Embedded Software Testing
Unit 4: Software Integration
Lecture 3
Seer AKademi-NPTEL MOU

(Refer Slide Time: 00:07)

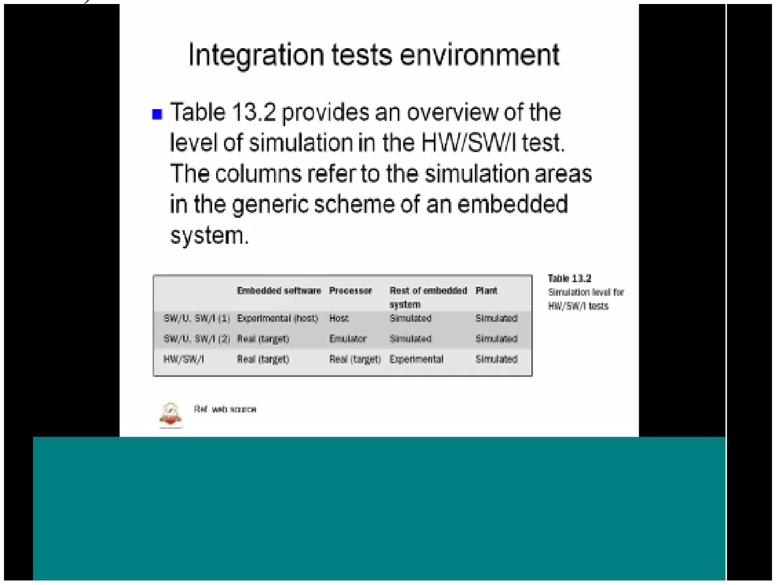
Embedded Software Testing
Unit 4: Software Integration

Lecture 3
Seer Akademi – NPTEL MOU

1

Welcome to the next session of the embedded software testing unit 4, this will be the last lecture session of software integration, we have a regression testing, we will study about the software integration and configuration and other aspects. The environment and followed by that we will discuss.

(Refer Slide Time: 00:36)



Integration tests environment

- Table 13.2 provides an overview of the level of simulation in the HW/SW/I test. The columns refer to the simulation areas in the generic scheme of an embedded system.

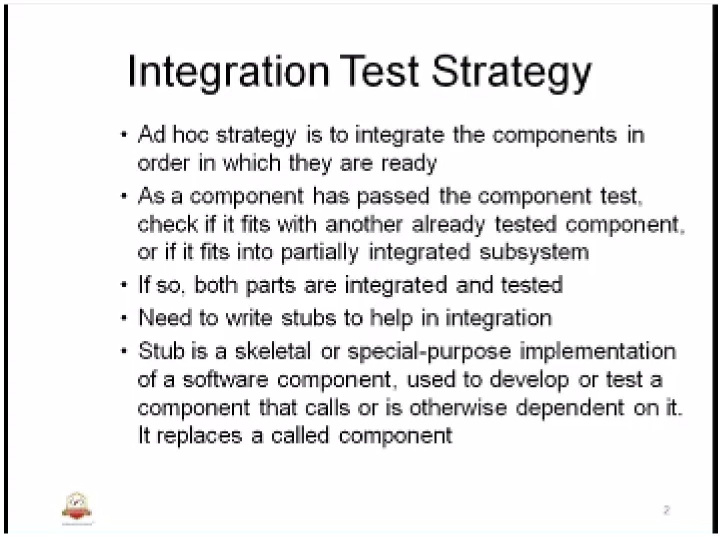
	Embedded software	Processer	Rest of embedded system	Plant
SW/U, SW/I (1)	Experimental (host)	Host	Simulated	Simulated
SW/U, SW/I (2)	Real (target)	Emulator	Simulated	Simulated
HW/SW/I	Real (target)	Real (target)	Experimental	Simulated

Table 13.2 Simulation level for HW/SW/I tests

Ref. web source

Study about integration testing okay, to recap on software integration,

(Refer Slide Time: 00:44)



Integration Test Strategy

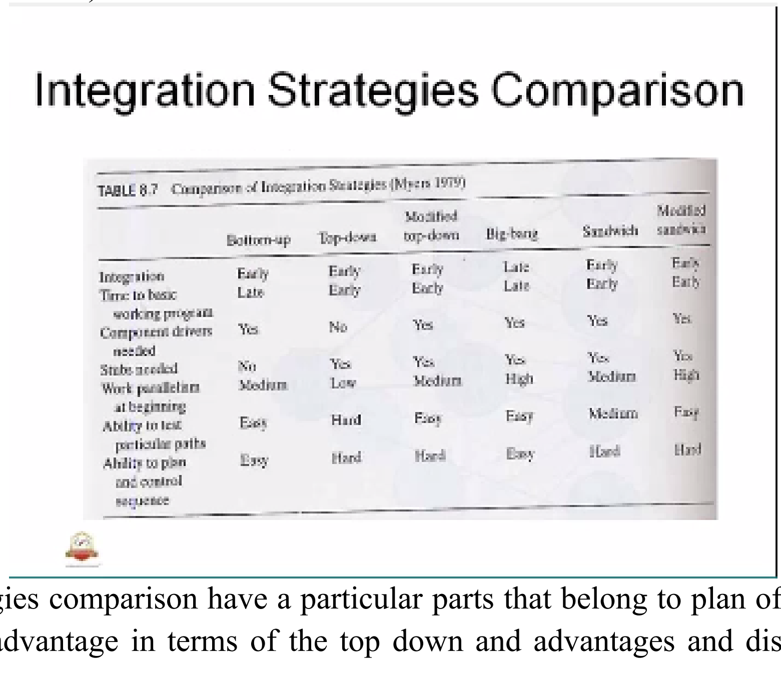
- Ad hoc strategy is to integrate the components in order in which they are ready
- As a component has passed the component test, check if it fits with another already tested component, or if it fits into partially integrated subsystem
- If so, both parts are integrated and tested
- Need to write stubs to help in integration
- Stub is a skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component

2

we know that the integration can be done in software level, is called system integration test, software integration test, hardware software integration test, so to recap the integration strategy, will not just enough to have a component level testing, but to have an adherence at the system level, how the components interact in module level when they are logically moved, we need to work out those strategy, to integrate those component.

So there are different analogies, like in strategy as they developed we can give or, we can start with the software modules using lowest possible units. So it will used stuffs and drivers, is called test drivers, yes we can make sure that, were the different stages are their development, integration can be taken up, so different strategy we had a comparison in terms of time and maintenance of component terms.

(Refer Slide Time: 02:04)



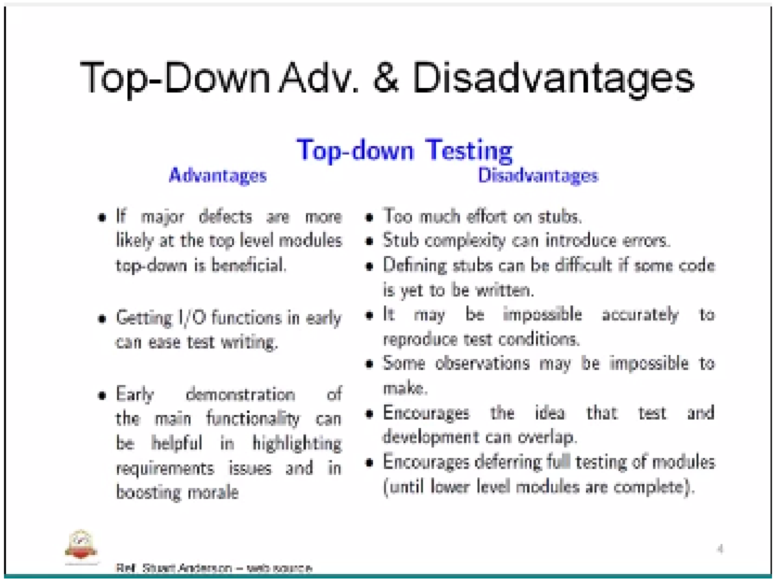
Integration Strategies Comparison

TABLE 8.7 Comparison of Integration Strategies (Myers 1979)

	Bottom-up	Top-down	Modified top-down	Big-bang	Sandwich	Modified sandwich
Integration	Early	Early	Early	Late	Early	Early
Time to basic working program	Late	Early	Early	Late	Early	Early
Component drivers needed	Yes	No	Yes	Yes	Yes	Yes
Stubs needed	No	Yes	Yes	Yes	Yes	Yes
Work parallelism at beginning	Medium	Low	Medium	High	Medium	High
Ability to test particular paths	Easy	Hard	Easy	Easy	Medium	Easy
Ability to plan and control sequence	Easy	Hard	Hard	Easy	Hard	Hard

Integration strategies comparison have a particular parts that belong to plan of sequence also we had gone to the advantage in terms of the top down and advantages and disadvantages of top down testing.

(Refer Slide Time: 02:24)



Top-Down Adv. & Disadvantages

Top-down Testing

Advantages

- If major defects are more likely at the top level modules top-down is beneficial.
- Getting I/O functions in early can ease test writing.
- Early demonstration of the main functionality can be helpful in highlighting requirements issues and in boosting morale

Disadvantages

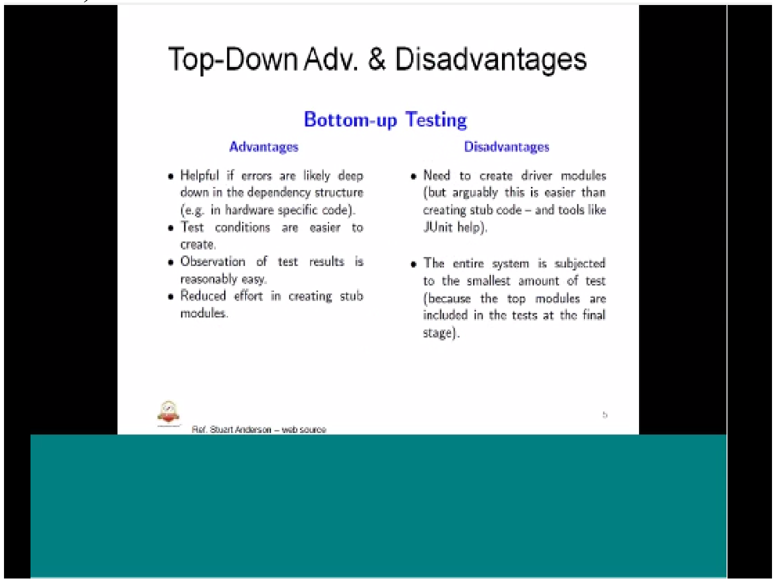
- Too much effort on stubs.
- Stub complexity can introduce errors.
- Defining stubs can be difficult if some code is yet to be written.
- It may be impossible accurately to reproduce test conditions.
- Some observations may be impossible to make.
- Encourages the idea that test and development can overlap.
- Encourages deferring full testing of modules (until lower level modules are complete).

Ref. Stuart Anderson – web source

4

Whether it is bottom of, at the lowest level of possible, in terms of having issues, better do the bottom of testing, so basically we look in feel of the system in terms of demonstrating or it could be as a structure, embedded system can be well demonstrated and testing from the top set module were in the bottom of testing is started with the lowest possible modules.

(Refer Slide Time: 02:37)



Top-Down Adv. & Disadvantages

Bottom-up Testing

Advantages

- Helpful if errors are likely deep down in the dependency structure (e.g. in hardware specific code).
- Test conditions are easier to create.
- Observation of test results is reasonably easy.
- Reduced effort in creating stub modules.

Disadvantages

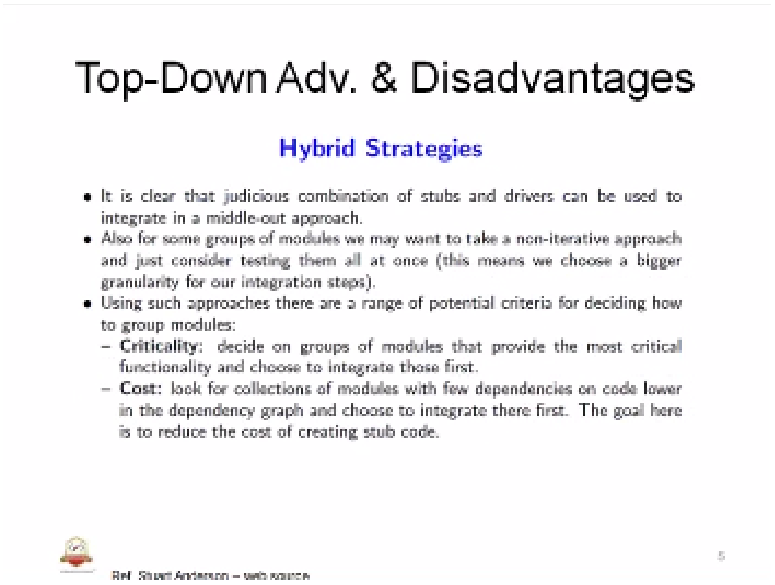
- Need to create driver modules (but arguably this is easier than creating stub code – and tools like JUnit help).
- The entire system is subjected to the smallest amount of test (because the top modules are included in the tests at the final stage).

Ref. Stuart Anderson – web source

5

As they are completed drivers, because drivers used to call them so that is where we used bottom of test strategy, but we need to have, that is why this the disadvantage is the, because, we do not know what the tool is used for creating the layer, you cannot use the disadvantage of the bottom of testing, were are in top down testing, there are lot of small stubs that may be required in disadvantage, it cannot be accurate, there could be some errors, it takes significant and effort and time, in terms of stubs. The other type of test called hybrid test which is used both of as bottom of so this is preferred in were the largest system has given.

(Refer Slide Time: 04:22)



Top-Down Adv. & Disadvantages

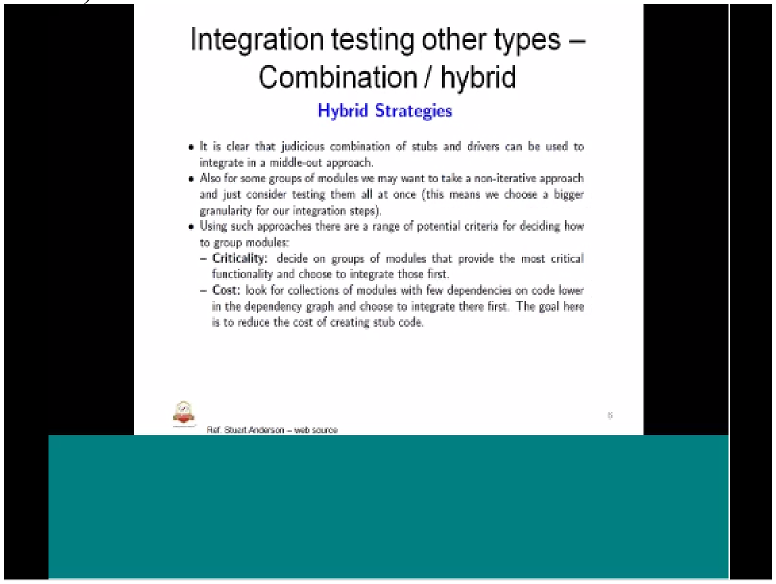
Hybrid Strategies

- It is clear that judicious combination of stubs and drivers can be used to integrate in a middle-out approach.
- Also for some groups of modules we may want to take a non-iterative approach and just consider testing them all at once (this means we choose a bigger granularity for our integration steps).
- Using such approaches there are a range of potential criteria for deciding how to group modules:
  - **Criticality:** decide on groups of modules that provide the most critical functionality and choose to integrate those first.
  - **Cost:** look for collections of modules with few dependencies on code lower in the dependency graph and choose to integrate there first. The goal here is to reduce the cost of creating stub code.

Ref. Stuart Anderson – web source

5

(Refer Slide Time: 04:23)



Integration testing other types – Combination / hybrid

Hybrid Strategies

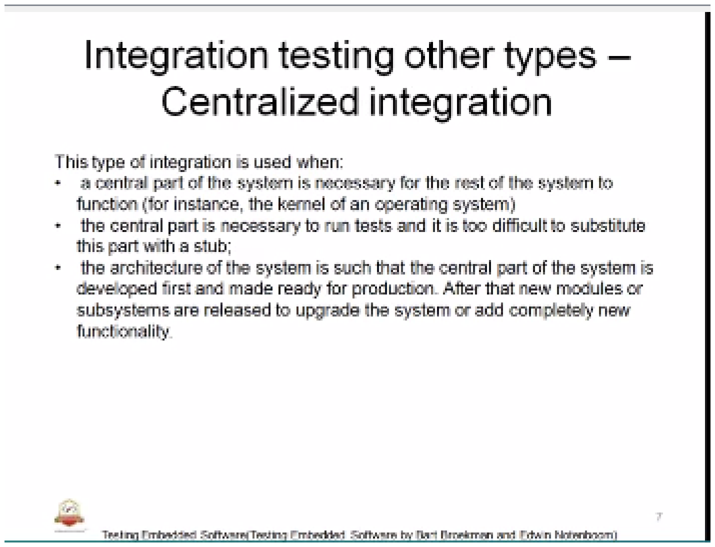
- It is clear that judicious combination of stubs and drivers can be used to integrate in a middle-out approach.
- Also for some groups of modules we may want to take a non-iterative approach and just consider testing them all at once (this means we choose a bigger granularity for our integration steps).
- Using such approaches there are a range of potential criteria for deciding how to group modules:
  - **Criticality:** decide on groups of modules that provide the most critical functionality and choose to integrate those first.
  - **Cost:** look for collections of modules with few dependencies on code lower in the dependency graph and choose to integrate there first. The goal here is to reduce the cost of creating stub code.

Ref. Stuart Anderson – web source

6

Were lot of sufficient are here ,small to moderate complex, and embedded systems are , so it is beneficially have hybrid strategy which has combination of both test stubs and both the drivers, it is a middle of approach , and both of top and bottom of , is out of mechanism, so this approach are basically based on the criticality, that the system has, so it is very subjective, so based of it you can take a call, using hybrid strategy, were to applied is one of testing mechanism in terms of integration.

(Refer Slide Time: 05:20)



Integration testing other types – Centralized integration

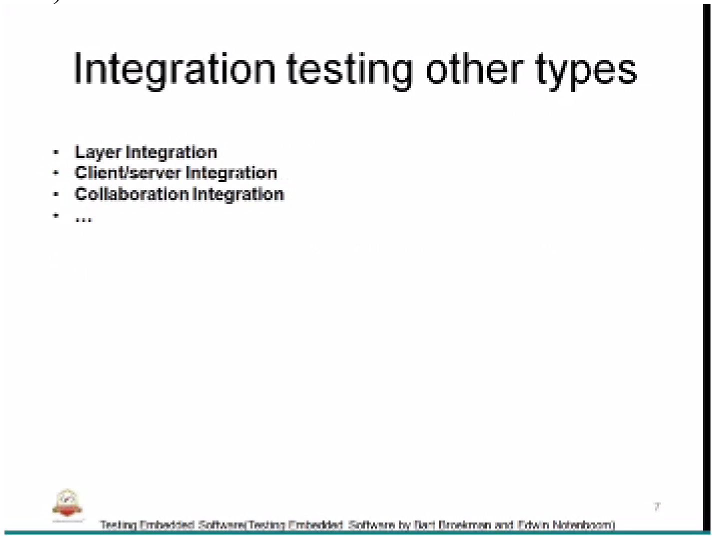
This type of integration is used when:

- a central part of the system is necessary for the rest of the system to function (for instance, the kernel of an operating system)
- the central part is necessary to run tests and it is too difficult to substitute this part with a stub;
- the architecture of the system is such that the central part of the system is developed first and made ready for production. After that new modules or subsystems are released to upgrade the system or add completely new functionality.

Testing Embedded Software(Testing Embedded Software by Bart Broekman and Edwin Notenboom) 7

So another type of integration are testing we have gone through called centralized integration, these are more appropriate of wires, because we cannot replace those hard of the testing system so solving that we have going to develop anything that required, we cannot substitute , it is too difficult to substitute the part with a stub, so these type of approach is called centralized integration, so surrounding the center line the part of the system, it developed , first and made re fall production.

(Refer Slide Time: 06:05)



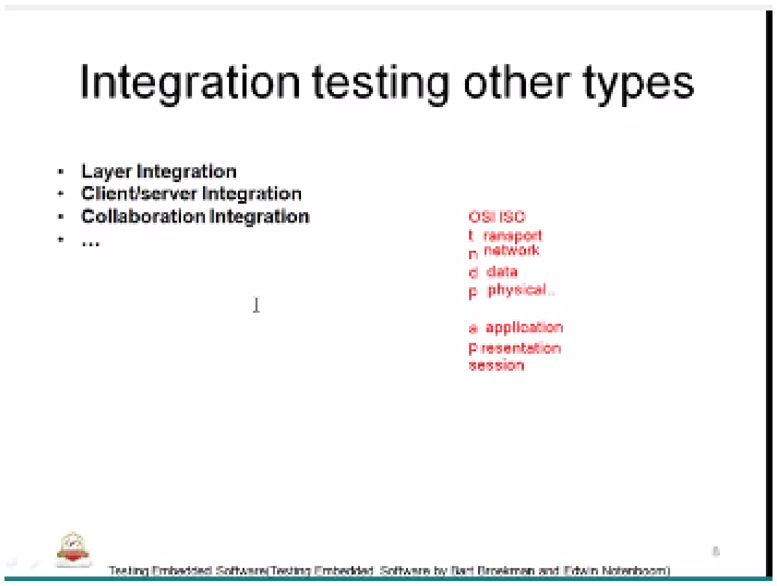
Integration testing other types

- **Layer Integration**
- **Client/server Integration**
- **Collaboration Integration**
- …

Testing Embedded Software(Testing Embedded Software by Bart Broekman and Edwin Notenboom) 7

The other type of an integration is, layer, layer integration has a different layers, application layer lower layer and hardware checking layer, this is particularly used, which want to tell us that, we reaches the protocol.

(Refer Slide Time: 06:32)



Integration testing other types

- Layer Integration
- Client/server Integration
- Collaboration Integration
- ...

OSI ISO
t ransport
n network
d data
p physical..

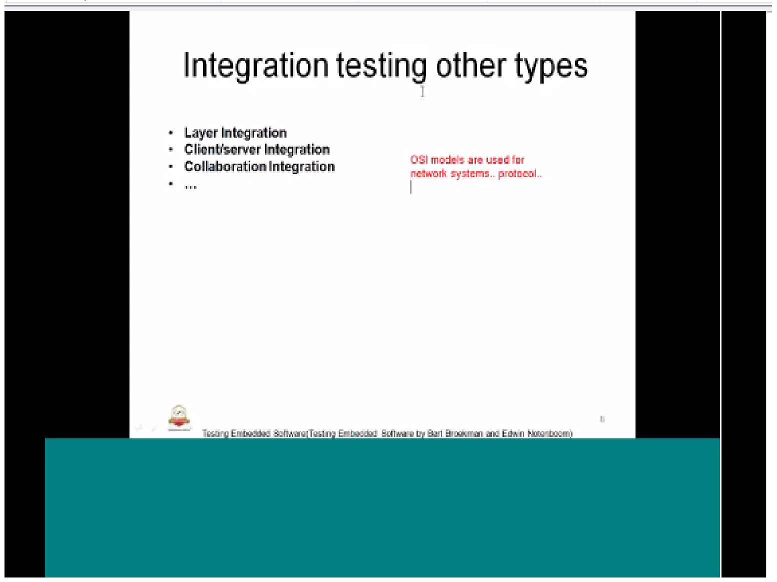
a application
p resentation
session

Testing Embedded Software(Testing Embedded Software by Bart Broekman and Edwin Notenboom)

Such as, we need a protocol ready to have layer and integration testing, so you know that, OSI have a scope network data layer, these are the important type elements of course we have application, presentation excess layer, there are top most of that.

So it assumes the presentation layers, application layer, then we have a transport, network, data and physical like that we can segregate so something like layer.

(Refer Slide Time: 07:48)



Integration testing other types

- Layer Integration
- Client/server Integration
- Collaboration Integration
- ...

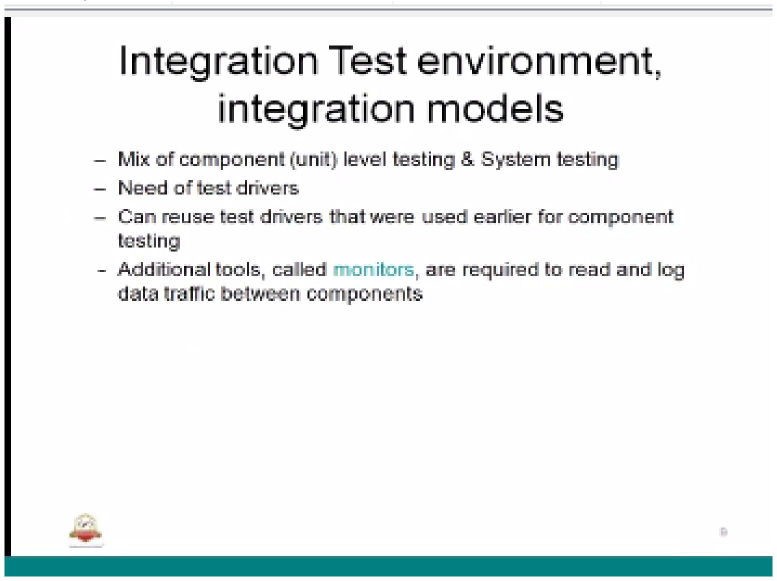
OSI models are used for network systems.. protocol..

Testing Embedded Software(Testing Embedded Software by Bart Broekman and Edwin Notenboom)

So we have seven layers, approach were protocols are networking application used this OSI models are used for network systems were protocol, network protocol, IFTP, IPU, UTP, all this were protocol mechanism we have approach.

So in this case better to use type of integration were will be developed in integration testing, so that type of integration testing, client server integration, we had study about this were the server is starts, were the client is addressed, were the client is stub, when server is be tested, so it is like collaborative, mechanism in terms of client in server is typically used were the injectors and huge application data based oriented, object oriented system. We have, my last session is

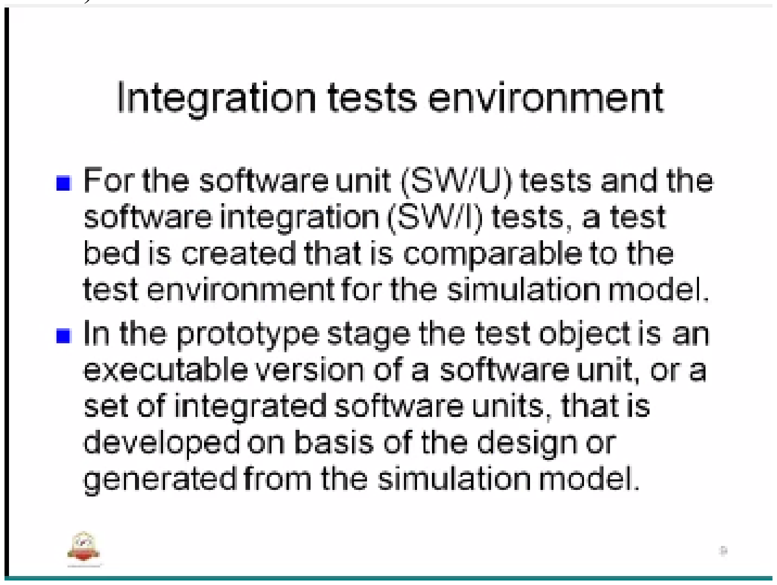
collaboration integration, here we interact different collaboration models, those models can be integrated together this is called as collaboration integration
So those have been different types of integration testing that we have studied then.
(Refer Slide Time: 09:10)



Integration Test environment, integration models

- Mix of component (unit) level testing & System testing
- Need of test drivers
- Can reuse test drivers that were used earlier for component testing
- Additional tools, called monitors, are required to read and log data traffic between components

We came to integration test environment, how it looks like, so basically it should have a mix of both unit level testing as well as system testing, because it develop a driver with a help of.
(Refer Slide Time: 10:32)

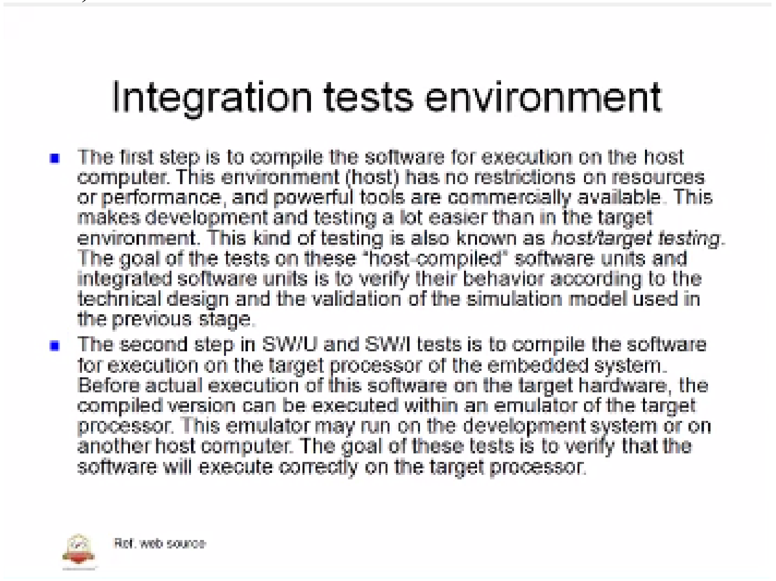


Integration tests environment

- For the software unit (SW/U) tests and the software integration (SW/I) tests, a test bed is created that is comparable to the test environment for the simulation model.
- In the prototype stage the test object is an executable version of a software unit, or a set of integrated software units, that is developed on basis of the design or generated from the simulation model.

Integrated environment, that environment could be used for developing the system and testing the each system, of identification, similarly we used a lowest possible, large test driver, so those which are also needed, so both sort of environment required from the integration testing, it just like integration model, which is comprise of, which will be comprising of both unit level as well as system level and environment.
So basically use some of test drivers, which have been working, which are used for induction through logically test in different models together. Additionally we may need a tools like

monitors to log and read the data traffic protocol, there will be a log and time standard all that, to do that we need to have an appropriate additional tools, it just monitors.
(Refer Slide Time: 11:28)

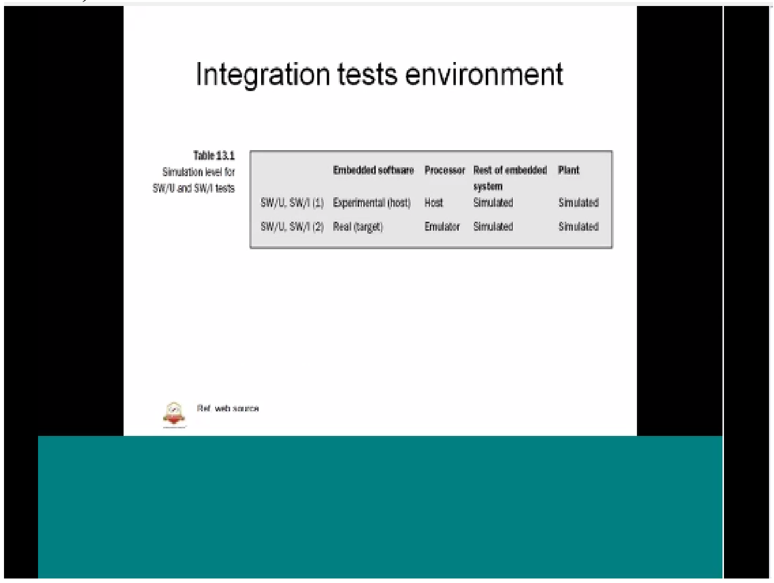


Integration tests environment

- The first step is to compile the software for execution on the host computer. This environment (host) has no restrictions on resources or performance, and powerful tools are commercially available. This makes development and testing a lot easier than in the target environment. This kind of testing is also known as *host/target testing*. The goal of the tests on these "host-compiled" software units and integrated software units is to verify their behavior according to the technical design and the validation of the simulation model used in the previous stage.
- The second step in SW/U and SW/I tests is to compile the software for execution on the target processor of the embedded system. Before actual execution of this software on the target hardware, the compiled version can be executed within an emulator of the target processor. This emulator may run on the development system or on another host computer. The goal of these tests is to verify that the software will execute correctly on the target processor.

Ref. web source

Integration test environment, so we know that the software unit test and the software integration test, it has bed which is created, that is comparable test and environment for the simulation model, so we know that simulation and invention and actual hardware that target were testing or used, so accordingly we are going to have a test environment, so that is there, we develop the integration test term.
(Refer Slide Time: 11:40)



Integration tests environment

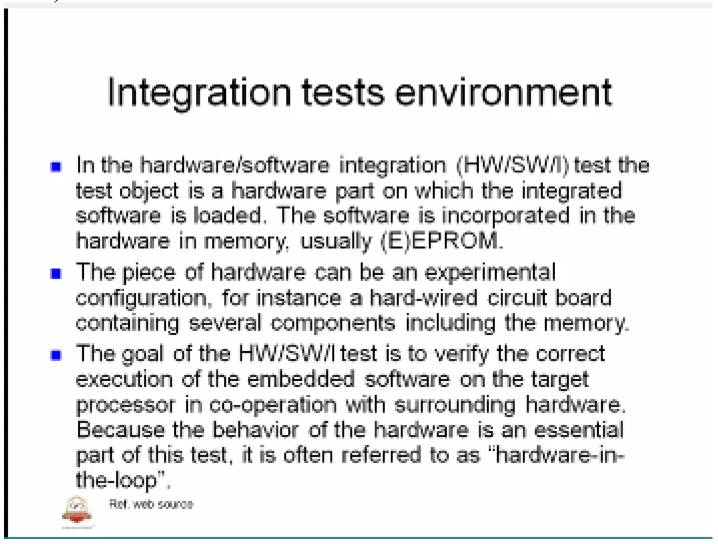
Table 13.1
Simulation level for SW/U and SW/I tests

	Embedded software	Processor	Rest of embedded system	Plant
SW/U, SW/I (1)	Experimental (host)	Host	Simulated	Simulated
SW/U, SW/I (2)	Real (target)	Emulator	Simulated	Simulated

Ref. web source

So basically proto stage and production stage are different type of integration we have, so both the version we can need a test object and executable version of a software unit, or a integrated software units that we used, so that is the basically developed on the basis of the design or generated from the simulation model, So based on that those integration test environment can be developed.

(Refer Slide Time: 12:05)



Integration tests environment

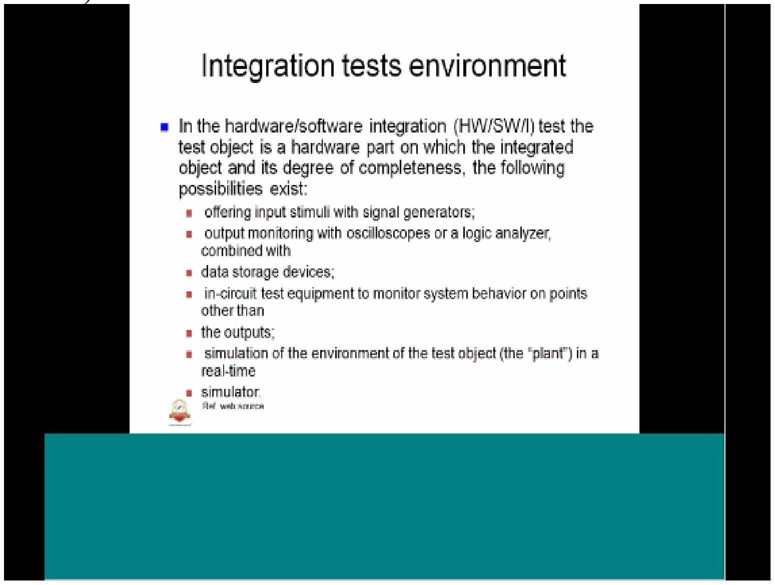
- In the hardware/software integration (HW/SW/I) test the test object is a hardware part on which the integrated software is loaded. The software is incorporated in the hardware in memory, usually (E)EPROM.
- The piece of hardware can be an experimental configuration, for instance a hard-wired circuit board containing several components including the memory.
- The goal of the HW/SW/I test is to verify the correct execution of the embedded software on the target processor in co-operation with surrounding hardware. Because the behavior of the hardware is an essential part of this test, it is often referred to as "hardware-in-the-loop".

Ref. web source

So in continuation of this, the host environment target base environment and then host target can be used as well in terms of integration test environment also we have concentrating from the book, in terms of software , simulation how it can be done? And walked were it can be used, host can be used while doing the experimental integration testing, were we use the enumerator it need to have a real target , having the target processors used.

So actual hardware does not have any transceivers or application host, it need to have a mechanism , such as hardware in loop , or hardware software in loop are sort of a test environment for integration test . specifically is useful for hardware and software, now this software's are in integration, software integration are the host space in term were the test environment, the target base environment we need to have a hardware software terms, target environment are bound to be integration test.

(Refer Slide Time: 12:44)



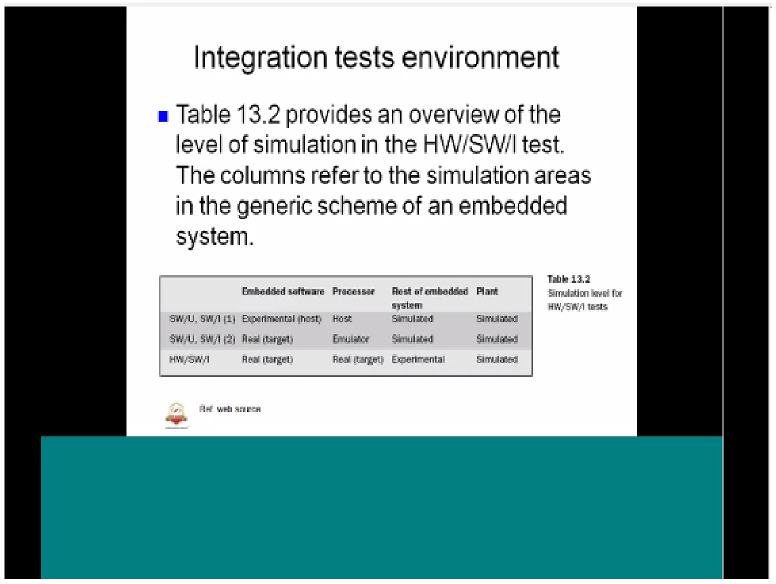
Integration tests environment

- In the hardware/software integration (HW/SW/I) test the test object is a hardware part on which the integrated object and its degree of completeness, the following possibilities exist:
  - offering input stimuli with signal generators;
  - output monitoring with oscilloscopes or a logic analyzer, combined with
  - data storage devices;
  - in-circuit test equipment to monitor system behavior on points other than
  - the outputs;
  - simulation of the environment of the test object (the "plant") in a real-time
  - simulator.

Ref. web source

So that is about the nil instruction today you will study more on.

(Refer Slide Time: 12:56)



Integration tests environment

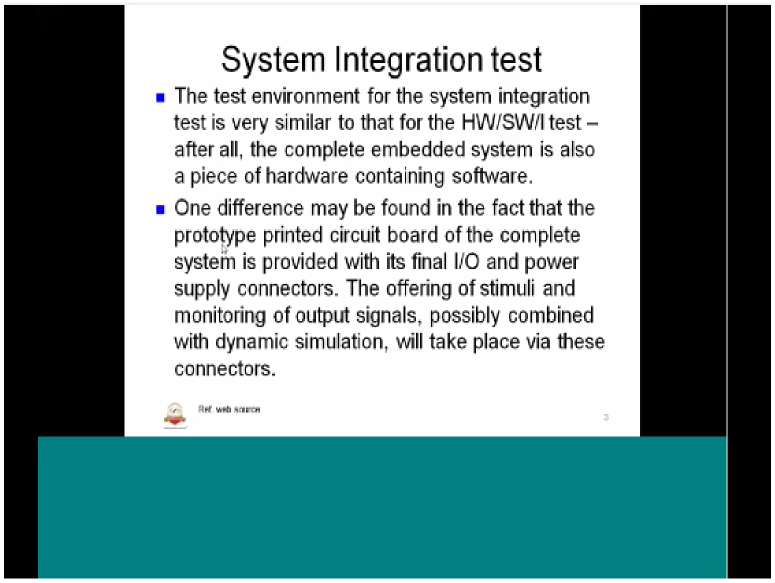
- Table 13.2 provides an overview of the level of simulation in the HW/SW/I test. The columns refer to the simulation areas in the generic scheme of an embedded system.

	Embedded software	Processor	Rest of embedded system	Plant
SW/U, SW/I (1)	Experimental (host)	Host	Simulated	Simulated
SW/U, SW/I (2)	Real (target)	Emulator	Simulated	Simulated
HW/SW/I	Real (target)	Real (target)	Experimental	Simulated

Table 13.2 Simulation level for HW/SW/I tests

Ref web source

For the integration test environment, the below table is a simulation level of software integration test, it can have sort of simulation in hardware and software integration, were there is simulation scheme; the embedded software can be experimental host. Some of the pieces of embedded software can be experimental or run on the host, the next one could be a real hardware , which uses some of the software units, the first one also software can be used in the hardware mode, the last one is software techniques can complete the environment, and system environment having the embedded software.
So the various processor that has been used for simulation in the hardware and software integrating the first base of test is based SW/U SW/1,2 ,HW/SW/1 these are the embedded software processor, the OS can be windows, Linux, whatever it could be, but the host execution will be based on the processor, and inter case rest of the system are formulated, a plant is something like where we are going to , what are the unit called simulation the second one will have a unit of.
Because we are using the real target, and we need to simulate the unit system itself completely with help of this simulator, and a last one we have the real target and
The processor is real target, there is no simulation of the target processor, that is how we going to have the layers of the simulation in terms of integration testing, so which ever provides OS of the simulation test , because refer simulation in the generic scheme of an embedded system.
(Refer Slide Time: 15:25)


System Integration test

- The test environment for the system integration test is very similar to that for the HW/SW/I test – after all, the complete embedded system is also a piece of hardware containing software.
- One difference may be found in the fact that the prototype printed circuit board of the complete system is provided with its final I/O and power supply connectors. The offering of stimuli and monitoring of output signals, possibly combined with dynamic simulation, will take place via these connectors.

Ref: web source

So continuation of the integration test, there is a system integration test which is in the higher level on the integration test of hardware software integration test, so what are you do here? The test environment are have to system integration test is kept; the system integration test is very similar of a hardware software integration test of the complete embedded system also a piece of hardware containing the software.

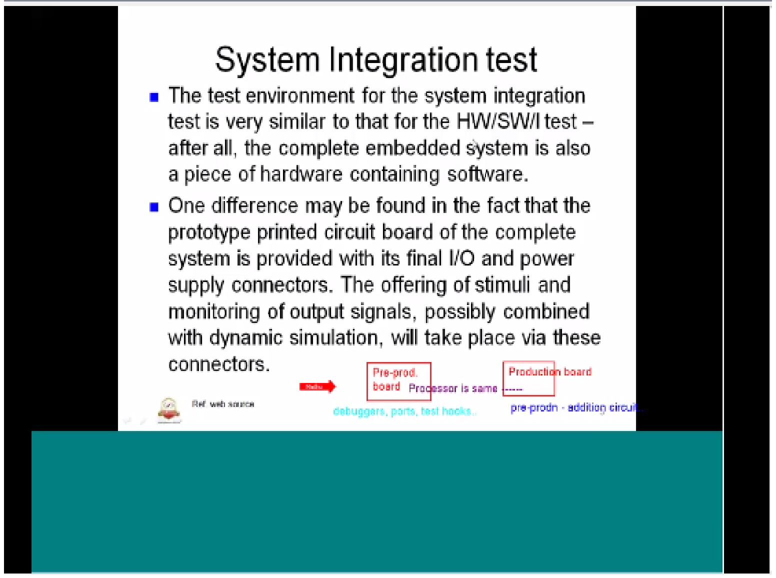
So it is not something different it is a whole hardware software and integrated once combine together, something like a system test, within also can be called as system integration test, they impressive and focus on the integration of various logically model in the embedded system or the embedded software, one difference may be found in the fact that, the prototype printed circuit board of the complete system is provided with it is final input and output and power supply connectors. The offering of stimuli and monitoring of output signals possibly combined with dynamic simulation. It takes place via these connectors.

So actual target board we are gone to use and prototype of inter circuit mode of the complete system , will be founded and their production testing were we use the actual input and output power supply can access , and we use the system integration on test strategy for doing this and developed mode. Basically the pre production mode has system integration mode, what it will have basically.

We use that, we used to draw that pre production board, we can called it as a preproduction board, so what is the difference between both, what will happen means ? if there are going to be any stubs or any fetches are doing there, all those final stages has to be work in preproduction and it test that, definitely we need a, any of the test hooks or in terms of loops, all this intermediate stubs will be odd of this preproduction board, which is slightly a larger than the actual production board.

Both will have the same processor, basically processor is same , in the both cases what will happen when the system circuits will required will be taken off , so basically what will happen means ? the pre production – additional circuit will be removed, in a production mode, so it can be used slowly for them, they providing the target, system or the feel actually, which will receive to the customer, it will not go back to the factory, so we will found the final production mode, so all this system does accepting the production mode, were as should be integration we are going to use the pre production mode, so that is the difference line okay,

(Refer Slide Time: 17:24)



System Integration test

- The test environment for the system integration test is very similar to that for the HW/SW/I test – after all, the complete embedded system is also a piece of hardware containing software.
- One difference may be found in the fact that the prototype printed circuit board of the complete system is provided with its final I/O and power supply connectors. The offering of stimuli and monitoring of output signals, possibly combined with dynamic simulation, will take place via these connectors.

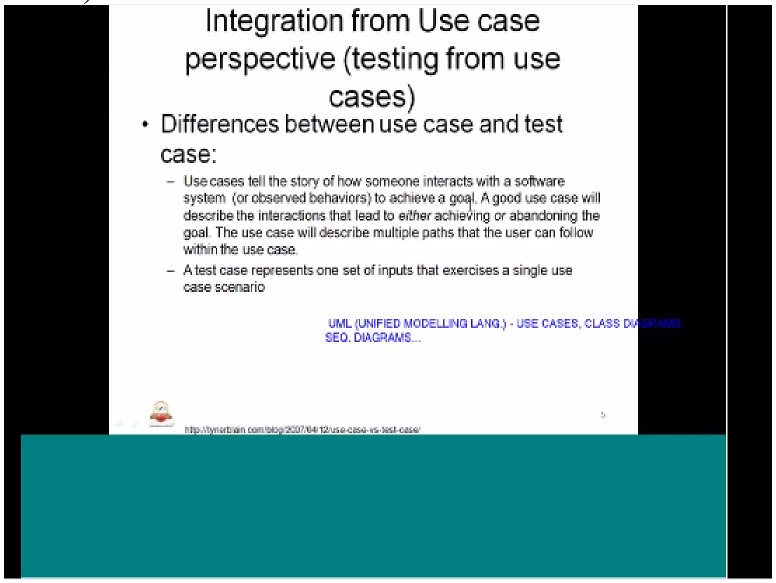
Ref. web source

Pre-prod. board
Processor is same -----
Production board
debuggers, ports, test hooks..
pre-prodn - addition circuit...

So the next one is the system integration test, this table basically defeats the system integration test, we can see them last term is adding here, and the real target system integration, and rest of the system are prototype.

And actual client is simulated, so which ever provides an overview of the level of simulation in the system integration test, the columns refers to the simulation areas in the generic scheme of the embedded systems, so the earlier one , we do not have the system integration in terms of actual target and actual system will be there, so that what is in the prototype, so this will be used for system integration test, as well as the preproduction testing it is called as okay, So let us see the next topic on the integration test.

(Refer Slide Time: 22:20)



Integration from Use case perspective (testing from use cases)

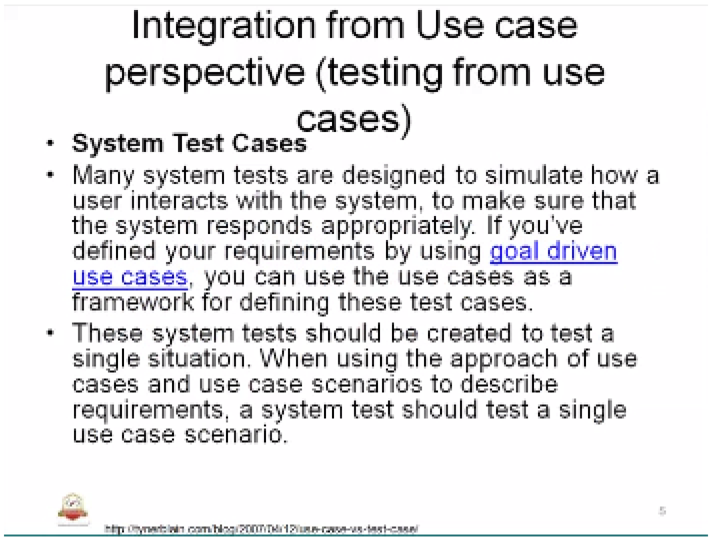
- Differences between use case and test case:
  - Use cases tell the story of how someone interacts with a software system (or observed behaviors) to achieve a goal. A good use case will describe the interactions that lead to *either* achieving or abandoning the goal. The use case will describe multiple paths that the user can follow within the use case.
  - A test case represents one set of inputs that exercises a single use case scenario

UML (UNIFIED MODELLING LANG.) - USE CASES, CLASS DIAGRAMS, SEQ. DIAGRAMS...

http://tynerblain.com/blog/2007/04/12/use-case-vs-test-case/

5

So this is basically the use case is derived or use case, integration test, so what we do here? You may knowing about use cases, for basically use case is derived in terms of the UMN, that thing we have a explanation about that we have know, perspective , because we are going to discuss on the UMN ,why? Because of the UMN study required basically let us understand that, MODELING LANGUAGE which uses cases CLASS, SEQUENCE etc.
So basically we are going to take it how the system looks like? so something like a model we are going to have, so from the users perspective, here the cases are going to developed or the classes are going to developed are called use cases, so definitely we know that cases are being developed, we have an input , what is the expected all that are excursed, so it will have for cases we can develop it test cases, and that use cases are constant on way on the higher level models of the boards for the functionality of the features of the system, so that is all the we can derive the test cases, on this use cases.
That is why is called a integration from use cases perspective, the same testing from use case, which are further going to be broken down in to test cases. So the use case tell the story of how someone interacts with a software system or the observed behaviors that is what these use cases to achieve a goal , a goal could be accepting some value, and computing something some other value.
Or achieving some of the results, that could be a functional war , for that goal , a good use case will describe the interactions that lead to either achieving or abandoning the goal , so basically use case describe the interactions for achieving the goal, or resulting a path it will going to take for that functionality, so use case will describe the multiple parts, the uses can be followed within the use case, so use cases can have different paths, we know that the follow of the program control level and data level follows.
And it is subjective to the complexity and the flow in to the types, similarly multiple path, use case can receive, so you will also can be used to draw the test cases.
A test case can represents one set of inputs that exercise a single use case , at least use case will have multiple scenarios, so one test case per each scenario , they gone to have , so that is were test case have been derived on the use cases, let us see more examples on integration on use case perspective.
(Refer Slide Time: 25:10)



Integration from Use case perspective (testing from use cases)

- **System Test Cases**
- Many system tests are designed to simulate how a user interacts with the system, to make sure that the system responds appropriately. If you've defined your requirements by using goal driven use cases, you can use the use cases as a framework for defining these test cases.
- These system tests should be created to test a single situation. When using the approach of use cases and use case scenarios to describe requirements, a system test should test a single use case scenario.

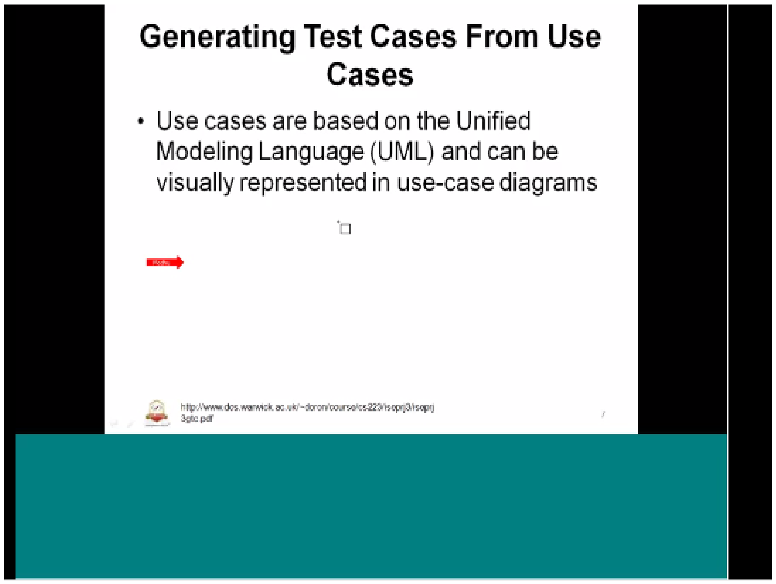
http://tynerblain.com/blog/2007/04/12/use-case-vs-test-case/ 5

Integration from use case perspective system test cases how are going to function? Many system test are designed to simulate, how a user interacts with the system, to make sure that the system responds appropriately, if the system needs to be subjected to some from an interaction and the responds, if you have defined your requirements by using goal driven, are use to achieve some goal, on so many paths.

You can use the use cases as a frame work of a design of a test case probably the use case is important and useful why? Because this frame work on top of which a lot of test cases and it goes for frame work, the frame work is basically use cases, also called as boundary one use cases, so what are they going to come up with ramp of the boundary cases, is nothing but the system test cases, the system test should be created to test a single situation, if each test will try to test the integer scenario or the situation.

When using the approach of use cases and use case scenarios to describe the requirements, a system test should a single use case scenario, so what is trying to say his, each use case, every single use case scenario to have attest case, in that test case we are going to have system test case know where addressing to the requirements. So that is where the requirements are going to be tested from the use case perspective.

(Refer Slide Time: 27:14)



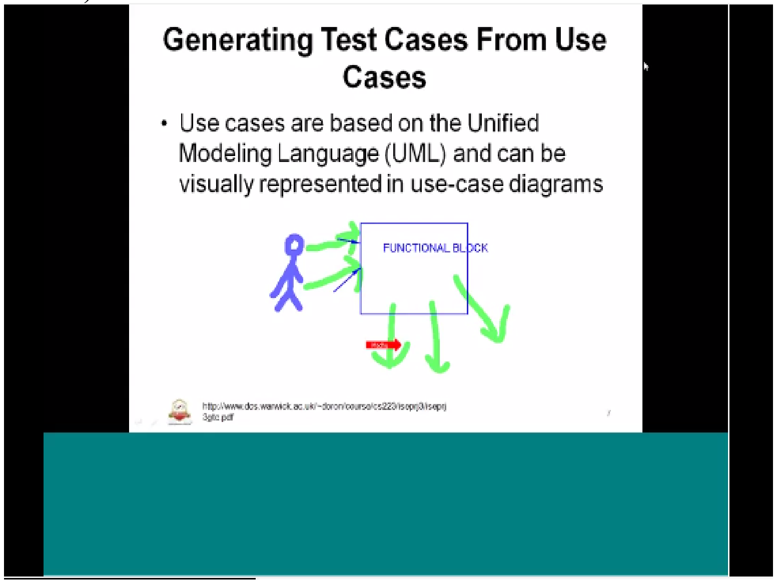
Generating Test Cases From Use Cases

- Use cases are based on the Unified Modeling Language (UML) and can be visually represented in use-case diagrams

http://www.dcs.warwick.ac.uk/~doron/course/cs223/isoprj3/isoprj3gtc.pdf

So the next one is how we are going to generate? Generating test case from use cases, use cases are based on the unified modeling language (UML), UML is used, can be visually represented in use case diagrams, I will not detail out the use case, may be an example of seeing to have an practical class, a sample UML , so the use case basically have a users, interacting with the system, suppose I will take you there, so this is a function block, and that will be have a user, user represent in a different way.

(Refer Slide Time: 28:08)



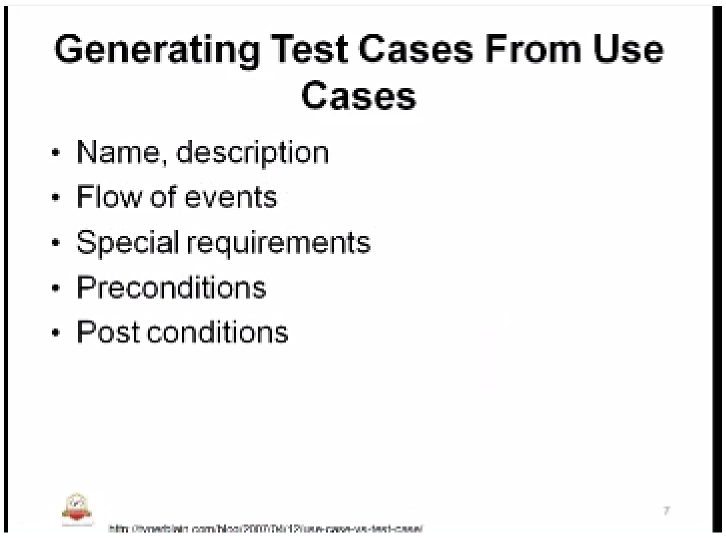
Generating Test Cases From Use Cases

- Use cases are based on the Unified Modeling Language (UML) and can be visually represented in use-case diagrams

http://www.dcs.warwick.ac.uk/~doron/course/cs223/isoprj3/isoprj3gtc.pdf

So basically it is a use case, so what it does? It will interact to the system and it expects from this system, so all this scenario like we have one or two and some results like 2 and 3,4, whatever it would be for the function block all this will be founded here, We can draw the test cases, test cases can be covered from this, likewise, so that is how we are going to generate the test cases for each of this path, were the use cases are being used with a user range of phases, with the system and expected output, how the user per assumes from that scenario okay, and the use case

(Refer Slide Time: 30:01)

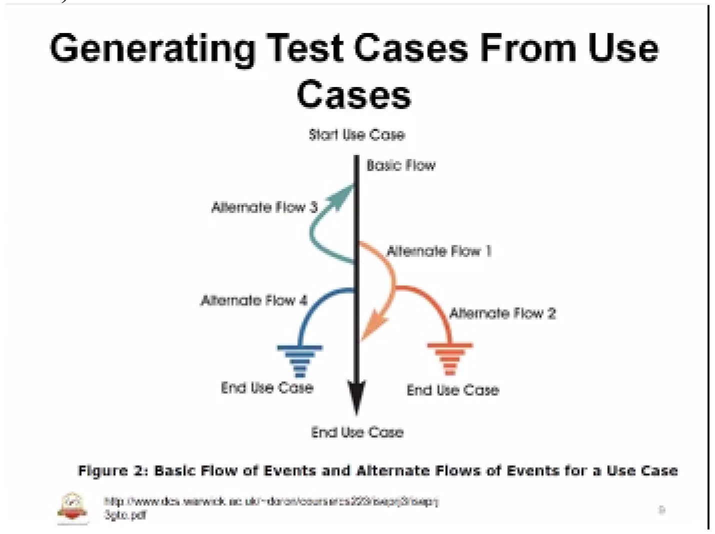


Generating Test Cases From Use Cases

- Name, description
- Flow of events
- Special requirements
- Preconditions
- Post conditions

http://tynerblain.com/blog/2007/06/12/use-case-vs-test-case/ 7

Will have the below items, we can see name and descriptions of the use case, flow of events that use case can take or different work, for that events how is going to drive? And special requirements execute to cover, because having the user per perspective, we may cover more functionality or more requirements, and specific requirements, there also addressed when any pre conditions that we study taking here, typically executing the use case and any pre conditions approving of test case have been executed, so all this aspects of use case to test case will have to be taking here.
Okay, so that is there we use the test cases generating the test case from use cases so typically what I have try to draw that users, we also called a actuators, actuators are nothing but, users, users can be elevators of the system or UML of actual users, so the lines what I have trying to draw, within the actors and the use case in terms of the test case, so use case diagrams provide , will be a picture in terms of, how the function block is used or perceives, and ascending will be tested, against that actually okay,
(Refer Slide Time: 32:37)



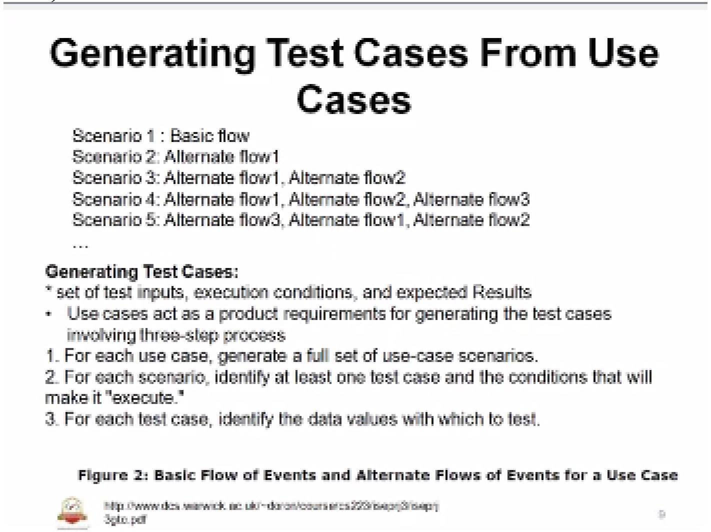
Generating Test Cases From Use Cases

Figure 2: Basic Flow of Events and Alternate Flows of Events for a Use Case

http://www.dcs.warwick.ac.uk/~daron/courses/cs223/[illegible] 3gto.pdf

The next one is example of generate with test case, we can see a different paths, or user case flow basically, so this diagram tells , a basic flow of events and alternate flows, of events for a use

case, so one could happen in, the basic flow will same, user going to use the function block, is a output, to achieve that, there could be a different flows, alternate 1, 2,3,4, different class get function, and this alternate flows can be derived in to sub alternate flows as well, so in this essay that way, we can have a multiple flows, as well.
So this multiple flows will have an each one address mail, and in different test cases, it called flow of events, and each of piece flow of events, it is called test case. And we need to apply by all these items are description all this points, and just addressing the use cases.
(Refer Slide Time: 33:52)



Generating Test Cases From Use Cases

Scenario 1 : Basic flow
Scenario 2: Alternate flow1
Scenario 3: Alternate flow1, Alternate flow2
Scenario 4: Alternate flow1, Alternate flow2, Alternate flow3
Scenario 5: Alternate flow3, Alternate flow1, Alternate flow2
...

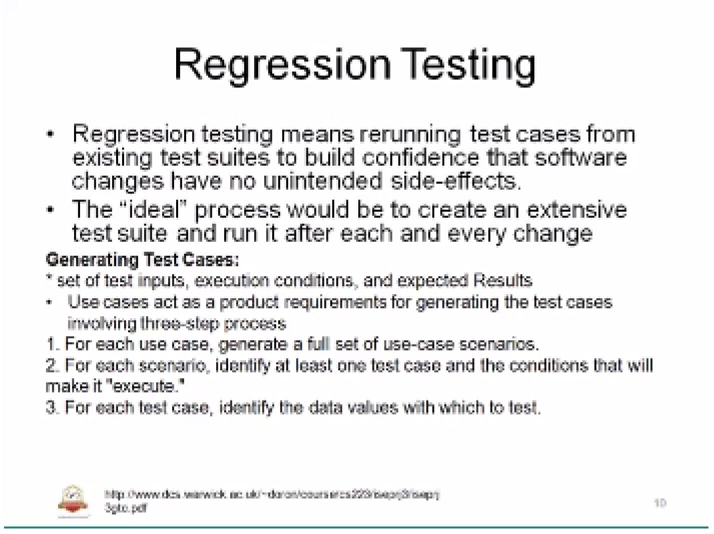
Generating Test Cases:
* set of test inputs, execution conditions, and expected Results
- Use cases act as a product requirements for generating the test cases involving three-step process
1. For each use case, generate a full set of use-case scenarios.
2. For each scenario, identify at least one test case and the conditions that will make it "execute."
3. For each test case, identify the data values with which to test.

Figure 2: Basic Flow of Events and Alternate Flows of Events for a Use Case

http://www.dcs.warwick.ac.uk/~daron/courses/cs223/bsepr/3/bsepr/3gto.pdf

We can see there are different scenarios, and scenario 1 could be a basic flow, scenario 2 could be alternate flow, and scenario 3 could be alternate flow 1, alternate flow 2, scenario 4 could be alternate 1, alternate 2 alternate 3, you can see a multiple scenario right, see it can take this path, if they flows alternate 1, alternate2, and alternate 3, it will goes back, and again come, likewise we can have multiple scenario, for each scenario 1 test case we are going to have it, so once we identify these two rows, or for in this example five flows are here, we are going to generate the test cases.
So how we are going to test? Generating test cases is set of inputs, executions, conditions, and expected results, uses cases or anything type use case act as a product requirements for generating the test cases, this way is very important function that, we have understood the requirements, and we have return the each group of scenarios based on the use cases, now this the time for write the test cases, and for test case writing will not care about the requirement now? Or a major aspect, we care about the use cases, so you can use cases something like a product requirements,
Those products requirements have to be addressed, for developing the test case, and executing them, so this is the important thing. So if you know these basically three steps process for each case generates a full set of use case scenarios, for each scenario identify at least one test case and the conditions that will make and execute, it can be more than one test case also, subject to the visibility of the particular use case if it is required and derived more test cases, so nothing have more than in one test cases, it should be invertible those test cases

That is where they going to have, as well as, practical procedures for this scenarios, for each test case identify the data values with which it has to be tested, so step 1 is to generate the scenarios, step 2 is to generate the test cases , step 3 is identify the data values, this is very important, so one thing I repeat briefly use case description, and identify each combination of alterate flows and scenarios, identify and understand the scenarios and creative scenario matrices, we can draw a table and identify matrix, it could be a peritoneal matrix or full matrix or scenario or whatever it could be, make sure that all the use cases scenarios have been addressed with a test cases, that is how we are going to draw the scenarios.
(Refer Slide Time: 40:05)



Regression Testing

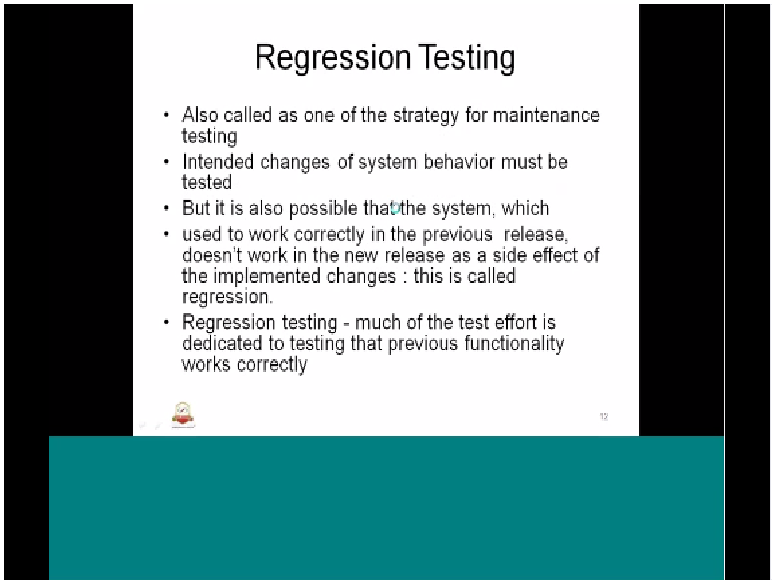
- Regression testing means rerunning test cases from existing test suites to build confidence that software changes have no unintended side-effects.
- The "ideal" process would be to create an extensive test suite and run it after each and every change

Generating Test Cases:
* set of test inputs, execution conditions, and expected Results
- Use cases act as a product requirements for generating the test cases involving three-step process

1. For each use case, generate a full set of use-case scenarios.
2. For each scenario, identify at least one test case and the conditions that will make it "execute."
3. For each test case, identify the data values with which to test.

http://www.dcs.warwick.ac.uk/~daron/courses/cs223/swproj3/swproj3gto.pdf 10

This is the first step, second step, once the full set of scenarios have identify the test cases, we can do this, by analyzing the scenarios, and use case excuse the description as well, so use case will have the description and result in one of the earlier, session have been shown in this slide, each use case should have description, and this description is used and a input for the analyzing the scenarios and generating the test cases from the use cases, so there at least one test case for each mode, probably more actual, it depends subjecting.
So the additional test cases are basically forward all the possibilities in the boundary, equivalence whatever we have study, in a reason we must to add a test cases for analyzing anything that, it could make it for all those scenarios, so that is used to be table block, this is where what with do in a test cases, identification process, the third step will be identifying the data values, for each of the step.
Because data values are very important , once all of these test cases, there should be reviewed basically understand and valued to ensure accuracy and identify, or missing test case, if anything is missing, then once all are understood viewed approved, the final structure are is to actually substitute the practical values or data values for the inputs and the expected outputs , so with this data accuracy can be implemented or executed right, so they are the description of simulation scenarios and paths, actually the test data going to identify the practicality of the testing so therefore it is to identify, actual values to used in the final test, so it can be done with a test case matrix for each of his okay.
(Refer Slide Time: 43:59)



Regression Testing

- Also called as one of the strategy for maintenance testing
- Intended changes of system behavior must be tested
- But it is also possible that the system, which
- used to work correctly in the previous release, doesn't work in the new release as a side effect of the implemented changes : this is called regression.
- Regression testing - much of the test effort is dedicated to testing that previous functionality works correctly

12

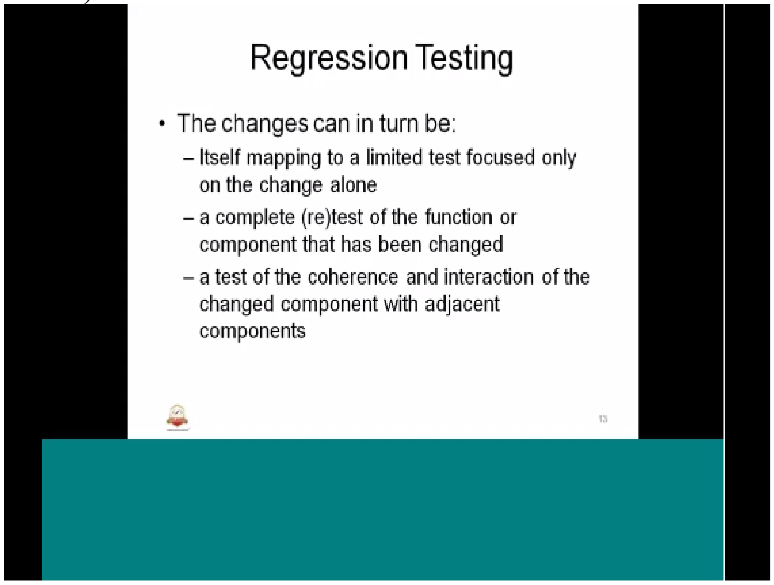
So with that the generating test case are from the use case comes to an end, so next topic regression continuation of testing, continue integration testing are it also called as regression testing, regression testing so what is mean by regression testing, it means returning test cases from existing test suites to build a confidence that software changes have no unintended side effects, so why? Need to have software changes why? Because when we have done the primary testing or returning testing, they are suddenly issues and stubs inform that founded on the embedded system of the software

To overcome that we have fixe an software and recoverment are not change, the test case should not change, so what as to change the user and to achieve that, am going to have a regression testing, so regression testing means returning the test cases, from the test suits, so basically while doing this we can get a comparison testing software as well as some changes, but does not have any unintended side effects, the idea process would be to create an extensive test suit and run it after each and every change, that means a test should be such a way that any changes in a system will not alter the execution process, so it have to re touch or re work on that test suit again and again,

Though the system is and has changes, changes means? Within the certain limit on the scope of the program, that is where regression testing is the important term , type of testing it is not type of process, ever one will follow the online list is , but some people will do the add nation of a regression, they have dedicated in for regression, they do batch, automation, so that every version of software testing it independently, that is the regression testing is very important, the primary testing has to be solid and accurate , so the regression testing will be success, based on that, so definition of re testing? Re testing based on the standards of BS 79251) it says that running it has more than once, regression testing can be repeat.

So another definition is re testing to a previous tested program, following modification and to ensure the falls have not be introduced, for uncovered whether result of the changes made, so call data's are re testing and regression testing, re testing is, testing again, regression testing is previously tested some modification have been done , of the faults are indentified and we are going to re test the same thing, so that we are going to check and those faults have been uncovered of changes have not been checked okay.

(Refer Slide Time: 46:21)



Regression Testing

- The changes can in turn be:
  - Itself mapping to a limited test focused only on the change alone
  - a complete (re)test of the function or component that has been changed
  - a test of the coherence and interaction of the changed component with adjacent components

13

Let us see in detail regression testing, why it is important? The different definition of techniques that are used for in testing, maintenance testing would have heard it is same as regression testing , whatever thing we are going to have for maintenance testing, basically this is the one, this is also called as one of the strategy for one more testing, there are other aspects also in this testing, the main aspect that regression testing, for re testing is done, intended changes of system behavior must be tested.

But it is also possible that the system which used to work correctly in the previous release, it does not work in the new release, as a side effect of the implemented change it is called regression, so we know that incremediately it is some issues, that because of it is not working, so why? Because the following issues in the next place, and that issue is fixed, so it is working fast , but we doing the testing of x, but we also testing Y, it will find and no changes on that, so all this will be brought up, that is called the regression, regression testing, much of the test effort is dedicated to testing that previously functionality works correctly.

So let see what it is, we make sure that the testing will be again, probably no issues, because another tricks that have been done for the identify the faults and errors in a test to, if you but the emphases, the regression is that, which against the work will be ended.

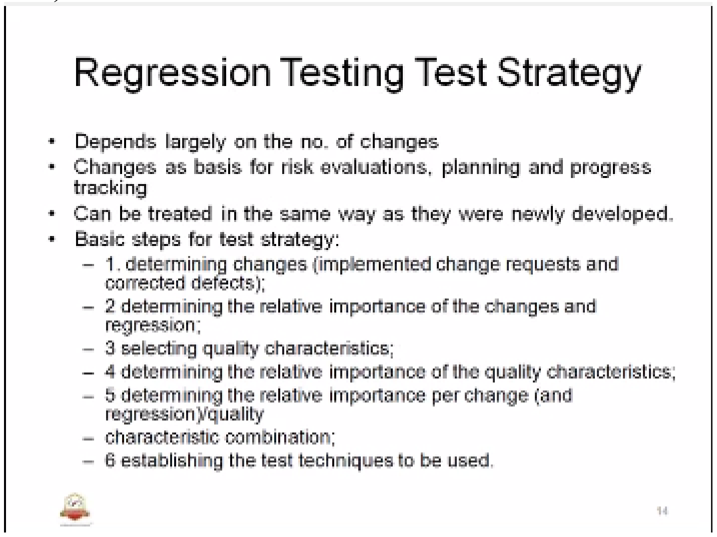
The changes can in turn be, itself mapping to a limited test focused only on the change alone, or a complete test of the function or component that has been changed , it takes a test of the coherence and interaction of the changed component with adjacent components form, so in regression testing, changes in testing takes place on the direct point, an implemented test focus on the change form, this use changes going to adjust , we will see the interrupt, so it is not be working, but that cannot be done, it was the changes can be impacting the sever in models

So the second type of changes can be complete test of the function, the entire function has some small stubs, it does not mean the impact of path or the interact body with function, or the component.

Or it is better to test the entire contd. That is what it means, the third one is pointing the testing of particular component, we need to check the OS and interaction of other components, the other modules, which having the interaction, will be crime one, which is the test, so that is where the regression testing the very much important, so with every implementation of change we know that, the risk of regression will be introduced, the regression testing is the standard elements in the entire response, usually the set of test cases is maintained for this purpose, generally they doing sample of the maintained, depending on the risks and test budget will available

The choice has to be either or execute full regression test it, for make a selection of the most relevant test cases, test issues are used effectively, to support the execution of the regression test, the regression test will be larger, selecting the regression test, here has to be skip, because the regression test can be executed with the limited effort, that is why? We need to skip okay,

(Refer Slide Time: 49:02)



Regression Testing Test Strategy

- Depends largely on the no. of changes
- Changes as basis for risk evaluations, planning and progress tracking
- Can be treated in the same way as they were newly developed.
- Basic steps for test strategy:
  - 1. determining changes (implemented change requests and corrected defects);
  - 2 determining the relative importance of the changes and regression;
  - 3 selecting quality characteristics;
  - 4 determining the relative importance of the quality characteristics;
  - 5 determining the relative importance per change (and regression)/quality
  - characteristic combination;
  - 6 establishing the test techniques to be used.

14

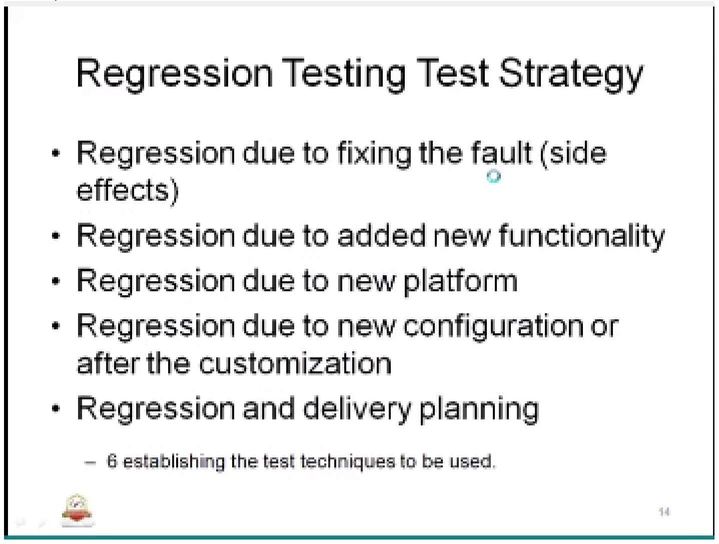
Regression testing test strategy, so what are the test have, it is basically depending on the number of changes, so definitely regression testing can be kind of subjective, how much is there, and what is that, kind of change that is also important, not a number of changes 10 numbers ,20 numbers does not match of that the one change can also replay huge a risk or the regression, so that is very important changes can get it for risk valuation, planning and progress tracking, so the changes that have been there,, on the embedded software using suffixes has to be a risk value, planning and progress tracking also, to be taking here, on the regression testing,

So it can be treated in a same way as they work with, means some of the personality is to be adjust for the regression testing such a way that the functionality doing developed, and it test it approach, so test strategy would be determining the changes, first portion, implemented change request and corrected defects, so what is mean by implemented, what is mean by change? That we need to determine.

The first, second is determining the relative importance of the changes and regression, that means it is to identify important of changes and regression importance, heard on this correcting quality have a characteristics, what are the quality aspects that need to be take care, all this have to be take care, both of determine the relative importance of the quality characteristics, we know that enough to identify the, or select the characteristics in a particular scenario it is also important to understand the rating of the quality aspects

The next one is the determine the relative importance for change, and regression or quality, so each changes, what is the relative important of the entire system, so that particular change, of the particulars on the characteristics, all this above steps need to be combine together to complete the conclusion, that is called characteristics combination, the risks are been establishing on the test techniques to be used, okay, so then we going to establish the test techniques for that identify changes that is what the steps are involved for regression testing.

(Refer Slide Time: 52:20)



Regression Testing Test Strategy

- Regression due to fixing the fault (side effects)
- Regression due to added new functionality
- Regression due to new platform
- Regression due to new configuration or after the customization
- Regression and delivery planning
  - 6 establishing the test techniques to be used.

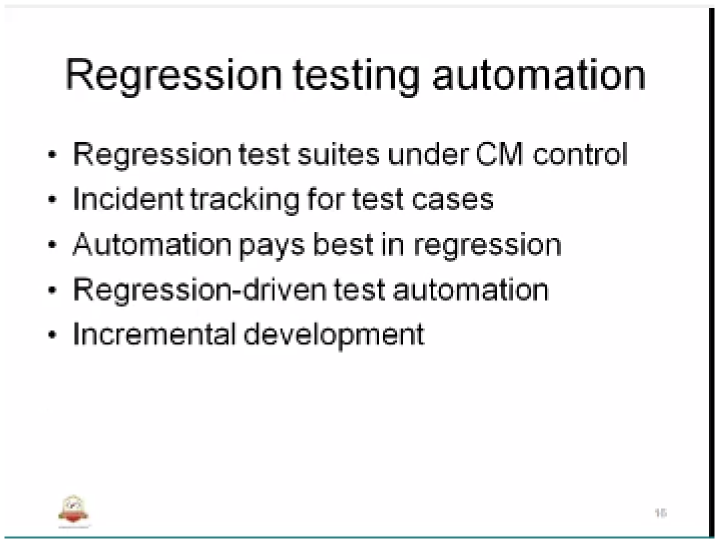
14

So what are the areas in regression testing, it need to be taken or take care, regression is due to fixing the fault side effects, we need top check, we know that, we found the problem, in terms of fault, the error and that is fix and we are going to release the regression test on this fifth part, so we need to check the side effects of the sixth part, regression due to have the functionality, new types of software has come, so that does not it mean that we need to have entire testing in these paths.

So what we doing to this regression, the regression testing in terms of a particular added functionality, or the new functionality, regression due to the new platform, that is used reserves of the same testing, will be carried out on a new plat form this is basically important where the voting activities, so especially OS is representing, operating system supports to work same on different platforms.

To be internal could be , it could be or any other plat form Linux you take for an example the Linux OS suppose to work on different hardware platforms, so all are going to be is do with a regression test strategy, regression can be due to give new configuration, or also the customization, some changes some configuration there is no software changes but the environment of configuration the program, of this software and the test as change and regression will be required for the same, the regression and deliver and , so those are one of the area, that interview taken here, configuration testing, so how we are going to deliver , so they are what incremental or non incremental for the regression testing areas okay,

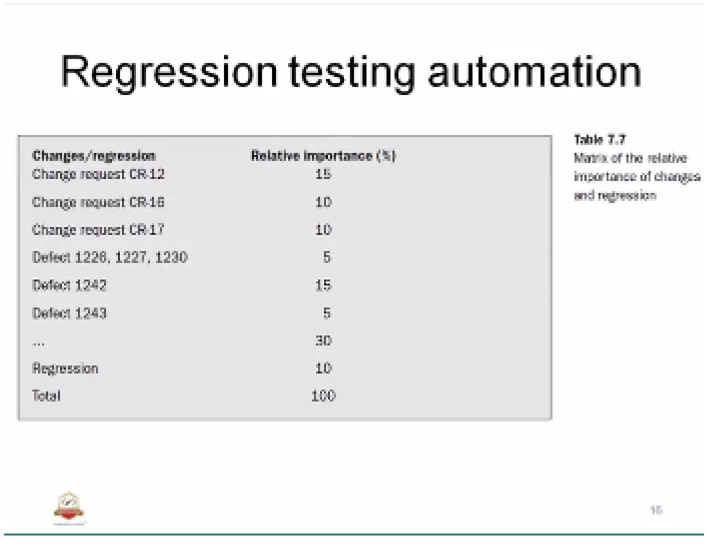
(Refer Slide Time: 54:42)



Regression testing automation

- Regression test suites under CM control
- Incident tracking for test cases
- Automation pays best in regression
- Regression-driven test automation
- Incremental development

The next part is regression testing automation, is very important to have automation as we progress the different steps, so any system automotive, recall or auto space , does not stop at one level testing, the product will be going to the involving again and again, requirements is going to update the module change, the faults are going to be receive fixed again and again, with a different scenarios of software , software's difference whatever platform units, definitely is a high time for identifying the automation.
Why planning the testing, especially it is useful for testing where manual internal is very minimum and not effort to have actually, so that is where the automation is very important and the test suits which are going to develop, for automation of should be control the configuration, so will study more about the configuration elements test management in the next class.
Incident tracking of test cases where the different incident is happening? while developing the test cases in terms of a test case of course strategy, all this can be derived, automation is based on this integration, basically automation is very much best in regression test, because again and again people cannot effort to the test cases, modifying all those that, and execution especially, so you can just use this script what we used earlier, modified according to the needs, whatever test case we needs will be execution is possible okay.
Regression driven test automation, regression can drive, test automation as restricted, you can use the batch file execution, multiple scripts , I have seen some batch execution , where thousands of test cases has been taken, this is the time for expression and all that, we used to keep the batches for overnight or next day , again keep forward base like that, so this kind of automation are very important in regression testing, the human interval is very less, and automation also used for implemented , where the development is goes to the each world again, okay.
(Refer Slide Time: 57:37)



Regression testing automation

Changes/regression	Relative importance (%)
Change request CR-12	15
Change request CR-16	10
Change request CR-17	10
Defect 1226, 1227, 1230	5
Defect 1242	15
Defect 1243	5
...	30
Regression	10
Total	100

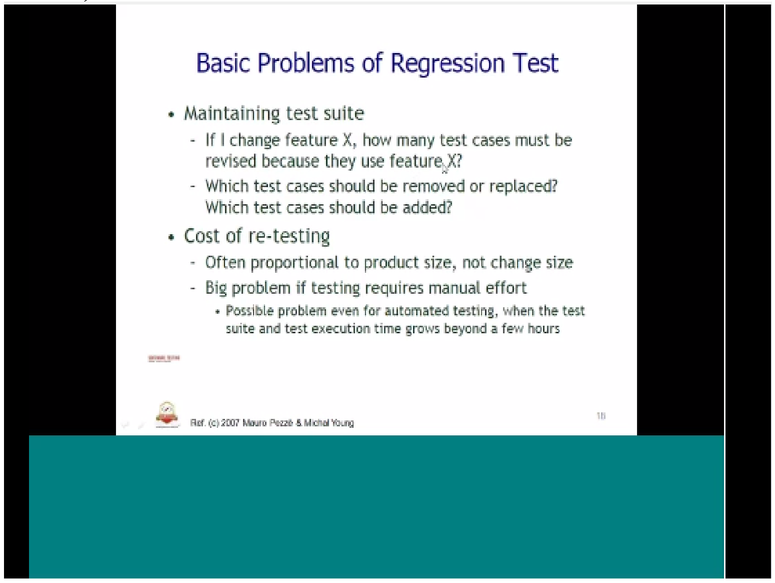
Table 7.7
Matrix of the relative importance of changes and regression

15

The next one regression testing strategy, that is very important of changes in regression is basically, table analyze the based on, how the different servers are important for testing test strategy, regression testing here they say as 10 percent recorded to be each 25 percent, but the previous example is given , based on a particular aspects, we can see what all the changes of implementation and the importance is represented, so all together the percent is what test strategy they were to operate it.

The importance changes could be there , in percent change the request 17, each 10 percent, defects are there , are very important , and very importance 5 percent, defects to 1242 is 15 percent of important, defect the 1243, 5 percent is important, these important for subjecting based on the type of defects or changes that are intended, the defects also can replace the request, defects are also called as problem reports, so likewise there are different types of categories changes and regression aspects, so rest of them they are keeping as 40 percent, during the regression result.

(Refer Slide Time: 59:39)



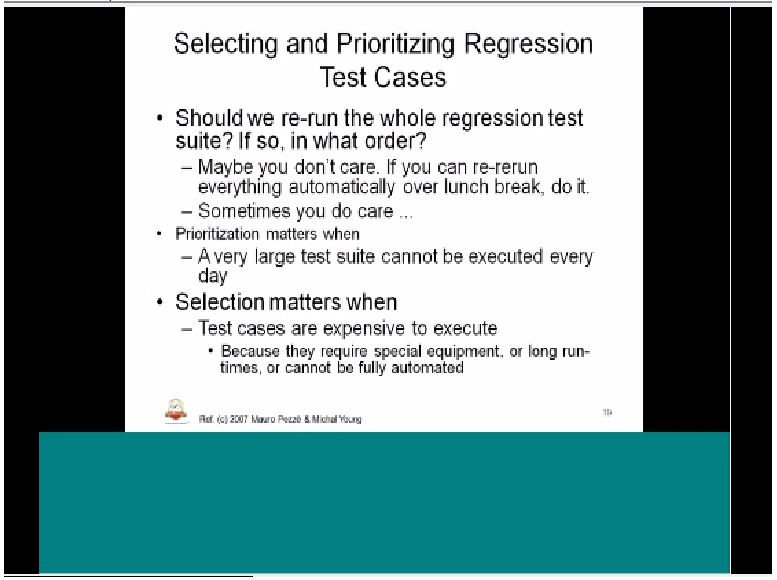
Basic Problems of Regression Test

- Maintaining test suite
  - If I change feature X, how many test cases must be revised because they use feature X?
  - Which test cases should be removed or replaced? Which test cases should be added?
- Cost of re-testing
  - Often proportional to product size, not change size
  - Big problem if testing requires manual effort
    - Possible problem even for automated testing, when the test suite and test execution time grows beyond a few hours

Ref. (c) 2007 Mauro Pezzè & Michal Young

16

Basic problems are regression test maintaining test suite, we need to have a test suit of which has to be maintained so that regression test can goes smooth, if I change feature x how many test cases must be revised, because they use feature x, so some features are common, they common template or a they can play certain part of aware, the module that has been used for multiple test cases, and all that need to revised.
Because change in the feature x, so that is what it mean the maintenance of issue for the regression test, which test case should be removed or replaced, which test cases should be added, so this is also very important criteria for maintenance the test suits, test suits are used to a huge for regression testing, then.
Cost of re testing, how much is going to cost, often proportional to product size, not the change in size, basically product says 1000, if the changes is one or two is not going to much cost mean perhaps the product size is 10, or 20 and change size 1 or two definitely the ratio is more and the cost is very high, so this is also manages, the problem is testing requires manual effort, suppose we do not have an automation.
For the some variable testing or manual analysis etc. those kind of regression is very tough, going in terms of manual effort, the idea behind is the original testing , the same idea has to be added out and people or the resources are used, as to be re produced and the philosophy is going to be manual, because that type of testing where totally manual and it has to be repeated, possible problem even for automation testing, when the test suits and test execution time grows behind the a few hours, so those are the problems for regression testing okay.
(Refer Slide Time: 1:02:02)

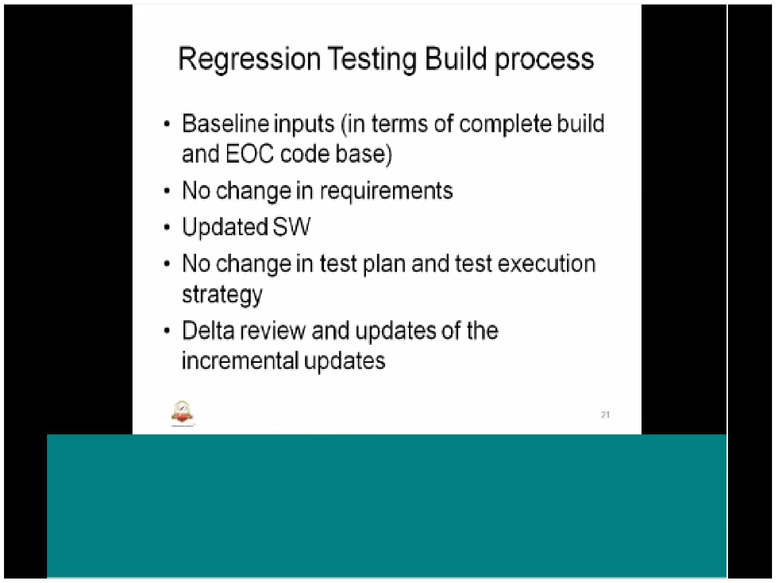


Selecting and Prioritizing Regression Test Cases

- Should we re-run the whole regression test suite? If so, in what order?
  - Maybe you don't care. If you can re-rerun everything automatically over lunch break, do it.
  - Sometimes you do care ...
- Prioritization matters when
  - A very large test suite cannot be executed every day
- Selection matters when
  - Test cases are expensive to execute
    - Because they require special equipment, or long run-times, or cannot be fully automated

Ref. (c) 2007 Mauro Pezzè & Michal Young 19

So let us study more on selecting and prioritizing regression test cases, test case maintains.
(Refer Slide Time: 1:02;07)



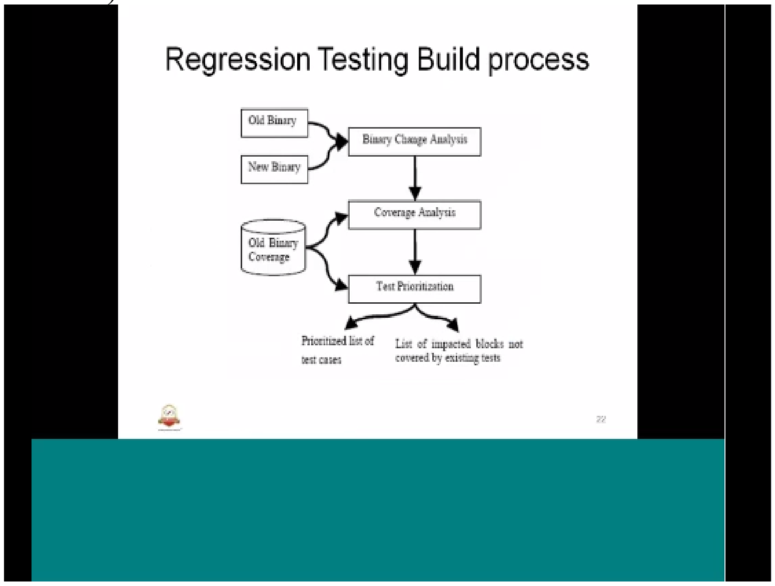
Regression Testing Build process

- Baseline inputs (in terms of complete build and EOC code base)
- No change in requirements
- Updated SW
- No change in test plan and test execution strategy
- Delta review and updates of the incremental updates

21

How are going to build process for regression test cases, you have to see a example of , how the regression test.

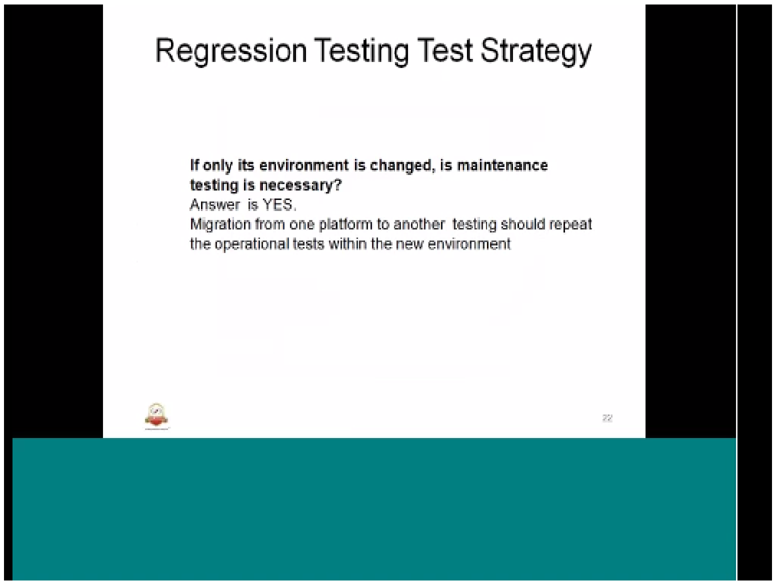
(Refer Slide Time: 1:02:11)



Regression Testing Build process

Is being process in the next class.

(Refer Slide Time: 1:02:13)



Regression Testing Test Strategy

If only its environment is changed, is maintenance testing is necessary?
Answer is YES.
Migration from one platform to another testing should repeat the operational tests within the new environment

22

So we will conclude on the regression testing in next class, we conclude the unit 4, the intersession test and regression test thank you.