Course Title: - Management Information Systems **Course No: - MCA-209-EA**

Unit 11

Need for System Analysis:

- In any type of organization, analysis of the existing system before computerizing is very important. This analysis helps the designer to define and design the new system efficiently.

The following are the important needs of system analysis.

It helps in defining system objectives:

- Generally most of the year old organization don't have any concrete objectives. This is because they are historical in operation.
- Therefore the employees are not in a position to define the objectives. Therefore system analysis is very important to define the objectives of the system.

It helps to identify the system boundaries:

- Each and every organization has number of sub systems. Normally the people employed in each sub system do not know their actual responsibilities, input for their system and the output produced.
- But system analysis helps the people to understand their responsibilities, limitations, inputs and the ouput.

It helps to understand the system importance:

- System analysis helps the designer to understand the importance of each and every system. So the designer can place the system in their correct position to achieve the objectives of the system.

It helps to decide the nature of the system:

- Design of a new system depends mainly on the nature of the existing system. System analysis helps the designer to understand whether the existing system is of open, closed, deterministic or probabilistic type.

It helps to maintain the interface of the existing system with the other systems:

- Sometimes the system to which we are going to re-design may act as an interface to other systems. So proper system analysis must be done in such a way that any changes made to this system will not affect the objectives of these systems. (That is the systems to which it is interfaced).

It helps to motivate the users to the new environment:

- Naturally the people involved in the system will not accept sudden working environment changes. But, system analysis makes the people in the system to involve in the system design. Therefore the people in the system feels free to work in the new environment.

It helps to understand the resource needs:

- System analysis helps to define the needed resources such as hardware and software.

It helps to understand the feasibility of the new system:

- System analysis of the system helps to study the feasibility of the new system. There are three types of feasibility. They are,

1. Technical
2. Economic
3. Operational

- In many cases, the systems are feasible in technical and economic but they are infeasible in operational point of view.

<u>Stages in system analysis</u>

While software system analysis is a complex and hard activity, people use it many years, so there should be some patterns and guides. There are a lot of books, some of them very good, most of them almost useless for agile developers, since based on solid Requirements Definition Phase with The Spec in the end.

In fact the process can be described with just several steps. All of them are important and in general could not be skipped.

1. Identify System Users

This is the most important question. If you miss with users, you will build the wrong solution. All further analyses will relay on defined user roles, so be very careful with this first step.

Typical questions:

- Who will use the system?

2. Define Main Users Goals

Each user in the system has specific goals. One user role will use system not very frequently to solve just a few problems, while another user role will use system about 2 hours per day and resolve many complex problems. Why user will use this system? What problems will it solve?

Typical questions:

- What I (as a user ___) want to achieve with help of the system?

3. Define System Usage Patterns

Each user has common behavioral patterns. For example, Manager comes at work and start checking yesterday results. Or Frequent Flyer wants to optimize travel expenses for the next month. Or Sales Person having a call with existing unhappy customer. Or Resource Manager handles requests on additional developers for 3 projects simultaneously. All these are typical usage patterns. This is the best starting point for functional solution. You clearly see the real problems, you understand them well, you have all you need to be creative and invent simple, effective and elegant solution.

Typical questions:

- What are the typical user behaviors (daily, specific situations, etc.)?

4. Invent Functional Solution to Meet Users Goals and Usage Patterns

This is just a logical continuation of previous step, but maybe the most complex step. Here you think how to solve exact problem, discuss the solution, jump to steps 5-6, refine the solution, write down it and move to the next usage pattern.

Typical questions:

- What is the best way to satisfy usage pattern?

5. Define Main Navigation Paths

This and next steps usually performed together with step 4. Usually it is hard to invent great solution without tracking user paths and sketching some UI areas. In fact it is better to stay off UI as far as you can. In discussions replace all phrases like "Then User clicks Leads link in the top menu and sees leads list" with "Then User sees leads list". Concentrate on information user need on each step, not on links and clicks. Good navigation path looks like this:

User:

1. Quickly add new lead into the system
2. See leads list
3. Find leads added yesterday
4. See additional details for each yesterday lead

There is no UI in the list above, just information.

Typical questions:

- What functional areas/action should user execute to complete usage pattern?
- How many areas required to complete user goal in specific pattern?

6. Create UI Mockups

UI mockups are great to have a quick look at possible users/system interaction. Dashboard sketches are perfect. Forget about cool tools like Visio, Dreamweaver or any other UI prototyping tool. Dashboard sketches are the fastest, easiest and exciting thing to work with. You will group around dashboard with markers in hands and discuss where to place system settings link with great enthusiasm. Everyone can draw with a marker, but it is just not possible with computer. With marker in hand everyone feels that s/he can improve this UI and bring in cool ideas. Here is the place where team spirit rises! Draw UI sketch, make a photo on digital camera and put all photos in a single shared space.

7. Polish UI Elements

There is always a possibility to make things better, to improve something here and there. It is a good attitude. You should think about UI details of most important features that will be implemented right after the project start. But be careful, don't spend much time on UI perfection, likely it will change during development anyway. And never polish UI for features that will be implemented in 3+ months ahead of current date, with great possibility it will be a waste of time.

Typical questions:

- Can we improve UI to reduce clicks, provide better visibility, etc?

After all the steps above you will end up with solid understanding of future system, with the most important artifacts in hands and clear starting point. And yes, you may start development and plan the first iteration!

System Development Models

Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

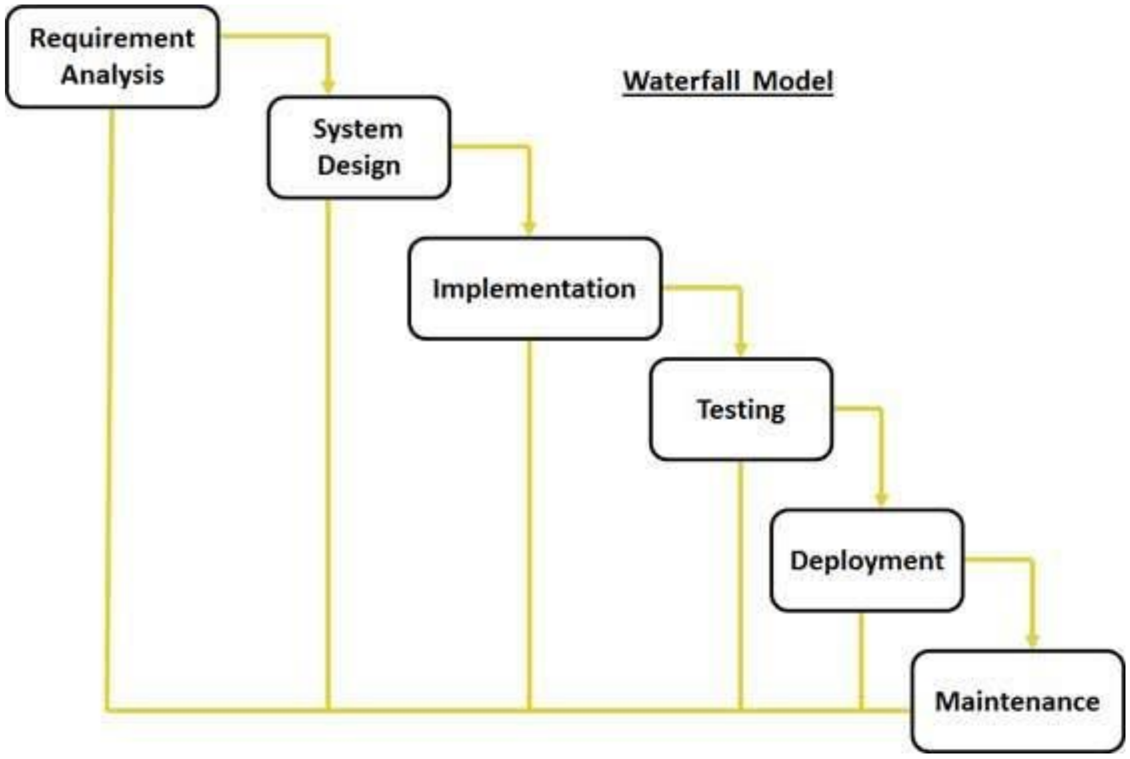
Waterfall model is the earliest SDLC approach that was used for software development .

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.

- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model Pros & Cons

Advantage

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantage

The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The following table lists out the pros and cons of Waterfall model:

Pros	**Cons**
• Simple and easy to understand and use	• No working software is produced until late during the life cycle. • High amounts of risk and uncertainty.

- Easy to manage due to the rigidity of the model . each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Prototype Model

The Software Prototyping refers to building software application prototypes which display the functionality of the product under development but may not actually hold the exact logic of the original software.

Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

What is Software Prototyping?

- Prototype is a working model of software with some limited functionality.
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is the stepwise approach to design a software prototype:

- **Basic Requirement Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

- **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.
- **Review of the Prototype:**The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.
- **Revise and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like , time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

Prototypes can have horizontal or vertical dimensions. Horizontal prototype displays the user interface for the product and gives a broader view of the entire system, without concentrating on internal functions. A vertical prototype on the other side is a detailed elaboration of a specific function or a sub system in the product.

The purpose of both horizontal and vertical prototype is different. Horizontal prototypes are used to get more information on the user interface level and the business requirements. It can even be presented in the sales demos to get business in the market. Vertical prototypes are technical in nature and are used to get details of the exact functioning of the sub systems. For example, database requirements, interaction and data processing loads in a given sub system.

Software Prototyping Types

There are different types of software prototypes used in the industry. Following are the major software prototyping types used widely:

- **Throwaway/Rapid Prototyping:** Throwaway prototyping is also called as rapid or close ended prototyping. This type of prototyping uses very little efforts with minimum requirement analysis to build a prototype. Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.
- **Evolutionary Prototyping:** Evolutionary prototyping also called as breadboard prototyping is based on building actual functional prototypes with minimal functionality in the beginning. The prototype developed forms the heart of the future prototypes on top of which the entire system is built. Using evolutionary prototyping only well understood requirements are included in the prototype and the requirements are added as and when they are understood.
- **Incremental Prototyping:** Incremental prototyping refers to building multiple functional prototypes of the various sub systems and then integrating all the available prototypes to form a complete system.

- **Extreme Prototyping :** Extreme prototyping is used in the web development domain. It consists of three sequential phases. First, a basic prototype with all the existing pages is presented in the html format. Then the data processing is simulated using a prototype services layer. Finally the services are implemented and integrated to the final prototype. This process is called Extreme Prototyping used to draw attention to the second phase of the process, where a fully functional UI is developed with very little regard to the actual services.

Software Prototyping Application

Software Prototyping is most useful in development of systems having high level of user interactions such as online systems. Systems which need users to fill out forms or go through various screens before data is processed can use prototyping very effectively to give the exact look and feel even before the actual software is developed.

Software that involves too much of data processing and most of the functionality is internal with very little user interface does not usually benefit from prototyping. Prototype development could be an extra overhead in such projects and may need lot of extra efforts.

Software Prototyping Pros and Cons

Software prototyping is used in typical cases and the decision should be taken very carefully so that the efforts spent in building the prototype add considerable value to the final software developed. The model has its own pros and cons discussed as below.

Following table lists out the pros and cons of Big Bang Model:

Pros	Cons
• Increased user involvement in the product even before implementation	• Risk of insufficient requirement analysis owing to too much dependency on prototype
• Since a working model of the system is displayed, the users get a better understanding of the system being developed.	• Users may get confused in the prototypes and actual systems.
• Reduces time and cost as the defects can be detected much earlier.	• Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
• Quicker user feedback is available leading to better solutions.	• Developers may try to reuse the existing prototypes to build the actual system, even when its not technically feasible
• Missing functionality can be identified easily	• The effort invested in building prototypes may be too much if not monitored properly
• Confusing or difficult functions can be identified	

Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.

Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis.

It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

Spiral Model design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

- **Identification:**This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

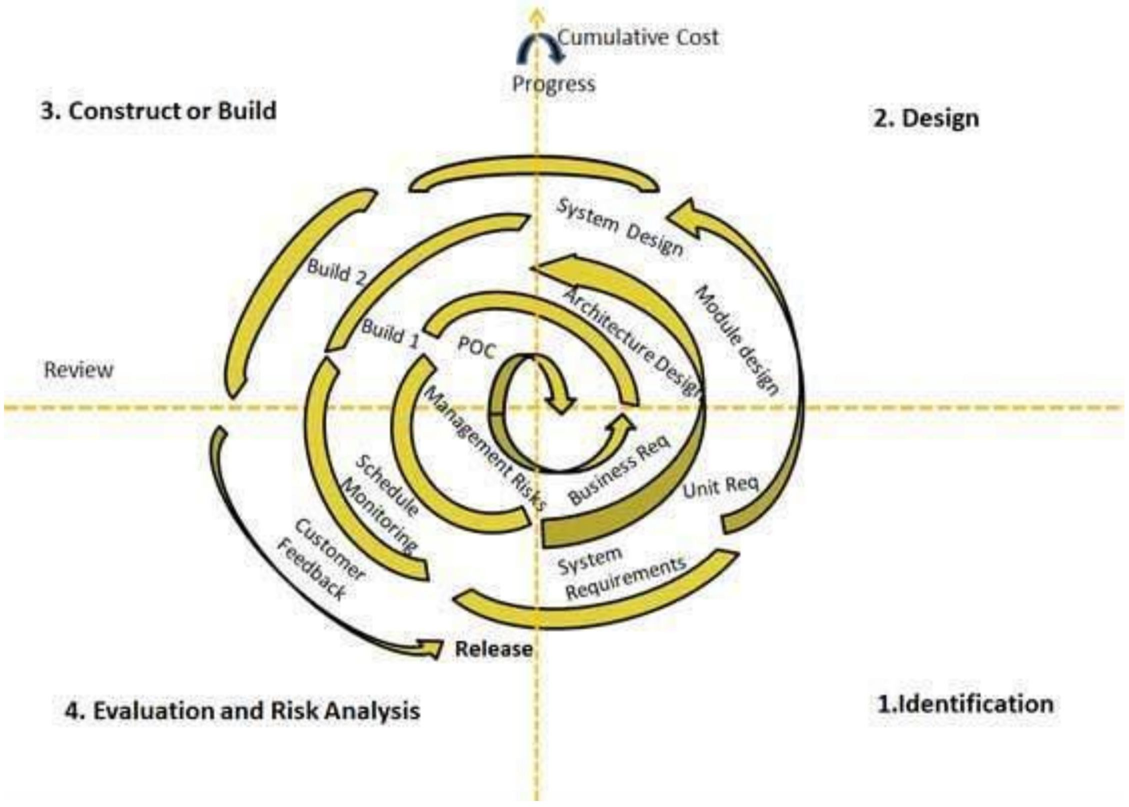
  This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

- **Design:**Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.
- **Construct or Build:**Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

  Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

- **Evaluation and Risk Analysis:**Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Following is a diagrammatic representation of spiral model listing the activities in each phase:



Based on the customer evaluation, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Spiral Model Application

Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:

- When costs there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Spiral Model Pros and Cons

The advantage of spiral lifecycle model is that it allows for elements of the product to be added in when they become available or known. This assures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases and allows for making an orderly transition to a maintenance activity. Another positive aspect is that the spiral model forces early user involvement in the system development effort.

On the other side, it takes very strict management to complete such products and there is a risk of running the spiral in indefinite loop. So the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

The following table lists out the pros and cons of Spiral SDLC Model:

Pros	Cons
• Changing requirements can be accommodated. • Allows for extensive use of prototypes • Requirements can be captured more accurately. • Users see the system early. • Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.	• Management is more complex. • End of project may not be known early. • Not suitable for small or low risk projects and could be expensive for small projects. • Process is complex • Spiral may go indefinitely. • Large number of intermediate stages requires excessive documentation.

RAD Model

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

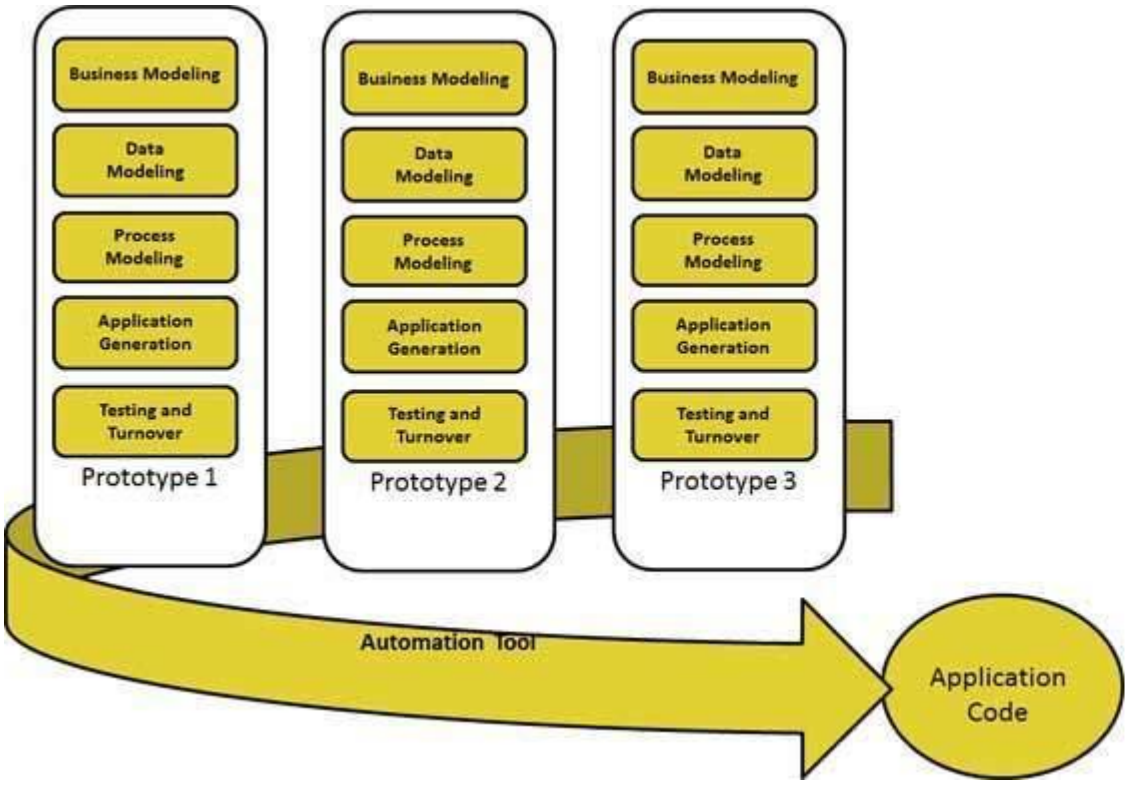
The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

RAD Model Design

RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. Following are the phases of RAD Model:

- **Business Modeling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.
- **Data Modeling:** The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.
- **Process Modeling:** The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding , deleting, retrieving or modifying a data object are given.
- **Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.
- **Testing and Turnover:**The overall testing time is reduced in RAD model as the prototypes are independently tested during every iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

Following image illustrates the RAD Model:



RAD Model Vs Traditional SDLC

The traditional SDLC follows a rigid process models with high emphasis on requirement analysis and gathering before the coding starts. It puts a pressure on the customer to sign off the requirements before the project starts and the customer doesn.t get the feel of the product as there is no working build available for a long time.

The customer may need some changes after he actually gets to see the software, however the change process is quite rigid and it may not be feasible to incorporate major changes in the product in traditional SDLC.

RAD model focuses on iterative and incremental delivery of working models to the customer. This results in rapid delivery to the customer and customer involvement during the complete development cycle of product reducing the risk of non conformance with the actual user requirements.

RAD Model Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail. Following are the typical scenarios where RAD can be used:

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
-
- It should be used if there.s high availability of designers for modeling.
-
- It should be used only if the budget permits use of automated code generating tools.
-
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
-
- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

RAD Model Pros and Cons

RAD model enables rapid delivery as it reduces the overall development time due to reusability of the components and parallel development.

RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.

Following table lists out the pros and cons of RAD Model:

Pros	**Cons**
• Changing requirements can be accommodated. • Progress can be measured. • Iteration time can be short with use of powerful RAD tools. • Productivity with fewer people in short time. • Reduced development time. • Increases reusability of components • Quick initial reviews occur • Encourages customer feedback • Integration from very beginning solves a lot of integration issues.	• Dependency on technically strong team members for identifying business requirements. • Only system that can be modularized can be built using RAD. • Requires highly skilled developers/designers. • High dependency on modeling skills. • Inapplicable to cheaper projects as cost of modeling and automated code generation is very high. • Management complexity is more. • Suitable for systems that are component based and scalable. • Requires user involvement throughout the life cycle. • Suitable for project requiring shorter development times.

<u>Roles and responsibilities of System Analyst</u>

Systems analysts bridge the gap between computer users and computer programmers.

Computer system analysts play several key roles in developing computer applications and systems. This job requires excellent communication skills and interpersonal skills, as well as the ability to organize information

Gathering Users' Application Requirements

Systems analysts must interview users and gather their requirements for new software systems. The analyst then organizes and documents her findings into design specifications for a new computer system. This is a difficult task, since many users do not have enough technical knowledge about computers. Many times they will leave out a requirement because they think the computer cannot possibly do it.

System Design

Design specifications help the systems analyst logically work through the various requirements and develop programming specifications for the computer programmer. The systems analyst’s understanding of information technology helps her determine the best technical solution for the problem at hand. Design specifications include text documents and flow charts for visual understanding.

System Documentation

While a programmer writes code, the systems analyst produces system documentation. This includes inputs, outputs and user-initiated processes. Documentation helps users learn to use the new application, and is also useful for training sessions. System documentation also helps programmers troubleshoot when the application does not behave as expected.

End-user Training

End-user training is where the systems analysts communication and interpersonal skills are put to the test. Users often resist change, opting for the old system rather than adapting to the newer, perhaps more challenging software. Good systems analysts are able to present the new system in a positive light and make difficult processes look easy.

<u>Database Administrator</u>

What is a Database Administrator?

A Database Administrator is a specialized type of Computer Systems Administrator. Also known as: DBA, Network and Database Administrator, Network & Database Administrator.

A database administrator is someone who uses software to store and organize data, such as financial information and customer shipping records. They make sure that data is available to users and is secure from unauthorized access. Database administrators work in many different types of industries, including computer systems design and related services firms, insurance companies, banks, and hospitals.

What does a Database Administrator do?

Database administrators, often called DBAs, make sure that data analysts can easily use the database to find the information they need and that the system performs as it should. DBAs sometimes work with an organization's management team to understand the company's data needs and to plan the goals of the database.

Database administrators often plan security measures, making sure that data is secure from unauthorized access. Many databases contain personal or financial information, making security important. Database administrators are responsible for backing up systems in case of a power outage or other disaster. They also ensure the integrity of the database, guaranteeing that the data stored in it comes from reliable sources.

DBAs must be able to monitor a database system's performance to determine when action is needed. They must be able to evaluate complex information that comes from a variety of sources. Most database administrators work on teams and must be able to communicate effectively with developers, managers, and other workers. Working with databases requires an understanding of complex systems, in which a minor error can cause major problems. For example, mixing up a customer's credit card information can cause someone to be charged for a purchase he or she didn't make. Database administrators use software to make sense of information and to arrange and organize it into meaningful patterns. The information is then stored in the databases that these workers administer, test, and maintain. When problems with a database arise, administrators must be able to diagnose and correct them.

Some responsibilities of a database administrator include:

- Identifying user needs to create and administer databases
- Ensuring that the database operates efficiently and without error
- Making and testing modifications to the database structure when needed
- Maintaining the database and updating permissions
- Merging old databases into new ones
- Backing up and restoring data to prevent data loss

Many database administrators are general-purpose DBAs and have all these duties. However, some DBAs specialize in certain tasks that vary with the organization and its needs.

Two common specialties are as follows:

- **System DBAs** are responsible for the physical and technical aspects of a database, such as installing upgrades and patches to fix program bugs. They typically have a background in system architecture and ensure that the database in a firm's computer system works properly.
- **Application DBAs** support a database that has been designed for a specific application or a set of applications, such as customer service software. Using complex programming languages, they may write or debug programs and must be able to manage the aspects of the applications that work with the database. They also do all the tasks of a general DBA, but only for their particular application.

Enterprise Resources Planning

An Enterprise Resource Planning (ERP) solution, basically integrates both internal and external communication with a single system. That is, ERP solution can collaborate information and processes (both external & internal) for various business functions like manufacturing, finance & accounts, human resource and the supply chain management. ERP is valuable for any business for its ability to provide accurate and updated data across multiple functions. Technical definition:

"Enterprise resource planning (ERP) is a category of business-management software—typically a suite of integrated applications—that an organization can use to collect, store, manage and interpret data from many business activities"

Basic Features:

1. Financial management

The financial module in ERP provides financial functionality and analysis reports for different departments and cost centers.

2. Human Resource Management

ERP solution offers many different sub-systems under the HR module. The subsystems are Personnel Management, Organization Management, Payroll Management, Time Management and Personal Development**.**

3. Manufacturing

It constitutes of number of functionalities, mainly, Bill of Material, Engineering Change Note, Shop Floor Control, Sales and Distribution Plan, Master Production Schedule, and many more**.**

4. Supplier and Purchase Order Management

The purchase module is well integrated with the Production Planning and Inventory Control Modules and also the supply chain process.

5. Inventory & Material Management Module

The functions of Inventory Control Module involves identifying inventory requirements, setting targets, providing replenishment techniques and options, monitoring item usages, reconciling the inventory balances and reporting inventory status**.**

6. CRM

A full fledged CRM module ensure proper management of lead, customer, opportunities and altogether better support to your customer and further growth in your business.

Benefits of ERP

1. Increased Productivity

Since all the data can be accessed from one location, it easier for the employees to manage and perform day-to-day tasks.

2. Managing day to day activity made easy

Moving to cloud reduces the cost of managing and maintaining servers. Reduces the overhead costs like IT staff, power, data storage and bandwidth

3. Improved decision making

ERP in place will give easy access to the data, hence making it easier for the management to take crucial decision with ease. This is easy for the management as all the data will be of quality and updated.

4. Reduced Operation cost

With ERP, all the department's system are connected to one integrated system. By integrating, the organization can do away with the issue of duplicate data or information. ERP also reduces the production cost and the inventory cost.

Conclusion

Enterprise Resource Planning as the name suggests is a technique that helps you get the best out of your business with proper management of your resources, business activities, product planning, cost, manufacturing or service delivery. It is a technique that helps you save a lot of money and save the most out of all your resources.

If ERP is not implemented in an organization, communication will go to-and-fro from department to department and also leave a long paper trail. Main purpose of an ERP solution is to connect all the different department and provide relevant, accurate and updated information at all times.

ERP Selection Criteria

By studying, evaluating, and documenting these five key criteria, our clients make the educated decisions that are best for their company.

Benefit from ERP Selection Criteria – Evaluate Five Criteria

1. Company
2. Technology
3. Function fit
4. Support
5. Cost of ownership

Once the vendors have been identified, the team should go about a process of evaluating these criteria. Here is a definition of each criterion:

Company

Company size (annual revenues and number of employees) becomes very important in your evaluation. Who will make the best partner? Who knows my industry the best and has the most references of companies like mine? Are they committed to serving my industry? Who will be able to keep my company abreast of technology changes for the next 20 years? Who will make the best vendor partner?

Technology

We find that most companies have developed a technology strategy favored by top management and IT. You will also find that ERP vendors have their own technology strategy. Even though most vendors profess they are open systems, in truth, each vendor has their technology "sweet spot." Understand the technology platform and architecture for each vendor and measure it against your strategy.

Function Fit

Even though there are dozens of vendors that have a good function fit, you will find there are only about three or four vendors that are the best fit for your business. The key in your evaluation is to quickly find those vendors that best address business best practices for your industry.

ERP Support

ERP vendors have a number of ways to support their client. The larger firms have significant support ecosystems to support their market. Buyers should evaluate all of their support systems including the following:

- Consulting organization
- Implementation methodology
- Education
- ERP implementation methodology
- Maintenance and phone support
- User groups
- Partner network

Cost of Ownership

Learn from the customers of the vendors about their cost of ownership. We find that at the "end of the day," most software vendors all get to the same price for software. A number of other factors differentiate vendors with cost of ownership. Annual support fees vary, implementation rates and fees vary, and ongoing need for support varies from vendor to vendor. Look to understand and confirm the total cost of ownership over a five year period.

Expert ERP Selection Criteria Assessment

Today's modern ERP system provides manufacturers the tools necessary to improve business performance. Robust business intelligence, dashboard reporting, mobile access, real-time data access, integrated inventory control, quality, MRP, and other features can help companies work smarter, make informed decisions, and improve business processes.

When carefully selected, an ERP system helps companies succeed and prosper in changing environments, setting them apart from competitors.

Challenges in ERP Implementation

7 Common Challenges Faced in ERP Implementation

Challenges in implementing ERP solutions are quite normal. Though it is not completely a technical job, a lot of planning and proper communication is very much essential to implement ERP across the organization. Below are the 7 common challenges we have noticed companies experience, when ERP Implementation

1. It is very important, that **implementation is done in stages**. Trying to implement everything at once will lead to a lot of confusion and chaos.
2. **Appropriate training is very essential** during and after the implementation. The staff should be comfortable in using the application or else, it will backfire, with redundant work and functional inefficiencies.

3. **Lack of proper analysis of requirements** will lead to non-availability of certain essential functionalities. This might affect the operations in the long run and reduce the productivity and profitability.
4. **Lack of Support from Senior Management** will lead to unnecessary frustrations in work place. Also, it will cause delay in operations and ineffective decisions. So, it is essential to ensure that the Senior Management supports the transformation.
5. **Compatibility Issues with ERP Modules** lead to issues in integration of modules. Companies associate different vendors to implement different ERP modules, based on their competency. It is very essential that there is a way to handle compatibility issues.
6. **Cost Overheads** will result, if requirements are not properly discussed and decided during the planning phase. So, before execution, a detailed plan with a complete breakdown of requirements should be worked out.
7. **Investment in Infrastructure** is very essential. **ERP applications** modules will require good processing speed and adequate storage. Not allocating suitable budget for infrastructure will result in reduced application speed and other software issues. Hardware and Software Security is also equally important.

These are certain generic challenges while implementing **ERP solutions**. Depending on the sector in which the company operates in, the extent of complications may vary. So, it is very essential to bring onboard, an expert team of consultants. This will ensure the implementation process is smooth without any glitches.