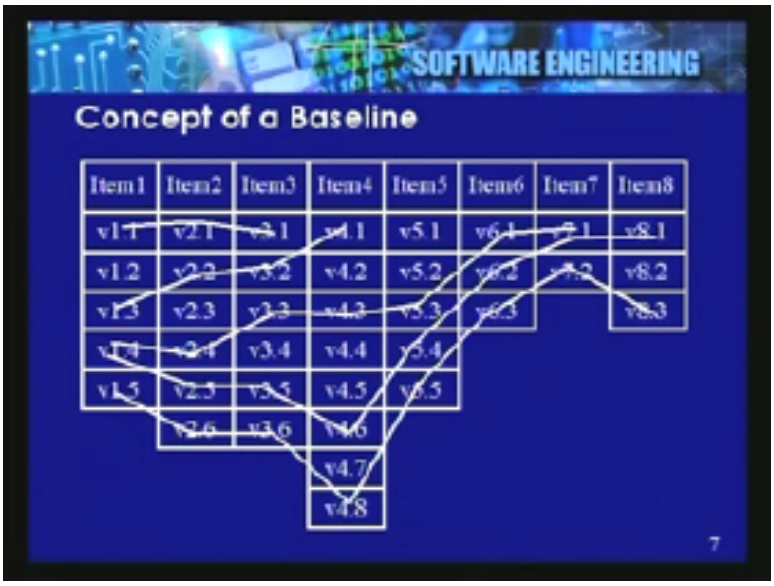
Now we have another particular concept that we need to understand and that is a concept of base line. What is a base line? Base line refers to a set of configuration items which together depict a product at a specific point of time. To take a very simple kind of example, let us take an example of Microsoft Office kind of a product. The numbers that I am mentioning may not be correct but I am just illustrating on a particular thing. So suppose you had a Office 97 and then you had a Office 2000 and you had other versions and when you had Office 97 in the market. Office 97 consisted of an excel and a word

and a power point and so many other particular pieces and each of these products themselves may not have been at the same version.

So you might have one product at version five and another at version nine and the third one at version 6 and we say these together constituted one particular product. And when we went to the next product some products might have undergone three version changes and another product may not have gone any version change at all. So the new base line would consist of the current versions of all the products that are involved. So, base line is a collection of set of configuration items which together depict the product. It may be associated with each base line in addition to the item may also be a build sequence. Build sequence is the sequence in which you put the product together. typically when you are installing a software all these functions are performed by a utility called quote and quote install and when you say install this utility goes and picks up the right components from the right place and puts them in your machine at the right place.

And in case it finds that a particular number or component number or something is missing then it gives you an error message saying that such and such file is not available or something something like that. So the build sequence of putting the items together makes it a working product. Remember what you are doing in a build, you are making a collection of items and putting them together to make and look like a single particular item. So the base lines are tested and certified version of a particular product and only approved items are accepted into the base line. Making a change to a base line has to go through a very controlled and documented kind of a procedure. Now let us look at the slide now. Conceptually let us look at what is the base line.

(Refer Slide Time: 22:21)



So what you see here is you have identified a series of items. Remember, all the items in a particular product may not get identified at time one. Typical example would be that, at the time when you are doing the requirements you obviously do not even know which source codes are going to exist, you have not yet made a higher level design so you do not even know which components are going to exist. But your policy says that each source code will be a CI and a corresponding object code will be your CI and so on and so forth. So what happens is, the items like requirements, specifications and all that may become available early in the game and things like design, coding, test data and test reports all these things may become available as you progress. Suppose you have to make a beginning and we say suppose when we made a beginning we had item 1, 2, 3 with us so we make a base line and we say to begin with it is a starting base line, we say version 1.1 of item number one and version 2.2 etc, actually the version particular part for item 2 is just for convenience has been depicted as 2.1 and version 3.1, it is not meant to be version 3.1.

So we have items and corresponding to it we have the versions. So you have 1.1, 1.2, and 1.3 as the changing versions of item 1 and similarly we have changing versions for the same. After a pre-determined time when you wanted to take another particular base line item 4 had also come into existence. Item number 1 had undergone two changes, item number 2 and 3 had undergone one change of version and of course the item number 4

came into existence for the first time so your next particular base line is going to be V 1 to V 1.3 and V 2.2 and V 3.2 and 4.1.

Similarly when you go a little left, now as you go deeper into the project you will find that the changes to the initially defined items may not be as serious or may not be as frequent whereas the subsequent items may undergo changes. So the third particular base line has this particular collection of items and then you have the fourth and the fifth base line and all that. So the schematic depiction of the base lines is shown in this particular fashion. We ask a very simple question; when do we do the base line? You look at your old photograph album and you say do you have many of your photographs in your albums, you say yes over your lifetime you have several photographs right starting from the first one in your birthday suit to all along you have lots of photographs taken at different instances.

Now you might ask a simple question; when do you take these photographs which are like line marked photographs and you will say yes may be one was done somewhere down the line when you were born and then another one was taken at a very important occasion when you went to school the first time and when you got through the college and then another one when you got married and may be your children and so on and so forth kind of a thing.

But by the same particular token typical application development base lines are; first is the initial base line of the project that is when you start the project you identify all the items that are handed over to you as .0 is the first particular base line.

Then we ask when is the next appropriate time, so it is like asking, in case you are going to take your picture once in five years would you take it on any day of the year or may be you prefer to take it on your birthday or some specific day which you tend to remember easily or something like that. Similarly another good point to take a base line is when the sis is completed and accepted so you have a srs base line. We may take another base line

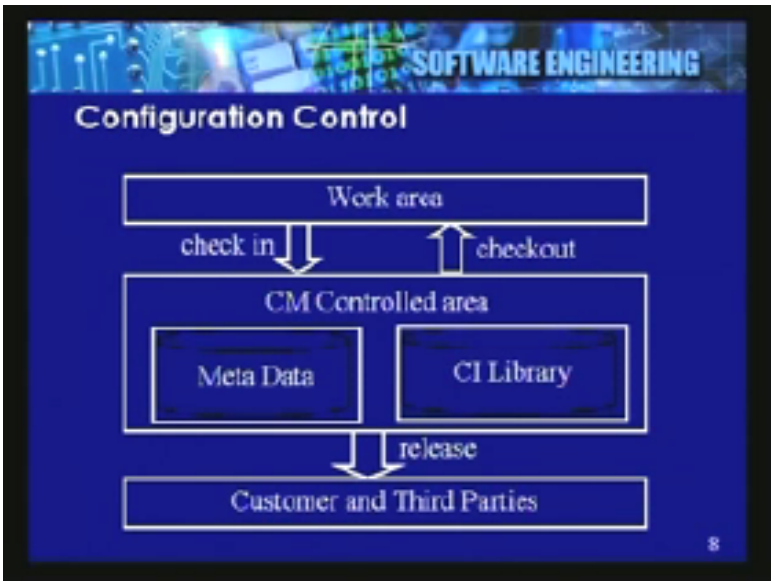
for completion of design. Now we do not have a base line for completion of coding for a simple reason that all coding never gets completed at the same time.

So the next base line that we have is when the integration of the product is done that is, the first time when all components are put together the entire product looks like and functions like a single system and then this particular system is subjected to testing namely the system testing, acceptance testing etc. So whenever the product is accepted and the product is handed over to the customer, like saying you wind up a particular product then you make it an acceptance base line.

Similarly in case you had a convergence project product project then the base lines would be like initial source base line will be there then a filtered source line etc. Filtered source line is passing all the source codes that you received through a program to alter it to the target environment which probably will do may be 85, 95% of your work and the remaining few percent of the work you will have to do it manually. So, once that work is done you will have a converted base line. And last but not the least when you release the product you will have a release base line. So the configuration base lines are very important from software project's point of view.

The next particular thing that we want to understand is configuration control. What do we understand in configuration control? It involves evaluation, coordination, approval or disapproval and implementation of changes to the configuration items. How is it organized? Look at the diagram. The simple thing is, on one side you have the work area and on the other side you have the customer and the third party and we see that no job will ever leave the work area and directly find its way to the customer, it must always through go through a CM configuration control.

(Refer Slide Time: 28:58)



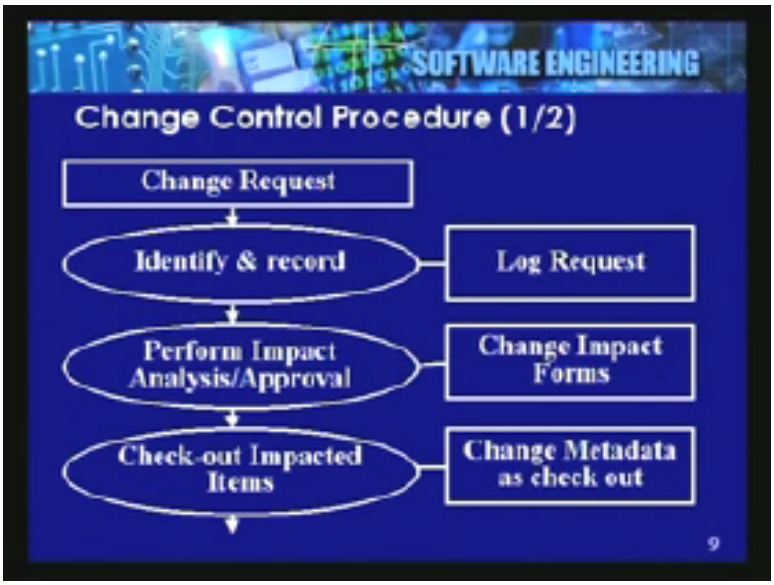
What that particular thing means is, any work from the work area will go to CM if required it will be re-issued to the work area and this process may continue as appropriate and in appropriate time you will have this particular product being delivered out to the customer or third party whatever appropriate. In the controlled area you have a CI library and Metadata. What is a CI library? CI library is the physical place where the storing of configuration items takes place and the Metadata is about the data.

Now let us look at the CI library. For instance, if you are doing a convergence project your software components may reside in two machines. So your CI library now will be one logical library but two physical libraries; one on one machine and another on another machine. The same thing may happen in case you are working on a multi-site kind of a contract. You may have one particular CI library at one's location and another one at another particular location. Then you have problems like some of them are software components but there are lots of non software components which also need to be put under configuration control. These could be physical documents like contract and so on and so forth. Those particular documents will also be a part of the CI library but probably it may sit in a cupboard. We have lots of other things.

Suppose the hardware for a particular project has been supplied by the customer and keeping track of that is a particular problem then the physical hardware may also become

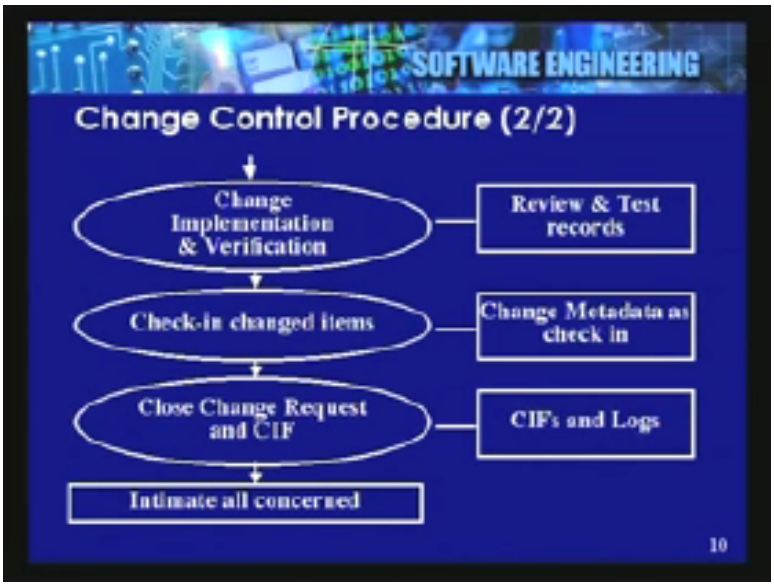
a part of the technically part of the library. So we have configuration control mechanism for that. So the configuration control mechanism starts by operating with the initiation from a defect report. So you get either a defect report from the development team or you have a change request which comes from the customer. So each of these particular reports will consist of several items like configuration item identity, the date that defect was found, the description of the defect, the effect of the defect, the action taken to contain or recover from the defect like what is the suggested solution, what is the urgency of the problem and all kinds of things. Similarly a change request may also say which one is the item that needs to be changed or added, date of request, the descriptions, the reasons for request and the likely effect on the change etc. So, all these particular things will be initiating that particular one. Once we have got that how does the physical change control process operates? For that let us look at the slide again. Hence we begin with a change request.

(Refer Slide Time: 31:27)



Once we got the change request what we need to do is first to identify and record the request in some kind of a request log. So request log will have a list of all the change requests that we have lodged with a particular product. Once we have got this log the next particular thing is somebody has to sit and do impact analysis and approve whether or not this particular change is worth it.

(Refer Slide Time: 32:34)



So, performing impact analysis and approval or otherwise is the next job. So this particular job is called change impact forms and you would like to make sure that this particular change which has been suggested is it worth doing, what is its implication. If you approve the change the next thing that you do is, the item to be modified is checked out from the CS the Configuration Store. Once you do the change in the Metadata you will mark that this particular item has been issued for modification.

Once you mark the Metadata another copy of the same particular item cannot be accidentally issued to another particular person so you are not likely to have a problem that two people working differently on the same component make a modification. Once you got this particular modification done then the person who did the modification needs to make sure that the change was made correctly the way it was supposed to be made. So, adequate amount of reviews and testing will have to be performed become a part of the at the corresponding reports generated. Once that change will be made probably the next thing is to accept the change in to the store and make changes to the Metadata.

This particular thing may look trivial but it is not, it is like how many times we had a packing list which was different from the contents of the box and the answer is very

simple, by the same token the developer may tell you that he is submitting version 1.8 of a particular component and what accompanies in the accompanying file is a version 1.7 or such particular problems can happen. So once we have the change check in then the corresponding version number is also changed in that particular thing.

Once you have done that we close the change request and we write so in the log saying that the change request has been implemented. And last but not the least once that has been done we intimate all concerned that the change xyz has actually been made. And important entry in the change control procedures and configuration management procedures is the concept of Change Control Board which is often called as CCB. Change Control Board is the formal group which is responsible for approving and of course otherwise rejecting the change request. Please understand all the change requests are not of same particular priority and the analogy like the car not starting is a category. A problem which means immediate fixing and that the car not having a headlight or the horn or they are getting stuck in a particular gear that means it works but not to full performance level would be the second particular level and the third would be having a scratch on the paint. You will say yes, whenever the car goes for repair around such kind of changes can be taken care of that point of time.

So the CCB's decision to make to accept or reject a change is not merely dependent on your giving a request and the genuineness of the request because the CCB has to take into account several considerations. The primary function of the CCB is to provide guidelines for preparing the change request, how to make it then providing the guidelines for evaluating the change request and last but not the least managing the implementation of approved changes. So CCB may have procedures for retrospective handling changes also. Take a simple example; if there is an emergency kind of a change somebody may be authorized to make that particular change on the spot and take the approval from the CCB retrospectively.

Even this kind of thing can happen in a live production environment if something goes down and need automatically need immediate fixing then such a extreme action may

have to be taken. There may also be a concept of automatic approval for predefined categories of changes. Take a simple example; suppose we have a screen and in this particular screen the template of the format has some spelling mistake say name has been spelled as instead of n a m e you are spelled it n m a e and obviously such thing needs a change.

But making this change is not going to have any impact on the cost or schedule or any other customer relationship related kind of problems or impact on the quality then we say that this kind of request, the approval of such a request can be easily done at a lower level than the CCB. These changes need to be documented but at the same time they need not go all the way up to CCB and the designator's authority may be allowed or authorized to make some particular change. that brings us to another question; is CCB is a board or is it a single entity and the answer is again, it depends, in case there is a small project then CCB is a single person entity and if it is a very large project then CCB will be group of people or it may have a hierarchy under it.

If it is a multi site project you may have two power centers like we say in politics two power centers and CCB may be split at two locations. So please remember, CCB is the authority which decides whether to or not to make a change, when to make a change and at what appropriate point of time these particular changes need to be released. So, once we have done these particular changes the next particular thing is change control commendation. How do we make sure that all concern people have been intimated that the change has happened?

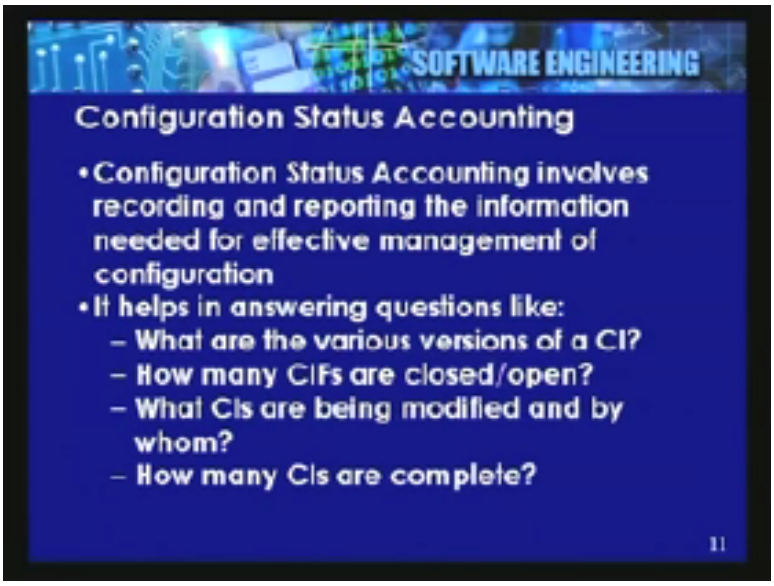
The responsibility of making sure that the change is communicated to everybody basically rests with the project manager. The project manger is responsible for integrating all changes and making sure that all concerned know about it. It is important to make sure that all concerned person are in sink about the latest project's information. No two people should think that they are working with respect to two different components or two different versions of the same components and so on and so forth. There are many mechanisms which are used for communicating the changes to all concerned. You may

have e-mails, www kind of situations or you may have special group of mails being sent to people depending on the kind of change that has happened or certain special project management software may also facilitate communicating tracking and communicating the changes to the software component.

Once we got done with the change control the next particular aspect that we are talking about is configuration status accounting. Configuration status accounting if you look in the slide involves recording and reporting the information needed for effective management of configuration for effective management of configuration.

For instance, we might want to know the list of approved configuration identifications, at any one point of time what is the current identification what are the current CIs, you might want to know how many which CI is in which version, how frequently are the versions changing for a particular CI, too many or frequent changes to CI indicates that there is something wrong, either somebody is not doing a good job in doing the initial specification or somebody is not doing a good job of review and that the half-baked kind of products are going to fill and once such a product goes to fill it keeps coming back like the old notes and coins being circulated faster than the new ones and by the same particular token you will have this thing.

(Refer Slide Time: 40:01)

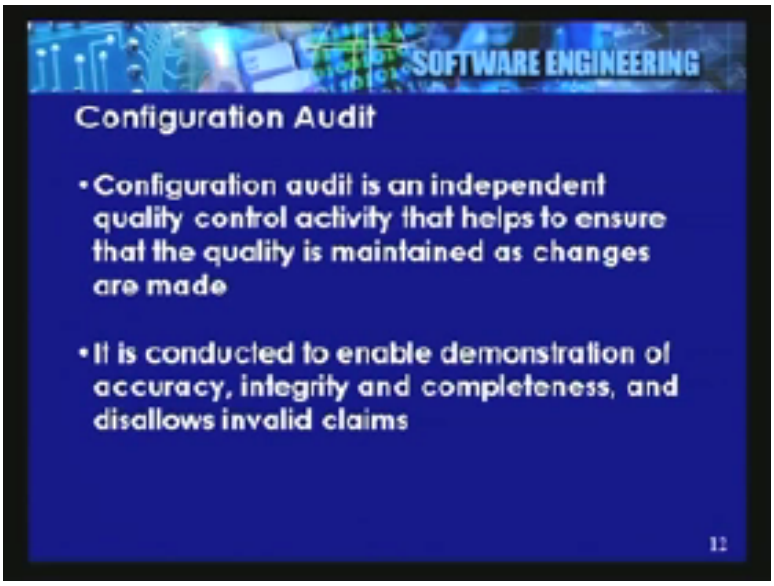


So you may also want to know how many requests have been received ,of what severity they are and severity is only one particular thing, how many have been checked out, how many are in process work and how many have been completed and returned. Once we have developed this configuration kind of a status accounting system then if you look at the slide again, then the next particular thing, is that it will help you in answering questions. It is the management's responsibility to answer this.

Just to remind you about the old saying which I keep repeating that "planning is everything plan is nothing by the same particular token; you need to ask question based on the past changes, what kind of changes you anticipate in the future. If you were to ask a question like what are the various versions of the CIs that we are having, how many CIs were open or closed, what CIs are being modified and by whom and how many CIs were completed so called then these kinds of questions can be answered from the database that you are going to be using for status accounting. So, configuration status reports should be generated on a regular basis and also provide a adhoc facility for inquiries.

This particular thing is what we do, once we got the configuration status accounting then the next thing we worry about is configuration audit. What is configuration audit? We have already seen what is audit.

(Refer Slide Time: 44:59)



Why do we have audit? To prevent delivery of a wrong product to the customer, to track incomplete implementation with reference to the original specifications and agreed changes, make sure that the agreed changes are made and stay made to confirm that the CIs have been appropriately base lined. Again base lining is, when the product is rechecked it does not mean it automatically gets base lined.

Take a simple example; suppose you were to make a change to a component about payroll system in the month of February you you might under circumstances want to hold back to this particular item and say I will issue this item in the next base line only which will be put into implementation from 1st of April the change of the financial accounting year. So such decisions may be there when the base line is ready. Another very important thing that you need to remember is that, when you have this data if required you should be able to back track from the proposed change or made change back to the original version as required.

So we need to have audits to confirm that the CS has been properly base lined, to confirm that the documentations maintain the traceability between the old and the new. Technically speaking the CI library must be in position of all the old versions of all the components. There is a very tall order, but for a variety of reasons if you could retain as many of them as possible it is going to be very essential. So configuration audit ensures

that, first checking whether all components are complete and correct and no pending issues are there in terms of identification and logging in and so on and so forth and verify that the integrity of the base line is maintained that what you say in your configuration status or accounting system as to where you are with respect to the base lines and the component versions, in reality that is what you are putting in practice.

So the practices and procedures that we have laid down are they being rigorously followed or not is one of the important aspects that you need to look at in configuration audit. If you look at the slide now we say configuration audit is independent. We have repeated this several times. It is an independent quality control; it is not a quality control activity but quality control like activity that helps in ensuring that the quality is maintained and the changes are made the way they are supposed to be made.

It is conducted to enable demonstration of accuracy, integrity, completeness and disallows any invalid claim about that particular product. These aspects are very important to know from our particular point of view. We have talked about the four functions. The last particular thing that we don't talked is what is the configuration management plan. Typically in software development organization you could have a single project plan or you could have project plan supported by knowledge areas, as we have mentioned eight knowledge areas and if the project is large each knowledge area may itself evolve into a separate plan by itself.

But many times all the knowledge areas may not be equally well represented. But the two knowledge areas which are very strongly represented in any project plan normally are quality assurance and configuration management. So quality assurance and configuration management plans in their own right may either be separate plans or annexure of a main project plan or may be associated as some sections of the project plan. The project plans basically have a specific objective. It helps the managing of the software development activity; it protects the configuration items from intruders and against accidental wipe out, of course protecting against intruders is more difficult and protecting against accidental wipe out can be done a little more thoroughly. And last but not the least you

will have to facilitate reporting on the state or stages of the project. So the project manager on an on-going basis in the project review will get a good idea as to how the configuration management progress is being presented.

(Refer Slide Time: 47:11)

So, CMP basically specifies who, where, when, how. All these will perform the CM functions. So in this particular aspect you will be having a configuration management plan. So, the configuration management plan will typically tell you what configuration management planwill tell you how to perform the following activities. First, how do you perform configuration identification, what constitutes an item. Now take a simple example; again it is a little repetition, suppose we have six volumes of a user manual do they constitute a single CI or is it six CIs and the answer is depending on how you intend to use it. If you always plan to keep all the six together and issue and collect them or whatever you do with them together then it is a single item. And in case you want to use them individually or they get modified individually then it becomes a separate particular CI.

So you will have to answer many questions like you have a simple thing like programs specification, source code, you have an object code, you have a test data input files, test data output files and so on and so forth. How do you break them up into configuration

items? Somebody has to give a clear guideline as to how to identify an item. So first is to decide what will and what will not be considered as a CI. Once you have said what will not be you will say who will decide what is the CI, who will give it a name, number and so on and so forth. Next particular thing is you will have to specify how configuration base lines are defined, identified, considered for recording and deployed.

This includes, that in case you need to have a back tracking then this particular back tracking will also have to be done. Another important part is maintenance of configuration library, both the physical library and maintaining the Metadata, both these particular things will have to be done. So who will do it, who is responsible, who will authorize, who will issue, who will take back, who will make sure that what is being supplied back or issued out is what you say you are issuing out or supplied back.

Access control and change control to the configuration; who is authorized, who is not authorized to suggest, request, make changes. Then status accounting, again all these things are there, how to perform, who will perform, where it will perform, when it will perform that is all we are answering. Back up control where it, how like every other particular software project back up you need to take back up of the configuration libraries also. That again is whether you want to keep it like disaster recovery or you want to keep it at the same place or at a remote site kind of issues are involved in specifying how to take a back up control.

Your procedures, standards, tools used that also is very simple, which procedure will be used, which standard etc. Suppose we have decided that you are going to take a starting base line, SIS base line and development based designed base line and integration base line and end of the project base line and you may have a concern that for a large project these activities may take long and in which case you might have a procedure or a practice which say that after initial base line take the SIS base line. But if that duration exceeds three months then in any case take one base line every three months and then take it at appropriate end of this. Similarly standards will have to be specified like how the items are identified, numbered and all kinds of things where they are kept, then the tools which

are used, lots of beautiful configuration management tools are available and these tools can be used for checking items in, out, storing versioning and all kinds of things.

Last but not the least you have the organizational structures and the authorities that are there under the configuration management framework. Because configuration management activity also requires its own resources, its organizational structures, authorities and so on and so forth. Last but not the least we will have to do the referencing of all the particular documents and from that particular point of view you will have to say that this is how we actually do this particular job.

So, if you were to look at this kind of a configuration management functions then we can now summarize and say that the configuration management function basically consists of five major activities configuration identification, configuration control, configuration accounting/status accounting, configuration audit and organizing for configuration management.