**Unit 1**

**Systems Analysis and Design** is the process of investigating an existing system (often a manual system) and designing a new replacement system (usually a computerised system) to carry out the same functions. Systems projects are complex and can last for many months, or even years. People who work in this area are known as Systems Analysts or

Systems Designers. In these notes, we will use the term "Systems Analyst", or simply "Analyst" to refer to the person who carries out both Analysis and Design.

Analysts require various skills. They must have social and interpersonal skills such as diplomacy and the ability to put people at ease, as well as being technically competent in order to translate user requirements into successful computer-based systems.

Analysts often begin their career as Computer Programmers, before moving into Analysis. Some organisations employ Analyst Programmers, who do both Programming and Analysis. However, some analysts start from a functional area, such as Sales or Finance and are trained in Computing concepts before transferring to Analysis.

A systems project normally begins with **Terms of Reference** provided by the user.

These outline the scope of the system to be investigated and specify any constraints under which the analyst must work. For example, a new system may have to run on existing hardware, or may have restrictions on file formats to ensure compatibility with other systems. If the system is a complex one, a **Feasibility Study** may be carried out to ensure that it is worth developing a new system and draw up the terms of reference.

During our discussions of this area, you're likely to come across some unfamiliar terms, or familiar ones used in a different context. Some of these are listed below:

* **data:** the raw ‘facts’ gathered;
* **information:** data with some structure imposed upon it, e.g. by grouping items together into records and/or by sorting in some order;
* **system:** a group of procedures which operate together as a coherent whole to carry out a specified task. Systems may be either manual or computerised. A computerised system may incorporate manual elements and will certainly have hardware and software components;
* **sub-system:** a constituent part of a system which carries out some clearly defined subtask;
* **systems analysis:** the process of investigating and recording the operation of an existing system;
* **system specification:** a detailed description of the tasks required of a new system;
* **systems design:** a breakdown of the system specification into logical and physical components;
* **systems boundary:** the limits of the system;
* **interface:** the connections between the system and its environment or other related systems, e.g. the link between an order acceptance system and an invoicing system;
* **environment:** the area outwith the system boundary;
* **information flow:** the routes by which information passes between sub-systems or between a system and its environment.



**Relation of information systems to levels of organization**

A systems project can be divided up into the following stages:

#### Analysis

Once the scope of the study has been agreed, a detailed investigation is undertaken to determine the operations carried out by the current system and the requirements for the new system. This will involve speaking with the users of the current system and examining all the paperwork involved.

#### Design

Design is the process of specifying the new system. This details the tasks to be carried out and the data to be input, output and stored on files. Initially a **logical design** is produced, specifying the functions to be carried out by the new system without tying these down to specific hardware or software. Then a **physical design** is produced, giving precise details of hardware, software, file formats etc.

#### Implementation

Implementation includes software development (programming) and the changeover to the new system. This can take place all at once or in stages. Implementation also includes training users in the operation of the new system.

#### Testing

Testing includes **program testing**, to ensure that the individual programs work correctly in isolation and system or **integration testing** to ensure that the programs work together correctly as a complete system. The analyst carries out both these types of testing. The final stage is **acceptance testing**, which is carried out by the user to ensure that the system meets the specified requirements.

#### Maintenance

Maintenance ids the process of amending a system after it is up and running. This includes both ad-hoc amendments to deal with undiscovered bugs as well as planned maintenance to cope with changing circumstances within the organisation (e.g. addition of new product ranges) or the environment (e.g. changes in tax structure).

We will now look at each of these stages in a little more detail.

### Analysis

Analysis provides the foundation for the remainder of the systems analysis and design cycle. During the analysis phase, the analyst investigates the existing system and begins to form impressions of the new system. The principal task in the analysis phase is the gathering and analysis of information about the current system.

The systems analysis and design cycle does not necessarily progress in a linear fashion, with each completed leading directly the next phase. It is often necessary to go back to a previous phase if continuing to the next phase would result in improper analysis, design or implementation. It is common to return to the analysis phase several times before completing the design phase, because these two are closely related. This process of iteration and revision is normal in most systems projects.

#### Information Gathering

The most important source information is the end users of the current system, who are often also the potential users of the new system. They may range from novices to highly-skilled individuals. The information gathered from end users will be crucial during the analysis and design phases. Later, the analyst will also discuss technical aspects of the system with programmers, network engineers and other technical staff.

A secondary source of information for the analyst can be found in the existing paper work or documents within the organisation. Documents represent the formal information flow through the current system. The analyst must collect sample copies of all relevant documents, e.g. input forms, output documents, reports, invoices etc to understand how data flows and is used in the current system. This information can be important in the subsequent design of files for the new system.

All the information gathered by the analyst must be recorded in a suitable format. Most organisations have standards specifying how this should be done. This often amounts to the use of a standard set of forms, similar to those you will use during this course.

#### Information Gathering Techniques

The analyst will use a range of techniques to gather information about the current system. The most important are **interviewing, questionnaires** and **observation.**

#### Interviewing

Interviewing is the commonest and most effective technique. An interview is an exchange of information between the analyst and the end user. It is planned in advance and has a specific purpose.

There are two basic types of questions: open ended and closed ended. **Open-ended questions** are neutral and non-restrictive. They allow interviewees to answer questions in any way they wish, and they encourage them to reveal information. For example:

‘How could the invoicing system could be improved?’

However, this can lead to the interview being controlled by the interviewee's answers rather than the interviewer's questions. Open-ended questions can sometimes result in the disclosure of irrelevant information.

**Closed-ended questions** are specific and provide the interviewer with greater control over the interview. However, they achieve only what they ask and discourage interviewees from opening up and revealing relevant information which the interviewer did not anticipate. For example:

‘What part of the invoicing system takes up the most time?’

The interviewer must take care that closed-ended questions are not loaded or leading.

Questions can be sub-divided into two categories: primary and secondary; both can be open or closed-ended. **Primary Questions** address a specific topic. **Secondary Questions** are follow-up questions designed to obtain more information than was given in response to a Primary question.

#### Questionnaires

Questionnaires allow the analyst to collect information from a large number of people, possibly spread over several sites. Standardised question formats can yield more reliable data than other fact-finding techniques, and wide distribution ensures that respondents remain anonymous. This can lead to more honest responses.

However, questionnaires don't let the analyst see the expressions or reactions of respondents. Respondents may not complete questionnaires as a high-priority task. If everyone doesn't reply, the respondents can become a self-selected group, which can lead to problems with data reliability.

Open-ended questionnaires allow people to express their feelings, opinions and experiences or explore a problem. Closed-ended questionnaires provide greater control by presenting respondents with specific responses to choose from. This format is excellent for obtaining factual information.

Questionnaires are expensive to develop and distribute. Analysts must consider the objectives of the questionnaire and determine what structure will be most useful and easiest to understand. Questionnaires should be tested before being printed and

distributed.

**SDLC**

The Systems Development Life Cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved including the wate rfall model (the original SDLC method), rapid application development (RAD), joint application development (JAD), the fountain model and the spiral model. Mostly, several models are combined into some sort of hybrid methodology. Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely particular plan was followed.

**Feasibility :** The feasibility study is used to determine if the project should get the go-ahead. If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development.

**Requirement Analysis and Design** : Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

**Implementation** : In this phase the designs are translated into code. Computer programs are written using a conventional programming language

or an application generator. Programming tools like Compilers, Interpreters, Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, Java are used for coding. With respect to the type of application, the right programming language is chosen.

**Testing :** In this phase the system is tested. Normally programs are written as a series of individual modules, these are subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

**Maintenance :** Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

### Explain the Waterfall Model.

Sometimes called the *classic life cycle or the linear sequential model,* the *waterfall model* is a systematic, sequential approach to software development in which development is seen as flowing downwards ( like a waterfall ) that begins at the system level and progresses through analysis, design, coding, testing and support. To follow the waterfall model, one proceeds from one phase to the next in a sequential manner. For example, one first completes "requirements specification". When the requirements are fully completed, one proceeds to design. The software is designed (on paper) and this design should be a plan

for implementing the requirements given. When the design is fully completed, an implementation of that design, i.e. coding of the design is made by programmers. After the implementation phases are complete, the software product is tested and debugged; any faults introduced in earlier phases are removed here. Then the software product is installed, and later maintained to add any new functions that the user needs and remove bugs. Thus in a waterfall model, we can move to the next step only when the previous step is completed and removed of all errors. There is no jumping back and forth or overlap between the steps in a waterfall model.

Requirements

Analysis

Design

Information Modeling

Delivery & Support

The model consists of six distinct stages, namely :

Code Generation

Testing

* + 1. In the *Information Modelling* phase
			1. Work begins by gathering information related to the existing system. This will consists of all items consisting of hardware, people, databases etc.
		2. In the *requirements analysis* phase
			1. The problem is specified along with the desired objectives (goals).
			2. The constraints are identified.
			3. All information about the functions, behaviour, and performance are documented and checked by the customers.
		3. In the *design phase*, all inputs, computations and outputs of the system should be converted into a software model so that it can be coded by programmers. The hardware requirements are also determined at this stage along with a picture of the overall system architecture.
		4. In the *code generation* phase, the design has to be translated into a machine-readable form using any of the programming languages available that is suitable for the project.
		5. In the *testing* phase stage
			1. Once code is generated, testing begins.
			2. It focuses on all the statements of the software and removes all errors.
			3. It ensures that proper input will produce actual results.
			4. Detailed documentation from the design phase can significantly reduce the coding effort.
		6. The *delivery and support* phase consists of delivering the final product to the customer and then taking care of the maintenance of the product. In this phase the software is updated to :
			1. Meet the changing customer needs
			2. Adapted to accommodate changes in the external environment
			3. Correct errors that were not previously known in the testing phases
			4. Enhancing the efficiency of the software

### Explain the Prototyping Process Model.

**Ans.:** The prototyping model begins with the requirements gathering. The developer and the customer meet and define the objectives for the software, identify the needs, etc. A „quick design‟ is then created. This design focuses on those aspects of the software that will be visible to the customer. It then leads to the construction of a prototype. The prototype is then checked by the customer and any modifications or changes that are required are made to the prototype. Looping takes place in this process and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continues till the user is satisfied with the system. Once a user is satisfied, the prototype is

converted to the actual system with all considerations for quality and security.

The prototype is considered as the „first system‟. It is advantageous because both the customers and the developers get a feel of the actual system. But there are certain problems with the prototyping model too.

* + 1. The prototype is usually created without taking into consideration overall software quality.
		2. When the customer sees a working model in the form of a prototype, and then is told that the actual software is not created, the customer can get irritated.
		3. Since the prototype is to be created quickly, the developer will use whatever choices he has at that particular time (eg, he may not know a good programming language, but later may learn. He then cannot change the whole system for the new programming language). Thus the prototype may be created with less-than-ideal choices.

### Describe the Rapid Application Development Model. State its disadvantages.

**Ans.:** Rapid Application Development (RAD) is an incremental software development process model that focuses on a very short development cycle. The RAD model is a „high-speed‟ version of the linear sequential model. It enables a development team to create a fully functional system within a very short time period (e.g. 60 to 90 days).

**Business Modeling :** The information flow among business functions is modeled in a way that answers the following questions :

What information drives the business process? What information is generated?

Who generates it?

Where does the information go? Who processes it?

**Data Modeling :** It gives all the details about what data is to be used in the project. All the information found in the business modeling phase is refined into a set of data objects and the characteristics and the relationships between these objects are defined.

**Process Modeling :** Here all the processes are defined that are needed to use the data objects to create the system. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

**Application Generation :** RAD makes use of the fourth generation techniques and tools like VB, VC++, Delphi etc rather than creating software using conventional third generation programming languages. The RAD reuses existing program components (when possible) or creates reusable components (when necessary). In all cases, automated tools (CASE tools) are used to facilitate construction of the software.

**Testing and Turnover :** Since the RAD process emphasizes reuse, many of the program components have already been tested. This minimizes the testing and development time.

If a business application can be divided into modules, so that each major function can be completed within the development cycle, then it is a candidate for the RAD model. In this case, each team can be assigned a model, which is then integrated to form a whole.

### Disadvantages :

* For Large projects, RAD requires sufficient resources to create the right number of RAD teams.
* If a system cannot be properly divided into modules, building components for RAD will be problematic
* RAD is not appropriate when technical risks are high, e.g. this occurs when a new application makes heavy use of new technology.

### Explain the Spiral Model. What are the advantages of this model?

**Ans.:** The spiral model, combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model, therein providing the potential for rapid development of incremental versions of the software. In this model the software is developed in a series of incremental releases with the early stages being either paper models or prototypes. Later iterations become increasingly more complete versions of the product.



As illustrated, the model is divided into a number of task regions.

These regions are :

* + 1. The **customer communication** task – to establish effective communication between developer and customer.
		2. The **planning** task – to define resources, time lines and other project related information..
		3. The **risk analysis** task – to assess both technical and management risks.
		4. The **engineering** task – to build one or more representations (prototypes) of the application.
		5. The **construction and release** task – to construct, test, install and provide user support (e.g., documentation and training).
		6. The **customer evaluation** task – to obtain customer feedback based on the evaluation of the software representation created during the engineering stage and implemented during the install stage.

The evolutionary process begins at the centre position and moves in a clockwise direction. Each traversal of the spiral typically results in a deliverable. For example, the first and second spiral traversals may result in the production of a product specification and a prototype, respectively. Subsequent traversals may then produce more sophisticated versions of the software.

An important distinction between the spiral model and other software models is the explicit consideration of risk. There are no fixed phases such as

specification or design phases in the model and it encompasses other process models. For example, prototyping may be used in one spiral to resolve requirement uncertainties and hence reduce risks. This may then be followed by a conventional waterfall development.

### Advantages of the Spiral Model :

 The spiral model is a realistic approach to the development of large - scale software products because the software evolves as the process progresses. In addition, the developer and the client better understand and react to risks at each evolutionary level.

 The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.

 It maintains a systematic stepwise approach, like the classic life cycle model, but incorporates it into an iterative framework that more reflect the real world.

 If employed correctly, this model should reduce risks before they become problematic, as consideration of technical risks

are considered at all stages.

# ITERATIVE AND INCREMENTAL DEVELOPMENT MODEL:

**Iterative and Incremental development** is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between.

Iterative and incremental development is essential parts of the Rational Unified Process, Extreme Programming and generally the various agile software development frameworks.

## Diagram : An iterative development model



* + 1. **Iterative/Incremental Development**

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The unified process groups increments/iterations into phases: inception, elaboration, construction, and transition.

* Inception identifies project scope, risks, and requirements (functional and non-functional) at a high level but in enough detail that work can be estimated.
* Elaboration delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
* Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
* Transition delivers the system into the production operating environment

Diagram: **Iterative/Incremental Development**

### Explain Information Gathering Process for System Development. OR

**Explain Fact Finding Method of System Analysis.**

**Ans.:** Fact finding means learning as much as possible about the present system.

The tools used in information gathering or fact finding are

* + 1. **Review of Written Documents :** In all organizations documents such as forms, records, reports, manuals, etc are available. These help in determining how the present system runs. The process of fact finding includes collection of all possible documents and evaluating them. Unfortunately, most manuals are not up to date and may not be readable. The analyst needs to find out how the forms are filled out, what changes need to be made and how easy they are to read.
		2. **On-Site Observation :** The purpose of on-site observation is to get as close as possible to the real system being studied. It is the process of recognizing and noting people, objects and occurrences to obtain information. As an observer the analyst must follow a set of rules. He/she must listen than talk and not give advice or pass a moral

judgment, must not argue or show friendliness towards others. The following questions can serve as a guide for on-site observations:

* What kind of system is it? What does it do?
* Who runs the system? Who are the important people in it?
* What is the history of the system?
	+ 1. **Interviews :** An interview is a face to face interpersonal situation in which a person called the interviewer asks a person being interviewed, questions designed to gather information about a problem. The analyst or interviewer can schedule interviews with key personnel of the organization. The analyst also needs to conduct detailed interviews with all the people who will actually use the system. This will provide all the details the analyst needs and also remove any fear from the users that the computers will replace the. Interviews help gather vital facts about the existing problems, such as lack of quality control or security, etc. Interviewing needs a friendly atmosphere so that the interviewer can ask questions properly, obtain reliable and correct answers and record the answers accurately and completely.
		2. **Questionnaires :** A questionnaire is a tool that has questions to which individuals respond. A questionnaire has the following advantages:
* It is economical and requires less skill than an interview.
* It can be used to gather data from large number of people simultaneously
* It is a uniform method in which all question asked are the same to all people
* The users are happy as they know that the answers they give are confidential
* User get time to think about the questions and so can give more accurate results than in an interview

### What is Feasibility? Describe the different types of Feasibility.

**Ans.:** Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility study. A feasibility study is carried out to select the best system that meets performance requirements. When conducting feasibility study, an analyst can consider 7 types of feasibility:

 **Technical Feasibility :** It is concerned with specifying the equipment and the computer system that will satisfy and support the proposed user requirements. Here we need to consider the configuration of the system which tells the analyst how many work stations are required, how the units are interconnected so that they can operate and communicate smoothly.

 **Operation Feasibility :** It is related to human organizational aspects. The points to be considered here are – what changes will be brought with the system?, what new skills will be required?, do the existing staff members have these skills and can they be trained?

 **Economic Feasibility :** It is the most frequently used technique for evaluating a proposed system. It is also called Cost/Benefit Analysis. It is used to determine the benefits and savings that are expected from the proposed system and compare them with the costs. If benefits are more than the cost, the proposed system is given an OK.

 **Social Feasibility :** It is a determination of whether the proposed system will be acceptable to the people or not. It finds out the probability of the project being accepted by the group of people who are directly affected by the changed system.

 **Management Feasibility :** It is a determination of whether the proposed system is acceptable to the management of the organization. The project may be rejected, if the management does not accept the proposed system.

 **Legal Feasibility :** It is a determination of whether the proposed project is under legal obligation of known Acts, Statutes, etc.

 **Time Feasibility :** It is a determination of whether the project will be completed within a specified time period. If the project takes too much time, it is likely to be rejected.

### System Analyst

The system analyst is overall responsible for the development of a software. He is the crucial interface between users, programmers and MIS managers. He conducts a system’s study, identifies activities and objectives and determines a procedure to achieve the objective. He has a very important role in the development of a system. The concerned person should also have some special qualities which we are going to discuss in this unit.

A Systems analyst is a person who is overall responsible for development of a software. He is the computer professional charged with analyzing, designing and implementing computer- based information systems. He is the crucial interface among users, programmers and MIS managers. A Systems analyst can be defined as follows:

A Systems analyst is a computer specialist who translates business problems and requirements into information systems and acts as liaison between IS (Information Systems) department and rest of the organization.

The analyst conducts a systems study, identifies activities and objectives and determines a procedure to achieve the objectives. He is the key member of both MIS organization and the software project team. He is a person with unique skills, experience, personality and common sense. His role has been emerging with advances in technology.

#### Roles of a Systems Analyst

The Systems analyst performs the following roles during various phases of SDLC. He works as a:

1. ***Problem Investigator:*** The analyst studies the problems and needs of an organization during feasibility and requirements analysis phases of SDLC. He visits the various departments of the organization and interviews the users. He analyses the problems of the current system and collects their new requirements. The analyst initially works as an investigator by extracting the real problems of the users.
2. ***Problem Solver:*** The analyst solves the problems of the current system faced by the users. He determines how people, method and technology can improve the current system. After feasibility analysis, he presents the system proposal to the management.
3. ***Systems Designer:*** The analyst creates a detailed physical (current) and logical (proposed) design of the system.
4. ***Motivator:*** The analyst motivates users to participate in development and implementation of the proposed system. This helps to understands user’s feelings about the proposed system. The analyst interprets the thoughts of users and hence, draws conclusions. He appeals management and users for getting the support in development and implementation of the proposed system.
5. ***Project Manager:*** The analyst monitors the development and implementation of software in relation to quality, cost and time. He works with the project leader for managing the project properly. For development of small systems, the Systems analyst is generally the project leader.

#### Qualities of Systems Analyst

Success in systems analysis requires interpersonal and technical skills of the analyst. The systems analyst is expected to possess the following qualities:

1. ***Qualified:*** The analyst must be highly qualified in software technology. Besides software, he should have a good knowledge of hardware and latest communication and networking technology. He must have a thorough awareness about the working (manual and computerized) of financial accounting, sales and marketing, invoicing, inventory control, production and other information systems of different organization.
2. ***Analytical Thinker:*** The analyst must be capable to extract real problems of the users by analyzing the existing system. He is expected to provide the best solutions to the problems. He should be able to provide more than one solution to a single problem so that the users can select the best one. The systems analyst must be capable of tackling any problem of the user. He must be a problem solver and not a problem creator.
3. ***Good Communicator:*** The analyst must have a good communication and presentation skills. He must have an excellent command on the language which the user can understand. There should not be any communication gap between the systems analyst and users.

**Software prototyping**, an activity during certain software development, is the creation of prototypes, i.e., incomplete versions of the software program being developed.

A prototype typically simulates only a few aspects of the features of the eventual program, and may be completely different from the eventual implementation.

The **conventional** purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Prototyping can also be used by end users to describe and prove requirements that developers have not considered, so "controlling the prototype" can be a key factor in the commercial relationship between developers and their clients.

## Prototyping Process:

The process of prototyping involves the following steps

## Identify basic requirements

Determine basic requirements including the input and output information desired. Details, such as security, can typically be ignored.

## Develop Initial Prototype

The initial prototype is developed that includes only user interfaces Review The customers, including end-users, examine the prototype and provide feedback on additions or changes.

## Revise and Enhance the Prototype

Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the contract/product may be necessary.

## Advantages of prototyping:

There are many advantages to using prototyping in software development – some tangible, some abstract.

**Reduced time and costs**: Prototyping can improve the quality of requirements and specifications provided to developers. Because changes cost exponentially more to implement as they are detected later in development, the early determination of *what the user really wants* can result in faster and less expensive software.

**Improved and increased user involvement**: Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications. The presence of the prototype being examined by the user prevents many misunderstandings and miscommunications that occur when each side believe the other understands what they said. Since users know the problem domain better than anyone on the development team does, increased interaction can result in final product that has greater tangible and intangible quality. The final product is more likely to satisfy the users desire for look, feel and performance.

## Disadvantages of prototyping:

**Insufficient analysis**: The focus on a limited prototype can distract developers from properly analyzing the complete project. This can lead to overlooking better solutions, preparation of incomplete specifications or the conversion of limited prototypes into poorly engineered final projects that are hard to maintain. Further, since a prototype is limited in functionality it may not scale well if the prototype is used as the basis of a final deliverable, which may not be noticed if developers are too focused on building a prototype as a model.

**User confusion of prototype and finished system**: Users can begin to think that a prototype, intended to be thrown away, is actually a final system that merely needs to be finished or polished. (They are, for example, often unaware of the effort needed to add

error-checking and security features which a prototype may not have.) This can lead them to expect the prototype to accurately model the performance of the final system when this is not the intent of the developers. Users can also become attached to features that were included in a prototype for consideration and then removed from the specification for a final system. If users are able to require all proposed features be included in the final system this can lead to conflict.

**Developer misunderstanding of user objectives**: Developers may assume that users share their objectives (e.g. to deliver core functionality on time and within budget), without understanding wider commercial issues. For example, user representatives attending Enterprise software (e.g. PeopleSoft) events may have seen demonstrations of "transaction auditing" (where changes are logged and displayed in a difference grid view) without being told that this feature demands additional coding and often requires more hardware to handle extra database accesses. Users might believe they can demand auditing on every field, whereas developers might think this is feature creep because they have made assumptions about the extent of user requirements. If the developer has committed delivery before the user requirements were reviewed, developers are between a rock and a hard place, particularly if user management derives some advantage from their failure to implement requirements.

**Developer attachment to prototype:** Developers can also become attached to prototypes they have spent a great deal of effort producing; this can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture. (This may suggest that throwaway prototyping, rather than evolutionary prototyping, should be used.)

**Excessive development time of the prototype**: A key property to prototyping is the fact that it is supposed to be done quickly. If the developers lose sight of this fact, they very well may try to develop a prototype that is too complex. When the prototype is thrown away the precisely developed requirements that it provides may not yield a sufficient increase in productivity to make up for the time spent developing the prototype. Users can become stuck in debates over details of the prototype, holding up the development team and delaying the final product.

**Expense of implementing prototyping**: the start up costs for building a development team focused on prototyping may be high. Many companies have development methodologies in place, and changing them can mean retraining, retooling, or both. Many

companies tend to just jump into the prototyping without bothering to retrain their workers as much as they should.

# QUESTIONAIRE:

A **questionnaire** is a research instrument consisting of a series of questions and other prompts for the purpose of gathering information from respondents. Although they are often designed for statistical analysis of the responses, this is not always the case. The questionnaire was invented by Sir Francis Galton.

Questionnaires have advantages over some other types of surveys in that they are cheap, do not require as much effort from the questioner as verbal or telephone surveys, and often have standardized answers that make it simple to compile data. However, such standardized answers may frustrate users. Questionnaires are also sharply limited by the fact that respondents must be able to read the questions and respond to them. Thus, for some demographic groups conducting a survey by questionnaire may not be practical.

## Question Construction:

**Question types**

Usually, a questionnaire consists of a number of questions that the respondent has to answer in a set format. A distinction is made between open-ended and closed-ended questions. An open- ended question asks the respondent to formulate his own answer, whereas a closed-ended question has the respondent pick an answer from a given number of options. The response options for a closed-ended question should be exhaustive and mutually exclusive. Four types of response scales for closed-ended questions are distinguished:

* + - * Dichotomous, where the respondent has two options
			* Nominal-polytomous, where the respondent has more than two unordered options
			* Ordinal-polytomous, where the respondent has more than two ordered options
			* (Bounded)Continuous, where the respondent is presented with a continuous scale

## Basic rules for questionnaire item construction

* + - * Use statements which are interpreted in the same way by members of different subpopulations of the population of interest.
			* Use statements where persons that have different opinions or traits will give different answers.
			* Think of having an "open" answer category after a list of possible answers.
			* Use only one aspect of the construct you are interested in per item.
			* Use positive statements and avoid negatives or double negatives.
			* Do not make assumptions about the respondent.
			* Use clear and comprehensible wording, easily understandable for all educational levels
			* Use correct spelling, grammar and punctuation.
			* Avoid items that contain more than one question per item (e.g. Do you like strawberries and potatoes?)

## Questionnaire administration modes:

Main modes of questionnaire administration are:

* + - * Face-to-face questionnaire administration, where an interviewer presents the items orally.
			* Paper-and-pencil questionnaire administration, where the items are presented on paper.
			* Computerized questionnaire administration, where the items are presented on the computer.
			* Adaptive computerized questionnaire administration, where a selection of items is presented on the computer, and based on the answers on those items, the computer selects following items optimized for the tester’s estimated ability or trait.

# JAD SESSIONS:

JAD (Joint Application Development) is a methodology that involves the client or end user in the design and development of an application, through a succession of collaborative workshops called JAD sessions. Chuck Morris and Tony Crawford, both of IBM, developed JAD in the late 1970s and began teaching the approach through workshops in 1980.

The JAD approach, in comparison with the more traditional practice, is thought to lead to faster development times and greater client satisfaction, because the client is involved throughout the development process. In comparison, in the traditional approach to

systems development, the developer investigates the system requirements and develops an application, with client input consisting of a series of interviews.

**Joint Application Design** (JAD) is a process used in the prototyping life cycle area of the Dynamic Systems Development Method (DSDM) to collect business requirements while developing new information systems for a company. "The JAD process also includes approaches for enhancing user participation, expediting development, and improving the quality of specifications." It consists of a workshop where “knowledge workers and IT specialists meet, sometimes for several days, to define and review the business requirements for the system” The attendees include high level management officials who will ensure the product provides the needed reports and information at the end. This acts as “a management process which allows Corporate Information Services (IS) departments to work more effectively with users in a shorter time frame.”

Through JAD workshops the knowledge workers and IT specialists are able to resolve any difficulties or differences between the two parties regarding the new information system. The workshop follows a detailed agenda in order to guarantee that all uncertainties between parties are covered and to help prevent any miscommunications. Miscommunications can carry far more serious repercussions if not addressed until later on in the process. (See below for Key Participants and Key Steps to an Effective JAD). In the end, this process will result in a new information system that is feasible and appealing to both the designers and end users.

"Although the JAD design is widely acclaimed, little is actually known about its effectiveness in practice." According to Journal of Systems and Software, a field study was done at three organizations using JAD practices to determine how JAD influenced system development outcomes. The results of the study suggest that organizations realized modest improvement in systems development outcomes by using the JAD method. JAD use was most effective in small, clearly focused projects and less effective in large complex projects.

Joint Application Design (JAD) was developed by Drake and Josh of IBM Raleigh and Tony Crawford of IBM Toronto in a workshop setting. Originally, JAD was designed to bring system developers and users of varying backgrounds and opinions together in a productive as well as creative environment. The meetings were a way of obtaining quality requirements and specifications. The structured approach provides a good alternative to traditional serial interviews by system analysts.

Brain-storming and theory Z principals in JAD: In 1984-5 Moshe Telem of Tel-Aviv University, developed and implemented a JAD conceptual approach that integrates brainstorming and Ouchi's "Japanese Management" theory Z principles for rapid, maximal and attainable requirements analysis through JAD. Telem named his approach Brainstorming a Collective Decision-Making Approach (BCDA) [4]. Telem also developed and implemented a BCDA technique (BCDT)[5], which was successfully used within the setting of a pedagogical management information system project for the Israeli educational system[3]. In this project brainstorming and theory Z principles in JAD proved to be not only feasible but also effective, resulting in a realistic picture of true users' information requirements.

## Conduct Jad sessions :

Joint Application Development or JAD as it is commonly known as a process originally developed for designing a computer based system. JAD focuses on the use of highly structured, well planned meetings to identify the key components of system development projects.

The JAD process is based on four simple ideas;

* + - * people who actually do a job have the best understanding of the job
			* People who are trained in information technology have the best understanding of the possibilities of that technology.
			* Information systems never exist alone
			* The best results are obtained when all these groups work together on a project.

The JAD technique is based on the observation that the success of a project can be hampered by poor intra team communication, incomplete requirements definition and lack of consensus. The training teaches the essential skills and techniques need to plan, organize and participate in JAD planning.

AD focuses on the use of highly structured, well planned meetings to identify the key components of system development projects. JAD centers on a structured workshop session. It eliminates many problems with traditional meetings which are like workshops. The sessions are

1. very focused
2. conducted in a dedicated environment
3. quickly drive major requirements

The participants include: facilitator, end users, developers, tie breakers, observers and subject matter experts. The success of JAD-based workshop depends on the skill of the facilitators.

## Need for to Conduct JAD sessions:

Everybody who is responsible for gathering requirements and developing business systems should attend the JAD training sessions. They are: workshop facilitators, business analysts, system analysts, process analysts, development project leaders, development team members, business managers and Information technology members.

## Advantages and Disadvantages

JAD is more expensive and cumbersome, compared to other traditional methods. Many companies find that JAD users participate freely in requirements modeling process. They feel a sense of ownership and support for the new system. One big disadvantage is that it opens up a lot of scope for interpersonal conflict.

Compared with traditional methods, JAD may seem more expensive and can be cumbersome if the group is too large relative to the size of the project. Many companies find, however, that JAD allows key users to participate effectively in the requirements modeling process. When users participate in the systems development process, they are more likely to feel a sense of ownership in the results, and support for the new system. When properly used, JAD can result in a more accurate statement of system requirements, a better understanding of common goals, and a stronger commitment to the success of the new system.

**Joint Application Design** (JAD) is a process used in the prototyping life cycle area of the Dynamic Systems Development Method (DSDM) to collect business requirements while developing new information systems for a company. "The JAD process also includes approaches for enhancing user participation, expediting development, and improving the quality of specifications." It consists of a workshop where “knowledge workers and IT specialists meet, sometimes for several days, to define and review the business requirements for the system.[1]” The attendees include high level management officials who will ensure the product provides the needed reports and information at the end. This acts as “a management process which allows Corporate Information Services (IS) departments to work more effectively with users in a shorter time frame.

Through JAD workshops the knowledge workers and IT specialists are able to resolve any difficulties or differences between the two parties regarding the new information system. The workshop follows a detailed agenda in order to guarantee that all uncertainties between parties are covered and to help prevent any miscommunications. Miscommunications can carry far more serious repercussions if not addressed until later on in the process. (See below for Key Participants and Key Steps to an Effective JAD). In the end, this process will result in a new information system that is feasible and appealing to both the designers and end users.

"Although the JAD design is widely acclaimed, little is actually known about its effectiveness in practice." According to Journal of Systems and Software, a field study was done at three organizations using JAD practices to determine how JAD influenced system development outcomes. The results of the study suggest that organizations realized modest improvement in systems development outcomes by using the JAD method. JAD use was most effective in small, clearly focused projects and less effective in large complex projects.

## Four Principle Steps :

1. **Define session objectives-** The first step for the facilitator together with the project leader is to define the session objectives and answering the questions as to what are the session objectives? What is wanted from the session? Who can help create the deliverables?
2. **Prepare for the session-** The facilitator has primary responsibility for the JAD preparation. Four categories of tasks are involved in preparing for the session.
	* Conduct pre-session research
	* Create a session agenda
	* Arrange session logistics
	* Prepare the participants



1. **Conduct the JAD session-** The facilitator conducts the JAD session, leading the developers and customers through planned agenda. Conducting the meeting involves:
* Starting and ending time,

-Distributing and following the meeting agenda

-Gaining consensus on the meeting purpose and round rules at the beginning of the meeting

-Keeping the meeting on track.

1. **Procedure the Documents-** It is critical to the success of any JAD session that the information on flip-charts, foils, whiteboard, and discussions be recorded and reviewed by the participants. Each day of the session, the facilitator and scribe should create a draft of the day’s results. The final documents from the JAD should be completed as soon as possible after the session. It is primary responsibility of the facilitator and the scribe to:
* organize the final document for easy use by project members
* complete a "Final Draft" document
* distribute it to selected individuals for review
* incorporate revisions as necessary
* distribute the final copy for participant sign-off

JAD improves the final quality of the product by keeping the focus on the upfront of the development cycle thus reducing the errors that are likely to cause huge expenses.

# VALIDATION:

**Validation** may refer to:

* Validity, in logic, determining whether a statement is true
* Validation and verification, in engineering, confirming that a product or service meets the needs of its users
* Verification and Validation (software), checking that a software system meets specifications and fulfills its intended purpose
* Validation of foreign studies and degrees, processes for transferring educational credentials between countries
* Validation (computer security), the process of determining whether a user or computer program is allowed to do something.

o Validate (McAfee), a software application for this purpose

* Validation (drug manufacture), documenting that a process or system meets its pre-determined specifications and quality attributes
* Validation (psychology), in psychology and human communication, the reciprocated communication of respect which signifies that the other's opinions are acknowledged, respected and heard
* Data validation, in computer science, ensuring that data inserted into an application satisfies defined formats and other input criteria
* Regression model validation, in statistics, determining whether a model fits the data well

# STRUCTURED WALKTHROUGHS:

In typical project planning, you must define the scope of the work to be accomplished. A typical tool that is used by project managers is the work breakdown structure (WBS). This walkthrough demonstrates a general approach to creating a WBS using Team Foundation Server and Microsoft Project.

This walkthrough is not based on any particular methodology. However, it does use the quality of service requirement and task work item types in the MSF for Agile Software Development process template. The approach used in this walkthrough should be adaptable your own organization's work item types and process.

In this walkthrough, you will complete the following tasks:

* Create a requirement using Team Foundation Server.
* Create tasks using Team Foundation Server.
* Create tasks using Microsoft Project.
* Link tasks and requirements.
* Create a work breakdown structure from tasks in Microsoft Project

The term is often employed in the software industry (see software walkthrough) to describe the process of inspecting algorithms and source code by following paths through the algorithms or code as determined by input conditions and choices made along the way. The purpose of such *code walkthroughs* is generally to provide assurance of the fitness for purpose of the algorithm or code; and occasionally to assess the competence or output of an individual or team.

## Types of Walkthroughs

* + - * Specification walkthroughs
				+ System specification
				+ Project planning
				+ Requirements analysis
			* Design walkthroughs
				+ Preliminary design
				+ Design
			* Code walkthroughs
			* Test walkthroughs
				+ Test plan
				+ Test procedure

## Prerequisites

The following prerequisites must be met to complete this walkthrough.

* + - * Microsoft Project must be installed.
			* A team project must be created that uses the MSF for Agile Software Development process template.

## Scenario

The scenario for this walkthrough is based on the example Adventure Works team project. Adventure Works is starting a project to set up a Web interface for ordering its products. One of the customer requirements states that customers be able to check on order status after orders are placed. The scope of this work must be defined in a work breakdown structure to a sufficient level of detail to enable project planning to be completed.

The following approach is used by Adventure Works. The project manager must create the WBS and has the help of the team to do this. One person on the team is a database expert and will provide details on what is needed in the database to support the new requirement. She will enter her work details using Team Foundation Server.

The project manager will work with other team members to define additional work to complete the Web interface. Then the project manager will enter those details using Microsoft Project.

Finally, the project manager will create a WBS in Microsoft Visio that can be used in the project planning document.

Throughout this walkthrough you will perform the steps of each role to create the tasks and WBS. When you complete the walkthrough, you will have created the following tasks and subtasks in a Gantt chart and a WBS.

* + - * Order Storage Subsystem
				+ Order Tables
				+ Order Stored Procedures
			* Order Web Interface
				+ Order Lookup Web Service
				+ Client Order Views

 term **"software quality metrics"** illustrate the picture of measuring the software qualities by recording the number of defects or security loopholes present in the software. However, quality measurement is not restricted to counting of defects or vulnerabilities but also covers other aspects of the qualities such as maintainability, reliability, integrity, usability, customer satisfaction, etc.

## Why Software Quality Metrics?



1. To define and categorize elements in order to have better understanding of each and every process and attribute.
2. To evaluate and assess each of these process and attribute against the given requirements and specifications.
3. Predicting and planning the next move w.r.t software and business requirements.
4. Improving the Overall quality of the process and product, and subsequently of project.

## What are the quality factors that define & impact the user experience?



## Software Quality Metrics: sub-category of Software Metrics

It is basically, a subclass of **software metrics** that mainly emphasizes on quality assets of the software product, process and project. Software metric is a broader concept that incorporates software quality metrics in it, and mainly consists of three types of metrics:



* **Product Metrics:**

It includes size, design, complexity, performance and other parameters that are associated with the product's quality.

* **Process Metrics:**

It involves parameters like, time-duration in locating and removing defects, response time for resolving issues, etc.

* **Project Metrics:**

It may include number of teams, developers involved, cost and duration for the project, etc.



## Methodology Of Software Quality Metrics

The methodology behind software quality metric is as under:

* Identify and prepare the list of possible requirements of quality, and subsequently, assigning direct metric, such as understanding, learning and operation time, to each of these requirements.
* Apply metrics framework, along with the cost-benefit analysis.
* Implementing metrics via collecting and defining data to compute metric values.
* Interpret and analyse the results, to ensure the fulfilment of requirements.
* Validate the metrics through validation methodology and thereafter proper documentation of the results.

## Features of good Software Quality Metrics

* Should be specific to measure the particular attribute or an attribute of greater importance.
* Comprehensive for wide variety of scenarios.
* Should not consider attributes that have already been measured by some other metric.
* Reliable to work similarly in all conditions.
* Should be easy and simple to understand and operate.

**RISK MANAGEMENT SYSTEM**

Risk is an inexorable and an unavoidable part of a software development process, which constantly evolves throughout the course of a project, affecting a project or software or both. Thus, it arise the necessity to deal and manage these risks in an efficient and effective manner.

In the field of software engineering, risk management is a methodology or a mechanism, carried out throughout the development process to identify, manage and control risks evolved before and during the development process.

Basically, three types of activities are covered under the risk management process.



* Risk Identification.
* Risk Analysis.
* Risk Control.

**You're All Set!**

**Please Check Your Email For Our Weekly Updates!**



## Risk Identification:

It is the first step of a risk management process, which involves the identification of potential risks that may affect a software product or a development project, and accordingly documenting them along with their characteristics.

It is a constant process, which is carried out throughout the development due to the fact that as the development process progresses, the more we get to know about the software product and based on it, we may able to explore and identify more unvisited or hidden risks.

Generally, this phase helps in identifying the two types of risks, product risk and project risk.

* [**Product risk**](https://www.professionalqa.com/product-risk): Risks pertaining to a software product or application, which may arise, due to its inefficiency, to function, desirably, to meet the expectation of the users.
* [**Project risk**](https://www.professionalqa.com/project-risk):These risks involves any sort of uncertain or unexpected event or action, which may likely to occur and degrade the progress of a project.

In this phase, usually client, stakeholders, business manager, project manager and test manager, collaborate and participates in brainstorming or small sessions, study and analyze the project documentation plan, etc., to make out the probable list of risks associated with the software development. Some commonly known techniques to identify risks may include risk templates, project retrospective, Failure Mode and Effect Analysis (FMEA), Failure Mode Effect and Criticality Analysis (FMECA), etc.

## Risk Assessment:

The next stage of a risk management process is risk analysis, which involves the assessment of the risks identified during the risk identification stage.

* This stage usually involves the analysis and prioritization of the risks, i.e. possible outcomes of each identified risk is being assessed based on which risks are categorized and accordingly, prioritized.
* Based on the degree of impact, possessed by each risk, they are being assigned severity levels, namely 'High', 'Medium' and 'low'. And based on their severity, they are prioritize i.e. High risks are considered as top priority whereas the low risk is regarded for the bottom most priority.

## Risk Control:

During this stage, risks are managed, controlled and mitigated, based on their priority so as to achieve the desired results. It is generally divided into three activities which may be seen below.

* **Risk Management Planning:** It involves a proper and effective plan to deal with the each identified risk.
* **Risk Resolution:**It refers to the execution of the plans, outlined during the risk management planning stage so as to either remove or fix identified risks.
* **Risk Monitoring:** It involves, regular monitoring and tracking, the development progress, in the direction, of resolving risk issues, which may include revaluation of the risks, their likelihood to occur, etc., and taking and implementing necessary & appropriate actions, wherever necessary.