

Principles of Engineering System Design
Dr. T. Asokan
Department of Engineering Design
Indian Institute of Technology, Madras

Lecture – 08

Originating Requirements: Examples Systems Engineering software – CORE

Hello friends. Welcome back to one more session on engineering system design. In the last few classes, we discussed about the first phase of a system development; that is, defining the system level design problem. And in this stage, we basically look at the requirements of the system from the customized point of view as well as from the system a design point of view, and identify all these requirements and prepare a originating requirements document; which is the output of the first level of the identifying the system level design problem.

We discussed about how do we do these how do we get the requirements, and then how do we document the requirements; what are the procedures to be followed in document this requirements. And we shown one case study example of system level design problem we took the example of an elevator, and then explained about the how do we actually develop the system level design problem and then get the originating requirements document. In order to emphasize the importance of requirements I will go through one more simple example I am not be going through the complete details. But I will just explain how do we approach the problem and then develop the requirements document.

(Refer Slide Time: 01:30)



So, here we mentioned about the failure of a air bag system, one of the classes we mentioned about this was mainly because of the a failure in identifying the actual requirements of the system. So, we look at this case study and see what are the basic requirements wrongly identified or why the system failed in identifying the or why the system developers, failed in identifying the requirements of this particular air bag system and how it failed.

(Refer Slide Time: 02:03)

Air bags, safety device appearing in automobiles in the early 1990's, became the cause of death for a noticeable number of individuals.

There were severe flaws in the design, testing and deployment requirements envisaged.


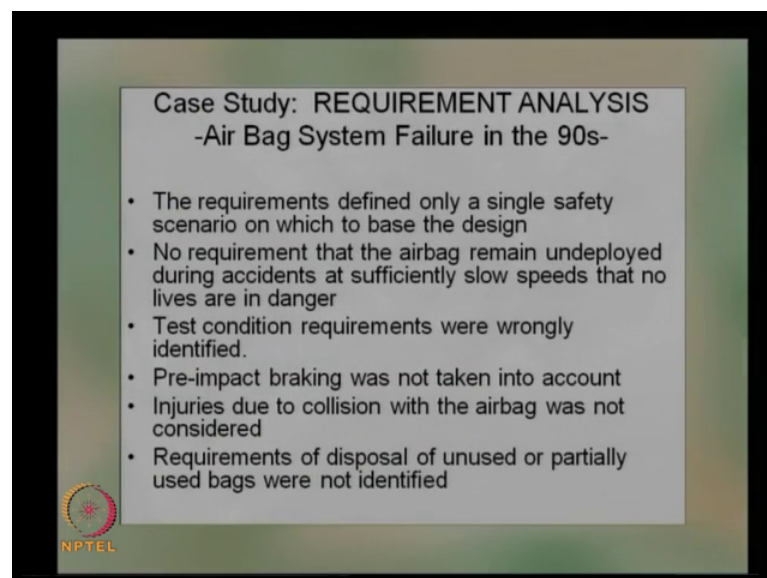


Photo Courtesy of Autoliv

If you look at this case, you will see that that safety devices which somewhere in early 90s it became the cause of death for a notable number of individuals, and there were severe flaws in the design testing and deployment of a deployment requirements envisaged during the design.

So, not only the a design requirements, but the testing requirements as well as the deployment requirements were wrongly identified, and that was the reason why this failed in the initial stages of the development of air bag system. So, whatever the basic requirements which actually failed in this case a listed here.

(Refer Slide Time: 02:36)



As you can see the requirements defined only a singled safety scenario on which to base the design. So, the multiple scenarios as we discusses earlier there were multiple scenarios of operation and a use of the product, but in this case of design only a single scenario was used. Basically it was designed for a mail driver driving at a speed of 30 kilometer 30 miles per hour, and diving a crash head on crash and that scenario only was considered while developing the system requirements. So, that was one of the faults in identification of the design requirement where the only singled safety scenario was considered. And then the second one no requirement that the airbag remain undeployed during accidents at sufficiently slow speeds that no lives are in danger.

So, this also was not envisaged. So, it was assumed that any case where there is an accident the a airbag should deploy. But in most of the cases when the speed was

sufficiently low, there was no need for the air bag deploy, but the since that was not envisaged and then it actually led to failure of the airbag system. Third one the test condition requirements were wrongly identified. Again the under which the and what are the conditions under which the air bag to be tested for safety was again wrongly identified again. Here a dummy driver with a weight of around 70 to 80 kg, and sitting in an applied position, with the hands on a 9-o clock and 3 o clock positions, running at a speed of 30 miles per hour, having a head on collision where the forces are always straight on the front of the car.

So, only this scenario was considered for testing. And most of the cases this was not the condition, because the head on it is not always head on collision there may be forces act in non-parallel to the vehicle as well as the driver may not be in his applied position, and the scenario of head on collision when this a driver is in applied position was actually very rear. So, all those conditions were not considered. And only a one single test condition was a tested, and system pause the test and it was allowed to use in vehicles. So, that was again a failure because the requirements of test were not a properly identified. Then again pre-impact braking was not taken into account.

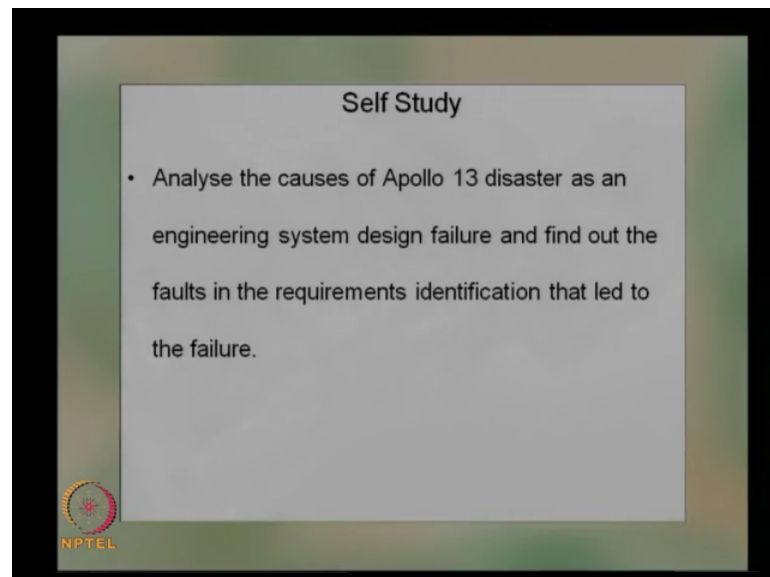
So, normally before the accident always brakes will be applied by the a driver, and that actually reduces the speed of the vehicle. And because of this speed reduction the driver will be always moving forward and hitting the steering assembly, and that was not actually envisaged in the a test condition. It was always assume that the airbag should deploy only after there is a collision. So, this pre-braking condition actually required the airbags to deploy, well before the accident or the time delay in deploying the airbag given to be decided based on this pre-impact braking. That was not considered when designing the airbag.

Similarly, injured is due to collision with the airbag was not considered. So, when the air bag is deployed and the driver is moving forward because of the impact, there was a possibility for the a driver to get injured because of the airbag. The sufficient elasticity of the a airbag over the pressure to be maintained, all those things were not considered when air bag was developed. And another one was the requirements of disposal of unused or partially used bags were not identified. Though this was not a reason for a failure of airbag, this was also not taken into account and actually the deployment or the

disposal of the system was not taken consider into account and the requirement for disposal was also not considered.

So, all these actually shows that when we design a system it is important to look at all the aspects of the system, and then identify the requirements. So, here you can see that the test conditions, a pre-braking a impact, and then the disposal, all those things were not given primary importance and that all those actually led to the failure of airbag system. This shows that the importance of requirement analysis. We will go through one more example how do we actually do the requirement analysis and prepare the case.

(Refer Slide Time: 07:08)

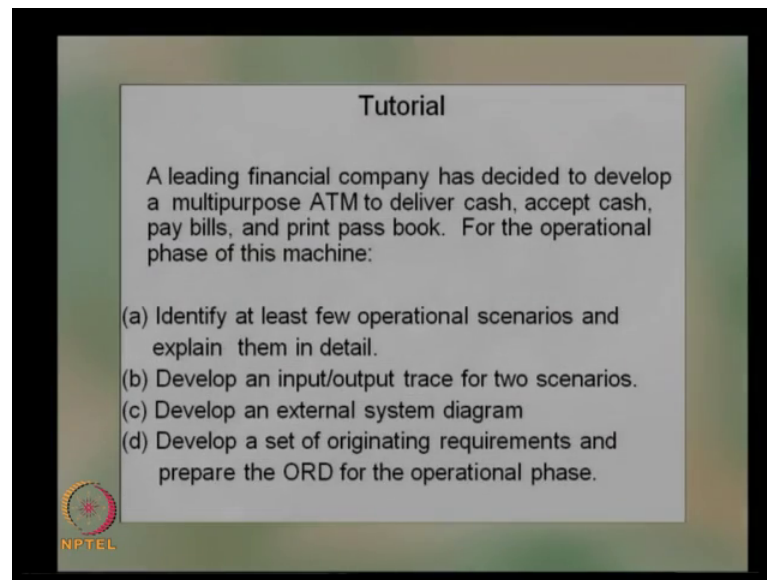


But in this case, I will leave this case to the as a self-study for you.

So, you can actually look at the causes of apollo 13 disaster as an engineering system design failure, and find out the faults in the requirements identification that led to the failure. Again, you can actually see lot of case studies in the of engineering disasters. You can look at this particular case study and then see what actually went wrong in apollo 13, what are the especially from the requirement analysis point of view, there may be either faults also, but look at the requirements and analysis identification of requirements, and what actually went wrong in the failure of apollo 13.

So, this is actually you can take it as a self-study, and identify there a faults in requirement analysis.


(Refer Slide Time: 07:54)



Tutorial

A leading financial company has decided to develop a multipurpose ATM to deliver cash, accept cash, pay bills, and print pass book. For the operational phase of this machine:

- (a) Identify at least few operational scenarios and explain them in detail.
- (b) Develop an input/output trace for two scenarios.
- (c) Develop an external system diagram
- (d) Develop a set of originating requirements and prepare the ORD for the operational phase.

 NPTEL

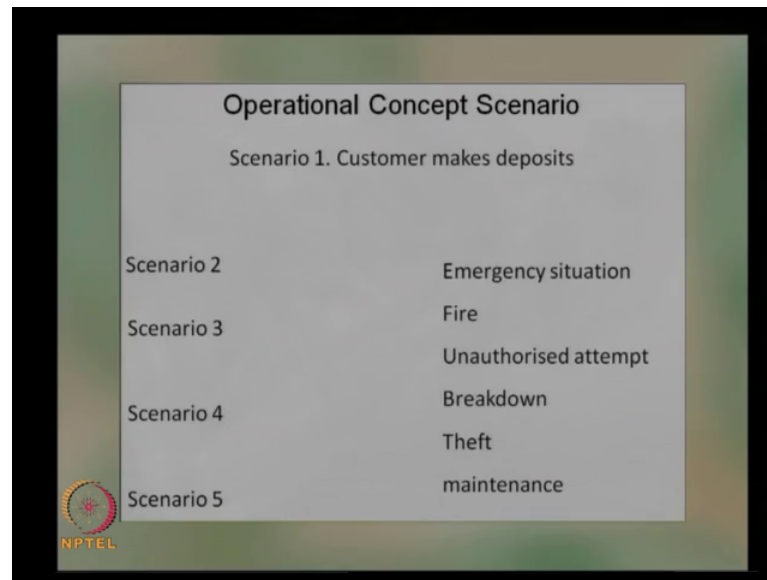
I would like to give you another tutorial, basically you can take this as a tutorial for your even this course, and try to attempt this problem, and identify all those what are the list at the here basically, I am looking at a product. Basically, an ATM machine, a leading financial company as decided to develop a multipurpose ATM to deliver cash, accept cash, pay a bills and print pass book, for the operational phase of this machine. So, it is only for the operational phase what you need to address in this tutorial. Identify at least few operational scenarios and explain them in detail.

So, as I told you about how do you identify the operation scenarios in the case studies and examples, we actually show how to do this. So, for this ATM machine try to identify a few operational scenarios and explain them in detail. So, we know how to describe an operational scenario. So, explain them in detail. And develop an input output trace for 2 scenarios. So, at least for 2 scenarios you can a develop an input output trace. And develop an external system diagram for this or identify all the external systems which actually interact with the main ATM system, and then prepare an external system diagram. And develop a set of originating requirements, and prepare the originating requirement document for the operational phase.

So, you can identify the originating requirements, and as per the format explained earlier you can prepare the originating requirement document. So, this is the work you need to do as part of the tutorial. I will be giving you some hints on how to do it, but I suggest

you that you do not go through the next few slides, try to solve this yourself, then you can check with the slides coming after this, to see whether you are in the you are doing it the in the right way.

(Refer Slide Time: 09:52)

A slide titled "Operational Concept Scenario" with a list of five scenarios and their corresponding descriptions. The scenarios are: Scenario 1 (Customer makes deposits), Scenario 2 (Emergency situation), Scenario 3 (Fire), Scenario 4 (Unauthorised attempt), and Scenario 5 (Breakdown, Theft, maintenance). An NPTEL logo is visible in the bottom left corner of the slide.

Operational Concept Scenario	
Scenario 1.	Customer makes deposits
Scenario 2	Emergency situation
Scenario 3	Fire
Scenario 4	Unauthorised attempt
Scenario 5	Breakdown Theft maintenance

So, to give you some hints the operational concept scenarios. So, you need to identify the operational concept scenarios. So, here you can identify many scenarios. For example, one is the customer making a deposit. So, a customer is coming to the ATM machine, he wants to deposit some cash.

So, what all the different activities, or different procedures you will be following in order to make the deposits. So, similarly you can identify other scenarios also, can have scenario 2 3 4 5. Or any number of scenarios. So, some examples are 4 scenarios are like and there is an emergency situation. So, it could be a fire in the ATM room, or there could be a theft or it can be an electrical short circuit, or whatever it is. So, what will happen in their emergency situation?

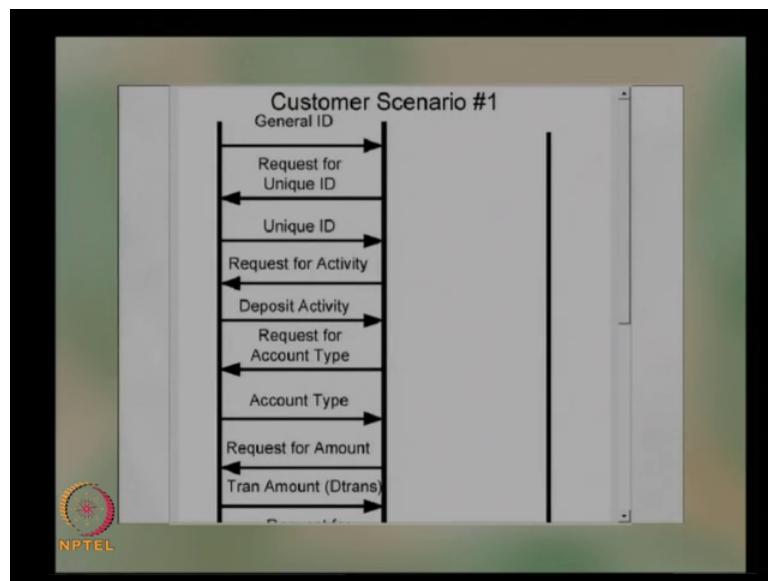
So, what are the requirements to be provided in the ATM machine or the external system which actually interact with the ATM machine in order to face this emergency situation? Or there is a fire in the system? Or there is an unauthorized attempt by someone to take cash or to loot the ATM. So, what should be the scenario? What actually the ATM machine should do or what actually the software or the external systems? Or how do they react to a such a situation? So, that also can be another scenario, or the machine as bog

down. So, what should be the how that scenario to be handled to the machines bog down how to inform the a central server or the service agents or the customer.

So, what will happened to the customer if it is in between and it breaks down? Or what will happen if there is a breakdown over the system when the cash transaction is going on? So, all those scenarios can be identified here. Similarly, there is a theft in the or an attempt to a theft and then maintenance scenario. So, the machine is under maintenance. So, what are the system is to shut down? And what are the system it should allow for service people to access? So, all those scenarios can be a explained in detail and you can actually identify all the requirements for these operation scenarios.

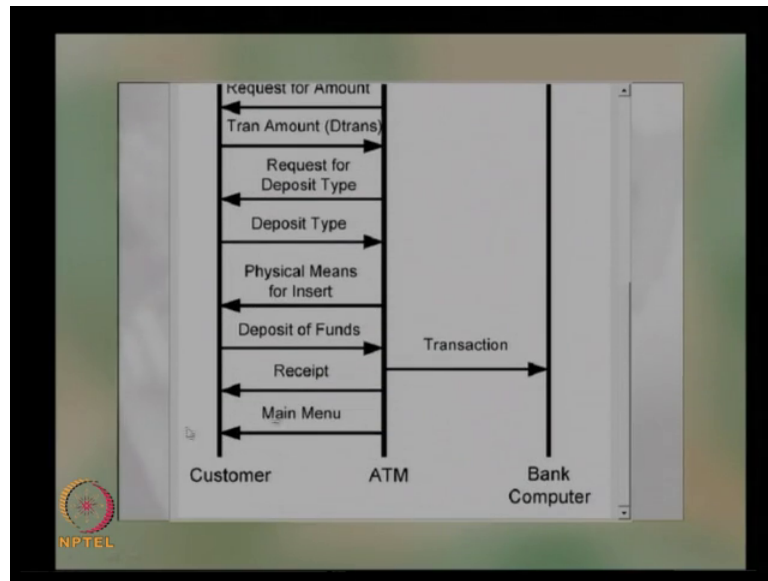
And to get the input output requirements you can actually go for an input output trace.

(Refer Slide Time: 11:56)



As you can see here, this is a customer scenario one. So, when where you actually the customer wants to make some deposits, you can see here there is a customer this customer you can see over here.

(Refer Slide Time: 12:10)



So, this is the customer, this is the ATM, and this is the bank computer or the central server. So, what kind of interactions takes place between these entities? You can see that customer is an external system over here. And then ATM is the main system of interest and bank server is again a external system which actually interacts with the ATM.

So, we can see here the customer will provide a general identification to the ATM, and then these in the form of a card an ATM card or it can be a thumb impression or whatever form the ATM you would like to have. And then the ATM will ask for a unique id in terms of a password or form of identification and the customer will provide the id, and then which is satisfied. If there is a database it will verify it with respect to it is database, otherwise there will be another line coming from here to the server, which will actually verify the id and then give a feedback. So, in this case it is assume that the ATM itself has got the ids or the passwords stored for a particular general id.

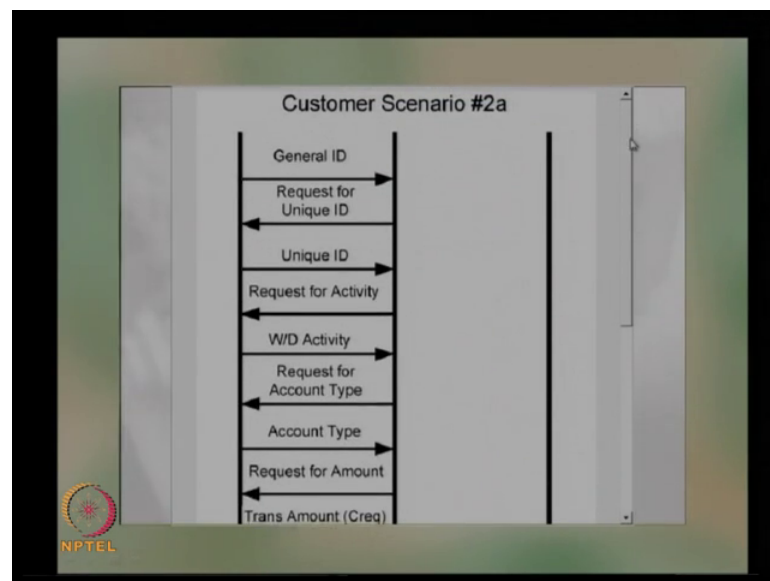
So, in that case it will ask for the activity what kind of a transaction you would like to have, and then the customer selects the deposit activity, and then the request for type a account type is given by the ATM. Then the account type is provided by the customer, and the amount is asked how much amount you want to deposit, the transaction amount is given by the customer. And then deposit type, and the deposit type is given. As in terms of cash or check or a DD whatever the type of deposit and then there is a physical means for insert. So, we can see here that the ATM machine provides a physical means

for insert. So, this kind of a transaction or the input output trace tends you the requirements.

When you say there is a physical means of insert; that means, the machine should have the facility or there is a requirement for a physical opening in the machine to accept the cash or check, and then the customer deposit the funds. And then there should be transaction information to the central server. And that will count the amount or verify the cash. And that will be transaction will be given to the central server. And then the receipt will be provided by the ATM, and the ATM will go back to it is main menu. And the customer can choose to have another transaction, or he can leave the place. So, these are the activities taking place in one of the scenarios.

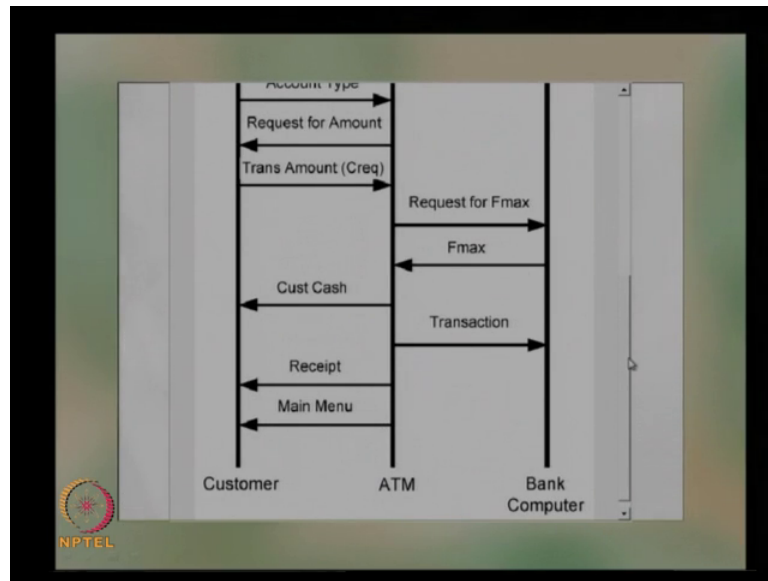
So, like this we can have all for the input output trace we can have for all most all the scenarios and provide find out the input output requirements.

(Refer Slide Time: 14:51)



So, this is another input output requirement for another scenario. They basically here is a withdrawal activity; the customer wants to withdrawal some amount. So, the initial activities will be the same. So, once you tell the request for a activity the customer chooses the withdrawal activity, and then request for account type then account type is given then request for amount transaction amount, and then once you give the amount.

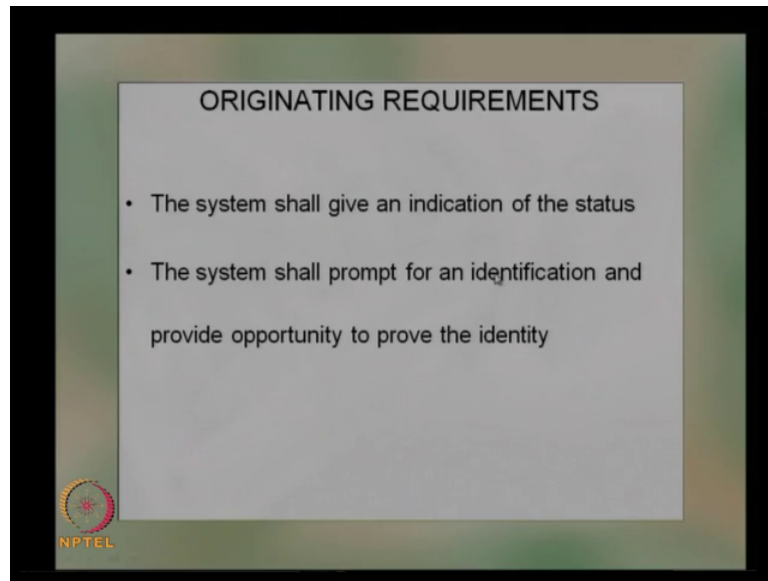
(Refer Slide Time: 15:19)



Then there will be a verification what is the maximum amount the person can withdraw, and that amount will be given back to ATM and then ATM will decide whether to give the cash or not. And once it is accepted that will be providing the cash.

So, the customer is given a cash a given the cash from the ATM. Again, it tells you that there will be a provision for the ATM to count the currency and then dispose or give the currency to the customer. And the receipt you provided the transaction details are sent back to the central server, and the receipt is provided and then it will go backs to main menu. So, these are the different activities taking place in during this particular transaction. Similarly, all the scenarios we can identify the input and output requirements, and this requirements can be placed as the in the pass part of the originating requirements.

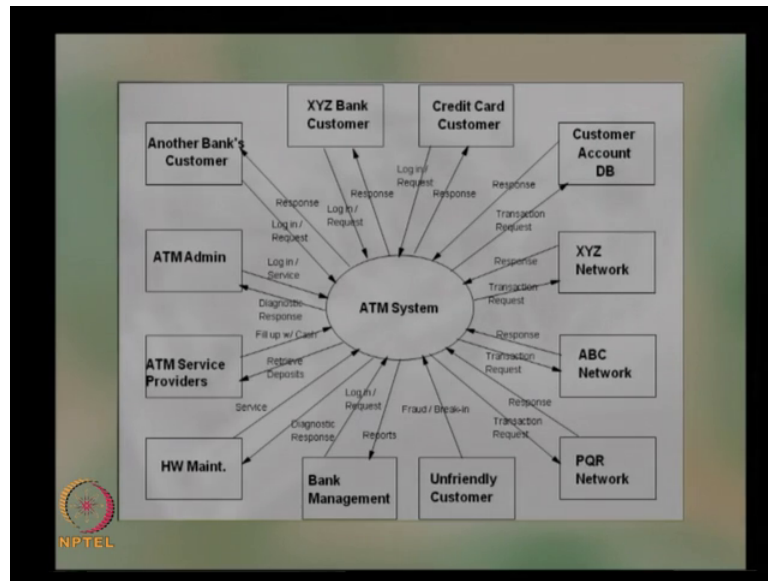
(Refer Slide Time: 16:13)



So, you originating requirements some of the originating requirements what we can identify the system shall give an indication of the status. I mean the status of the machine. So, what is where it is in operating condition, where it is in maintenance condition or it is in a breakdown condition. So, this indication of the status to be provided to the customer; the system shall prompt for an identification and provide opportunity to provide the identity. So, since the machine has check the identity. So, there should be a prompt for an identification, and provide an opportunity to prove the identity. So, these are some of the requirements you can identify.

So, similarly you can have many requirements. These are just examples of some of the requirements which you can identify using the input output trace.

(Refer Slide Time: 16:55)

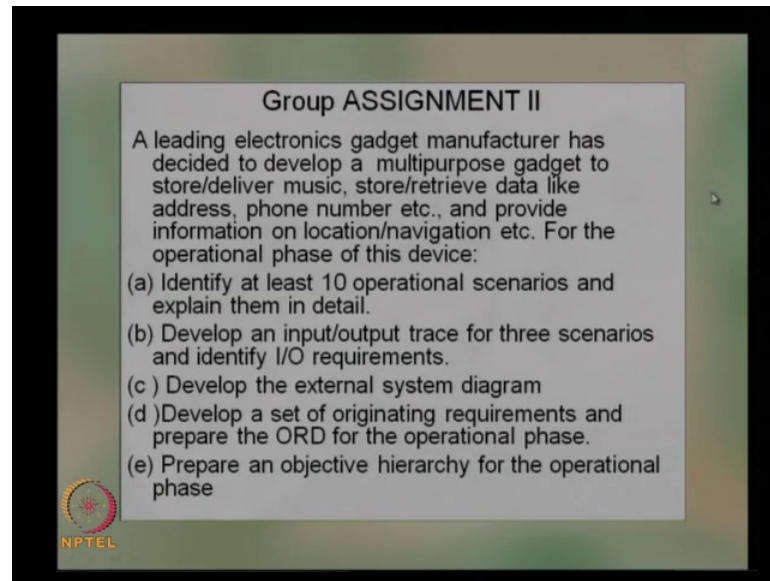


So, these actually shows a very top level external system diagram, can you see these are all the external system which will be interacting with the ATM system you can have ATM admin which will be interacting with the ATM system for the maintenance. As well as a service and other requirements, and the customers from other banks will be coming and there will be different banks customer like then will be a credit card customers. And then there will be different networks. Like, there are different international networks like master visa and that kind of a networks.

So, there will be interacting with the ATM, then there will be a unfriendly customer basically who are trying to either attempt to take money unauthorized of without any authority. Or there will be a some theft and other situations. And there will be bag management interacting with the system for basically to find out the transactions and to see whether there is enough cash or other things.

So, there will be hardware maintenance also which actually will be maintaining a system, and whenever there is a problem there will be attending as along with the service providers. So, these are the different systems interacting with the ATM. So, there are the basically the external systems. And you can actually go further down to this level, and identify what kind of interaction takes place. So, we discussed about the external system diagram. So, we can prepare the more detailed external system diagram identifying the type of interaction taking place between these entities.


(Refer Slide Time: 18:25)



Group ASSIGNMENT II

A leading electronics gadget manufacturer has decided to develop a multipurpose gadget to store/deliver music, store/retrieve data like address, phone number etc., and provide information on location/navigation etc. For the operational phase of this device:

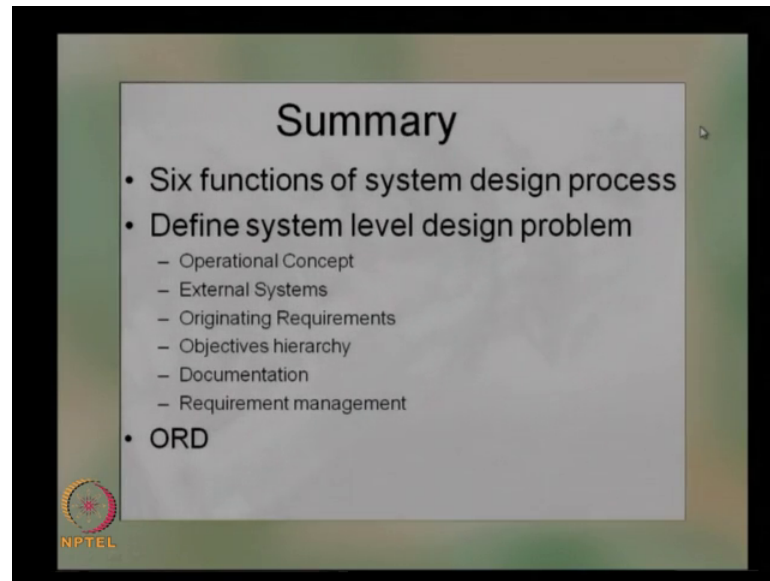
- (a) Identify at least 10 operational scenarios and explain them in detail.
- (b) Develop an input/output trace for three scenarios and identify I/O requirements.
- (c) Develop the external system diagram
- (d) Develop a set of originating requirements and prepare the ORD for the operational phase.
- (e) Prepare an objective hierarchy for the operational phase

 NPTEL

So, that was just to tell you how to attempt the problem of the tutorial problem. And this will be an assignment for you. You can attempt this leading electronics gadget manufacturer has decided to develop a multipurpose gadget to store deliver music store retrieve data like address phone number etcetera. And provide information on location navigation. And for the operational phase of this device identify at least 10 operational scenarios and explain them in detail. Develop an input output trace for 3 scenarios and identify input output requirements. Develop the external system diagram for these develop a set of originating requirements, and prepare the ORD for the operational phase. And prepare an objective hierarchy for the operational phase also.

So, what are the basic objectives? And what is the hierarchy of these objectives, also can be prepared. So, this is a group assignment for you can prepare this and you can send it to me if you want any feedback from this.

(Refer Slide Time: 19:23)

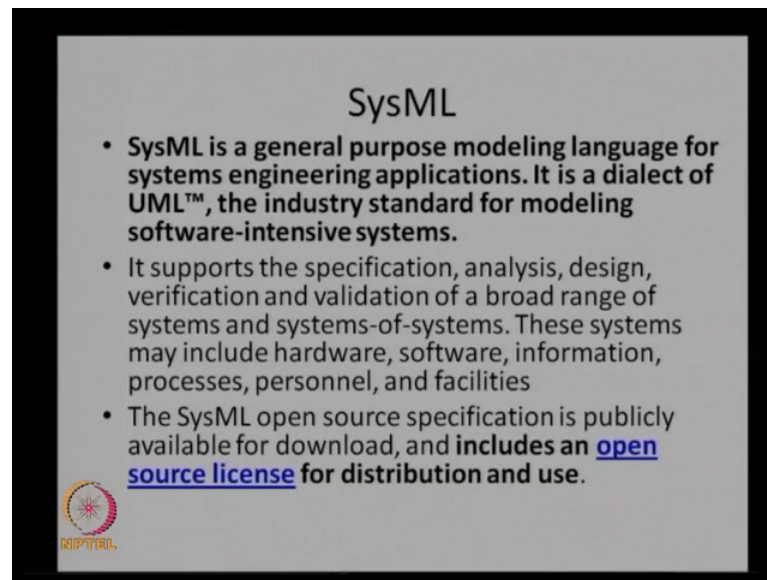


So, to summarize what were discussed in the last few lectures, we started with 6 functions of system design process; where we started with the first problem of that is the defining the system level design problems. So, that was the out of the 6 this is the first one and we attempted the 2 go into the details of this phase.

So, basically you will discussed about the operational concept, and how do we develop the operational concept, and then how do we identify the external systems. How do we look at the originating requirements, and then the objectives hierarchy, and documentation and requirements management? And at the end of this level what we are getting as an output is the originating requirements document, and we saw how do we actually prepare the ORD, and what is the format for preparing the originating requirements document. So, with this we complete the first level of design problem that is the defining the system level design problem. Before going to the next level that is the functional development of the system, I would like to briefly explained to you about some of the softwares available for a system design.

So, I will briefly explain these softwares basically it would give you an idea of what actually the software can do in helping you to design a an engineering system. So, there are as I mentioned earlier there are multiple software tools for a system engineering, and some of them are commercially available. Some of them are mean some of them are fully available some of them you need to get some license to use.

(Refer Slide Time: 21:03)

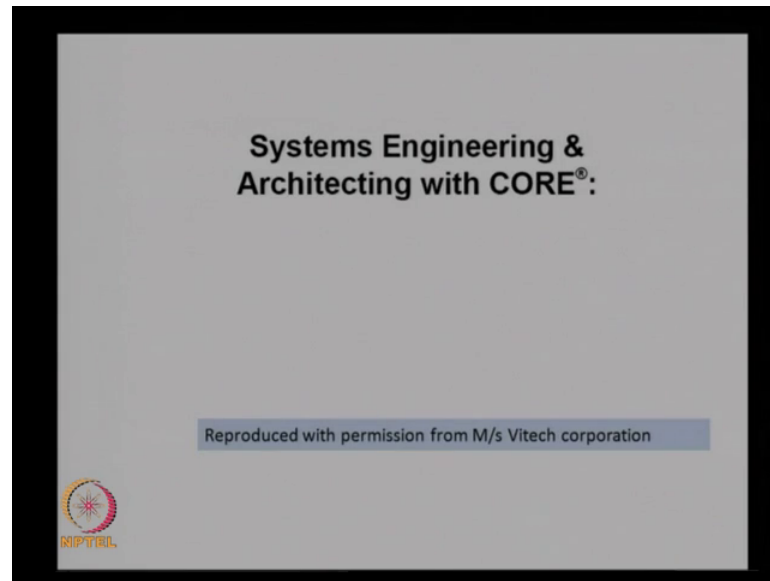


As I mentioned earlier there is one of the software is known as a SysML. It is a general-purpose modeling language for systems engineering application it is a dialect of UML. It is a trademark the industry standard for modeling software intensive systems.

So, this is the basically modeling language, a dialect for industry standard. Basically, used for software intensive a system. they are basically now used in a system engineering because lot of system engineering applications include software and hardware integration. So, this is very well used in a system engineering applications. So, it supports the specification analysis design, verification and validation of a broad range of systems and systems of systems. This systems may include hardware software information processes personnel and facilities. So, we can actually use this a SysML language for analysis and design or verification and validation of systems; which actually include hardware software information process personnel and facilities.

This is an open source which is publicly available for download, and includes an open source license for distribution and use. So, if you are interested or if you want to download it you can actually get it is a free and open source. You can use this one for your system design applications. And most of the softwares are come with it is own user manual and explanations on how to use different functions in the software. So, you can downloading the software most likely you will be getting the users manual and explanation on how to use it.

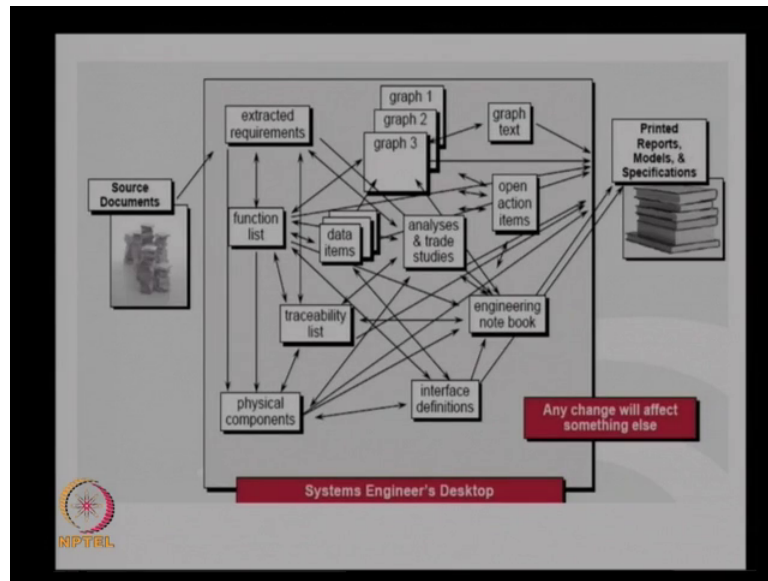
(Refer Slide Time: 22:40)



Another one is known as core. It is again a software coming from vitech corporation. I explained you in one of the classes that IIT Madras has got a arrangement with vitech corporation. So, we are actually part of their university education program. And therefore, we are actually eligible for downloading the software few of course, especially the education version. Only thing you need to have some registration with this company.

So, once you register with this company you will be given a password to download, and a most likely you will have to approach me to get the password because they send the password to me if you ask for registration. And then you can actually download it and used for your education application. So, in case you want to download the software you can contact me and we will make a arrangement for you to get the password for downloading the software. So, this software is known as core basically looking at the different aspects of system level design problem, and then how this problem can be simplified using the software.

(Refer Slide Time: 23:48)

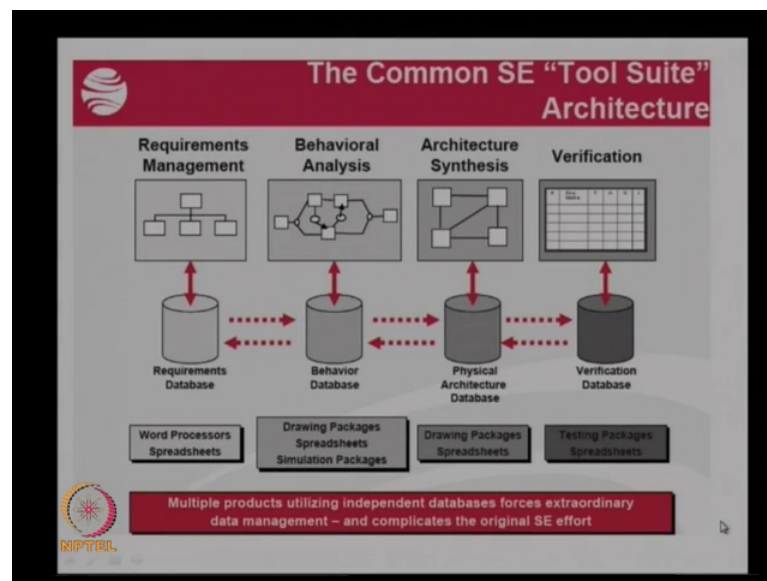


I will give you a brief explanation about the software, what it can do and what are the capabilities or what actually makes it interesting to use the software for a system level design problem. As you can see system engineers desktop, there are lot of things in the desktop of a system engineer. So, he has got the source document from where he will actually extract the requirements, and then based on this requirement it will be finding out the function list, and then the physical components. And for the physical components you need to have a traceability, and the traceability will go back to function and then from the function it will go to the requirements. And similarly, there are different data items and there are interface definition when you have physical component we will be lot of interfaces between components.

So, you need to define this interfaces that again coming from the function and the data items. Similarly, there will be a lot of graphical output coming from the function list in terms of different way of modeling the a behavior of the system, and this actually all this need to be recorded as printout reports or models or specifications. So, again the note book engineering note book data what there will be whatever you stable on different design activities or a concept development. So, all these things are need to be finally, printed as a reports. So, as you can see there are lot of interaction between all these functions and any change will affects something else on the system.

So, how do we have a properly ordered and a systematic way of representing all these thing in a using the capabilities of computation and information technology? How do we actually develop this a particular over a an application? Or we can have all these things in a very order and systematic way, without much complexity you can make easy changes you can have proper traceability. So, how do we do this is the is a major task for a system engineer?

(Refer Slide Time: 25:41)



So, if you look at the a common system engineering tool suite architecture. A normally there will be a database for the requirements. This will be in terms of it will be a written it has a word processors or using spreadsheets.

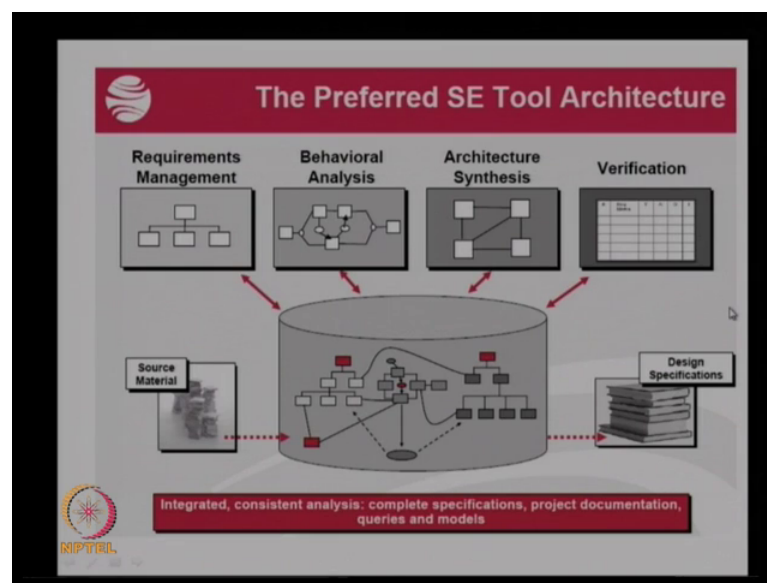
So, all the requirements database will be developed. And there will be a behavior database in terms of the functions and functional interactions and the interfaces. And then you will be having physical architecture database, the components subassemblies and how they actually satisfy the functions. And then there will be a verification and validation database. So, there will be multiple databases, and using this database the requirements will be managed using the requirement database. And the behavior analysis will be carried out using the behavior database, and architecture synthesis will be done using the physical architecture, and verification done will be using verification database.

So, as we can see here different source for these databases will be from behavior database will be from drawing packages, spreadsheets, simulation packages and drawing

packages spreadsheets over here for physical architecture, testing packages and spreadsheets for verification database. So, here you can see multiple products utilizing independent databases; which forces extraordinary data management and complicates the original system engineering effort. So, here you can see there are multiple databases, and using this multiple databases become makes the data management a very difficult task.

So, here is the importance of having a good software to help you to manage these databases, and there actually the software helps you to provide a common database.

(Refer Slide Time: 27:17)

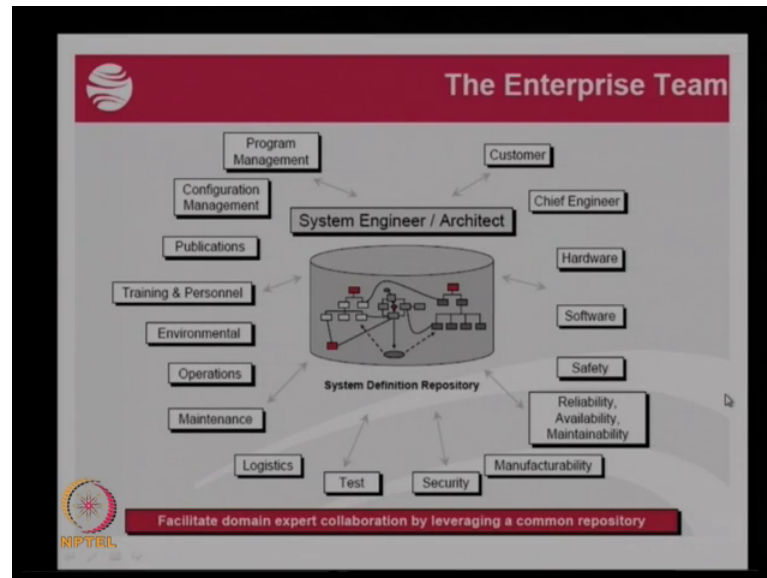


So, that is the preferred system engineering architecture. Where you have a common database where it is properly arranged with proper traceability, and all those management requirements management behavior analysis, architecture synthesis and verification, they are actually coming from this database. And there will be only one input to this database that is a source material, and the output will be the design specification. So, this interaction they all the other interactions will be through the database. And therefore, this common database helps to have a proper control of the data and data management becomes very easy in this case.

So, in integrated consistent analysis, which actually gives a complete specifications, project documentation, queries and models everything will be directly coming from the database, or this a common database. So, that is the important or that is the preferred architecture where you provide a common database and from this database you have all

the other analysis, and once you have any changes to one of these data will actually be captured by all these systems or all these sub functions, and any changes can be easily located or it can be traced without any difficulty, because there is only one data source and this data source can be easily managed. So, that is the idea of having this particular software.

(Refer Slide Time: 28:37)



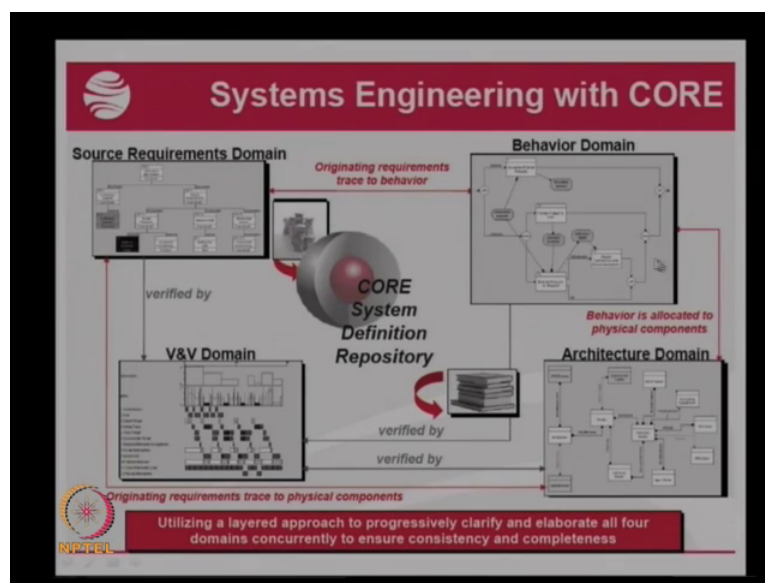
So, we can see that in this database, there will be many interactions apart from the functions, there will be interaction with program management, the configuration management publications then training personnel, then environmental interactions operations, maintenance, logistics, test, security, manufacturability, reliability, a availability and maintainability of the systems safety, software, hardware and engineers and the customers. So, all of them will be interacting with this database or directly or indirectly they will be interacting. And the system engineer or architect will be at the top level looking at the database and providing necessary update as well as a interaction with the other agencies.

So, all these people will be always looking at a single database instead of having multiple databases, a single database will be therefore, everyone to access or everyone to have the others to have controlled access so that there will not be any ambiguity about the changes made in the system, or any variations in the design. Because all the a system all the subsystems or the development teams will be using the same database. Therefore,

it is very easy to manage. So, there will be a better integration or better interaction with the different teams for maintaining uniformity in the development process.

So, it facilitate domain expert collaboration by leveraging a common repository. So, there are different domain experts. They will be always having the same design repository and a confusion or the problems created because of multiple databases and multiple changes can be easily be avoided using this kind of a common repository for design engineers.

(Refer Slide Time: 30:22)



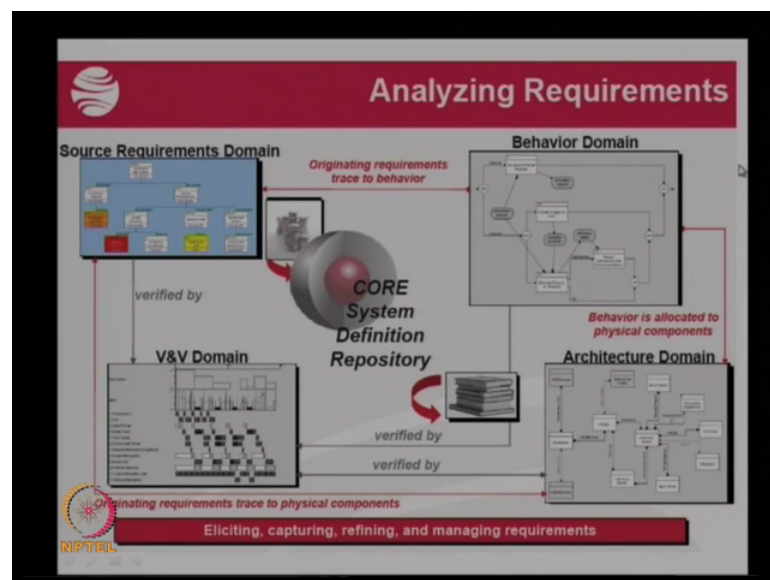
So, the particular software core actually looks at these from the 4 sub functions; basically, the source requirements domain, behavior domain, architecture domain and v and v that is verification and validation domain.

So, all these domains will be using the same database, and independently you can develop these requirements a using the database, and based on the requirements domain you can develop the behavior domain; where you can always trace to the behavior the originating requirements trace to behavior and from behavior you can actually go to the architecture design. So, behavior is basically looking at the functions, and then functional decompositions. So, you look at from this behavior domain you can develop the architecture domain how the behavior is allocated to different physical components can be identified. And then you can have the verification and validation using the architecture developed for the system.

So, here you can see that there is an originating requirements, trace to physical components. So, you can look at the requirements and then how these requirements are mapped using the physical components. Similarly, you can have this traceability between the requirements and the behavior and the architecture and the architecture and the verification and validation requirements. So, all these function will be always using a common database and using this common database all these will be developed, and any changes in one of these will be automatically reflected into the core database and the database will be accessed by the other domains like behavior or architecture in order to make the necessary changes.

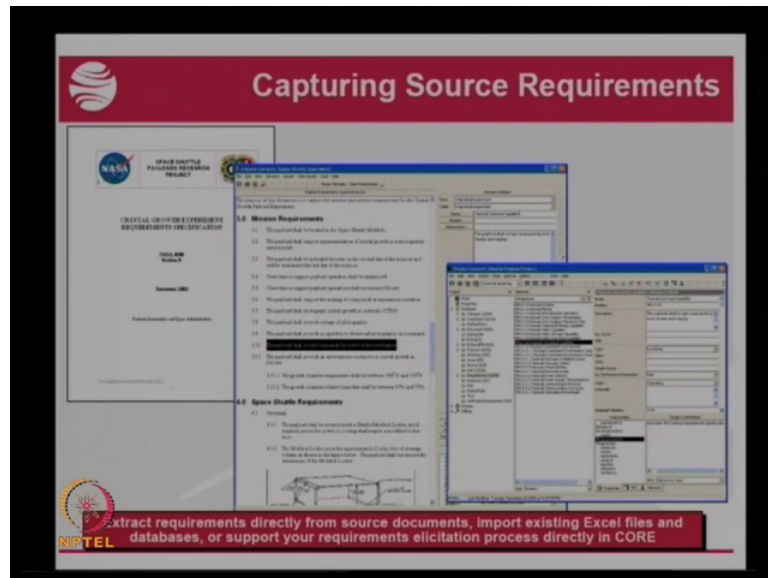
So, utilizing a layered approach to progressively clarify and elaborate all 4 domains concurrently to ensure consistency and completeness. So, there are 4 layers and this layered approach to progressively clarify and elaborate all the domains (Refer Time: 32:14) even concurrently and ensure consistency. So, that is the advantage of doing these. Let us see how these domains are developed using the database. And there are software tools available, which can actually help you to develop these domains very easily without much difficulty in identifying the a sources as well as the a traceability of those requirements.

(Refer Slide Time: 32:35)



So, let us look at the source requirements domain. Basically, we will be looking at the requirements.

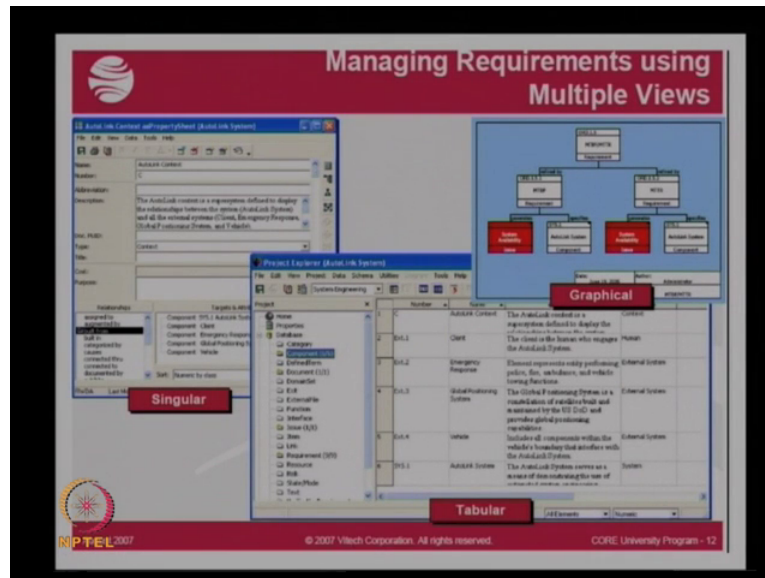
(Refer Slide Time: 32:42)



So, while developing the requirements the software allows you to extract requirements directly from the source documents, the documents could be an excel sheet or a word processor using a word processing tool. So, any of these tools can be used to extract the requirement. So, these tools can be used to extract the requirements, and this requirement will be put into the database of the software, and support you requirements excitation process directly in core.

So, you can use the software, and directly get the requirements from these input documents and then managing the requirements using multiple views.

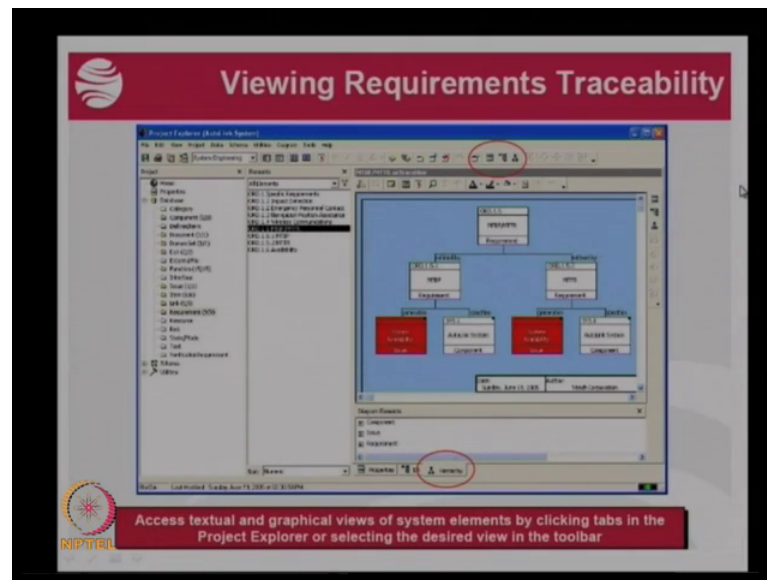
(Refer Slide Time: 33:18)



So, you can have multiple view of the requirements. You can have a singular view over a particular a requirement, you can identify you can look the actual requirement the description of the requirement. And then whatever the targets the components which actually provide this one from where the requirement came what is the traceability of the requirement. So, all those thing can be identified from this singular representation. Or you can have a graphical representation which only tell you the hierarchy of the requirements.

So, what is the main requirement, and then a how does the sub requirements come, and then how can actually we divide this requirements into another sub class. So, all these hierarchical requirement also can be shown in the a software. Or you can have a tabular format for the requirements. So, for the different requirements, you can tabulate it. And then look at the details of these requirements if you want to go into the details of the requirement you can actually click on to this requirement, then it will come into a singular format. So, this is possible with the software, because it actually allows you to save the requirements in different formats singular graphical or tabular and that actually helps you to have a proper control of the requirements.

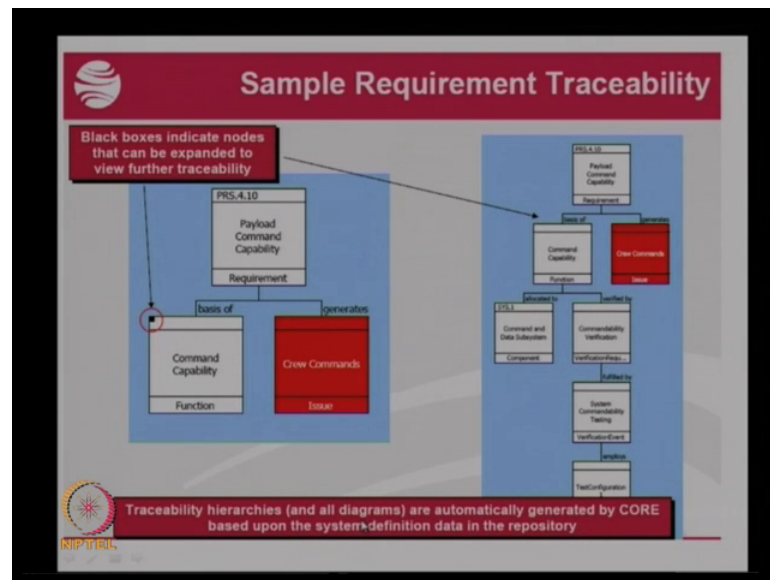
(Refer Slide Time: 34:35)



So, here actually you can see that there are different options available using this option you can see the requirements, and then its traceability can also be looked at. So, if you want to know more about this particular requirement, you can actually go on to that requirement then see from where this requirement came what was the original requirement and then how this was modified. So, all the details of the requirements can be easily obtained using this format. So, it access textual and graphical views of system elements by clicking tabs in the project explorer or selecting the desired view in the toolbar.

So, you can use the desired toolbar and then I mean you click on the desired toolbar and get the desired view of the requirement hierarchy.

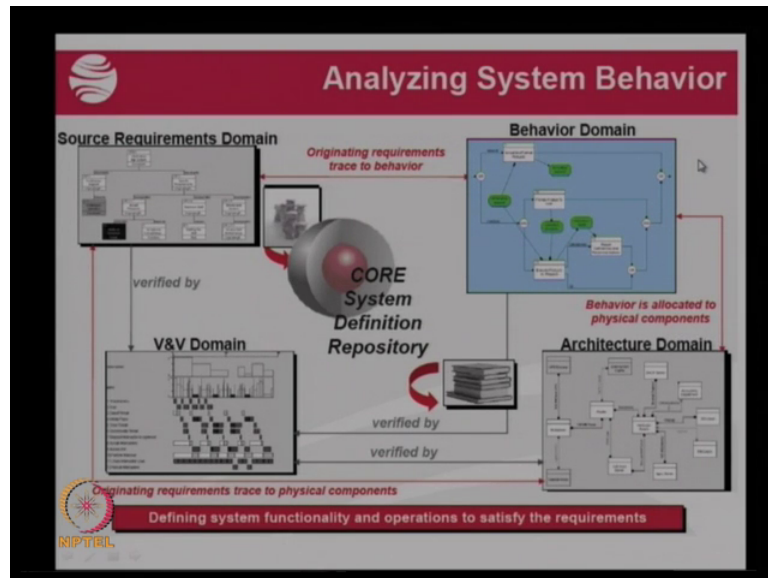
(Refer Slide Time: 35:18)



Again, you can traceability is basically you can look at the requirement hierarchy, and there is a more to this particular requirement there will be a some integration. A black boxes indicate nodes that can be expanded to view further traceability. So, if you want to know more about this command capability requirement, can click on to this and it will go it is sub phase, and it will tell you all the other requirements it can be identified.

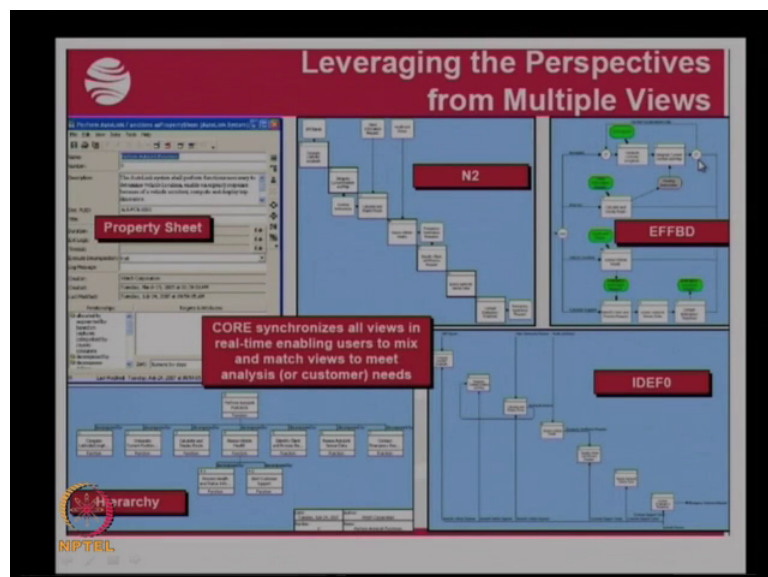
So, the traceability hierarchies are automatically generated by core based upon the system definition data in the repository. So, this always will be automatically generated. So, you do not need to really prepare all these things. Once you give a link to the in the while preparing the repository, if you link all these requirements automatically this the traceability will be generated. So, it is easy to go into the details of that requirement then find out the origin of that particular requirement.

(Refer Slide Time: 36:15)



And the next one is the behavior domain. Behavior domain is basically how do you convert these requirements to the functional blocks, or the decomposition of this into sub functions. So, this can actually be done in different ways, there are multiple ways of modeling the behavior of the system and some of the methods are like you can have a hierarchy of the functions.

(Refer Slide Time: 36:36)

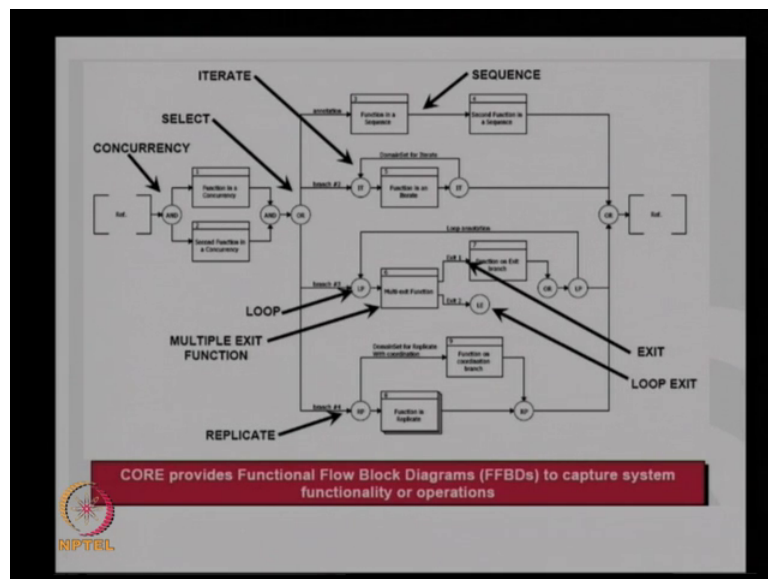


You can have the top-level function, and then subdivide this into all sub functions or again divide into it is sub functions.

So, this actually will give you a functional hierarchy of the system. Or you can actually have a property sheet for each function you can discover the details. And then find out the from where this particular function came what was the requirement on which the function was developed. Or there are other methods called n 2 diagrams. We will learn about this n 2 diagrams, similarly functional block diagram or IDEF 0 diagram. These are all different methods of representing the behavior of the system, or the requirement of a function. We will discuss about this in the coming lectures, but you can just understand that the software can be used to develop or represent the functions in different ways.

So, depending on the convenients of the user, you can either access the IDEF 0 diagram, or the n 2 diagram or the a functional block diagram, and then look at the functions and it traceability and the interaction between different function.

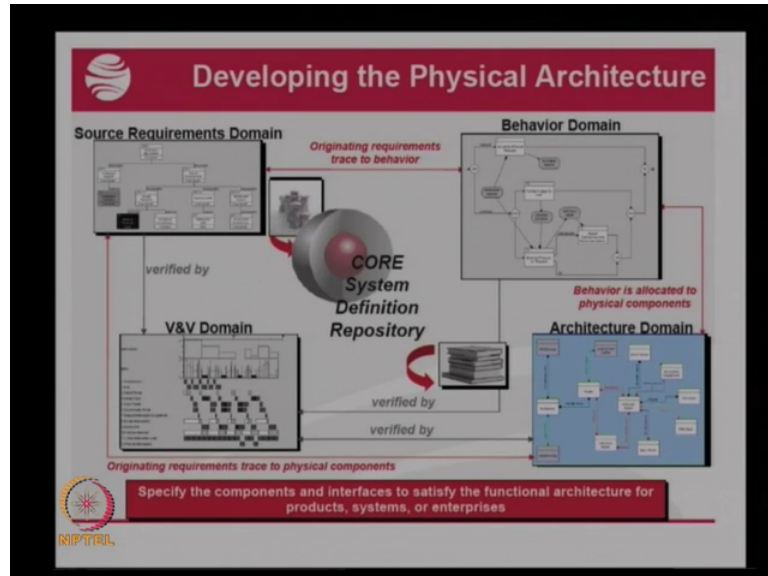
(Refer Slide Time: 37:46)



For example, if you look at the functional block diagram FFBD, you can see that there are functions like a and or iterations and exit. So, these all tell you what actually happens about this particular function, what is which are the functions to be executed parallely or together or which are the function to be a at serially using and or blocks or which are need to be iterated for a multiple times. So, all those things can be easily represented using this diagram.

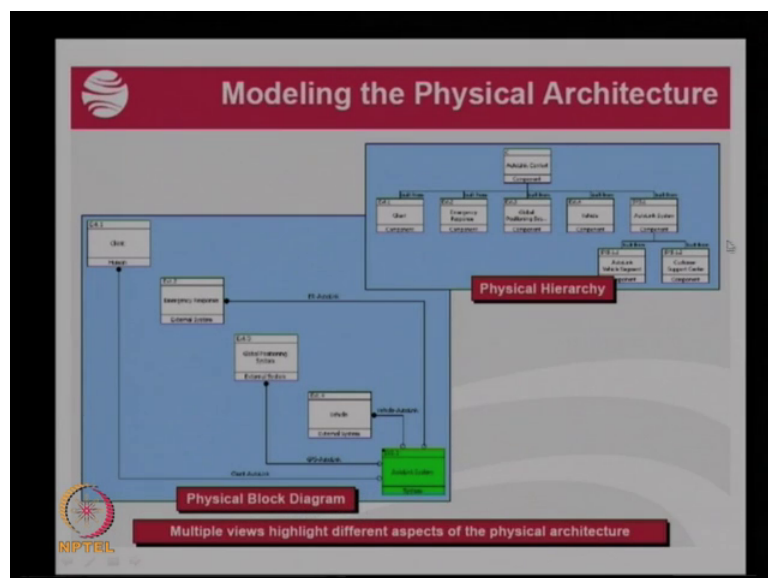
So, core actually provides you the facility to model the behavior of the system in this format using FFBD.

(Refer Slide Time: 38:27)



Developing the physical architecture. So, after the behavior domain, once you identify the function, you need to go for the architecture domain; that is, converting these functions into blocks or the physical components. So, these again need to be trace to the functions as well as to the requirements. So, you can see that you can develop using the behavior domain.

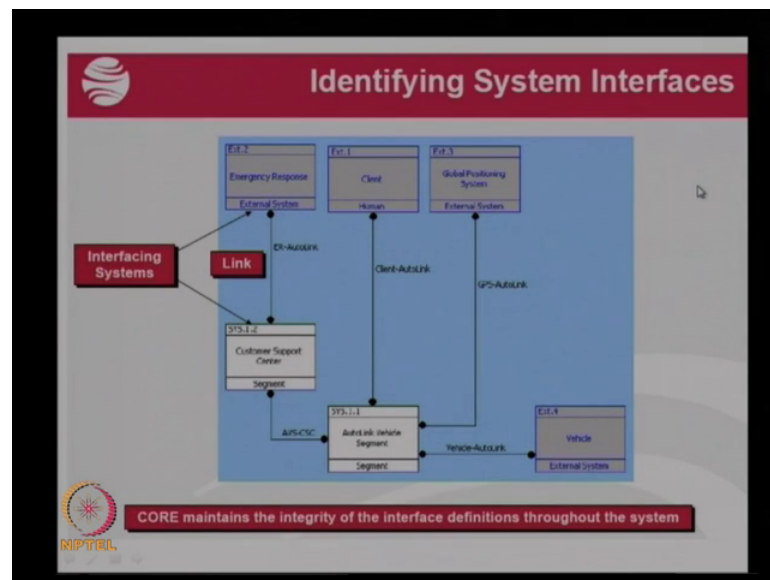
(Refer Slide Time: 38:53)



From the behavior domain, you can develop for the architecture domain. Again, architecture domain you can have a physical hierarchy.

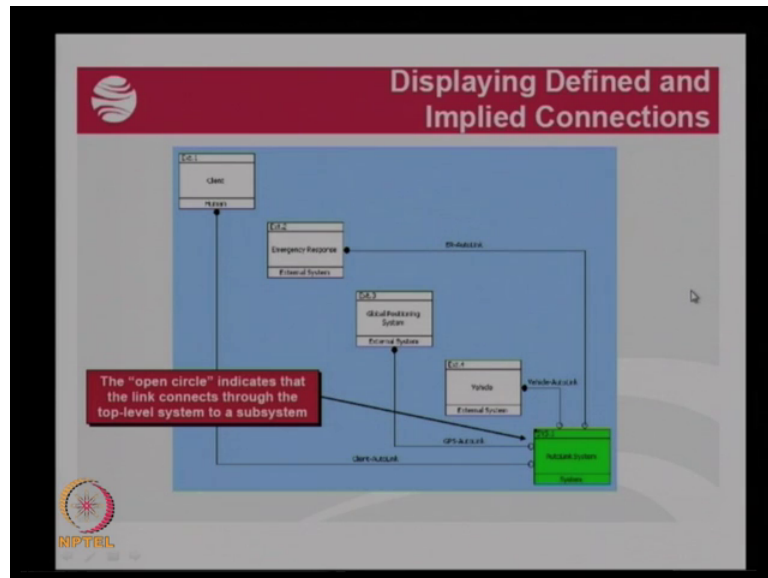
So, depending on the components, we can have the physical architecture hierarchy of components, the main assembly and then sub-assemblies and then components to this way a physical hierarchy can be developed. Or you can have the physical block diagram with its own interaction with the different blocks. This actually shows that these blocks are connected to or this particular system is connected to these blocks emergency resources. And the multiple views highlight different aspects of the physical architecture. So, you can have multiple views. So, this is one aspect of the physical hierarchy, and this is another aspect of the block diagram.

(Refer Slide Time: 39:33)



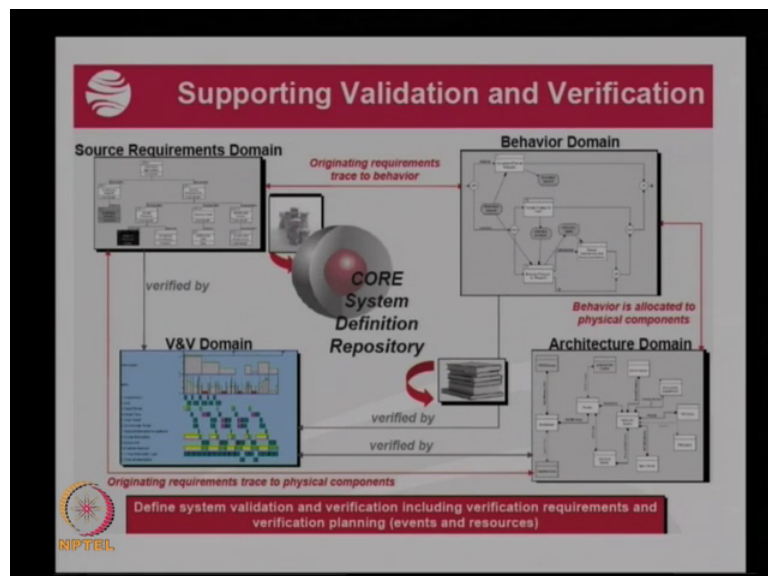
And then the interfaces also can be identified using the physical hierarchy. Because we will be having different components and subsystems and they knew to interact with each other. And this interaction interfaces through which the systems interact can be easily identified and represented using interfacing systems. That is actually show you the core maintains the integrity of the interface definitions throughout the system. So, whatever the interface you define. So, that can be actually be the integrity of it can be maintained throughout the system development process and then displaying defined and implied connections.

(Refer Slide Time: 40:04)



So, different connection can be defined using the physical hierarchy. So, you can see that there are different methods of representing the connections. An open circle indicates that the link connects through the top-level system to a sub system. So, this is the top-level system. So, it connects to a subsystem through a link. So, this is the open circle similarly, there are different ways of representing the connections. So, all those things can be easily represented using this physical architecture in the core.

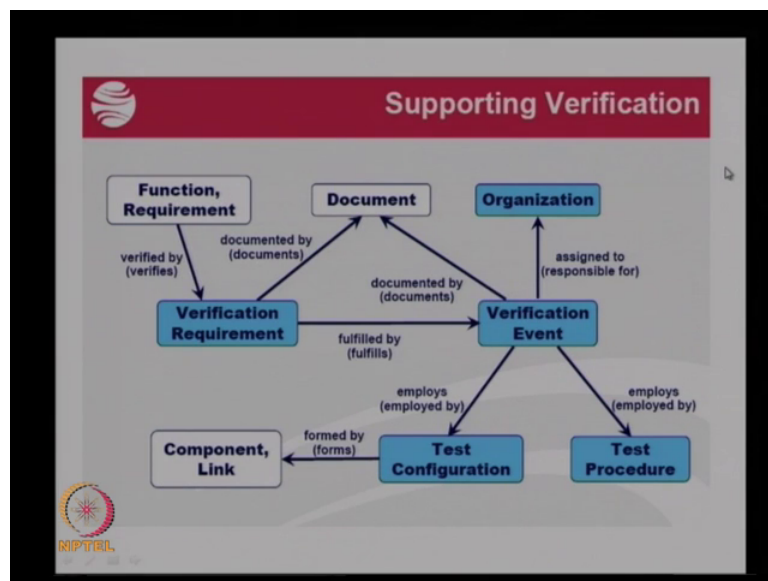
(Refer Slide Time: 40:37)



The other one is the last one is the validation and verification.

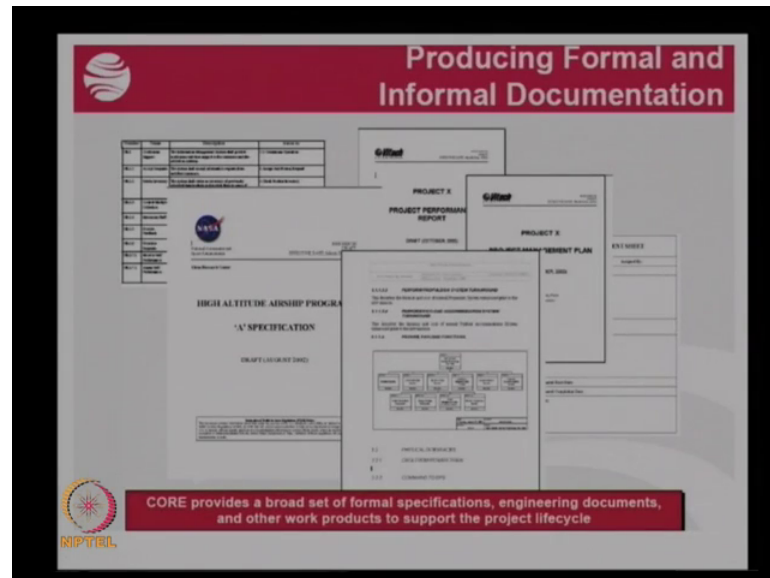
We have this behavior domain and then architecture domain. As we have the component then we need to validate and verify the components, whatever is selected for providing this function. Again, this has to be verified by the system requirements. That is the interaction shown here. Again, here you can see that, the function requirements, the documentation, the organization, components, test configuration and test procedures, all these are linked to the verification event and a verification requirement.

(Refer Slide Time: 40:57)



So, we have the verification events. So, how do we verify? And what are the requirements needed to do that particular verification? That again comes to the verification function requirement. And all these things need to be documented using the verification documents. So, these also can be easily incorporated in the software. Once we have the components we can during the development of the components you can have a verification procedure for this one. So, automatically this will be link to the verification document, and this system will provide you a verification chart for that particular subcomponent or the subsystem.

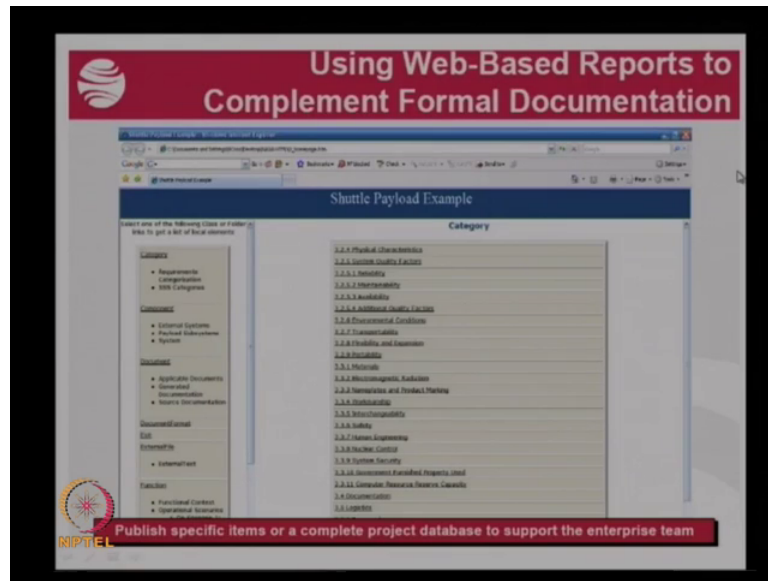
(Refer Slide Time: 41:47)



And once you have this all these you can actually have a formal and informal documentation. So, if you are not able to read all this data this actually comes from the software. Basically, it tells you that you can have different kinds of formats of the document for the system. So, it provides a broad set of formal specifications, engineering documents, and other work products to support the project life cycle. So, you can have document for the functional requirement, or you can have the document for the verification requirements, or you can have a document for the originating requirements document.

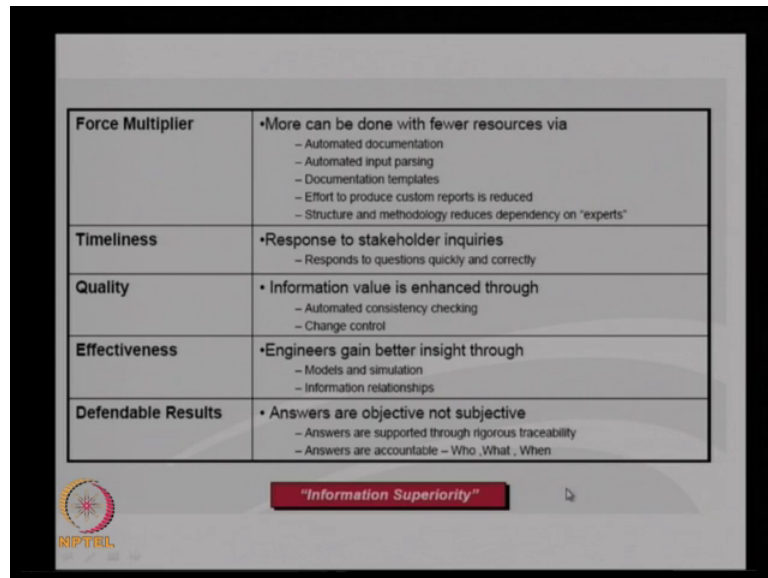
So, all those documents can be easily generated from the software without much difficulty you do not need to go back and forth to find out the traceability and from where the requirement came or from where the document can be traced. So, you can easily develop these documents using the software. The software provides you the features to develop documents, and are with the a broad set of formal specifications and engineering documents. So, these are the advantages of using a software for development of engineering system.

(Refer Slide Time: 42:53)



And using web based reports to the complement formal documents, if you want a web based reports. So, if you want to categorize the reports and keep it as a document for a further analysis, then that also can be done you can have different categories of the document. And then a probably specific items or a complete project database to support the enterprise team. So, if any enterprise team ask for a particular document you can easily generate it on the web, and then can be transferred to that particular team. So, that they can be go through the document and then identify what is the present status of the a requirements identified or the changes made in the configuration. All those things can be easily carried out using the software, the advantages of using the software, whether it is a core or SysML or any other software, you can always used as a force multiplier.

(Refer Slide Time: 43:37)



Force Multiplier	<ul style="list-style-type: none">• More can be done with fewer resources via<ul style="list-style-type: none">– Automated documentation– Automated input parsing– Documentation templates– Effort to produce custom reports is reduced– Structure and methodology reduces dependency on "experts"
Timeliness	<ul style="list-style-type: none">• Response to stakeholder inquiries<ul style="list-style-type: none">– Responds to questions quickly and correctly
Quality	<ul style="list-style-type: none">• Information value is enhanced through<ul style="list-style-type: none">– Automated consistency checking– Change control
Effectiveness	<ul style="list-style-type: none">• Engineers gain better insight through<ul style="list-style-type: none">– Models and simulation– Information relationships
Defendable Results	<ul style="list-style-type: none">• Answers are objective not subjective<ul style="list-style-type: none">– Answers are supported through rigorous traceability– Answers are accountable – Who, What, When

"Information Superiority"

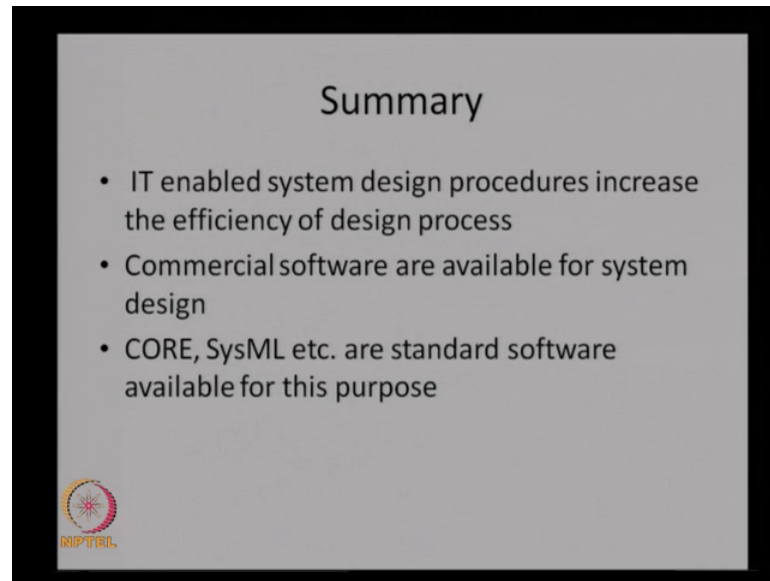
RIPTROL

So, more can be done with fewer resources, via automated documentation, automated input passing documentation templates effort to produce customer reports, and structure and methodology which reduces dependency on experts. So, once you have all these data you can even non-expert can actually use it very easily and get the reports. And timeliness that is response to stakeholder inquiries or very easily you can get the reports and you can respond to the inquiries on the stakeholders.

Quality the information value is enhanced through automated consistency checking and change controls. So, these are all automatically they are all part of the software. So, you can have a consistency checking and change control always. So, whenever you make a change you ensure that this change is reflected in all the related components and systems. And the effectiveness engineers gain better insight through models and simulation information relationships. And the defendable results answers are objective not subjective. So, you can always answer to the queries based on the data available. So, it is always objective results. And the answers are accurate who did what and when they did and what changes made and who made all those details are easily available in the software. So, it can actually trace the changes and modifications easily and find out the requires.

So, it basically it gives you an information superiority in the development of the system.

(Refer Slide Time: 45:13)



The slide is titled "Summary" and contains the following bullet points:

- IT enabled system design procedures increase the efficiency of design process
- Commercial software are available for system design
- CORE, SysML etc. are standard software available for this purpose

In the bottom left corner, there is a circular logo with a star-like pattern and the text "MPTBL" below it.

To summarize a we have a different software for doing help development of engineering system. So, the information technology enable to system design procedures, increase the efficiency of design process, commercial software are easily available for system design or they are available you can actually either a get it from the open source or you can actually get for go for commercial softwares. And it can be used for system development. And core SysML etcetera are some of the standard softwares available for this purpose.

So, with this we conclude this session on requirement analysis as well as the softwares for the system development. And the next class onwards we will start functional decomposition or behavior modeling of the system. And then we will go to the physical architecture development then verification and validation.

So, till we meet good bye.