CH-5 Acquisition, Development and Implementation of Information Systems

This chapter is Divided in to 3 Divisions

Division-1 - SYSTEM DEVELOPMENT METHODOLOGY

Waterfall Model	Prototyping Model	Incremental Model	Spiral Model	RAD Model	Agile Model

Division -2 – SDLC – 8 Phases

Preliminary Investigation	Systems Requirements Analysis	Systems Design	Systems Acquisition	Systems Development	Systems Testing	Systems Implementation	Post Implementation Review and Maintenance

Division -3- Miscellaneous

Achieving System Development Objectives
Accountants' Involvement in Development Work
System Development Team

Division-1: SYSTEM DEVELOPMENT METHODOLOGY

SDM is a formalized, standardized, well-organized and documented set of activities used to manage a system development project.

Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. The **methodology is characterized by the following:**

- The project is divided into a number of identifiable processes, and each process has a starting point and an ending point.

Each process comprises several activities, one or more deliverables, and several management control points.

The division of the project into these small, manageable steps facilitates both project planning and project control.

-Specific reports and other documentation, called Deliverables must be produced periodically during system development.

-Users, managers, and auditors are required to participate in the project, which generally provide approvals, often called signoffs, at pre-established management control points.

-The system must be tested thoroughly prior to implementation to ensure that it meets users' needs as well as requisite functionalities.

-A training plan is developed for those who will operate and use the new system.

-Formal program change controls are established to preclude unauthorized changes to computer programs.

-A post-implementation review of all developed systems must be performed to assess the effectiveness and efficiency of the new system and of the development process.

MODEL-1 Waterfall Model

The waterfall approach is a traditional development approach in which each phase is carried in sequence or linear fashion.

In this traditional approach of system development, activities are performed in sequence.

These phases include

requirements analysis,
specifications and design requirements,
coding,
final testing, and
release.

3-key characteristics are the following

1. Project is divided into sequential phases, with some overlap and splash back acceptable between phases.

2. Emphasis is on

planning,
time schedules,
target dates,
budgets and implementation of an entire system at one time.

3. Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval

by the user and IT management

occurring at the end of most phases before beginning the next phase.

4-Strengths

1.It is ideal for supporting less experienced project teams and project managers, whose composition fluctuates.

2.The orderly sequence of development steps and design reviews help to ensure the quality, reliability, adequacy and maintainability of the developed software.

3.Progress of system development is measurable.

4.It enables to conserve resources.

10- Weaknesses

1. It promotes the gap between users and developers with clear vision of responsibility.

2.Written specifications are often difficult for users to read and thoroughly appreciate.

3. It leads to excessive documentation and often time-consuming.

4. It is difficult to respond to changes, which may occur later in the life cycle, and if undertaken it proves costly and are thus discouraged.

5. System performance cannot be tested until the system is almost fully coded.

6. Problems are often not discovered until **system testing.**

7.Requirement inconsistencies, missing system components and unexpected development needs are often discovered during **design and coding.**

8. There is a little to iterate, which may be essential in situations.

9. Project progresses forward, with only slight movement backward.

10. It is criticized to be inflexible, slow, costly, and cumbersome due to significant structure and tight controls.

MODEL-2 Prototyping Model

The goal of prototyping approach is to develop a small or pilot version called a prototype of part or all of a system.

A prototype is a usable system component that is built quickly and at a lesser cost, and with the intention of modifying/replicating/expanding or even replacing it by a full-scale and fully operational system.

As users work with the prototype, they learn about the system criticalities and make suggestions about the ways to manage it.

These suggestions are then incorporated to improve the prototype, which is also used and evaluated.

Finally, when a prototype is developed that satisfies all user requirements, either it is refined and turned into the final system or it is scrapped.

If it is scrapped, the knowledge gained from building the prototype is used to develop the real system.

Prototyping can be viewed as a series of 4 steps:

The generic phases of this model are explained as follows:

1. Identify Information System Requirements:

In traditional approach, the system requirements are to be identified before the development process starts.	However, under prototype approach, the design team needs only fundamental system requirements to build the initial prototype, the process of determining them can be less formal and time- consuming than when performing traditional systems analysis.

2. Develop the Initial Prototype:

The designers create an initial base model and give little or no consideration to internal controls, but instead emphasize system characteristics such as simplicity, flexibility, and ease of use.

These characteristics enable users to interact with tentative versions of data entry display screens, menus, input prompts, and source documents.

The users also need to be able to respond to system prompts, make inquiries of the information system, judge response times of the system, and issue commands.

3.Test and Revise:

After finishing the initial prototype, the designers **first demonstrate** the model to users and then give it to them to experiment and ask users to record their likes and dislikes about the system and recommend changes.

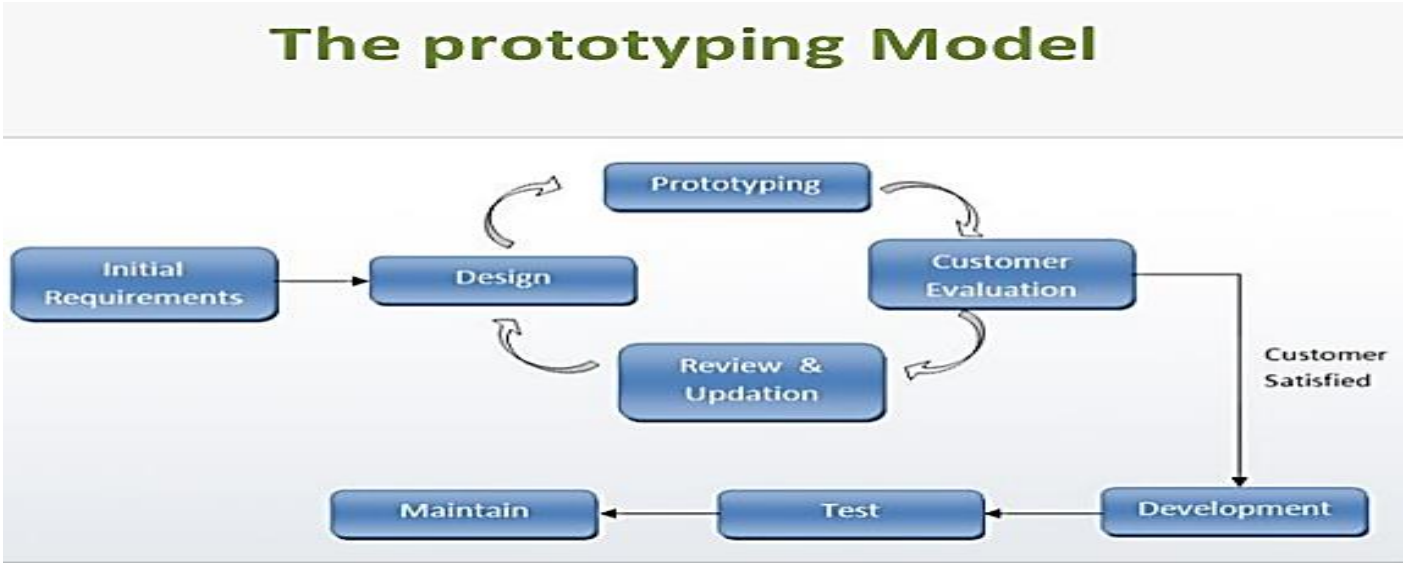
Using this feedback, the design **team modifies** the prototype as necessary and then resubmits the revised model to system users for reevaluation.

Thus iterative process of modification and reevaluation continues until the users are satisfied.

4.Obtain User Signoff of the Approved Prototype:

Users formally approve the final version of the prototype, which commits them to the current design and establishes a **contractual obligation** about what the system will, and will not, do or provide.

Prototyping is not commonly used for developing traditional applications such as accounts receivable, accounts payable, payroll, or inventory management, where the inputs, processing, and outputs are well known and clearly defined.



10-Strengths

1. It improves both user participation in system development and communication among project stakeholders.

2. It is especially useful for

resolving unclear objectives;
developing and validating user requirements;
experimenting with or comparing various design solutions, or
investigating both performance and the human computer interface.

3. Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed.

4. It helps to easily identify, confusing or difficult functions and missing functionality.

5. It enables to generate specifications for a production application.

6. It encourages innovation and flexible designs.

7. It provides for quick implementation of an incomplete, but functional, application.

8. It typically results in a better definition of these users' needs and requirements than does the traditional systems development approach.

9. A very short time period is normally required to develop and start experimenting with a prototype.

10. Since system users experiment with each version of the prototype through an interactive process, errors are hopefully detected and eliminated early in the developmental process.

9-Weaknesses

1. Prototyping may cause behavioral problems, if, system developers are unable to meet all user demands.

2. The interactive process of prototyping causes the prototype to be experimented with quite extensively.

3. Inadequate testing can make the approved system error-prone, and inadequate documentation makes this system difficult to maintain.

4. Prototyping can only be successful if the system users are willing to devote **significant time** in experimenting with the prototype and provide the system developers with change suggestions.

5. Prototype may not have sufficient checks and balances incorporated.

6. Designers may prototype too quickly, without sufficient upfront user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential.

7.Requirements may frequently change significantly.

8.Identification of non-functional elements is difficult to document.

9.Approval process and control are not strict.

MODEL-3 Incremental Model

The Incremental model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.

This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds as shown in diagram).

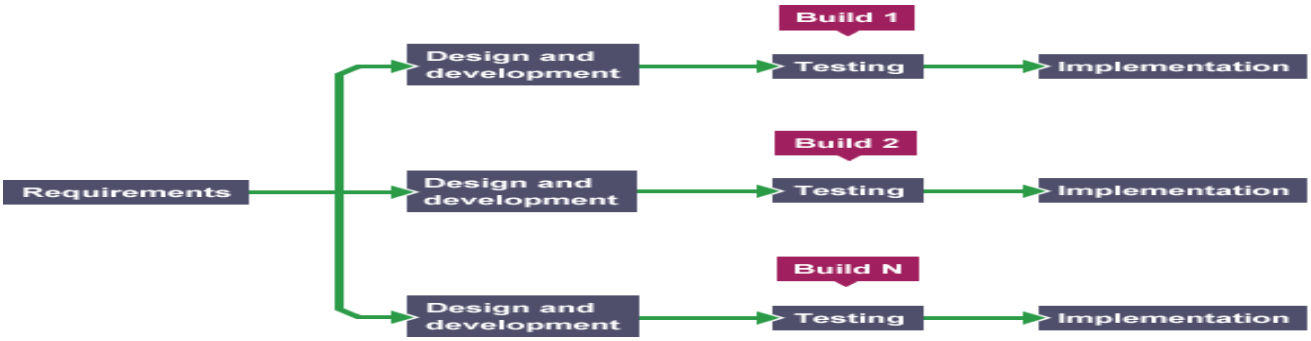
Each component is delivered to the client when it is complete.

3- few pertinent features are:

1. A series of mini-waterfalls are performed, where all phases of the waterfall development model are completed for a small part of the system, before proceeding to the next increment.

2. Overall requirements are defined before proceeding to evolutionary, mini – Waterfall development of individual increments of the system.

3. The initial software concept, requirement analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which comes to climax in installation of the final prototype.



7-Strengths

1. Potential exists for exploiting knowledge gained in an early increment as later increments are developed.

2. Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval by the

user and information technology management at designated major milestones.

3. Stakeholders can be given concrete evidence of project status throughout the life cycle.

4. It is more flexible and less costly to change scope and requirements.

5. It helps to mitigate integration and risks earlier in the project.

6. It allows the delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.

7. Gradual implementation provides the ability to monitor the effect of incremental changes, isolated issues and make adjustments before the organization is negatively impacted.

5-Weaknesses

1. lack of overall consideration of the business problem and technical requirements for the overall system, before moving onto the next increment.

2. Each phase of an iteration is rigid and do not overlap each other.

3. Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

4. Since some modules will be completed much earlier than others, well-defined interfaces are required.

5. It is difficult to demonstrate early success to management.

Model-4 Spiral Model

The Spiral model is a software development process combining elements of both design and prototyping-in-stages.

It tries to combine advantages of top-down and bottom-up concepts.

It combines the features of the prototyping model and the waterfall model.

The spiral model is intended for large, expensive and complicated projects.

Game development is a main area where the spiral model is used and needed, that is because of the size and the constantly shifting goals of those large projects. A list of **pertinent characterizing features** includes the following:

1. The new system requirements are defined in as much detail as possible.

This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

2. A preliminary design is created for the new system.

This phase is the most important part of "Spiral Model" in which all possible alternatives that can help in developing a cost effective project.

This phase has been added specially in order to identify and resolve all the possible risks in the project development.

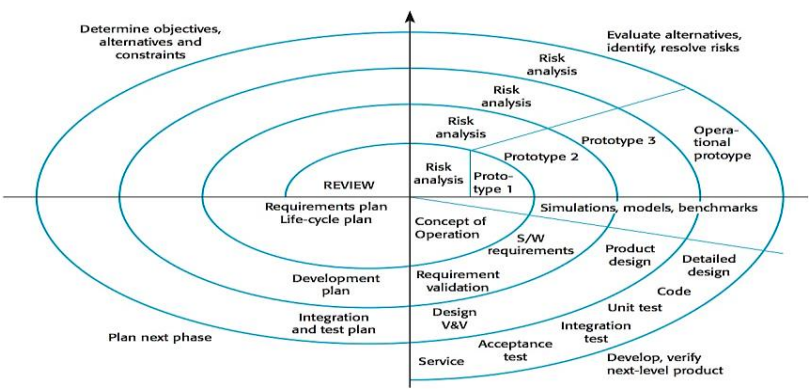
If risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.

3. A first prototype of the new system in constructed from the preliminary design.

This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

4. A second prototype is evolved by a 4 times procedure by evaluating the

first prototype in terms of its strengths, weaknesses, and risks;
defining the requirements of the second prototype;
planning and designing the second prototype; and
constructing and testing the second prototype.



3- Strengths

1. It enhances the risk avoidance.

2. It is useful in helping for optimal development of a given software iteration based on project risk.

3. It can incorporate Waterfall, Prototyping, and Incremental methodologies as special cases in the framework, and provide guidance as to which combination of these models best fits a given software iteration, based upon the type of project risk.

5- Weaknesses

1. There are no firm deadlines, cycles continue with no clear termination condition leading to, inherent risk of not meeting budget or schedule.

2. No established controls exist for moving from one cycle to another cycle.

Without controls, each cycle may generate more work for the next cycle.

3. A skilled and experienced project manager is required to determine how to apply it to any given project.

4. It may prove highly customized to each project, and thus is quite complex and limits reusability.

5. It is challenging to determine the exact composition of development methodologies to use for each iteration around the Spiral.

Model-5 Rapid Application Development (RAD) Model

Rapid Application Development (RAD) refers to a type of software development methodology;

which uses minimal planning in favor of rapid prototyping.

The planning of software developed using RAD is inserted pages with writing the software itself.

The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements.

10-Key features include the following:

1. Key objective is fast development and delivery of a high quality system at a relatively low investment cost.

2. Attempts to reduce inherent project risk by breaking a project into smaller segments and

providing more ease-of-change during the development process.

3. Aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools.

4. Key emphasis is on fulfilling the business need while technological or engineering excellence is of lesser importance.

5. Project control involves prioritizing development and defining delivery deadlines or “time boxes.”

6. Generally includes Joint Application Development (JAD), where users are intensely involved in system design, either through consensus building in structured workshops, or through electronically facilitated interaction.

7. Active user involvement is crucial.

8. Iteratively produces production software, as opposed to a throwaway prototype.

9. Produces documentation necessary to facilitate future development and maintenance.

10.Standard systems analysis and design techniques can be fitted into this framework.

8- Strengths

1. The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks.

2.Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at lower cost.

3. Quick initial reviews are possible.

4.Constant integration isolates problems and encourages customer feedback.

5.It holds a great level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or spiral frameworks.

6.It concentrates on essential system elements from user viewpoint.

7.It provides for the ability to rapidly change system design as demanded by users.

8.It leads to a tighter fit between user requirements and system specifications.

9- Weaknesses

1. Fast speed and lower cost may affect adversely the system quality.

2. The project may end up with more requirements than needed (gold-plating).

3. Potential for feature creep where more and more features are added to the system over the course of development.

4. It may lead to inconsistent designs within and across systems.

5. It may call for violation of programming standards related to inconsistent naming conventions and inconsistent documentation.

6. It may call for lack of attention to later system administration needs built into system.

7. Formal reviews and audits are more difficult to implement than for a complete system.

8. Tendency for difficult problems to be pushed to the future to demonstrate early success to management.

9. Since some modules will be completed much earlier than others, well–defined interfaces are required.

Model-6 Agile Model

It is a conceptual framework that promotes foreseen interactions throughout the development life cycle.

Agile Manifesto is based on following 12 features:

1. Customer satisfaction by rapid delivery of useful software.
2. Welcome changing requirements, even late in development.
3. Working software is delivered frequently.
4. Working software is the principal measure of progress.
5. Sustainable development, able to maintain a constant pace.
6. Close, daily co-operation between business people and developers.
7. Face-to-face conversation is the best form of communication.
8. Projects are built around motivated individuals, who should be trusted.
9. Continuous attention to technical excellence and good design.
10. Simplicity.
11. Self-organizing teams.
12. Regular adaptation to changing circumstances.

5- Strengths

1. Agile methodology has the concept of an adaptive team, which enables to respond to the changing requirements.

2. The team does not have to invest time and efforts and finally find that by the time they delivered the product, the requirement of the customer has changed.

3. Face to face communication and continuous inputs from customer representative leaves a little space for guesswork.

4. The documentation is crisp and to the point to save time.

5. The end result is generally the high quality software in least possible time duration and satisfied customer.

6- Weaknesses

1. it is difficult to assess the efforts required at the beginning of the SDLC.

2. There is lack of emphasis on necessary designing and documentation.

3. By nature, Agile projects are extremely light on documentation.

4. Agile requires more re-work and due to the lack of long-term planning and the lightweight approach to architecture, re-work is often required on Agile projects when the various components of the software are combined and forced to interact.

5. The project can easily get taken off track if the customer representative is not clear about the final outcome.

6. Agile lacks the attention to outside integration.