FEDERAL UNIVERSITY OF KASHERE
FACULTY OF SCIENCE
Department of Computer Science
www.fukashere.edu.ng
P.M.B 0182,
GOMBE,
GOMBE STATE

SYSTEM ANALYSIS AND DESIGN (CSC 3307)

MODULE (1)

Module Content:

- **System Concepts**
- **System Development Life Cycle**

Referral material: Systems Analysis and Design-An Object-Oriented Approach with UML-2015.

Introduction

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD), mainly deals with the software development activities. Systems development is a systematic process which includes phases such as planning, analysis, design, and implementation.

What is a System?

An organized relationship between any set of components to achieve some common cause or objective. A system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal. A system provides several definitions:

1. A regularly interacting or interdependent group of items forming a unified whole.
2. An organized set of doctrines, ideas, or principles, usually intended to explain the arrangements or working of a systematic whole.
3. An organized or established procedure.
4. Harmonious arrangement or pattern.
5. An organized society or social situation regarded as stultifying establishment.

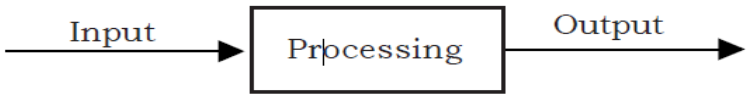

Figure 1. Basic Component of a System

A system has the following properties:

- **Organization**

Organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

- **Interaction**

It is defined by the manner in which the components operate with each other. For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

- **Interdependence**

Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

- **Integration**

Integration is concerned with how system components are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

Types of Systems

There are many different types of systems, but indeed, virtually everything that we come into contact with during our day-to-day life is either a system or a component of a system (both). It is useful to organize the many different kinds of systems into useful categories. The systems can be divided into the following types:

- **Physical or Abstract Systems**

Physical systems are tangible entities. We can touch and feel them. *Physical System* may be static or dynamic in nature. For example, desks and chairs are the physical parts of computer system workstation which are static. A programmed computer is a dynamic system in which programs, data,

and applications can change according to the user's needs. *Abstract systems* are non-physical entities or conceptual that may be formulas, representation or model of a real system.

- **Open or Closed Systems**

An open system must interact with its environment. It receives inputs from and delivers outputs to the outside of the system. For example, an information system which must adapt to the changing environmental conditions. *A closed system* does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

- **Adaptive and Non-Adaptive System**

Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals. *Non-Adaptive System* is the system which does not respond to the environment. For example, machines.

- **Permanent or Temporary System**

Permanent System persists for long time. For example, business policies. *Temporary System* is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

- **Natural and Manufactured System**

Natural systems are created by the nature. For example, Solar system, seasonal system. *Manufactured System* is the man-made system. For example, Rockets, dams, trains.

- **Deterministic or Probabilistic System**

Deterministic system operates in a predictable manner and the interaction between system components is known with certainty. For example, two molecules of hydrogen and one molecule of oxygen makes water. *Probabilistic System* shows uncertain behavior. The exact output is not known. For example, Weather forecasting, mail delivery.

- **Social, Human-Machine, Machine System**

Social System is made up of people. For example, social clubs, societies. In *Human-Machine System*, both human and machines are involved to perform a particular task. For example, Computer programming. *Machine System* is where human interference is neglected. All the tasks are performed by the machine. For example, an autonomous robot.

- **Man–Made Information Systems**

It is an interconnected set of information resources to manage data for particular organization, under Direct Management Control (DMC). This system includes hardware, software, communication, data, and application for producing information according to the need of an organization.

- **Computer Based System**

This system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

The systems development life cycle (SDLC)

This section introduces the Systems Development Life Cycle (SDLC), the fundamental four-phase model (planning, analysis, design, and implementation) common to all information system development projects. Secondly, it describes the three major system development methodologies, discussing their pros and cons.

The systems development life cycle (SDLC) is the process of understanding how an information system (IS) can support business needs by designing a system, building it, and delivering it to users. It is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. SDLC also means combination of various system development activities. In other words, we can say that various system development activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle also means software development life cycle. The key person in the SDLC is the *systems analyst*, who analyzes the business situation, identifies opportunities for improvements, and designs an information system to implement them. Being a systems analyst is one of the most interesting, exciting, and challenging jobs around. Systems analysts work with a variety of people and learn how they conduct business. Specifically, they work with a team of systems analysts, programmers, and others on a common mission. Systems analysts feel the satisfaction of seeing systems that they designed and developed make a significant business impact, knowing that they contributed unique skills to make that happen.

In many ways, building an information system is similar to building a house. First, the house (or the information system) starts with a basic idea. Second, this idea is transformed into a simple drawing that is shown to the customer and refined (often through several drawings, each improving on the last) until the customer agrees that the picture depicts what he or she wants. Third, a set of

blueprints is designed that presents much more detailed information about the house (e.g., the type of roofing, where the dining will be placed). Finally, the house is built following the blueprints, often with some changes directed by the customer as work progresses.

The SDLC has a similar set of four fundamental phases: planning, analysis, design, and implementation. Different projects may emphasize different parts of the SDLC or approach the SDLC phases in different ways, but all projects have elements of these four phases. Each phase is itself composed of a series of steps, which rely upon techniques that produce deliverables (specific documents and files that provide understanding about the project). For example, when you apply for admission to a university, all students go through the same phases: information gathering, applying, and accepting. Each of these phases has steps information gathering includes steps such as searching for schools, requesting information, and reading brochures. Students then use techniques (e.g., Internet searching) that can be applied to steps (e.g., requesting information) to create deliverables (e.g., evaluations of different aspects of universities).

The fundamental four-phase model common to all information system development projects are explained below.

1. Planning

The planning phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it. It has two steps:
1. During project initiation, the system's business value to the organization is identified: how will it lower costs or increase revenues? Most ideas for new systems come from outside the information system (IS) area (from the marketing department, accounting department, etc.) in the form of a system request. A system request presents a brief summary of a business need, and it explains how a system that supports the need will create business value. The IS department works together with the person or department that generated the request (called the project sponsor) to conduct a feasibility analysis. The feasibility analysis examines key aspects of the proposed project:

- The idea's technical feasibility (Can we build it?)
- The economic feasibility (Will it provide business value?)
- The organizational feasibility (If we build it, will it be used?)

The system request and feasibility analysis are presented to an information systems approval committee (sometimes called a steering committee), which decides whether the project should be undertaken.

2. Once the project is approved, it enters *project management*. During project management, the project manager creates a work plan, staffs the project, and puts techniques in place to help the project team control and direct the project through the entire SDLC. The deliverable for project management is a *project plan*, which describes how the project team will go about developing the system.

2. Analysis

The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used. During this phase, the project team investigates any current system(s), identifies improvement opportunities, and develops a concept for the new system. This phase has three steps

1. An *analysis strategy* is developed to guide the project team's efforts. Such a strategy usually includes an analysis of the current system (called the *as-is system*) and its problems, and then ways to design a new system (called the *to-be system*).

2. The next step is *requirements gathering* (e.g., through interviews or questionnaires). The analysis of this information in conjunction with input from project sponsor and many other people—leads to the development of a concept for a new system. The system concept is then used as a basis to develop a set of business *analysis models,* which describe how the business will operate if the new system is developed. The set of models typically includes models that represent the data and processes necessary to support the underlying business process.

3. The analyses, system concept, and models are combined into a document called the *system proposal,* which is presented to the project sponsor and other key decision makers (e.g., members of the approval committee) who decide whether the project should continue to move forward.

The system proposal is the initial deliverable that describes what business requirements the new system should meet. Because it is really the first step in the design of the new system, some experts argue that it is inappropriate to use the term analysis as the name for this phase; some argue a better name would be analysis and initial design. Most organizations continue to use the name *analysis* for this phase. Just keep in mind that the deliverable from the analysis phase is both an analysis and a high-level initial design for the new system, jointly called the system proposal.

3. Design

The design phase decides how the system will operate, in terms of the hardware, software, and network infrastructure; the user interface, forms and reports; and the specific programs, databases, and files that will be needed. Although most of the strategic decisions about the system were made in the development of the system concept during the analysis phase, the steps in the design phase determine exactly how the system will operate. The design phase has four steps:

1. The *design strategy* is first developed. It clarifies whether the system will be developed by the company's own programmers, whether the system will be outsourced to another firm (usually a consulting firm), or whether the company will buy an existing software package.

2. This leads to the development of the basic *architecture design* for the system, which describes the hardware, software, and network infrastructure to be used. In most cases, the system will add or change the infrastructure that already exists in the organization. The interface design specifies how the users will move through the system (e.g., navigation methods such as menus and on-screen buttons) and the forms and reports that the system will use.

3. The *database and file specifications* are developed. These define exactly what data will be stored and where they will be stored.

4. The analyst team develops the program design, which defines the programs that need to be written and exactly what each program will do.

This collection of deliverables (architecture design, interface design, database and file specifications, and program design) is the *system specification* that is handed to the programming team for implementation. At the end of the design phase, the feasibility analysis and project plan are reexamined and revised, and another decision is made by the project sponsor and approval committee about whether to terminate the project or continue.

4. Implementation

The final phase in the SDLC is the *implementation phase*, during which the system is actually built (or purchased, in the case of a packaged software design). This is the phase that usually gets the most attention, because for most systems it is the longest and most expensive single part of the development process. This phase has three steps:

1. *System construction* is the first step. The system is built and tested to ensure it performs as designed. Because the cost of bugs can be immense, testing is one of the most critical steps in implementation. Most organizations give more time and attention to testing than to writing the programs in the first place.

2. The system is installed. *Installation* is the process by which the old system is turned off and the new one is turned on. It may include a *direct cutover approach* (in which the new system immediately replaces the old system), a *parallel conversion approach* (in which both the old and new systems are operated for a month or two until it is clear that there are no bugs in the new system), or a *phased conversion strategy* (in which the new system is installed in one part of the organization as an initial trial and then gradually installed in others). One of the most important aspects of conversion is the development of a training plan to teach users how to use the new system and help manage the changes caused by the new system.

3. The analyst team establishes a *support plan* for the system. This plan usually includes a formal or informal post-implementation review as well as a systematic way for identifying major and minor changes needed for the system.

In a more detailed sense, the SDLC is decomposed into the following explicit steps:

- *Preliminary study:* Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration.
- *Feasibility study:* In case the system proposal is acceptable to the management, the next phase is to examine the feasibility of the system. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness.
- *Detailed system study:* The detailed investigation of the system is carried out in accordance with the objectives of the proposed system. This involves detailed study of various operations performed by a system and their relationships within and outside the system
- *System analysis:* Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning.
- *System design:* Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase

in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design. There are several tools and techniques used for describing the system design of the system. These tools and techniques are: Flowchart, Data flow diagram (DFD), Data dictionary, Structured English, Decision table, Decision tree.

- *Coding:* The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language
- *Testing:* Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system.
- *Implementation:* After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice.
- *Maintenance:* Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments. It has been seen that there are always some errors found in the systems that must be noted and corrected. It also means the review of the system from time to time.

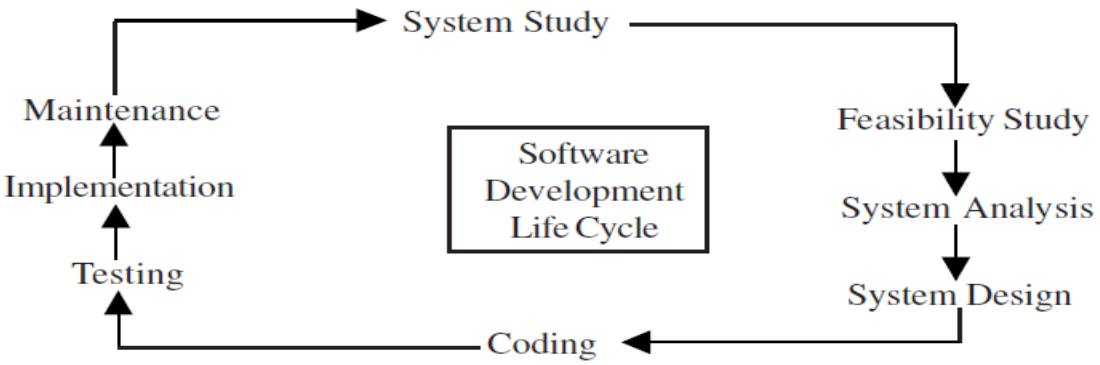

Figure 2: Phases of System Development Life Cycle

System Development Methodology

A methodology is a formalized approach to implementing the SDLC (i.e., it is a list of steps and deliverables). There are many different systems development methodologies, and each one is unique, based on the order and focus it places on each SDLC phase. Some methodologies are formal standards used by government agencies, whereas others have been developed by consulting firms to sell to

clients. Many organizations have internal methodologies that have been honed over the years, and they explain exactly how each phase of the SDLC is to be performed in that company.

There are many ways to categorize methodologies. One way is by looking at whether they focus on business processes or the data that support the business. A *process-centered methodology* emphasizes process models as the core of the system concept. For example, process-centered methodologies would focus first on defining the processes (e.g., assemble sandwich ingredients). *Data-centered methodologies* emphasize data models as the core of the system concept. In data centered methodologies would focus first on defining the contents of the storage areas (e.g., refrigerator) and how the contents were organized. By contrast, *object-oriented methodologies* attempt to balance the focus between process and data by incorporating both into one model.

Categories of Systems Development Methodologies

1. Structured Design

The first category of systems development methodologies is called *structured design*. Structured design methodologies adopt a formal step-by-step approach to the SDLC that moves logically from one phase to the next. Numerous process-centered and data-centered methodologies follow the basic approach of the two structured design categories outlined next.

a) Waterfall Development:

The original structured design methodology (still used today) is waterfall development. With waterfall development–based methodologies, the analysts and users proceed in sequence from one phase to the next. The key deliverables for each phase are typically very long (often hundreds of pages in length) and are presented to the project sponsor for approval as the project moves from phase to phase. Once the sponsor approves the work that was conducted for a phase, the phase ends and the next one begins. This methodology is referred to as waterfall development because it moves forward from phase to phase in the same manner as a waterfall. Although it is possible to go backward in the SDLC (e.g., from design back to analysis), it is extremely difficult (imagine yourself as a salmon trying to swim upstream against a waterfall, as shown in Figure below. The two key advantages of the structured design waterfall approach are that it identifies system requirement long before programming begins and it minimizes changes to the requirements as the project proceeds. The two key disadvantages are

that the design must be completely specified before programming begins and that a long-time elapses between the completion of the system proposal in the analysis phase and the delivery of the system (usually many months or years).

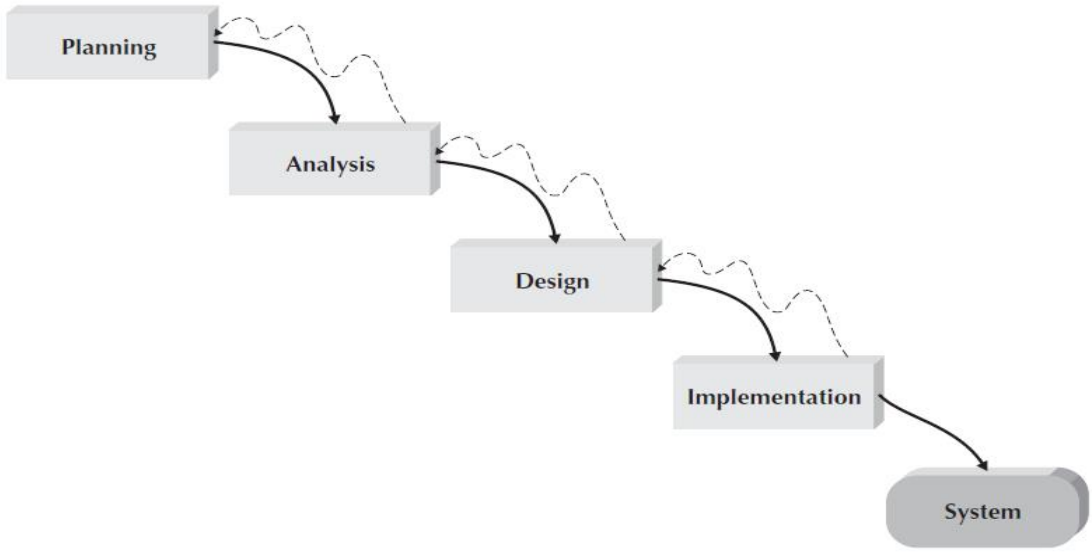

Figure 3: A Waterfall Development-Based Methodology

b) Parallel Development:

Parallel development methodology attempts to address the problem of long delays between the analysis phase and the delivery of the system. Instead of doing design and implementation in sequence, it performs a general design for the whole system and then divides the project into a series of distinct subprojects that can be designed and implemented in parallel. Once all subprojects are complete, there is a final integration of the separate pieces, and the system is delivered shown below.

The primary advantage of this methodology is that it can reduce the schedule time to deliver a system; thus, there is less chance of changes in the business environment causing rework. However, the approach still suffers from problems caused by paper documents. It also adds a new problem: Sometimes the subprojects are not completely independent; design decisions made in one subproject may affect another, and the end of the project may require significant integration efforts.

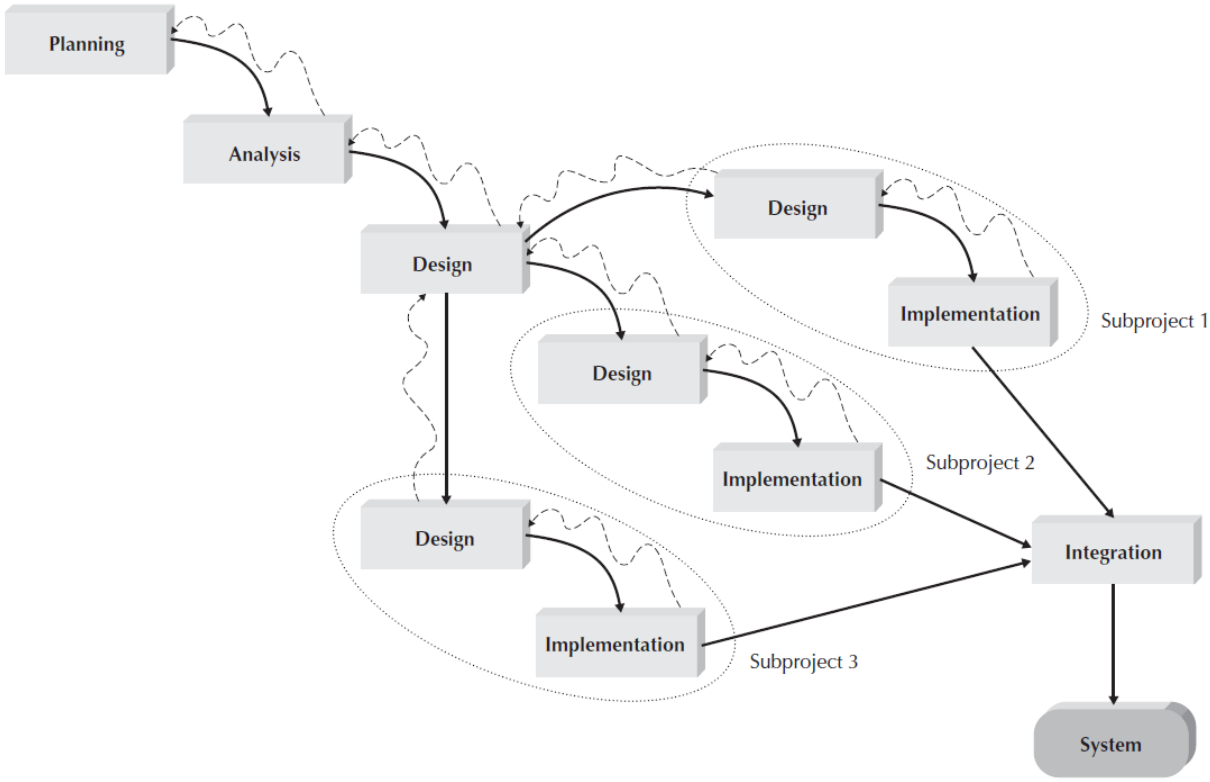

Figure 4: A Parallel Development-Based Methodology

2. Rapid Application Development (RAD)

A second category of methodologies includes *rapid application development (RAD)* based methodologies. These are a newer class of systems development methodologies that emerged in the 1990s. RAD-based methodologies attempt to address both weaknesses of structured design methodologies by adjusting the SDLC phases to get some part of the system developed quickly and into the hands of the users. In this way, the users can better understand the system and suggest revisions that bring the system closer to what is needed.

Most RAD-based methodologies recommend that analysts use special techniques and computer tools to speed up the analysis, design, and implementation phases, such as computer aided software engineering (CASE) tools, joint application design (JAD) sessions, fourth-generation/visual programming languages that simplify and speed up programming (e.g., Visual Basic), and code generators that automatically produce programs from design specifications. The combination of the changed SDLC phases and the use of these tools and techniques improve the speed and quality of systems development. However, there is one possible subtle problem with RAD-based methodologies: managing user expectations. Due to the use of the tools and techniques that can

improve the speed and quality of systems development, user expectations of what is possible may dramatically change.

a) Phased Development

A phased development–based methodology breaks an overall system into a series of *versions,* which are developed sequentially. The analysis phase identifies the overall system concept, and the project team, users, and system sponsor then categorize the requirements into a series of versions. The most important and fundamental requirements are bundled into the first version of the system. The analysis phase then leads into design and implementation but only with the set of requirements identified for version 1. as shown below. Once version 1 is implemented, work begins on version 2. Additional analysis is per-formed based on the previously identified requirements and combined with new ideas and issues that arose from the users' experience with version 1. Version 2 then is designed and implemented, and work immediately begins on the next version. This process continues until the system is complete or is no longer in use.

Phased development-based methodologies have the advantage of quickly getting a useful system into the hands of the users. Although the system does not perform all the functions the users need at first, it does begin to provide business value sooner than if the system were delivered aft er completion, as is the case with the waterfall and parallel methodologies. Likewise, because users begin to work with the system sooner, they are more likely to identify important additional requirements sooner than with structured design situations. Th e major drawback to phased development is that users begin to work with systems that are intentionally incomplete. It is critical to identify the most important and useful features and include them in the first version and to manage users' expectations along the way.

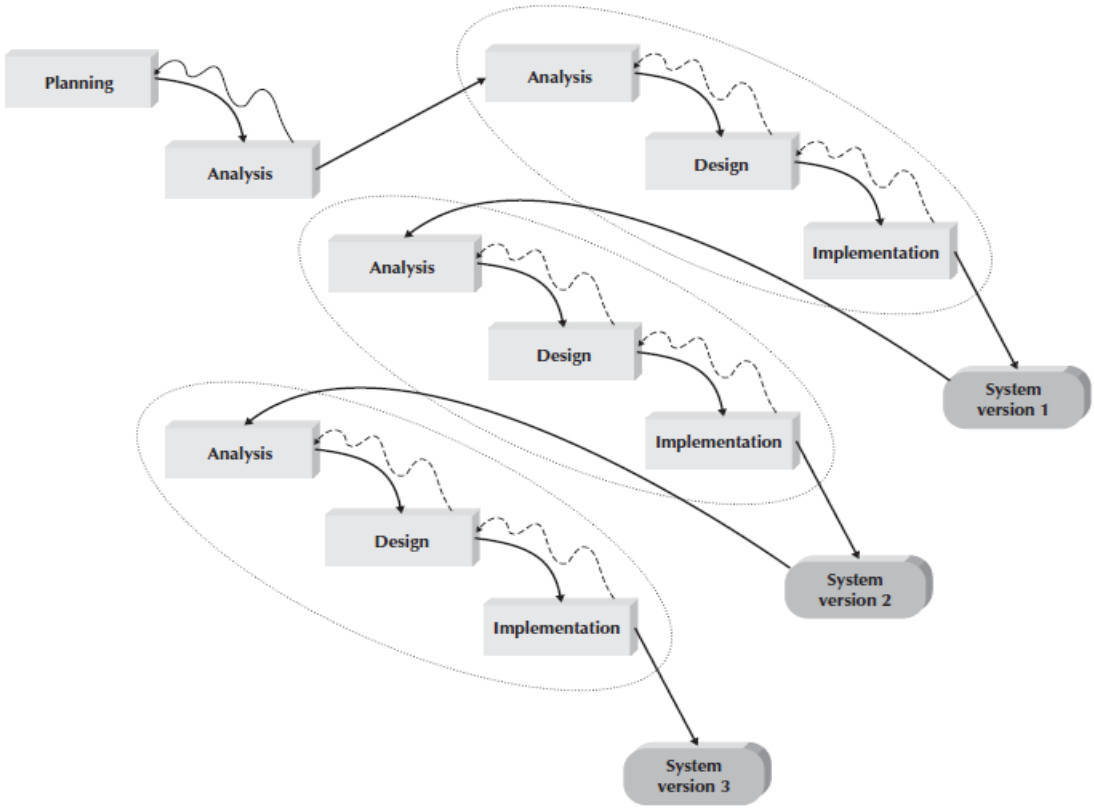

Figure 5: A Phased Development-Based Methodology

b) Prototyping

A *prototyping-based* methodology performs the analysis, design, and implementation phases concurrently, and all three phases are performed repeatedly in a cycle until the system is completed. With these methodologies, the basics of analysis and design are performed, and work immediately begins on a *system prototype,* a "quick-and-dirty" program that provides a minimal number of features. The first prototype is usually the first part of the system that is used. This is shown to the users and the project sponsor, who provide comments. These comments are used to reanalyze, redesign, and re-implement a second prototype, which provides a few more features. This process continues in a cycle until the analysts, users, and sponsor agree that the prototype provides enough functionality to be installed and used in the organization. After the prototype (now called the system) is installed, refinement occurs until it is accepted as the new system.

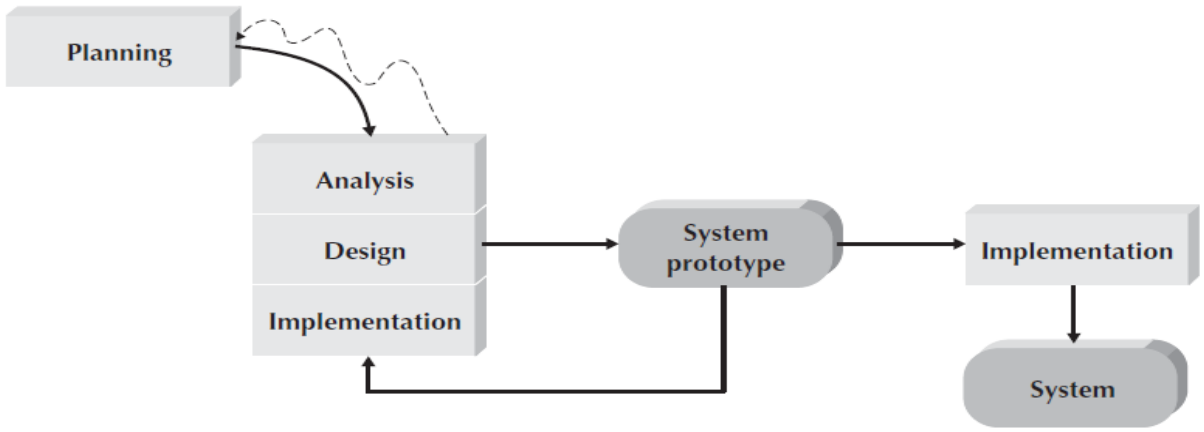

Figure 6: A Prototyping-based Methodology

The key advantage of a prototyping-based methodology is that it very quickly provides a system with which the users can interact, even if it is not ready for widespread organizational use at first. Prototyping reassures the users that the project team is working on the system (there are no long delays in which the users see little progress), and prototyping helps to more quickly refine real requirements.

c) Throwaway Prototyping

Throwaway prototyping–based methodologies are similar to prototyping-based methodologies in that they include the development of prototypes; however, throwaway prototypes are done at a different point in the SDLC. These prototypes are used for a very different purpose than those previously discussed, and they have a very different appearance. A system developed using this type of methodology relies on several design prototypes during the analysis and design phases. Each of the prototypes is used to minimize the risk associated with the system by confirming that important issues are understood before the real system is built. Once the issues are resolved, the project moves into design and implementation. At this point, the design prototypes are thrown away, which is an important difference between these methodologies and prototyping methodologies, in which the prototypes evolve into the final system.

Throwaway prototyping-based methodologies balance the benefits of well-thought-out analysis and design phases with the advantages of using prototypes to refine key issues before a system is built. It can take longer to deliver the final system as compared to prototyping-based methodologies, but this type of methodology usually produces more stable and reliable systems.

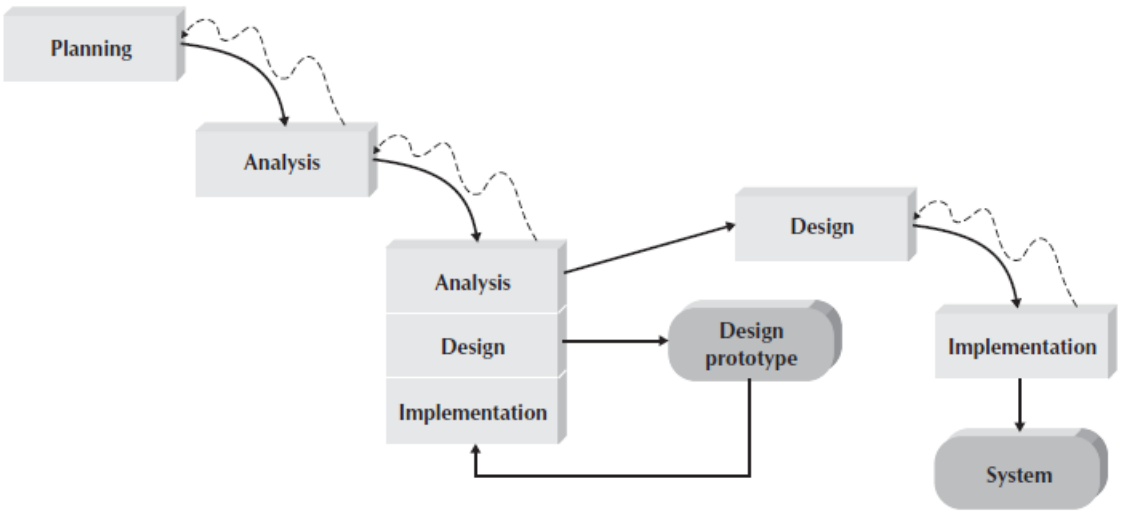

Figure 7: A Throwaway Prototyping-Based Methodology

3. Agile Development

A third category of systems development methodologies is still emerging today: agile development. All agile development methodologies are based on the agile manifesto and a set of twelve principles. The emphasis of the manifesto is to focus the developers on the working conditions of the developers, the working software, the customers, and addressing changing requirements instead of focusing on detailed systems development processes, tools, all-inclusive documentation, legal contracts, and detailed plans. These programming-centric methodologies have few rules and practices, all of which are fairly easy to follow. These methodologies are typically based only on the twelve principles of agile software. These principles include the following:

1) Software is delivered early and continuously through the development process, satisfying the customer.
2) Changing requirements are embraced regardless of when they occur in the development process.
3) Working software is delivered frequently to the customer.
4) Customers and developers work together to solve the business problem.
5) Motivated individuals create solutions; provide them the tools and environment they need, and trust them to deliver.
6) Face-to-face communication within the development team is the most efficient and effective method of gathering requirements.

7) The primary measure of progress is working, executing software.
8) Both customers and developers should work at a pace that is sustainable. Th at is, the level of work could be maintained indefinitely without any worker burnout.
9) Agility is heightened through attention to both technical excellence and good design.
10) Simplicity, the avoidance of unnecessary work, is essential.
11) Self-organizing teams develop the best architectures, requirements, and designs.
12) Development teams regularly reflect on how to improve their development processes.

Based on these principles, agile methodologies focus on streamlining the system-development process by eliminating much of the modeling and documentation overhead and the time spent on those tasks. Instead, projects emphasize simple, iterative application development.6 All agile development methodologies follow a simple cycle through the traditional phases of the systems development process. Virtually all agile methodologies are used in conjunction with object-oriented technologies. Two of the more popular examples of agile development methodologies are extreme programming (XP) and Scrum.

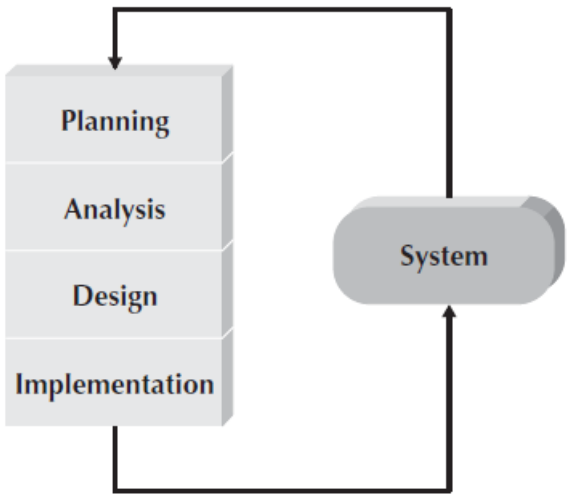

Figure 8: A Typical Agile Development Methodology

a) Extreme Programming

Extreme programming (*XP*) is founded on four core values: communication, simplicity, feedback, and courage. These four values provide a foundation that XP developers use to create any system. First, the developers must provide rapid feedback to the end users on a continuous basis. Second, XP requires developers to follow the KISS principle. Third, developers must make incremental changes

to grow the system, and they must not only accept change, they must embrace change. Fourth, developers must have a quality-first mentality. XP also supports team members in developing their own skills.

Three of the key principles that XP uses to create successful systems are continuous testing, simple coding performed by pairs of developers, and close interactions with end users to build systems very quickly. After a superficial planning process, projects perform analysis, design, and implementation phases iteratively.

Selecting the Appropriate Development Methodology:

Selecting a methodology is not simple, as no one methodology is always best. Many organizations have their own standards. Many organizations have standards and policies to guide the choice of methodology. You will find that organizations range from having one "approved" methodology to having several methodology options to having no formal policies at all. Figure 9 summarizes some important criteria for selecting a methodology.

	Structured Methodologies		RAD Methodologies			Agile Methodologies	
Ability to Develop Systems	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Excellent	Good	Good
That Are Reliable	Good	Good	Good	Poor	Excellent	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

Figure 9: Criteria for Selecting a Methodology

- **Clarity of user Requirements:**

RAD methodologies of prototyping and throwaway prototyping are usually more appropriate when user requirements are unclear as they provide prototypes for users to interact with early in the SDLC.

- **Familiarity with Technology:**

If the system is designed without some familiarity with the base technology, risks increase because the tools may not be capable of doing what is needed.

- **System complexity:**

Complex systems require careful and detailed analysis and design. Project teams who follow phased development-based methodologies tend to devote less attention to the analysis of the complete problem domain than they might if they were using other methodologies.

- **System Reliability:**

System reliability is usually an important factor in system development. Throwaway prototyping-based methodologies are most appropriate when system reliability is a high priority. Prototyping-based methodologies are generally not a good choice as they lack careful analysis and design phases.

- **Short Time Schedules:**

RAD-based methodologies are well suited for projects with short time schedules as they increase speed. Waterfall-based methodologies are the worst choice when time is essential as they do not allow for easy schedule changes.

- **Schedule Visibility:**

One of the greatest challenges in systems development is determining whether a project is on schedule. This is particularly true of the structured design methodologies because design and implementation occur at the end of the project. RAD-based methodologies move many of the critical design decisions earlier in the project; consequently, this helps project managers recognize and address risk factors and keep expectations high.

TYPICAL SYSTEMS ANALYST ROLES AND SKILLS

It is clear from the various phases and steps performed during the SDLC that the project team needs a variety of skills. Project members are change agents who identify ways to improve an organization, build an information system to support them, and train and motivate others to use the system. Six major skill sets an analyst should have include:

- Technical
- Business
- Analytical
- Interpersonal
- Management
- Ethical

In addition to these six general skill sets, analysts require many specific skills associated with roles performed on a project. In the early days of systems development, most organizations expected one

person, the analyst, to have all the specific skills needed to conduct a systems development project. Some small organizations still expect one person to perform many roles, but because organizations and technology have become more complex, most large organizations now build project teams containing several individuals with clearly defined responsibilities. Most IS teams include many other individuals, such as the programmers, who actually write the programs that make up the system, and technical writers, who prepare the help screens and other documentation (e.g., users manuals and systems manuals).

Categories of Analysts

1. *Business Analyst:* A business analyst focuses on the business issues surrounding the system.
2. *Systems Analyst:* A systems analyst focuses on the information system (IS) issues surrounding the system.
3. *Infrastructure Analyst:* An infrastructure analyst focuses on the technical issues surrounding how the system will interact with the organization's technical infrastructure (e.g., hardware, software, networks, and databases).
4. *Change Management Analyst:* A change management analyst focuses on the people and management issues surrounding the system installation
5. *Project Manager:* A project manager is responsible for ensuring that the project is completed on time and within budget and that the system delivers all benefits intended by the project sponsor