Software Engineering Requirements Slides:

Lecture#1

(Software Requirements Engineering)

**Introduction:**

* Requirements form the basis for all software products
* Requirements engineering is the process, which enables us to systematically determine the requirements for a software product

**Requirements:**

* Something required, something wanted or needed-Webster’s dictionary
* There is a huge difference between *wanted* and *needed* and it should be kept in mind all the time

**Software Requirements - 1:**

* A complete description of *what* the software system will do without describing *how* it will do it is represented by the software requirements

**Software Requirements - 2**

* Software requirements are complete specification of the desired external behavior of the software system to be built
* They also represent External behavior of the system

**Software Requirements - 3:**

* Software requirements may be:

-Abstract statements of services and/or constraints

-Detailed mathematical functions

**Software Requirements - 4:**

* **Software requirements may be:**
  + Part of the bid of contract
  + The contract itself
  + Part of the technical document, which describes a product

**IEEE Condition:**

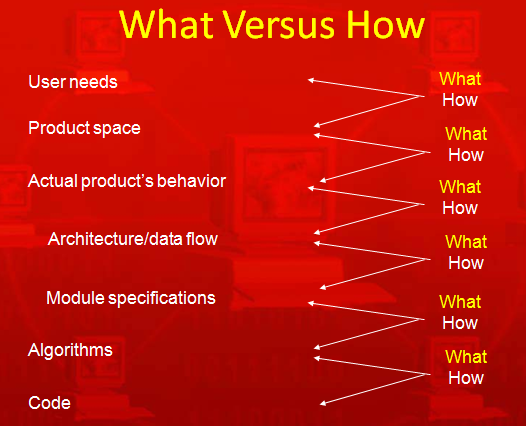
A condition or capability that must be met or possessed by a system...to satisfy a contract, standard, specification, or other formally imposed IEEE Std 729

**Sources of Requirements:**

* Stakeholders
  + People affected in some way by the system
* Documents
* Existing system
* Domain/business area

**Levels of Software Requirements:**

* Stakeholders describe requirements at different levels of detail
  + *“What versus How”*
  + *“One person’s floor is another person’s ceiling*”



**Importance of Software Requirements:**

* The hardest single part of building a software system is deciding what to build...No other part of the work so cripples the resulting system if done wrong. No other part is difficult to rectify later
  + Fred Brooks

**Examples of Requirements - 1:**

* The system shall maintain records of all payments made to employees on accounts of salaries, bonuses, travel/daily allowances, medical allowances, etc.

**Examples of Requirements - 2:**

* The system shall interface with the central computer to send daily sales and inventory data from every retail store

**Examples of Requirements - 3:**

* The system shall maintain records of all library materials including books, serials, newspapers and magazines, video and audio tapes, reports, collections of transparencies, CD-ROMs, DVDs, etc.

**Examples of Requirements - 4:**

* The system shall allow users to search for an item by title, author, or by International Standard Book Number
* The system’s user interface shall be implemented using a web browser

**Examples of Requirements - 5:**

* The system shall support at least twenty transactions per second
* The system facilities which are available to public users shall be demonstrable in ten minutes or less

**Kinds of Software Requirements:**

* Functional requirements
* Non-functional requirements
* Domain requirements
* Inverse requirements
* Design and implementation constraints

**Functional Requirements:**

**Functional Requirements - 1:**

* Statements describing what the system does
* Functionality of the system

**Functional Requirements - 2:**

* Statements of services the system should provide
  + Reaction to particular inputs
  + Behavior in particular situations

**Functional Requirements - 3:**

* Sequencing and parallelism are also captured by functional requirements
* Abnormal behavior is also documented as functional requirements in the form of exception handling

**Functional Requirements - 4:**

* Functional requirements should be complete and consistent
* Customers and developers usually focus all their attention on functional requirements

**Functional Requirements Example # 2:**

* The user shall be able to search either the entire database of patients or select a subset from it (admitted patients, or patients with asthma, etc.)

**Functional Requirements Example # 3:**

* The system shall provide appropriate viewers for the user to read documents in the document store

**Functional Requirements Example # 4:**

* Every order shall be allocated a unique identifier (ORDER\_ID) which the user shall use to access that order

**Functional Requirements Example # 5:**

* The system shall allow customers to return non-perishable items within fifteen days of the purchase. A customer must present the original sale receipt to return an item

**Comments on Examples:**

* Notice the level of detail in different requirements described above. Some are very detailed compared to others

**Comments on Examples:**

* Notice the ambiguity in the requirement, which uses the term ‘appropriate viewers’
* This requirement does not mention the formats of documents and types of viewers, which can be used

**Comments on Examples:**

* Incomplete and ambiguous requirements are open to multiple interpretations and assumptions
* This can lead to the development of poor quality, or faulty, software products

**Summary:**

* Requirements form the basis of all software engineering projects
* Functional requirements capture the behavioral aspects/functions of the proposed automated system
* Functional requirements are the backbone of all software products

**Lecture#02**

**Non-Functional Requirements:**

**Non-Functional Requirements - 1:**

* Most non-functional requirements relate to the system as a whole. They include constraints on timing, performance, reliability, security, maintainability, accuracy, the development process, standards, etc.

**Non-Functional Requirements - 2:**

* They are often more critical than individual functional requirements
* Capture the emergent behavior of the system, that is they relate to system as a whole

**Non-Functional Requirements - 3:**

* Must be built into the framework of the software product
* Failure to meet a non-functional system requirement may make the whole system unusable

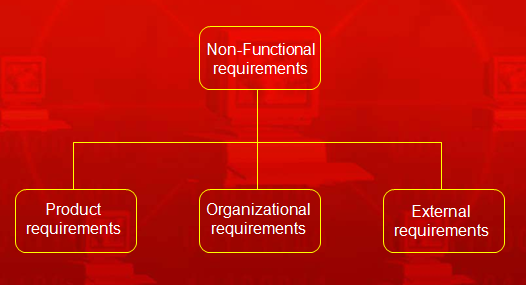
**Non-Functional Requirements - 4:**

* For example, if an aircraft system does not meet reliability requirements, it will not be certified as ‘safe’
* If a real-time control system fails to meet its performance requirements, the control functions will not operate correctly

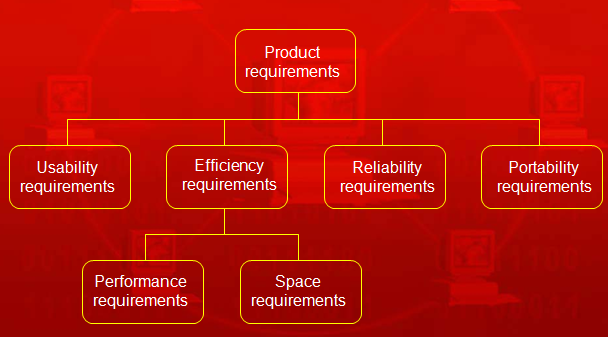
**Non-Functional Requirements - 5:**

* Non-functional requirements arise through user needs, because of budget constraints, because of organizational policies, because of the need of interoperability with other software and hardware systems, or because of external factors such as safety regulations, privacy legislation, etc.

**Non-Functional Requirements:**

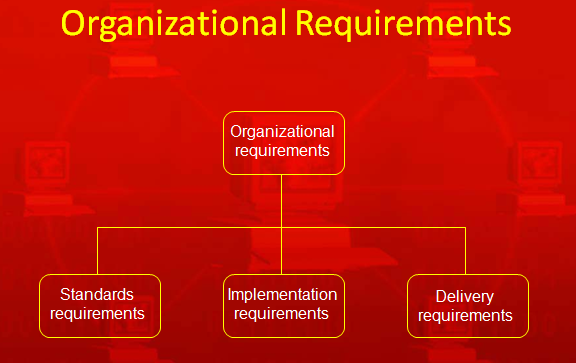


**Product Requirements:**



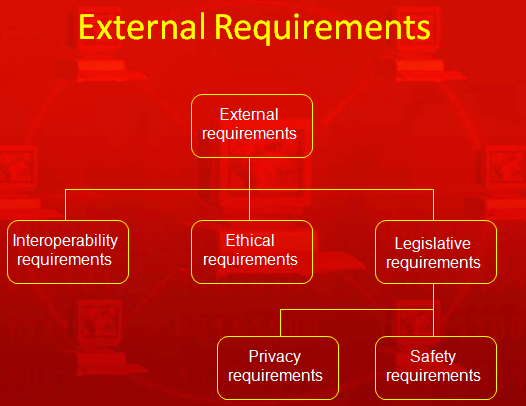
**Product Requirements Examples:**

* The system shall allow one hundred thousand hits per minute on the website
* The system shall not have down time of more than one second for continuous execution of one thousand hours



**Organizational Requirements Examples:**

* The system development process and deliverable documents shall conform to the MIL-STD-2167A
* Any development work sub-contracted by the development organization shall be carried out in accordance with Capability Maturity Model



**Observations on Non-Functional Requirements - 1:**

* Non-functional requirements can be written to reflect general goals for the system. Examples include:
  + **Ease of use**
  + **Recovery from failure**
  + **Rapid user response**

**Observations on Non-Functional Requirements - 2:**

* Goals are open to misinterpretation
* Objective verification is difficult
* Distinction between functional and non-functional is not always very clear

**Observations on Non-Functional Requirements - 3:**

* Non-functional requirements should be written in a quantitative manner as much as possible, which is not always easy for customers
* For some goals, there are no quantitative measures, e.g., maintainability

**Observations on Non-Functional Requirements - 4:**

* Goals can be useful to designers and developers, as they give clues to them about priorities of the customers

**Observations on Non-Functional Requirements - 5:**

* Chances of conflicts within non-functional requirements are fairly high, because information is coming from different stakeholders. For example, different stakeholders can give different response times or failure tolerance levels, etc.

**Observations on Non-Functional Requirements - 6:**

* Some negotiations must be done among different stakeholders, to achieve an agreement in these situations

**Summary:**

* Discussed different aspects of the non-functional requirements
* Non-functional requirements capture very important emergent behavior of the automated system
* Due importance, time, and resources should be given to non-functional requirements

**Lecture # 03**

**Kinds of Software Requirements Discussion:**

* NFRs are very important to capture the emergent behavior of the system in these there major dimensions
* **Product**
  + Usability, reliability, portability, efficiency (performance, space)
* **Organizational**
  + Standards, implementation, delivery
* **External**
  + Interoperability, ethical, legislative (privacy, safety)

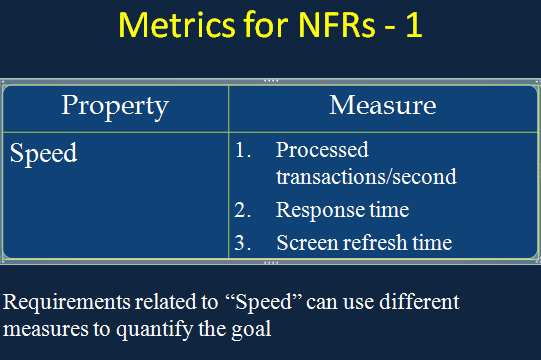
***NFRs as Goals:***

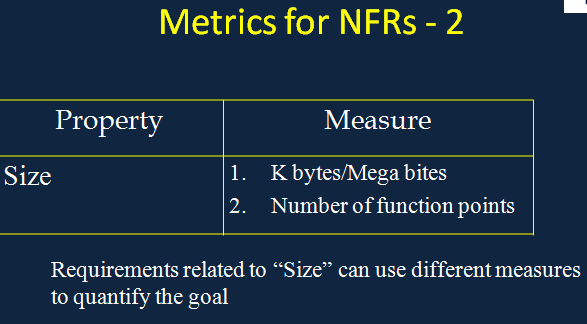
* Non-functional requirements are sometimes written as general goals, which are difficult to verify
* They should be expressed quantitatively using metrics (measures) that can be objectively tested

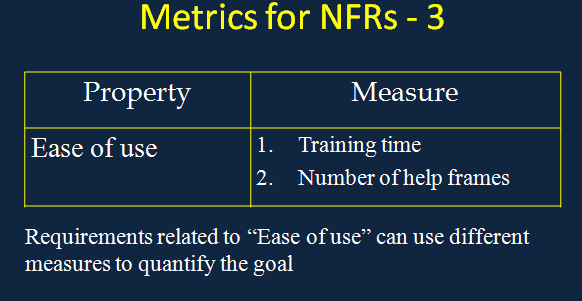
**Example:** Goal converted into an NFR:

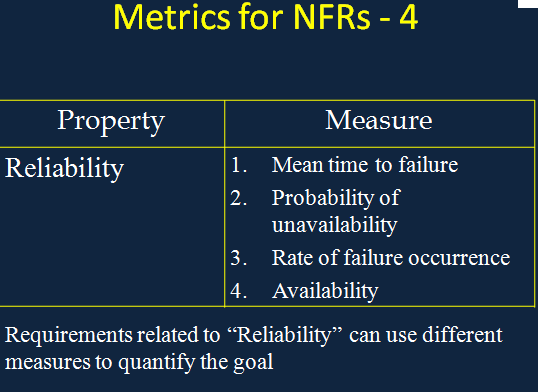
* Goal (unverifiable)
  + The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized
* Non-functional requirement (verifiable)
  + Experienced controllers shall be able to use all the system functions after a total of two hours’ training. After this training, the average number of errors made by experienced users shall not exceed two per day

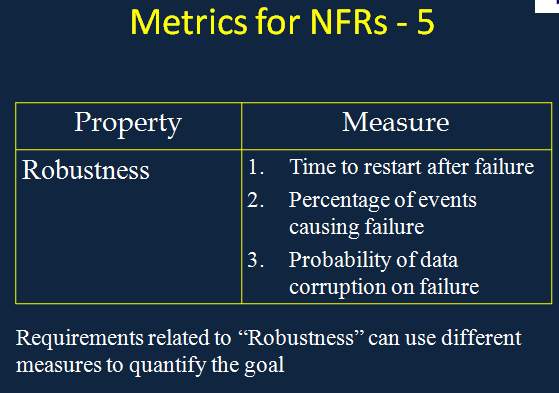
Metrics for Non-Functional Requirements (NFRs):

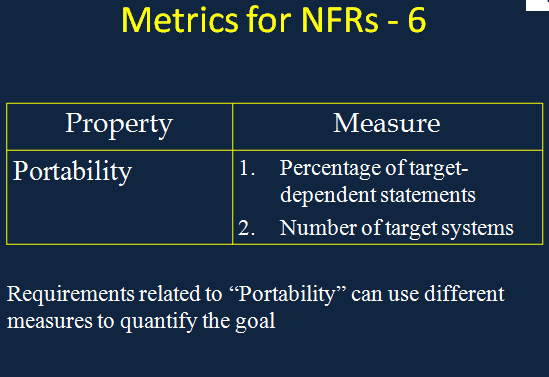










****

**Discussion on Metrics for NFRs:**

* With the help of these measures the NFRs can be verified quantitatively
* It should also be noted that the cost of quantitatively verifying each NFR may be very high.

**Domain Requirements**

**Domain Requirements - 1:**

* Requirements that come from the application domain and reflect fundamental characteristics of that application domain
* These can be both the functional or non-functional requirements

**Domain Requirements - 2:**

* These requirements, sometimes, are not explicitly mentioned
* Domain experts find it difficult to convey domain requirements
* Their absence can cause significant dissatisfaction

**Domain Requirements - 3:**

* Domain requirements can impose strict restrictions on solutions. This is particularly true for scientific and engineering domains
* Domain-specific terminology can also cause confusion

**Domain Requirements - 4:**

***Example:***

In a commission-based sales businesses, there is no concept of negative commission. However, if care is not taken novice developers can be lured into developing systems, which calculate negative commission

**Domain Requirements - 5:**

Banking domain has its own specific constraints, for example, most banks do not allow over-draw on most accounts, however, most banks allow some accounts to be over-drawn

**Inverse Requirements**

**Inverse Requirements - 1:**

* They explain what the system shall **not** do.

Many people find it convenient to describe their needs in this manner

* These requirements indicate the indecisive nature of customers about certain aspects of a new software product

**Inverse Requirements - 2:**

* ***Example:***

The system shall not use red color in the user interface, whenever it is asking for inputs from the end-user

**Design and Implementation Constraints**

**Design and Implementation Constraints - 1:**

* They are development guidelines within which the designer must work
* These requirements can seriously limit design and implementation options
* Can also have impact on human resources

**Design and Implementation Constraints Examples:**

* The system shall be developed using the Microsoft .Net platform
* The system shall be developed using open source tools and shall run on Linux operating system

**Lecture # 04**

***Recap of Last Three Lectures;***

* Kinds of requirements
  + Functional
  + Non-functional
  + Domain
  + Inverse
  + Design and implementation constraints

**Topics Covered In This Lecture:**

There also exists another view of requirements apart from different kinds of requirements we have studied so far.

* + Another view of requirements
  + There are some problems which occur in requirements, that are necessary to be identified and properly attended.
  + Problems in requirements

**Another View of Requirements:**

In general requirements can be viewed as

* + User/customer requirements

OR

* + System contract requirements

**User/Customer Requirements**

**User/Customer Requirements - 1:**

* Functional and non-functional requirements should be stated in natural language with the help of forms or simple diagrams describing the expected services of a system by the User under certain constraints

**User/Customer Requirements - 2:**

* These are understandable by users, who have no, or little, technical knowledge
* System design characteristics should be avoided as much as possible

**User/Customer Requirements - 3:**

* It is a good practice to separate user requirements from more detailed system requirements in a requirements document

**User/Customer Requirements - 4:**

* Including too much information in user requirements, constraints the system designers from coming up with creative solutions

**User/Customer Requirements - 5:**

* The rationale associated with requirements is very important. It helps in managing changes to requirements

**System Contract Requirements**

**System Contract Requirements - 1:**

* Sets out the system services and constraints in detail
* May serve as the basis of contract for implementation of the system
* Should be complete and consistent

**System Contract Requirements - 2:**

* They are used by the designers and developers as the starting point for system design
* They should be understood by technical staff of the customer organization and the development team

**System Contract Requirements - 3:**

* In principle, these requirements should also state ‘what’ the system does, rather than ‘how’ it is implemented
* However, with the level of details needed to specify the system completely, it is not possible to exclude all design information

**System Contract Requirements - 4:**

* An initial architecture of the system may be defined to help structure the requirements specification
* In most cases, systems interoperate with other systems
* Use of specific design may be included as an external requirement

**System Contract Requirements - 5:**

* Natural language is often used to describe system requirements
* Some specification languages may be used with natural language, which add structure to specifications and reduce ambiguity

**System Contract Requirements - 6:**

* Unified Modeling Language (UML) is a specification language, which has become the de-facto standard for modeling requirements

**Requirements Problems**

**Requirements Problems - 1:**

* The requirements don’t reflect the real needs of the customer for the system
* Requirements are inconsistent and/or incomplete
* It is expensive to make changes to requirements after they have been agreed upon

**Requirements Problems - 2:**

* There are misunderstandings between customers, those developing the system requirements, and software engineers developing or maintaining the system

**Problems with Natural Languages - 1:**

Requirement specification in natural language pose some problems which include

* Lack of clarity
* Requirements confusion
* Requirements consolidation

**Problems with Natural Languages - 2:**

* Natural language understanding relies on the specification readers and writers using the same words for same concept
* A natural language requirements specification is over-flexible.

“You can say the same thing in completely different ways”

**Problems with Natural Languages - 3:**

* It is not possible to modularize natural language requirements. It may be difficult to find all related requirements
  + To discover the impact of a change, every requirement have to be examined

**Impact of Wrong Requirements:**

* When requirements are wrong, systems are late, unreliable and don’t meet customers needs
* This results in enormous loss of time, revenue, market share, and trust of customers

**Summary:-**

* Discussed requirements from the user/customer’s perspective and also explored issues related to system contract requirements
* Discussed requirements problems

***Lecture #0 5***

**Process - 1:**

* A process is an organized set of activities, which transforms inputs to outputs
* We can use synonyms of process such as: procedure, method, course of action, etc.
* Processes are essential for dealing with complexity in real world

**Process - 2:**

* Processes document the steps in solving a certain problem
* They allow knowledge to be reused
* They Allow people to apply the process in their peculiar but similar problems

**Examples of Processes - 1:**

* An instruction manual for operating a microwave oven
* An instruction manual for assembling a computer or its parts
* A procedure manual for operating a motor vehicle radio and CD player

**Examples of Processes - 2:**

* A quality manual for software development.

Such a manual describes the processes, which should be used to assure the quality of the software

**Software Processes:**

* Software engineering, as a discipline, has many processes
* These processes help in performing different software engineering activities in an organized manner
* Requires creativity
* Provides interactions between a wide range of different people
* Helps in engineering judgment
* Requires background knowledge

**Examples of Software Processes:**

* Software engineering development process (SDLC)
* Requirements engineering process
* Design process
* Quality assurance process
* Change management process

**Software Requirements Engineering Process:**

Before discussing different aspects of requirements engineering process, let us discuss the concept of process models

**Process Models:**

* A process model is a simplified description of a process presented from a particular perspective
* There may be several different models of the same process
* No single model gives a complete understanding of the process being modeled

**Variations in Process Models:**

* A process model is produced on the anticipated need for that model. We may need
  + A model to help explain how process information has been organized
  + A model to help understand and improve a process
  + A model to satisfy some quality management standard

**Types of Process Model:**

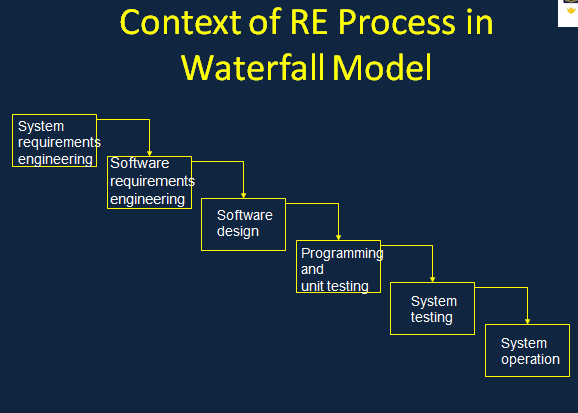
* Coarse-grain activity models
* Fine-grain activity models
* Role-action models
* Entity-relation models

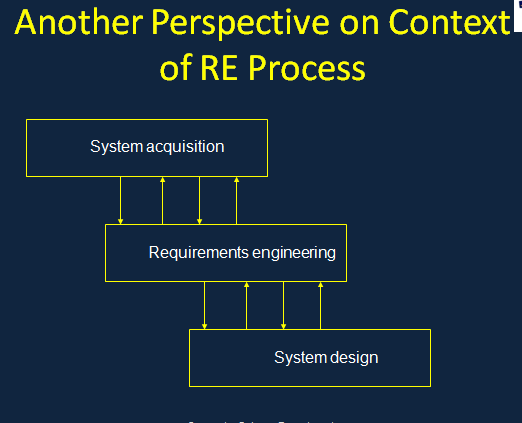
**Coarse-grain Activity Model:**

* This type of model provides an overall picture of the process
* Describes the context of different activities in the process
* It doesn’t document how to enact a process

**Context of Requirements Engineering:**

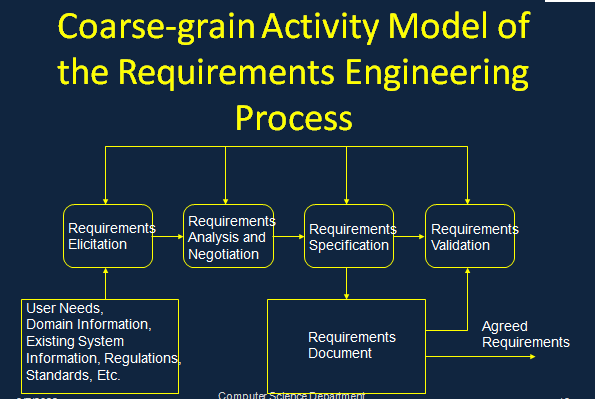
* Software requirements follow the “system requirements” and “system design”
* The primary goal is understanding
* Software requirements are followed by software design in a software development life cycle

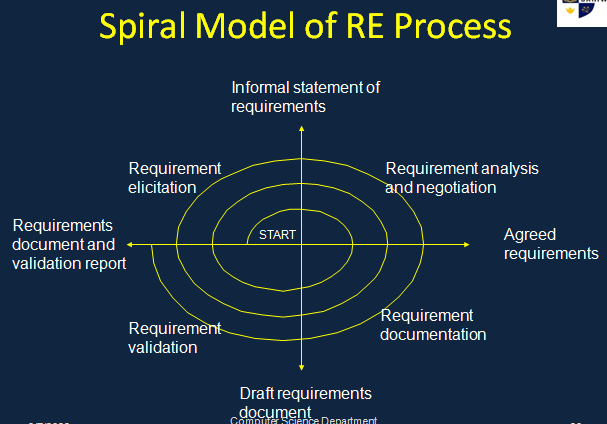




**Coarse-grain Activity Model of the Requirements Engineering Process:**

* Requirements engineering process is an example of coarse-grain activity model





**Fine-grain Activity Models:**

* These are more detailed models of a specific process, which are used for understanding and improving existing processes
* We’ll discuss some fine-grain processes within the general requirements engineering processes in later lectures

**Role-action Models:**

* These are models, which show the roles of different people involved in the process and the actions which they take
* They are useful for process understanding and automation

**Entity-relation Models:**

* The models show the process inputs, outputs, and intermediate results and the relationships between them
* They are useful in quality management systems

**References:**

* ‘Requirements Engineering: Processes and Techniques’ by G. Kotonya and I. Sommerville, John Wiley & Sons, 1998