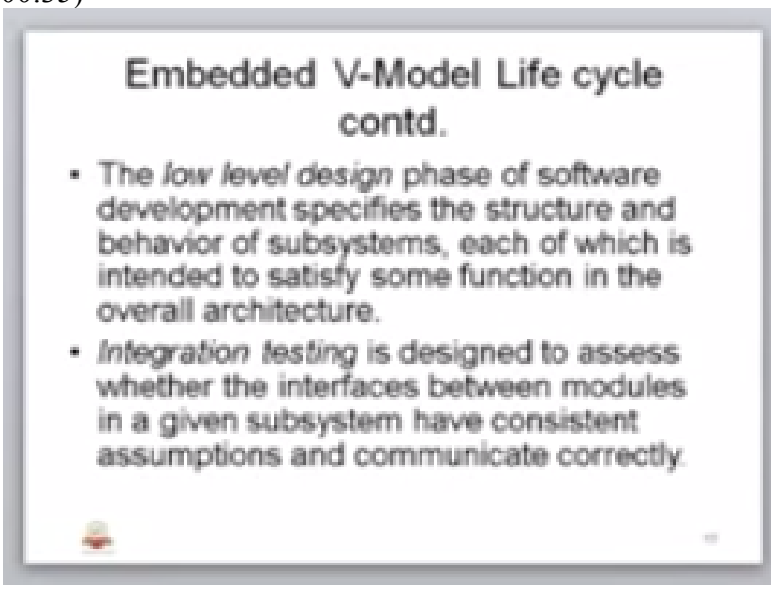
(Refer Slide Time 00:20)

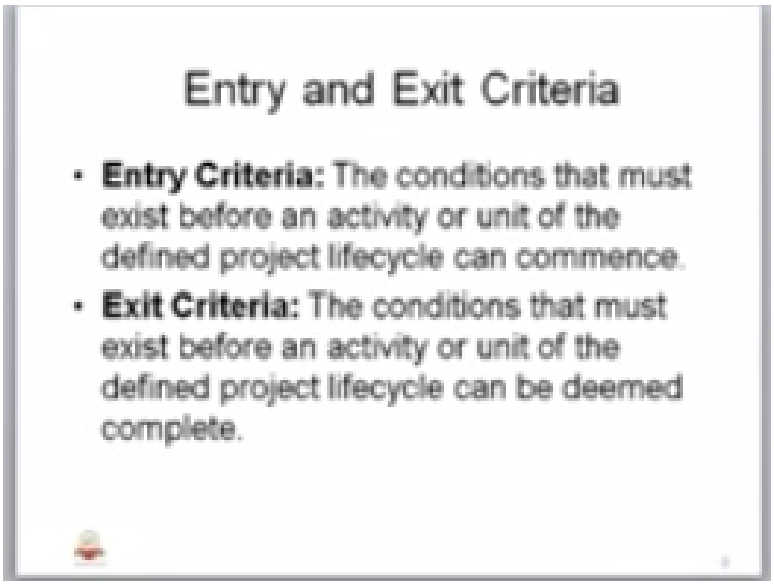
The next embedded software testing these lecture 10 this is in continuation of embedded the development lifecycle and testing lifecycle and different types of a life cycle like B models are multiple B model set up. So in earlier section, we studied about wheel encycle

(Refer Slide Time 00:35)

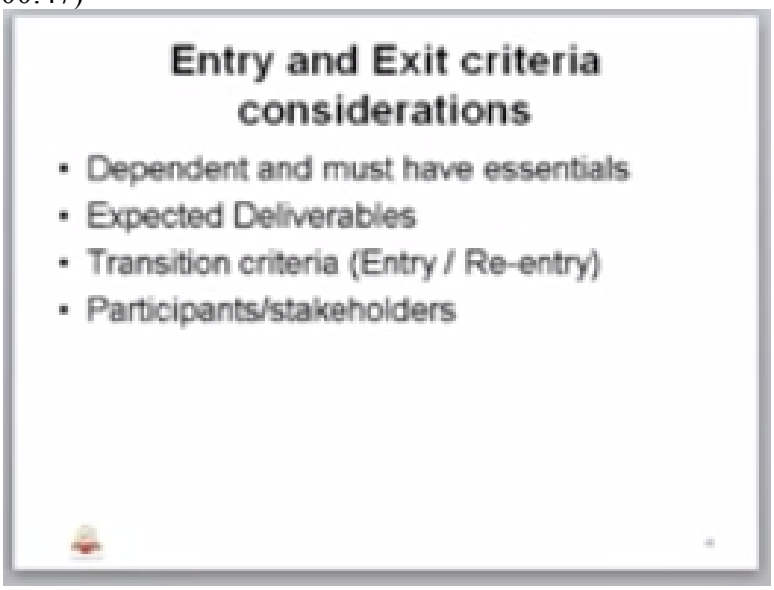


What are the entry criteria that is to be understood

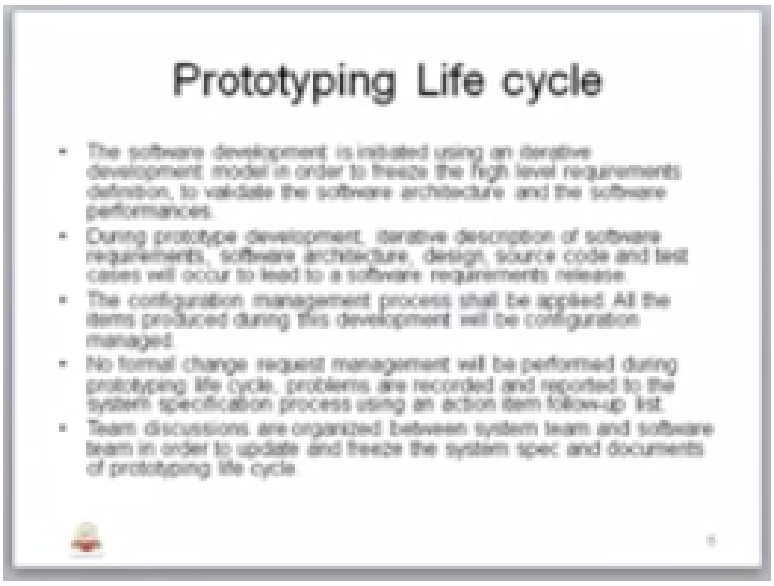
(Refer Slide Time 00:44)



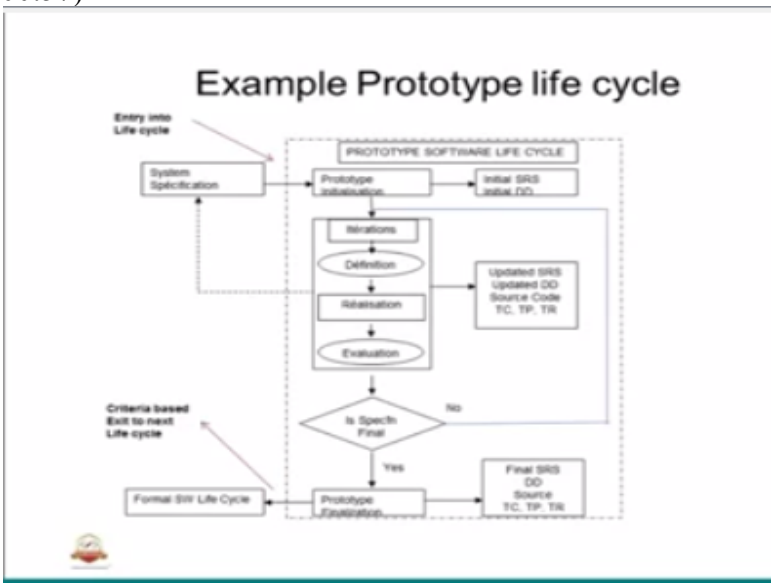
And what is a dependency how it is going to enter
(Refer Slide Time 00:47)



And we also studied about the two type's prototyping life cycle and formal life cycle of an example prototype life cycle we studied.
(Refer Slide Time 00:55)

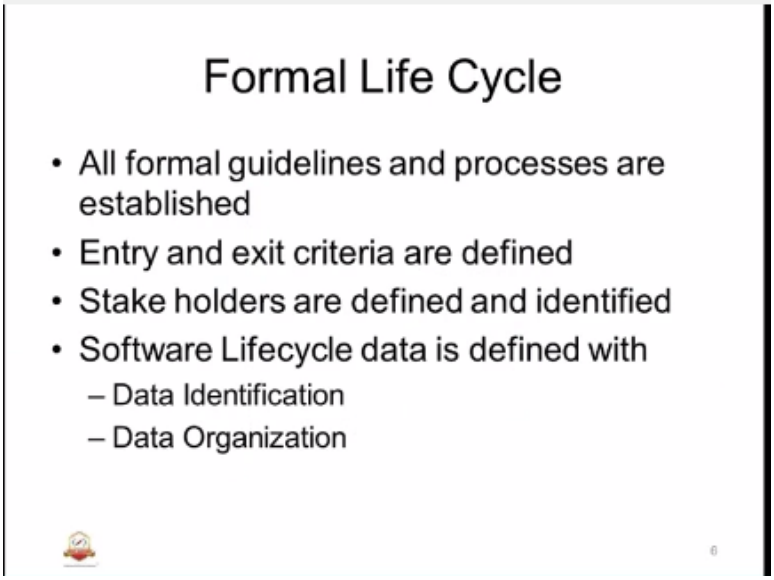


(Refer Slide Time 00:57)

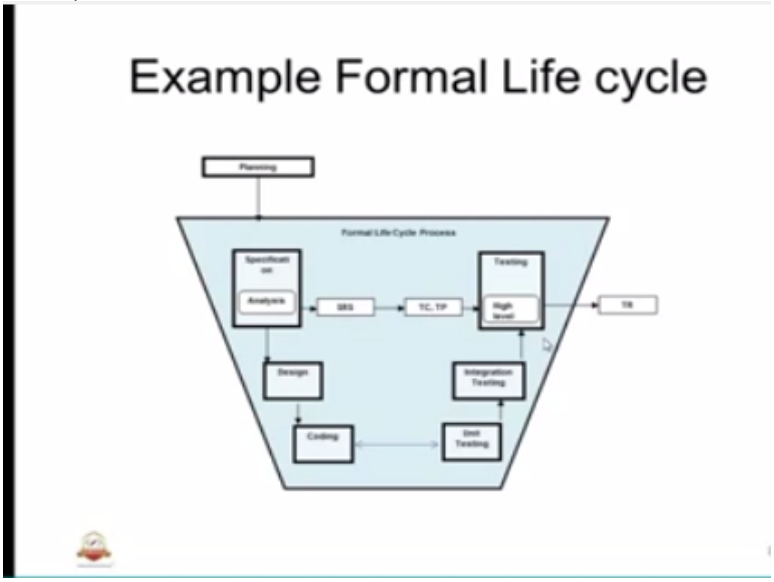


Just glance through the mission what we had in the earlier class this is an example of prototype life cycle have entry end exit as a finalized the prototype it is finalized SRS as a BD. Source code testing aspects which is enough for going for the formal life cycle.

(Refer Slide Time 01:28)

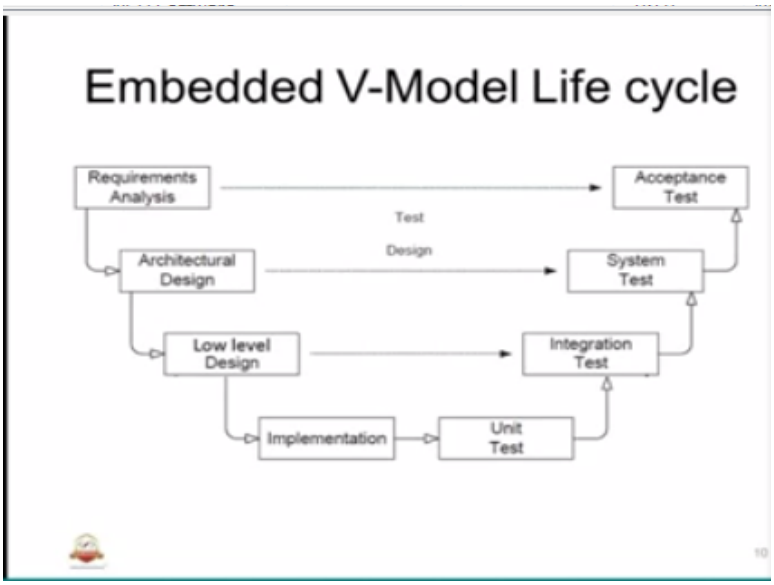


(Refer Slide Time 01:29)

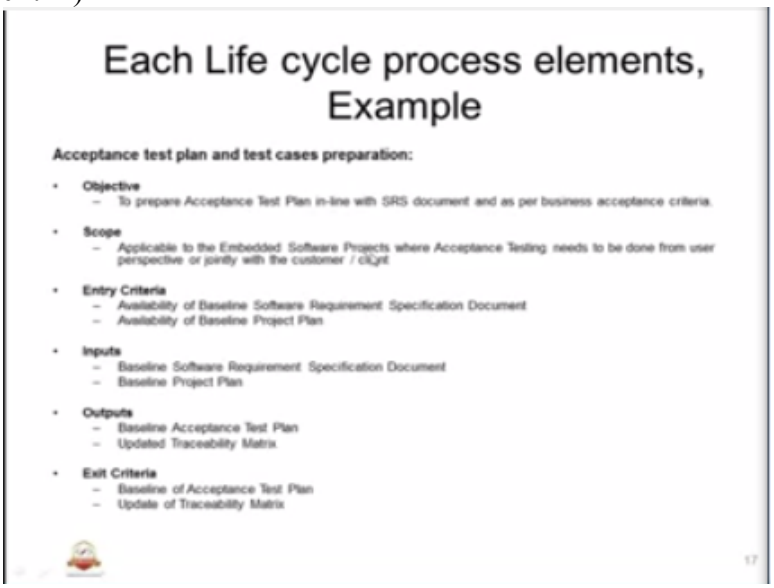


A formal life cycle we know that it's a v-shape why because the on the left hand side we have finalized requirements of specification then we have a design next to that then we have a coding and then in parallel to that the various aspects of testing I have been carried out of specification with a high level design integration coding it is a unit testing. So that is why it is called a B model as part of the exercise that we do for testing of these 11 producer test result. So that is a formula formal life cycle after we are done with an example after we had into pro typing.

(Refer Slide Time 02:21)



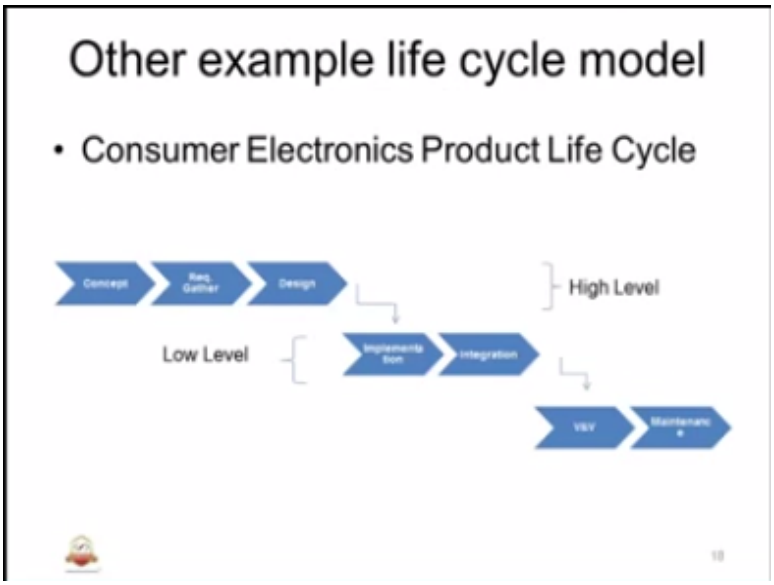
Few examples, of depicting the remodeling life cycle.
(Refer Slide Time 02:24)



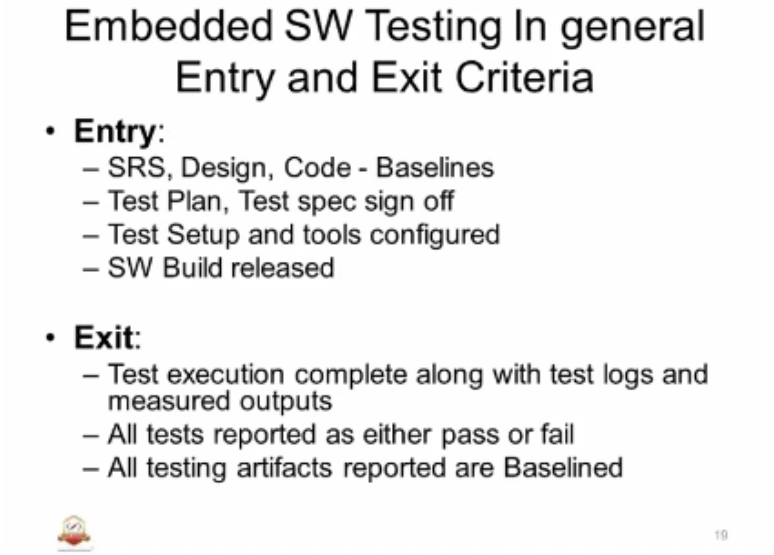
Each life cycle elements have it own processor like the process has those have to be having the objectives and scope in green inputs outputs exits defined for each of the processor can follow life cycle and in our section. We understood about each of this process items taking an example of.

Or test plan where we have integrate objective to prepare our text plan, scopes the project preparing the acceptance testing and entry we do with a high level requirement specification project and also is taken care then we have the inputs as a star as project plan outputs as a baseline discipline and traceability exit is a outcome of the testing what we do the preparation is that this plan is a outcome along with a traceability report.

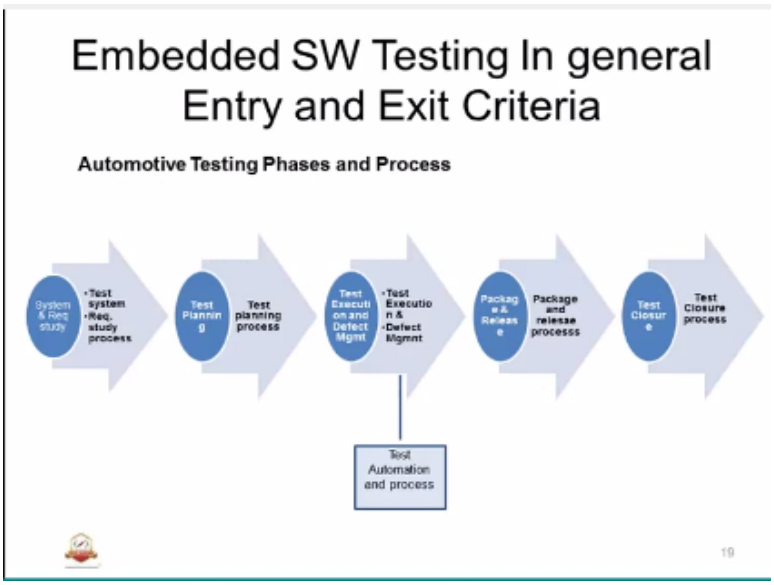
(Refer Slide Time 03:49)



Also we went through consumer electronic product life cycle.
(Refer Slide Time 03:58)



And we defined example of our other general entry and exit of embedded SW testing
(Refer Slide Time 04:01)

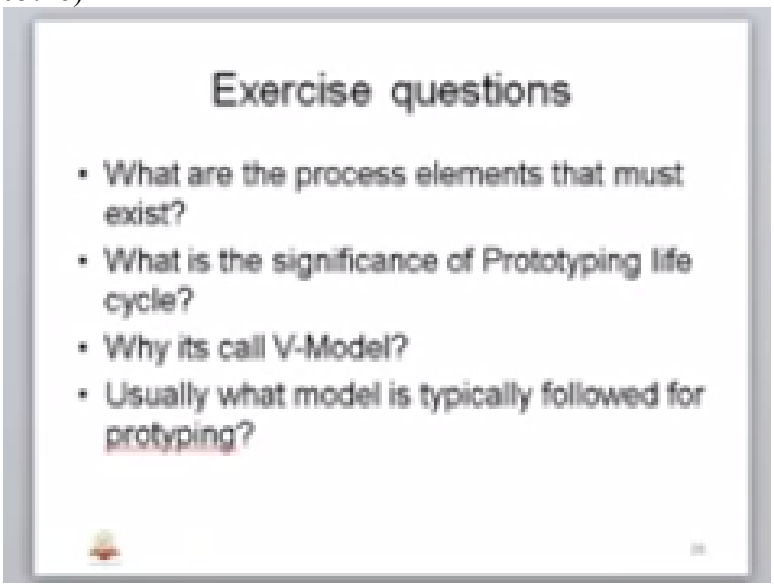


So testing life cycle we studied taking an example of automatic testing system requirement study test planning just execution in there are different one which also involves automotive testing package interleaves then that is the conclusion automotive testing.

So each one these five phases how its own process each of them are we studied about it is objectives, scope, entry, inputs, outputs, exit. Will not go through in detail because gone through this in detail project applicable in the future test for the embedded software

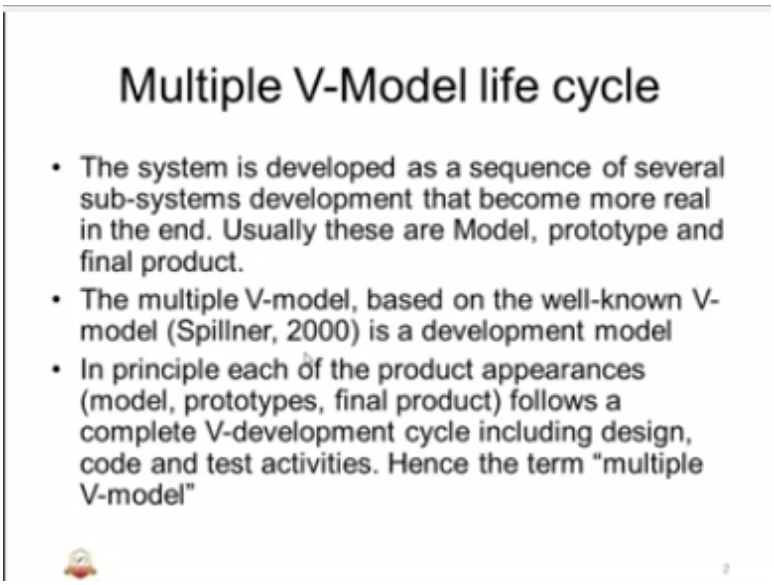
this is a taking an example of automobile test process how they follow for testing the embedded software also we studied about some of the embedded software words we will test base this again and again which starts we are aware of this also we had a few questions as X place so that is input of previous session

(Refer Slide Time 05:16)



Now we will detect background we will go through the next section okay, we know that V model life cycle how it is

(Refer Slide Time 05:27)



Once we have laid out V model life cycle there is another variant of V model life cycle what is called a multiple V model life cycle why it is required what is the importance of this so the essence of multiple V model different physical versions of the same system of development it means you take an example prototype we might have prototype one, two, three, etc, so say different versions are different feature wise the profits as a prototype we would have developed each one possessing.

The same required functionality different operatives the minimum functionality has to be they are actually because the in product or what we are going to develop you the same. But there will be a multiple variants versions in terms it is outcomes this means for instance what the complete functionality can be tested for the model as well as for the prototype and the final product.

On the other hand certain detailed technical properties cannot be tested very well on the model & must be tested on the prototype that means the end once we finalize the end model we may not be able to test us some of the technical elements of that product so we may have to go for doing a testing on the prototype itself for instance the impact of environmental conditions and we can best be tested on the final product but we don't have to do it or we may not be able to do that environmental condition completely on the prototype.

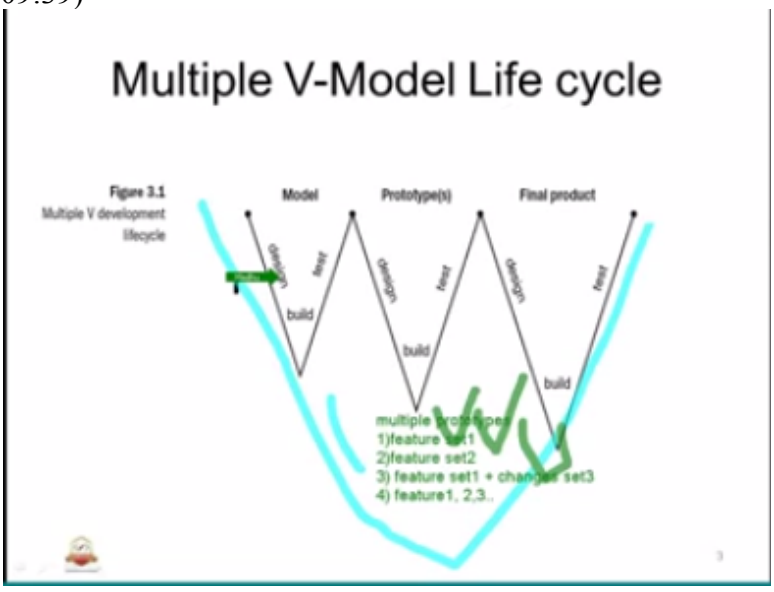
So likewise we have a scalable or incremental way of having the model used the testing the different physical versions of require this product requires specific techniques and specific testing environments. We know that different prototype will have a different I mean deviated slightly or the as per therefore a clear legend the multiple V model can be various testing environments. So system is developed as in sequence of thousands of systems development that become more really in the end. We have sub systems it is called multiple physical versions it is called sub systems usually piece per model prototype and final product could be anything what we develop.

So the multiple V model based on the multiple V model it's also called a spillner 2000 in the development model so it is called in there what is that called a lifecycle elements the government design coding, testing all together it is called in a developed model is the fundamental multiple

model is based on the v-model in principle. Each of the product appears product appearance follows completely development cycle including design code and testing hence the term multiple V model that means we have a different variants of the prototype developed as a multiple v cycle so that is why it is called a multiple V model.

So this is typically used in production environment where they may not be able to conclude on the final end product or they might not have designed such a way that one go they will develop the final product what they do is they will first identify they basic V model in that again they will identify subsystem V models so all these subsystems V model complex into multiple V model and check so this is the master plans for everything will definitely study about master plan maybe in the next session okay.

(Refer Slide Time 09:39)

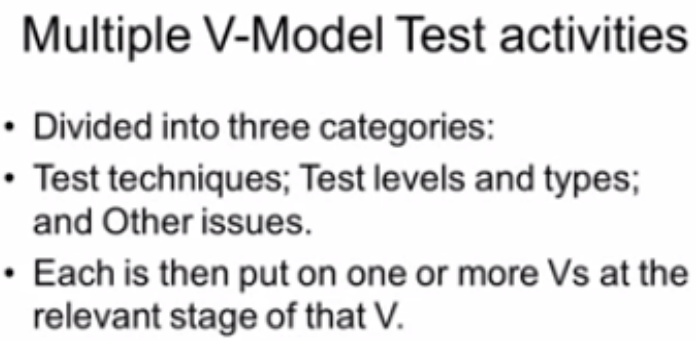


So this figure you can see how it looks like model V development lifecycle we know that we have studied one week in the earlier section it could be for a model prototype models for a formal one. So one each one has a design source code development and build and testing design-build test again bill test so you can see three V models one is the smaller one under the medium become so doesn't mean that it is same actually it can have a different aspects like we might have a detailed design all over logarithms and all the stuff taken care in the bigger one and the smaller one may not need it again depends on what sort of a development we are going to adapt.

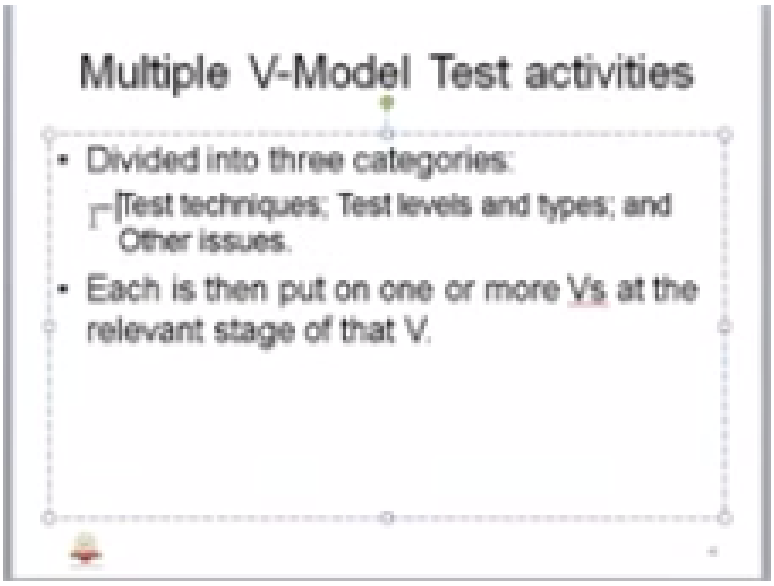
So basically the model or the initial prototype will have a smaller V model life cycle and prototype will have the next. So i have put S that booklet is mentioned as S why because there could be multiple prototypes. Having a feature set features set one in the first prototype and first prototype having a feature set one second one having a feature set two, third one could be having feature set one plus some changes or you can call it as set 3, both could be having feature1, 2, 3 partially set up so all this can be they are part of the prototypes.

So we can add multiple V like this I just run in between it can have a multiple V models before we draw on the final product. So that is the meaning of this multiple V model life cycle and I

again tell that if you really having multiple development lifecycle defined for each of the variants like we have a initial prototype depend as a model.
So often work curve before the feature set prototypes are something like we the users will develop a proof of concept application or the feature set 1, 2, 3 etc, and be accordingly we will have a life cycle that also, all this together we will draw a bigger V model that is the final product or a formula V model life cycle we can call. So this also can be part of a usually this will be a part of a master plan you can call this as a one single having multiple V set so it is called multiple V model life cycle.
(Refer Slide Time 13:35)



So in multiple V model life cycles we have our testing activities given below it's basically divided into three categories they are test techniques, test levels and types and other issues. Test techniques test level and types and other issue so these three categories physically he divided the V model test activities it is then put in one or more piece at the eleventh stage of it that means basically we will identify all these categories then we will apply into each of this so that is how it is been done for it is triangular.
(Refer Slide Time 14:34)



So we will study about each of these three categories so you can see this table, techniques test levels types other issues this is actually this whatever I have tested in this model V model multiple mode and in the next slides i have later V models this is all basically they read from the testing embedded software a book by drokman and edwin note book because there are several techniques.

But this is a book that is being referred for this course embedded software testing here okay. So test related activities and issue, issues that need to be allocated throughout the development and testing lifecycle what are those? Here development testing why it need to be mentioned is development also can be having some of this test because developers as a tester is very important for embedded software. Constant I will explain that in a different slide developer as a test algorithm how it is, techniques so what are the techniques that we there should be considered for a multiple V models. This is specific to multiple V model code coverage analysis consist to testing okay.

Fagan inspection a little more effective analysis it is also called as sections fault injection, fault analysis, formal verification, face restrict, model checking, location testing, random testing, these are the techniques that are used or it can be considered need not be all some of this based on the applicability and corresponding test level for types.

Where they can be applied for example core coverage analysis can be applied for design verification control flow testing code review. So there is a term called control flow very clear for control flow and data flow some of the very important items so that we need to be knowing for embedded software testing In order understand control flow is basically the and the controlling also or the way how the program is controlled embedded software product is getting controlled in terms of the logic or flowing on the entire system data flow is where now it is getting rolled on throughout the structures.

So Fagan inspection it is a type of instruction required Maybe I will we will discuss this some detail later for conform this testing. We follow this I am assuming for our design state verification, fault injection where we have integration where we inject the signal levels of any

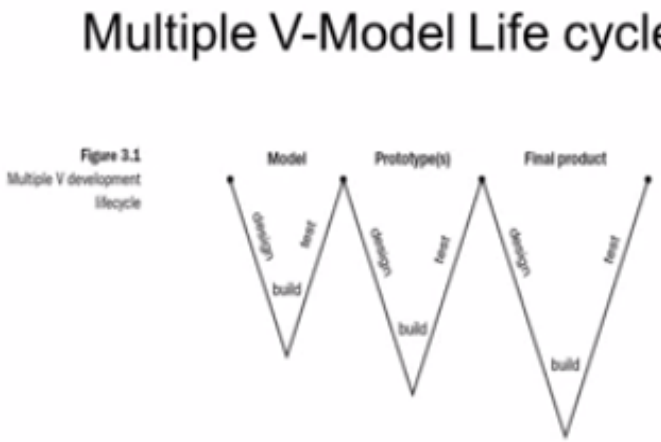
data errors in terms of system it will be to the integration similarly. This involves the host problem of target here what we do is we begin analysis of the fault so what is the root cause of a particular fault types where we will predict that means if I inject an error where and all is going to impact this the needs to be analyzed.

So against the requirement definitely requirement will take care of the fault recovery and multi-detection and recovery mechanism all this will it in terms of faulting analysis formal verification model integration test integrate testing, field test then we have a model checking with integration testing requirement certification, software acceptances, integration test unit test. So these are some of the best levels are types so based on the type of product that we use this test levels types corresponding to the techniques.

We can use it and we have other issues and card category architecturally designed, certification detailed design, test plan design and build tools, design and build simulator design and built stuffs. I will explain about stuffs and simulator design and build drivers and then for testability is one of the very important thing that now almost all industries their following especially product companies like Samsung the design phase so they will identify the testability part that means whether this design can be tested. So without that testability design is not complete or design will not have improved for mechanism.

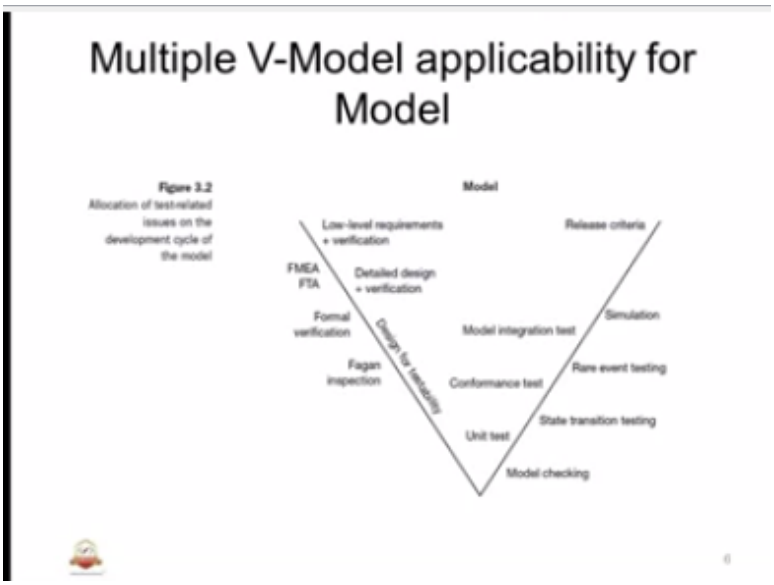
So they did not have some sort of a design we can support or help testing that could be test hooks or that could be test tops or drivers or test monitor whatever it is, and of course we have a high level requirement medium level requirements, low level requirements, master test plan we will continue master test plan later, production requirements, release criteria. So this is the third category and so this is a complete master list of three categories this all need to be considered before we take up the multiple V model.

(Refer Slide Time 21:20)



Those techniques levels and other issues can fit into any of this I mean so basically they we need to draw a picture when we do the V model when we do the multiple V model planning all this have to be considered so that is the main thing about multiple V model.

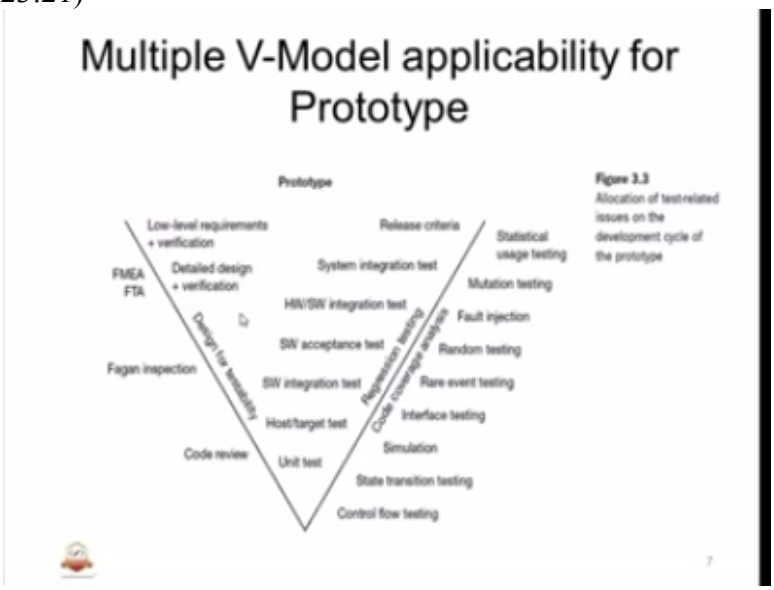
(Refer Slide Time 21:53)



So now we consider only the first one model, multiple V model, and applicability for model that means what is the example out of this how much we can draw for the model so it is nothing but to allocation is called allocation of test related issues on the developmental issue of the model. So low level requirements in verification, detailed design verification, design for testability so all this will be on the left hand side so the V model is in the curving shape and corresponding to this requirement we have a few criteria, we have an integration model integration we have a controllers test we have unit test these are some of the selected a techniques test level types and other issues.

So for this so what is outcome like we have a simulation we have these are some of the activities similarly on the left hand side there will be FMA, FTA formulation verification for the inspection so this is falling outside the where on forward the model so these are the applicability how we are going to draw based on this master table okay.

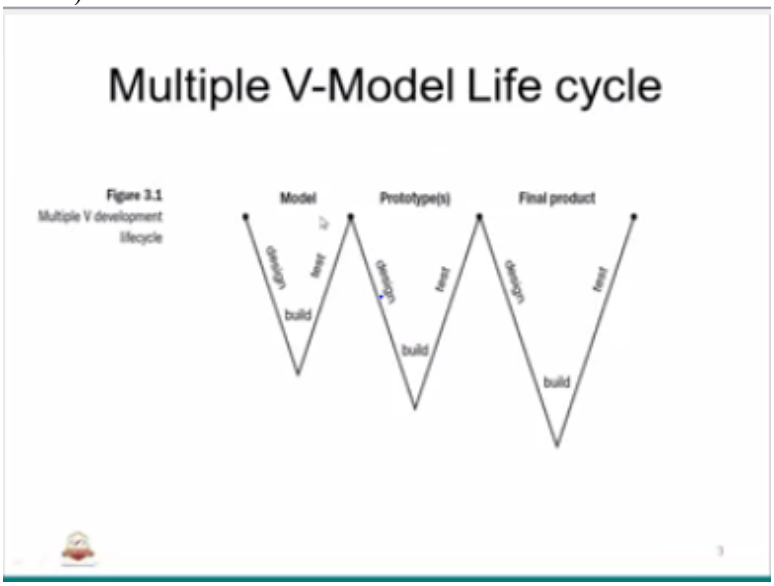
(Refer Slide Time 23:21)



Another one is V model applicability for prototype, what are the categories is that can i use for prototyping V model here again outside the V we have techniques in terms of that maybe A inspection code review within the V we have low level requirements detail design plus verification requirements, designs for testability then on the right-hand side we have any tests at the bottom host target based testing we have software integration software acceptance then we have harder software integration then we have system integration.

Then we have the package integrities on the right hand side so this can go for a multiple testing days on the applicability and all that help of technician testing and also we can hope of that we come up with a coverage analysis means how much we are covered what is left out what is past what is pay code coverage analysis, statistical I usage testing, mutation testing fault injection, random testing, rare event testing, interface testing, simulation, state transaction testing, control flow testing, so this on the right hand side is all about the techniques what we apply for this prototype, this is an example of prototype V model.

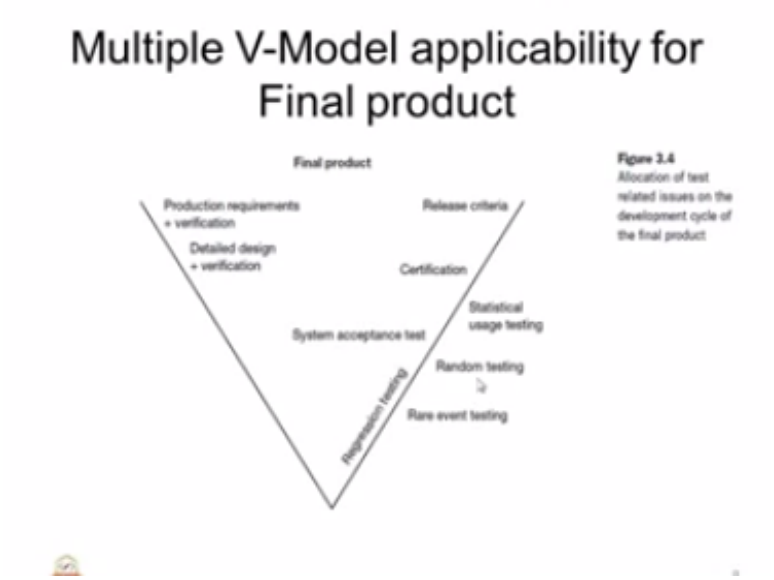
(Refer Slide Time 25:26)



When it comes under applicability under multiple V model of life cycle final product the formal product how we can apply the multiple V model product requirements that means N sort of requirement detailed design. And because V code all we already taken care in a prototype so final product of a, with only the requirement and detailed design and in terms of releasing or deploying formal things to the customer we have at least criteria acceptance and all that, certification is important.

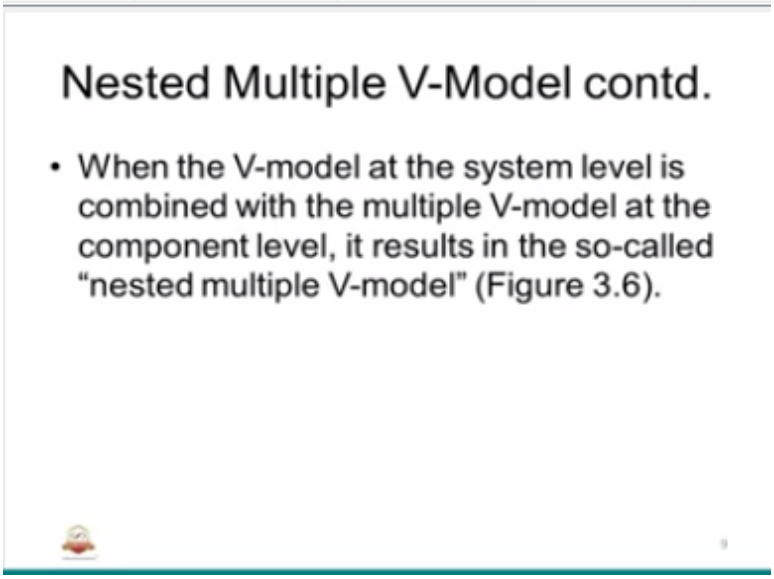
If we especially the products are going to be repaired in terms of safety aspects that is to be taken care certification like we have D17B from designated authority from Airbus or Boeing automatic we have autos also Boeing it is the certification process and in terms of the techniques statistical are you hormone testing so this will be a applied before we complete the V model this is a location for a final product or the formal product applicability.

(Refer Slide Time 26:40)



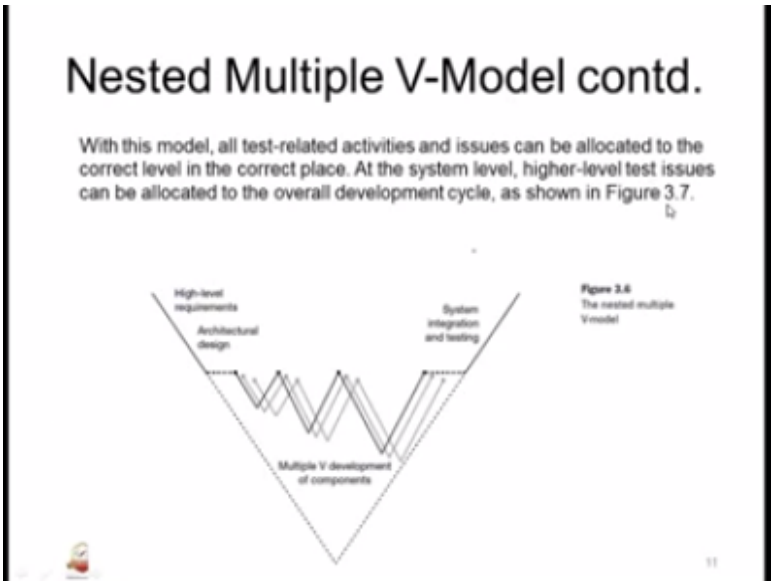
So that is about the multiple V model where we have initial prototype, prototype final product have been organized into multiple product each one having design code this design code test and finally we are going to have the final or the formal product and its applicability is that on a must list of techniques is levels types and other issues okay.

(Refer Slide Time 27:19)



The next one is a nested multiple V model we all studied about V model now studied about to multiple V model then another term called a nested multiple V model. The name itself tells the multiple models are nested basically definition is here when the v-model at, the system level is combine with the multiple V model at the component level its regards in the so called nested multiple V model. So agreeing is that system level we have V model it is combined with multiple V model having prototype final and before for prototype. So once this is a combined at the component level so we have defined in system level the combo power level multiple models which will integrate.

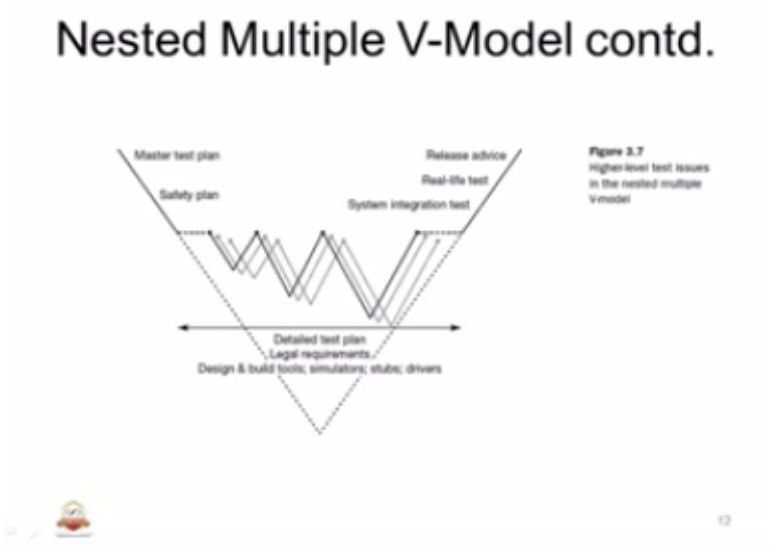
(Refer Slide Time 28:31)



So it is so called nested multiple V model so we have an example law here you can see this is a multiple V models one multiple V model two N etc so component one component, component two, component N things nested under the one V model. So this is an example of how one example of taking up a component as a vertical V model at the component level. So, parallel development is also called as parallel development interfacing V model.

So okay in continues of that then instead we tested multiple V model all test related activities ambitious can be allocated to correct level in the correct place against whatever we identify testing activities instead to be applied for the particular level the component one component two component three is specified test will be applicable for that specific V whatever we have followed nested multiple V model. So at the system level higher level test issues can be allotted overall development cycle is one more bigly three points 7

(Refer Slide Time 30:04)



We have seen a parallel development and in terms of V model component 1, 2, 3etc, when it comes to Nestor to multiple V model system level missed related activities can be allotted into

the correct level. They are for this actually you can see in the bottom little this plan legal require design built simulator stuffs etc, all this will be underneath that means this common same thing is going to be applied for V model one, V model two, V model three as an example 3 V you can see at each V at the system level component level of people going three times you can see.

So high level test issues so next multiple we model another example is taken carrier at this they have master test plan, safety plan we are at least applicable for all this and then recovery V model. So when it comes to the individual V model disposition is taken care in a nested multiple V model same thing is deputed in same way here because the 3.6 that might be will V model components put together it comes to high level test the common line extra basically. So the multiple V model with a three sequential recycle that we are using does not take into compounder the practice of other functional decomposition of the operate system development of such system.

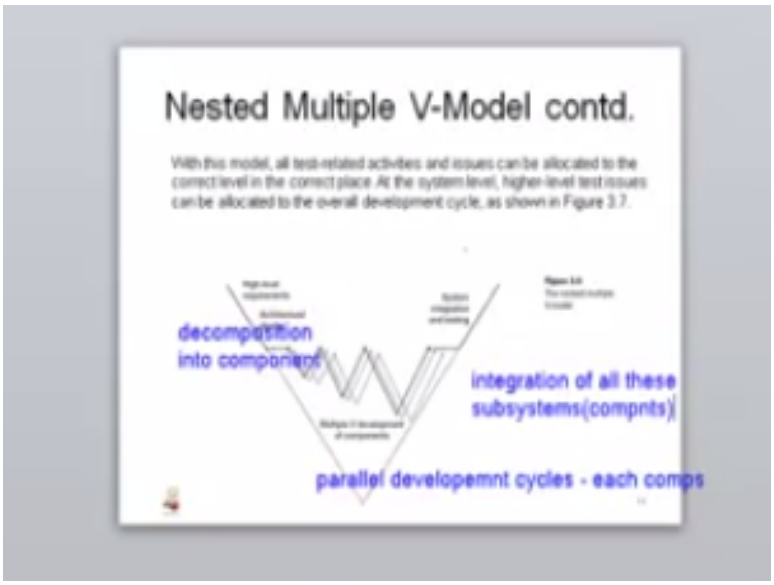
Followed by an tactical design is where it is determined its components hardware or software are required to realize this and those components are them develop this greatly and commonly integrated in full system in fact the simple V model can be applied to this development process at a high level the left side of a V model handling decomposition of the system into its component the middle part of the V model consists of parallel development cycles what we will see now here of all the components the right side of the v-model are testing the integration of the compounds so I repeat this as the left side of the V model decomposition of the system into various compounds.

And we said high level requirements design low level design coding etc, on the middle part of the construction for the development cycles like we have a prototypes 1 2 3 or components finally hitting below the right hand side we are going to have a validation or integration of all activities all these aspects. So this principle can be repeated for these components too big or too complex to develop as one entity for such a component and architecture design activities carried out to determine.

Which sub components are require a architecture design how many sub components sub components are required to maintain system working so it is also could result in another v-model within the V model we can know another V model and that is why it is called as nested. So multiple V model is not only helpful this has to be planned and executed for completion product but also be get some for your user's often current product. So in fact is the development project for a product is about the new release of just a few components. Then the multiple V model and full for the master plan related to the development of the complete product can help to identify the relevant test activities.

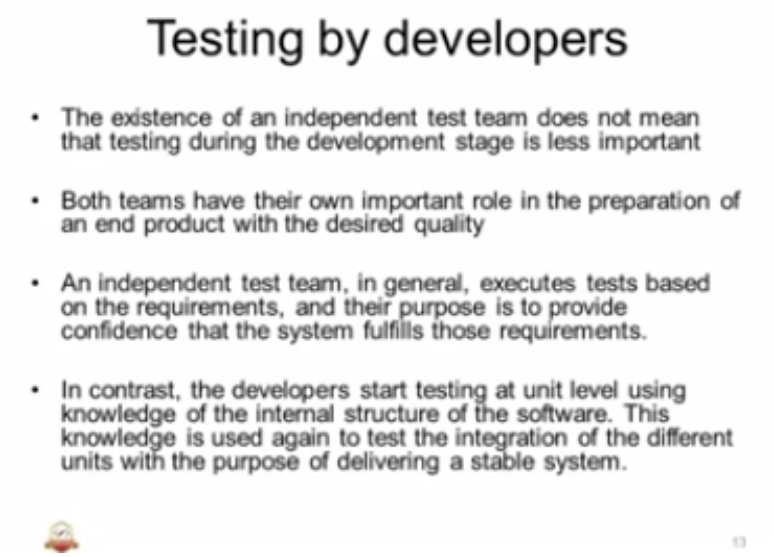
So basically when you have a complex or a bigger systems what we do is we go the parent development of different rephrases and listed on left hand side divided basically in two components we are in develop a system with multiple components. So the middle part we will identify the recycle for each of these components and the right hand side we will have the integration.

(Refer Slide Time 37:03)



So on the left-hand side of the optical nested V model what we decomposition into sub systems or components. The middle part we have the parallel development recycles because of each of the component for each other each component on the other side we have the integration of all these subsystems, components so that is the significant so for multiple V model in context with the listing.

(Refer Slide Time 38:02)



So basically it is a used in the bigger or complex systems where we have numerous features I differed for numerous component. So that is a significant of multiple V models, applicability and also nested nature okay having understood this all different types and their activities categorized now we move into the testing aspect irrespective of who or whatever the model we follow the picture. So we know that term host and target are available for both the test team as well as the development team, so now taking into consideration how testing can be done by developers how testing can be done by independent testing and then a few principles so that we to go through we'll go through in the end.

Testing by developers so we know that the systems are developed by a larger things sometimes divided into teams into multiple allocated on subsystem teams or it could be physically separated we go back go bigger systems or complex systems like embedded systems or aerospace embedded system not developed at one side it will be geographically separated though they have a inter connection and all that the team the testing that testing environment or the components or the inner lab and all the systems supply.

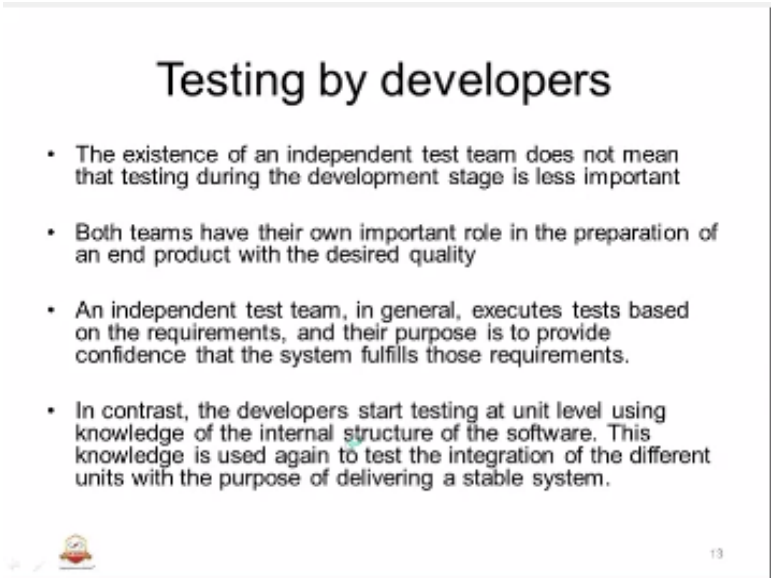
so the develop the systems are large and complex and that alone puts quality under the pressure all should work in random make sure that they work for the quality and the end result that is integrate for the product the demand for equality is increasing. so two aspects one is quantity market because with it also so developers available so we need to come up with a good quality also there is a safety aspect been take care so in order to avoid all this at a later stage so better to do a testing.

little thoroughly in the starting stage for the beginning stage, so beginning stage we know that there will not be any accessing of formal testing though there are this plans and all that, so the best way to do is testing done by developers, so that is the background why testing we need to have it, by the developer, so what are those?

The existence of independent test does not mean the testing during the development stage is less important that means which equally important to have a testing done at development time both teams have their own important role in the preparation an end product with the desired quality, it means they have their own testing team in the independent testing where as the developers or the tester they own responsibility, so the end product should be having a high quality with the desired call.

An independent test team general executes tests based on the requirements their purpose is to provide confidence that the system fulfills the requirements, so they go by what you are specified what the product is of course to do what you have documented. What you have specified in the requirements they just go by that, where as in contrast to that the developers start testing at unit level because they are very near to the input and they are the owners of the unit, because they have developed, because they own it.

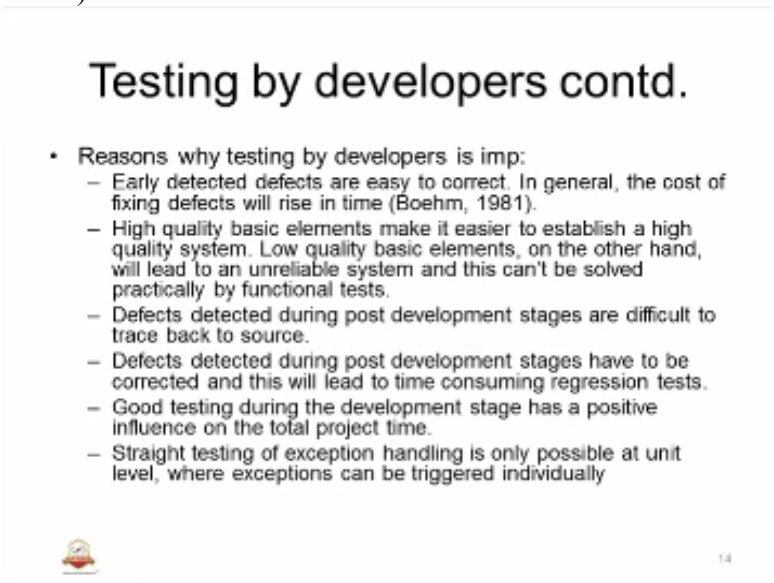
(Refer slide Time: 41:38)



And they are having knowledge that means, they know about a internal structure of compound, this knowledge used again to test the integration of the different units in the purpose of delivering stable system, so what they make sure is, they develop the units piece of software and since they are knowledgeable they use the same test mention are the develop or debug environment to test it or in terms of unit level or in terms of integration level to address two things one is quality.

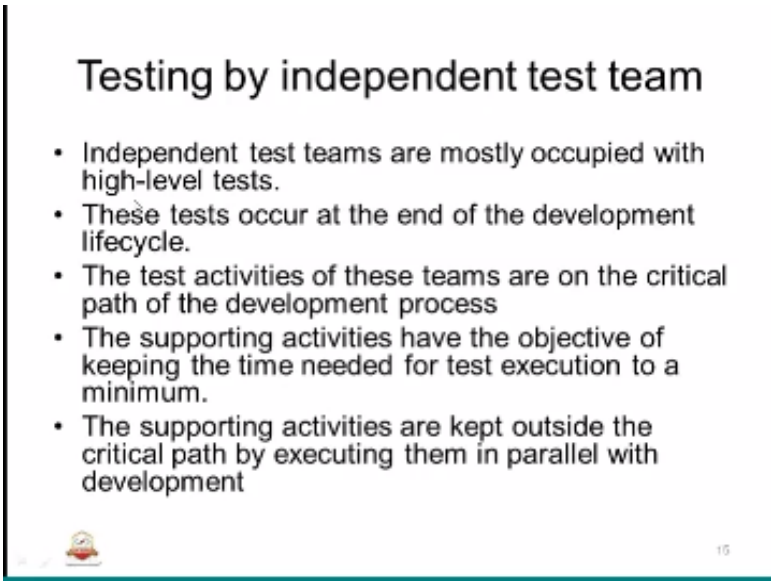
So that, there are no bugs creped un-knowingly certain thing is, the system that, they develop they should be stable, so those two purposes are attained, with the help of what kind of testing at the early stage.

(Refer slide Time: 42:35)



So that is why we need to have a testing by developers, the early stage of development, so continuation reasons why testing developer is important, early detected defects are easy to correct in general the cost of fixing defects will rise in time, we know that we have seen a chat

where the cost arises as the go deep into the product, before we use a product we cannot afford to have a box for issues at the young stage expensive.
High quality basic elements make it easier to establish a high quality, that means basic elements are put it right in the beginning only, so they are (N) product the whole product will come output, low quality basic elements or anything turn unplayable system and this can't be solved a practically by functional test.
It means you would have crept some quality issues small into different elements in the beginning, in the end it's very difficult to detect some of the box is that got just because of the low quality in the development, if x detected during post development stages are difficult to trace back to source that is it, we cannot afford to have this small detected during the development stage, so this is difficult to trace it back, so better identify in the beginning only, defects detected during a post developing stages have to be corrected and this will lead to time consuming regression test we keep the regression again it again because the detects that are being done on the product and again need to be tested, good testing during development stage has the positive influence on the total project time.
That means we have the confidence we have the productivity liability extra, written in the end stage because of the good testing we have done in development stage, state testing of a description handling only possible at the unit level where exceptions can be triggered individually, I will tell about exception and interruption some time in later stages, because embedded system development word you may call it as an interrupt as a result of exceptions, exception are somewhat unknown behavior of the system in certain conditions, where the unit level or the source level it may be difficult a later stage detected fix at the development, it may not be possible quickly to do it testing as well as the fixing at a later stage, that’s why we need to have some sort of testing done in the development in the product.
That means basically quality cannot be added the product with the end by testing, there should be built in right from the start, taking the quality at every development stages very much necessary, so how we can do to do is by testing, testing is a not pursued by many developers above the rewarding task didn't developers have seen ignoring the Telstra aspects they will never care about the testing aspects, so importance to a testing thinking listing the thoughts while doing the development, so to keep the efforts for the developers is a good practice we should effective and efficient , to do this integration (46:38)maybe we can go through that again sometime in the discussion. So essentially all activities have to be planned and should be controlled, so that is the ultra mock testing by development stage.
(Refer Slide Time: 47:12)



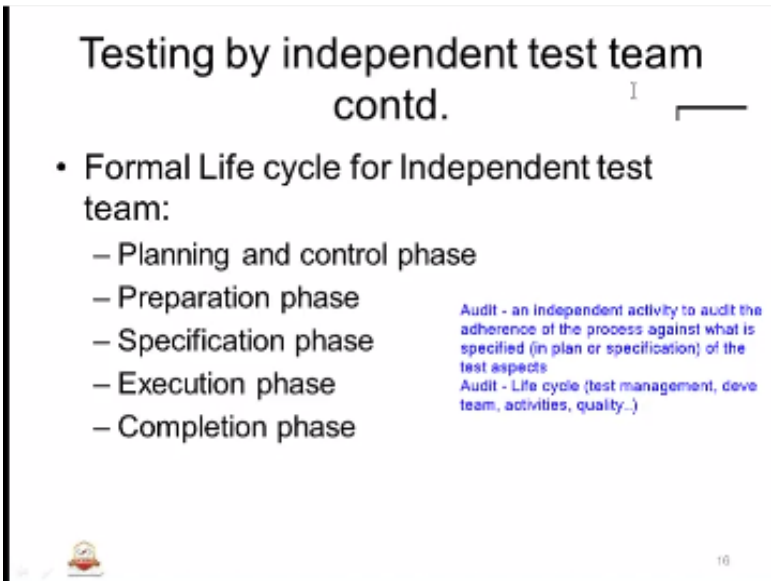
Now testing by independent test team, we all know that our testing is an important aspect there needs to be a testing plan, testing to see you within done by a separate team, we have a test plan also, so we'll just go through that what are those listing by independent testing, independent teams are mostly occupied with high level testing understood they go by the high level test having requirements aspects this test occur at the end of the development life cycle.

We know that development electrical is a getting closer the testing activities will start, the test activities these teams are on the critical path of the development process, that means it's very important element in the critical path with running critical it item that in under critical path on the whole lifecycle, supporting activities have the objective of for keeping the time you get to for testing execution to a minimum, it means.

Whatever the support activities is going to have the objective of keeping time or test execution came anywhere, that means when we are doing the development when we are identify test activities and planning, so we do small test hooks and all as a support the objectives will have a credible time.

How much it is needed for test execution at least a minimum or how much it is need for each of the tests so that is gone is not the idea behind supporting activities, supporting activities are kept outside the critical path by executing them that are with support activities latest books develop and maintain the strategy is the in parallel development all taken care and separate activity supportively forward development as well as testing so this is will not be in critical path what is used while testing execution or actual testing is happening which is under critical path formal life cycle independent testing basically.

(Refer Slide Time: 50:39)



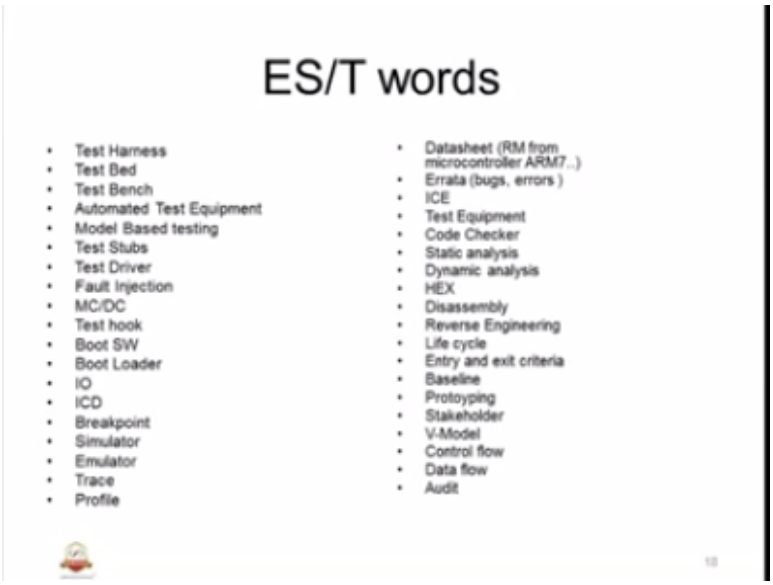
What are the life cycle items offices for a independent testing planning and control phase, we have preparation phase, specification, execution, completion this is basically if different phases that is been sending here these are all mapping to what we have discussed in the earlier classes other way but in terms of formal life cycle for independent testing taking requirements as a input and doing the high level test gives they are to plan and control they have to prepare they have to specify get into execute and complete it and so these are the phrases that are used for the independent testing okay.

This is basically a formal what is described here because here this involves independent testing including the test management and development the lead also will be involved because he needs to know what works what we wish to fix so what is a test how it is ticketing the pass or fail so most of the information needed by the testing and outside the testing that means whatever they need it on the product should be provided.

And each stakeholder have his own objectives and the end the expectations tango testing sometimes of the outcome of the subject to an extent audit that means there is a separate audit that is taken care against what is being specified. So there is an audit team or the independent team who does the audit on testing is that important is nothing like independent activity to audit the adherence of the process against.

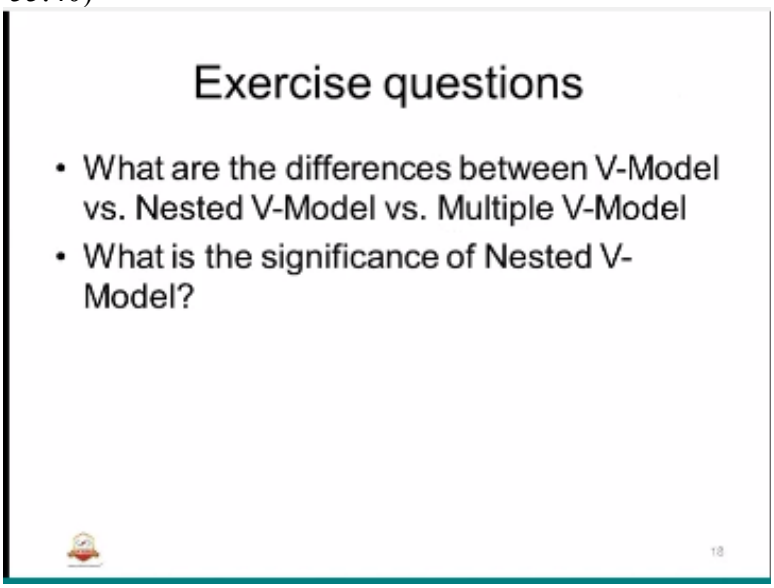
What is specified plan or specification of the test aspect? So this audit can happen or any aspects of the life cycle it can happen on the test management audit can be done on the test management development team and their activities quality etc, do not get confused with the audit and quality. Quality is a different thing audit is a different thing, so these are the formal electrical aspects that are used, so that is about a testing by (L) purpose testing by independent team,

(Refer Slide Time: 53:56)



Now we'll move on to ten principal of embedded testing I will take in next session as we are running out of session, so will go through the words which we have done before as well today. So we model a couple of things added control flow data flow, that what we have is audit if anything is needed we can add test harness, test bed, test bench, automated test equipment, model based testing, test stubs, test driver, fault injection, MC/DC, test hock, boot SW, boot leader, IO, ICO, break point, simulator, emulators, trace, profile, datasheet, errata, ICE, test equipment, code checker, static analysis, dynamic analysis, HEX, disassembly, rename engineering, life cycle, entry and exist criteria, based line, prototyping, stakeholder, V model, control flow, data flow, audit. we want to highlight a we can do it think they are taking care of all this V model where we show ok so that is about some of the words way these words are going through a section to have this in mind as a tester.

(Refer Slide Time: 55:40)



So that will not forget this will be thought embedded software testing that's why it is important ok so we have couple of questions proposition what are the differences between V model poses

nested V model multiple V model so we thought a difference between V model multiple V model and testing what is the significance of nested V model so why we need it so they are discussing about this please answer these questions so that is about this section next section we will see the next aspect of significance of nested V model.