



INFORMATION SYSTEMS BUILDING BLOCKS

Information systems architecture provides a unifying framework into which various people with different perspectives can organize and view the fundamental building blocks of information systems.

There are three focuses for information system development activities (building blocks).

DATA - the raw material used to create useful information.

PROCESSES - the activities including the management that carries out the mission of the business.

INTERFACES - how the system interfaces with its users and other information systems.

- **DATA Building Blocks**
- Data can and should be thought of as raw material used to produce information.
- **Data Requirement** are a representative of users' data in terms of entities, attributes, relationships and rules. Data requirements should be expressed in a format that is independent of the technology that can or will be used to store data.

Process Building Block

- Processes deliver the functionality of an information system. Processes perform work in a system. People perform some processes. Machines, including computers, perform others.
- **Example** Typical business functions include SALES, SERVICE, MANUFACTURING, SHIPPING, RECEIVING, ACCOUNTING etc. Each function is ongoing.

- **Since a process is an** activity that respond to an event. Its governed by;
- **Process requirements** – a user's expectation of the processing requirements for a process and its information systems.
- **Policy** – a set of rules that govern a process.
- **Procedure** – a step-by-step set of instructions and logic for accomplishing a process.
- **Work flow** – the flow of transactions through processes to ensure appropriate checks and approvals are implemented.

Communication Building Blocks

- Communication as an information systems building block is concerned with;
 - Who (which units, employees, customers, and partners) must interact with the system?
 - Where are these units, employees, customers, and partners located?
 - What other information systems will the system have to interface with?

- Concerned with the technical design of both the user and the system-to-system communication interfaces.
- **Interface specifications** – technical designs that document how system users are to interact with a system and how a system interacts with other systems.
- **User dialogue** – a specification of how the user moves from window to window or page to page, interacting with the application programs to

Information Systems Development

- System development is done in the system life cycle. This entails;
- (1) systems development and (2) systems operation and maintenance.
- **System development** – Is a formalized approach to the systems development process; a standardized development process that defines a set of activities, methods, best practices, deliverables, and automated tools that system developers and project managers are to use to develop and continuously improve information systems and software.

Representative System Development Methodologies

- Architected Rapid Application Development (Architected RAD)
- Dynamic Systems Development Methodology (DSDM)
- Joint Application Development (JAD)
- Information Engineering (IE)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Structured Analysis and Design
- eXtreme Programming (XP)

Principles of System Development

- Get the system users involved.
- Use a problem-solving approach.
- Establish phases and activities.
- Document through development.
- Establish standards.
- Manage the process and projects
- Justify systems as capital investments.
- Don't be afraid to cancel or revise scope.
- Divide and conquer.
- Design systems for growth and change.

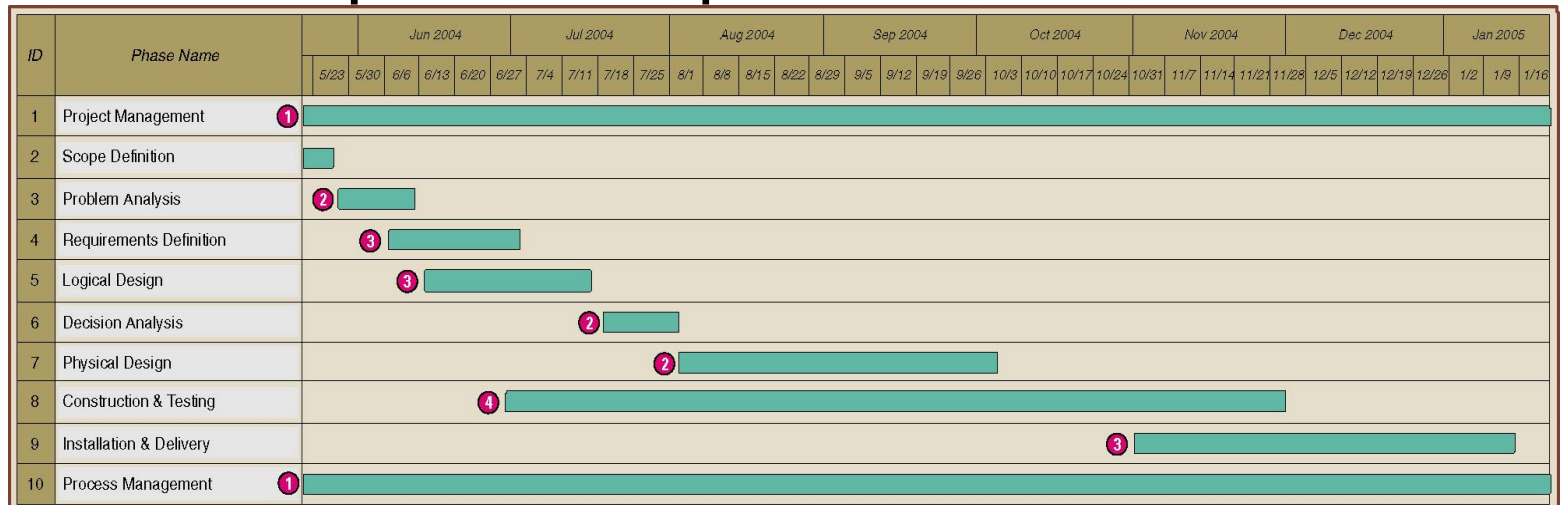
Use a Problem-Solving Approach

Classical Problem-solving approach

1. Study and understand the problem, its context, and its impact.
2. Define the requirements that must be met by any solution.
3. Identify candidate solutions that fulfill the requirements, and select the “best” solution.
4. Design and/or implement the chosen solution.
5. Observe and evaluate the solution’s impact, and refine the solution accordingly.

Establish Phases and Activities

- Every project is different depending on the size, complexity, and development methodology or route. The key point to emphasize is that the phases occur in parallel. It is important that analysts do not misinterpret that the phases in this chapter are sequential.



Manage the Process and Projects

Process management – an ongoing activity that documents, manages, oversees the use of, and improves an organization's chosen methodology (the “process”) for system development. Process management is concerned with phases, activities, deliverables, and quality standards should be consistently applied to all projects.

Project management is the process of scoping, planning, staffing, organizing, directing, and controlling a project to develop an information system at a minimum cost, within a specified time frame, and with acceptable quality.

Justify Information Systems as Capital Investments

Cost-effectiveness – The result obtained by striking a balance between the lifetime costs of developing, maintaining, and operating an information system and the benefits derived from that system. Cost-effectiveness is measured by a cost-benefit analysis.

Strategic information systems plan – a formal strategic plan (3-5 years) for building and improving an information technology infrastructure and the information system applications that use that infrastructure.

Strategic enterprise plan – a formal strategic plan (3-5 years) for an entire business that defines its mission, vision, goals, strategies, benchmarks, and measures of progress and achievement. Usually, the strategic enterprise plan is complemented by strategic business unit plans that define how each business unit will contribute to the enterprise plan. The information systems plan is one of those unit-level plans.

Don't Be Afraid to Cancel or Revise Scope

Creeping commitment – a strategy in which feasibility and risks are continuously reevaluated throughout a project. Project budgets and deadlines are adjusted accordingly.

Risk management – the process of identifying, evaluating, and controlling what might go wrong in a project before it becomes a threat to the successful completion of the project or implementation of the information system. Risk management is drive by risk analysis or assessment.

Where Do Systems Development Projects Come From?

- **Problem** – an undesirable situation that prevents the organization from fully achieving its purpose, goals, and/or objectives.
- **Opportunity** – a chance to improve the organization even in the absence of an identified problem.
- **Directive** - a new requirement that is imposed by management, government, or some external influence.

THE SOFTWARE DEVELOPMENT LIFE CYCLE

- Developing an information system is usually a large project. All projects need to be planned and managed. The SDLC shows the main activities normally associated with information systems development. It shows where to start, what to do next and where to end.
- Systems development is divided into phases
- Problem Identification
- Feasibility Study
- Systems Analysis
- System Design
- System Construction
- Systems Implementation
- Systems Maintenance and Review

The SDLC

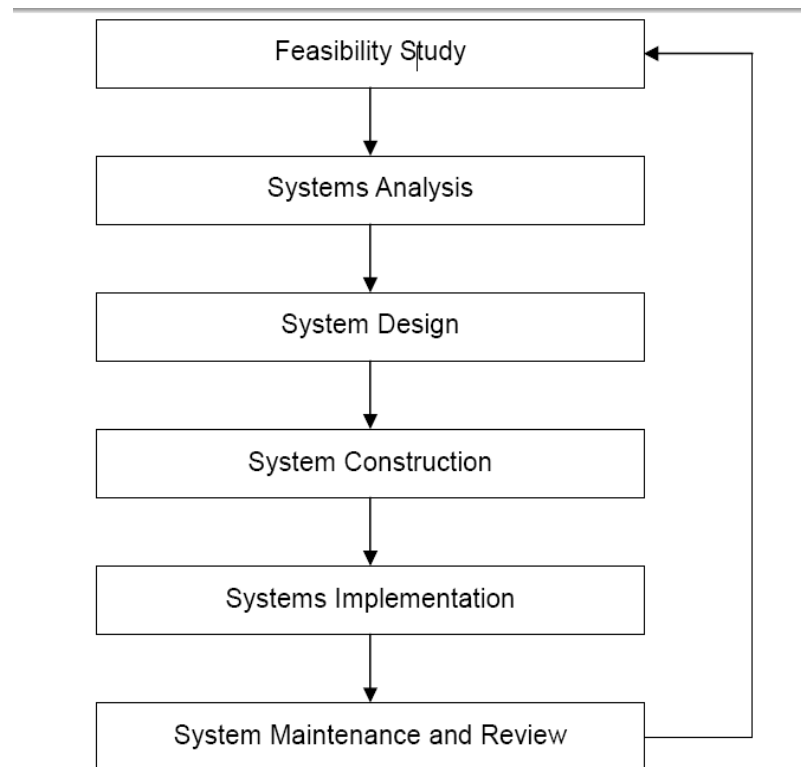





Figure 2.1: The Basic SDLC

- 
- This is a highly structured approach and implies that one phase cannot start until the previous one ends. Each phase has some sort of deliverable as an output which, in turn, acts as an input to the next phase.
 - ***Feasibility Study***

Before any project can start in earnest, it is essential to find out whether it is feasible or not.

- Basically a feasibility study will look at technical, personnel and cost issues and examine different ways in which the system can be developed.
- For example – can the system be developed in-house or will outsourcing be required, does the organisation have the necessary technical resources and expertise to undertake the project, will the system be of financial benefit to the organisation and is the money available to develop it?
- The output from this phase is a **feasibility report**, which summarizes the study and makes recommendations about the way forward.

- 
- This phase consists of a detailed investigation of the requirements of the system, invariably involving extensive consultation with the users of the system.
 - If the new system is to replace an old one, then it is normal to study the existing system in depth so that its objectives, outputs and the exact way in which it works can be understood, as well as identifying any problems associated with it so that these are not repeated.

- 
- At the same time, it is necessary to find out what new features and functionality should be included in the new system.
 - The output from this phase is a **requirements specification**.


System Design

- This phase takes the requirements specification and converts it into a **system design specification**. This involves the design of inputs, outputs, databases, computer programs and user interfaces.
- The system design specification contains all the detail required for the system builders to construct the system.

- **The design phase** is normally split into logical and physical design. **Logical design** concentrates on the business aspects of the system and is theoretically independent of any hardware or software.
- **Physical design** takes the logical design specification and applies it to the implementation environment. Most often the choice of programming language and database is already decided and these technologies are taken into account in physical design.

System Construction

- This phase is where the system is actually built. The system specifications are turned into a working system by writing, testing and, in due course, documenting the programs which will make up the whole system.
- Once the individual programs have been tested, the whole system needs to be put together and tested as a whole.


- 
- This whole phase requires extensive user involvement.
 - The output from this phase consists of detailed **program and file specifications** which, in total, describe exactly how the new system works.

Systems Implementation

- The objective of this phase is to produce a fully functioning and documented system. It involves training users, transferring data from the old system to the new and actually putting the new system into operation – "going live".
- A final system evaluation will also need to be performed to make sure the system works according to expectations.

System Maintenance and Review

- During the life of a system, continual review and maintenance will need to be performed in order to maintain its functionality. For example, new requirements may need to be implemented and errors in the system need to be rectified. Such maintenance is really a repetition of the other phases of the life cycle as a new requirement or a fix for an error needs to be analyzed, designed and implemented.

- 
- Eventually all systems become outdated and need to be replaced, so the cycle starts again, with the way in which the old system is operating and the requirements which now apply forming the backdrop to a new feasibility study to examine whether a new system should be developed.

- **The basic SDLC in Figure 2.1 implies a purely linear** approach in that one phase finishes before another one starts and there is no going back. In practice, of course, this is unrealistic.
- When working through a phase, it is often the case that something does not work out as planned or that there is an error or omission in the previous phase. It is, therefore, necessary to go back and modify the previous phase.

- So, there is an iterative nature to the SDLC, and this is shown in Figure 2.2. This approach is often called the **Waterfall Model**, and the dotted lines on the diagram show the interaction between phases, with the possibility of returning to a previous phase to make adjustments always available.

The Waterfall Model

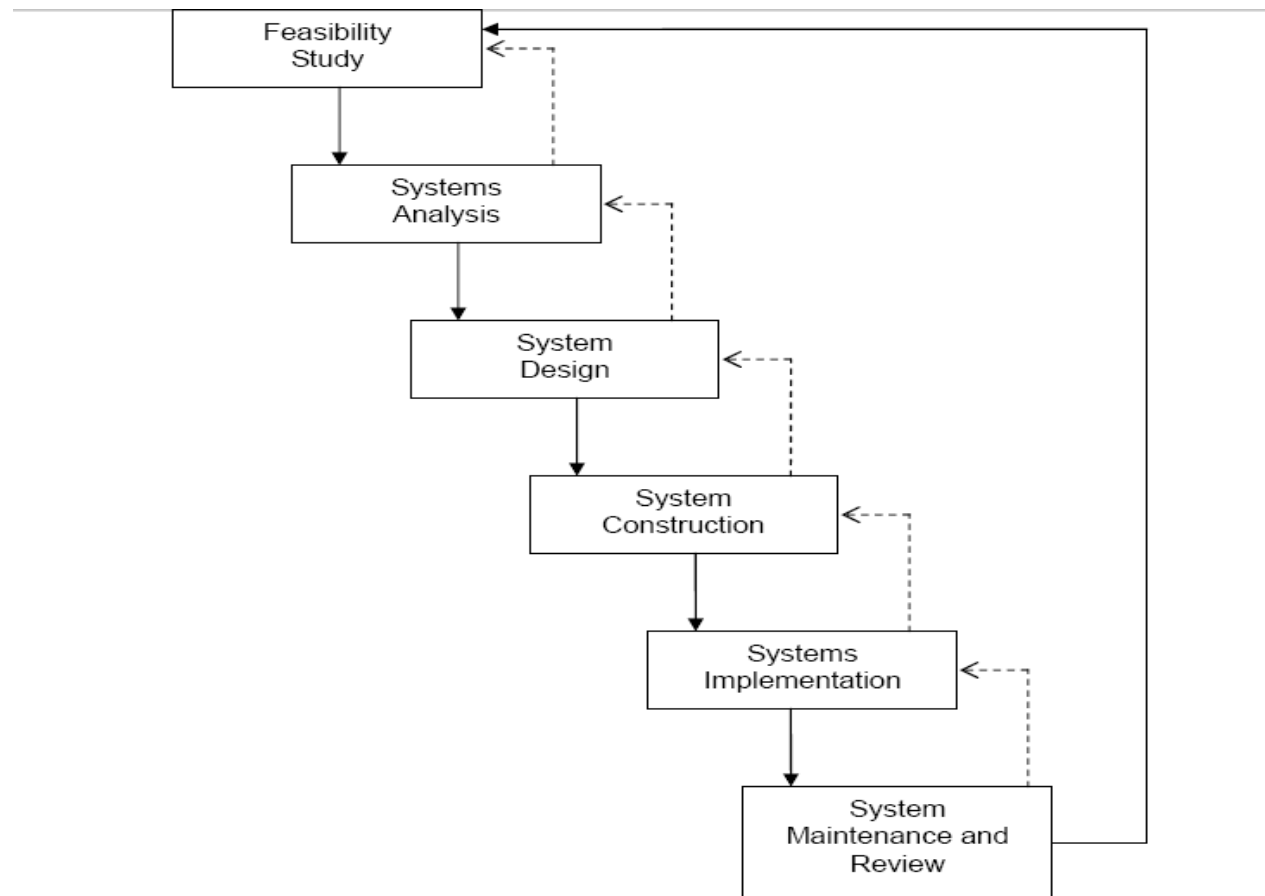



Figure 2.2: The Waterfall Model

Advantages of the SDLC

- ***Project management***
- Breaking down a project into phases has distinct advantages in that each stage can be specified, planned and evaluated before moving on to the next planned stage. This enables the whole project to be closely managed, and is the same approach extensively used in engineering projects such as building a bridge, and the same principles apply.
- ***Documentation***
- Each stage has its own documentation standards and these make the quality assurance processes much easier.

- 
- ***Tried and Tested Techniques***
 - Data flow diagrams, entity relationship diagrams and entity life histories are all widely accepted techniques which have been used in thousands of projects.

Disadvantages of the SDLC

- ***Inflexibility***
- Because the SDLC has distinct, sequential phases it was regarded as inflexible, especially for smaller systems.
- ***Narrow focus***
- The SDLC tended to be used to develop low level operational systems and largely ignored the needs of middle and senior management.
- ***Old-fashioned techniques***
- ***Time and cost overruns***

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.

Structured Design

- The first category of systems development methodologies is called *structured design*.
- These methodologies became dominant in the 1980s, replacing the previous, ad-hoc, and undisciplined approach.
- 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 as outlined below.

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.

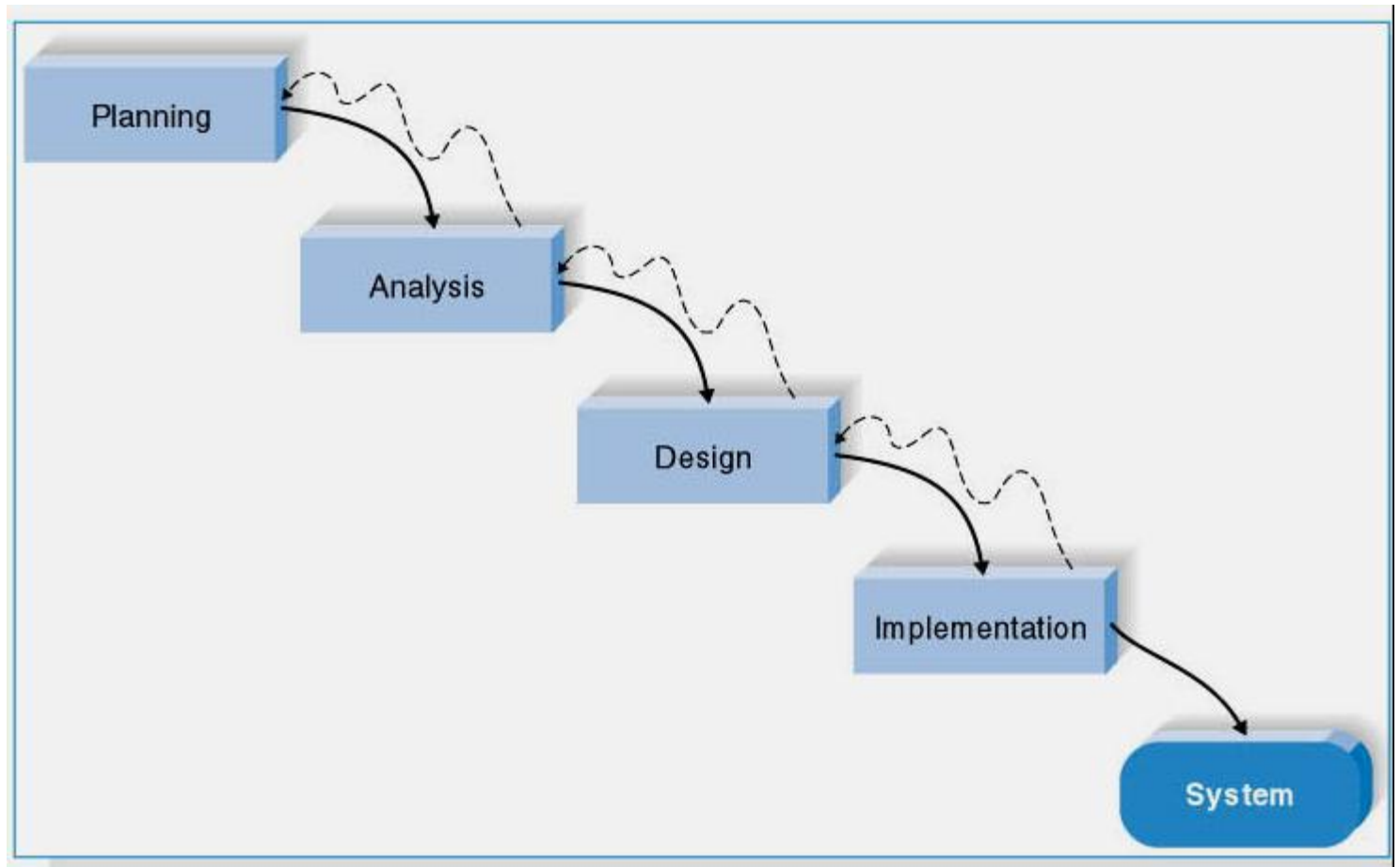
The key deliverables for each phase are typically very long (often hundreds of pages in length) and are presented to the project sponsor for approval as the project moves from phase to phase.




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 and costly.




- 
- 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).

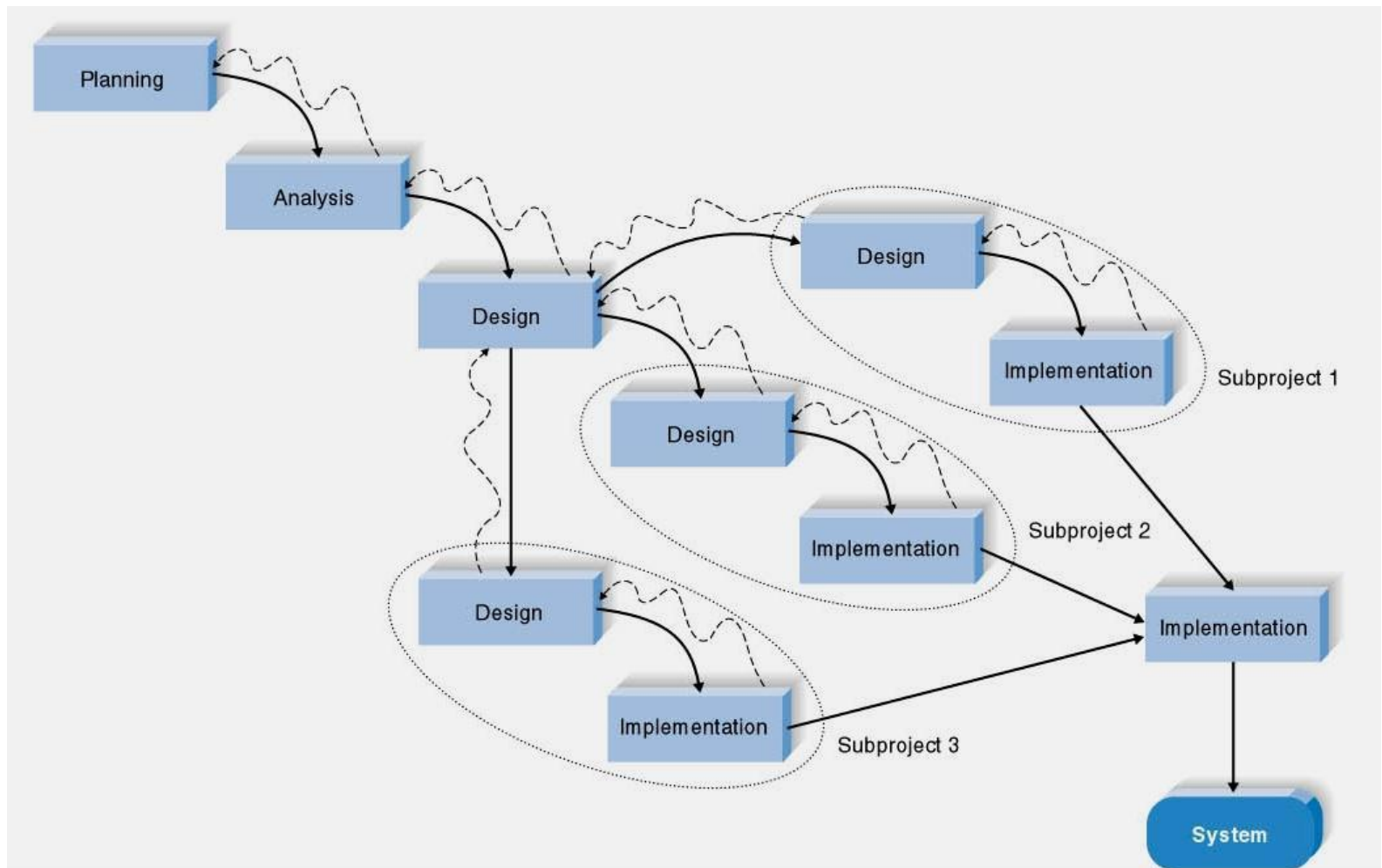
b). Parallel Development

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

- 
- 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.



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 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

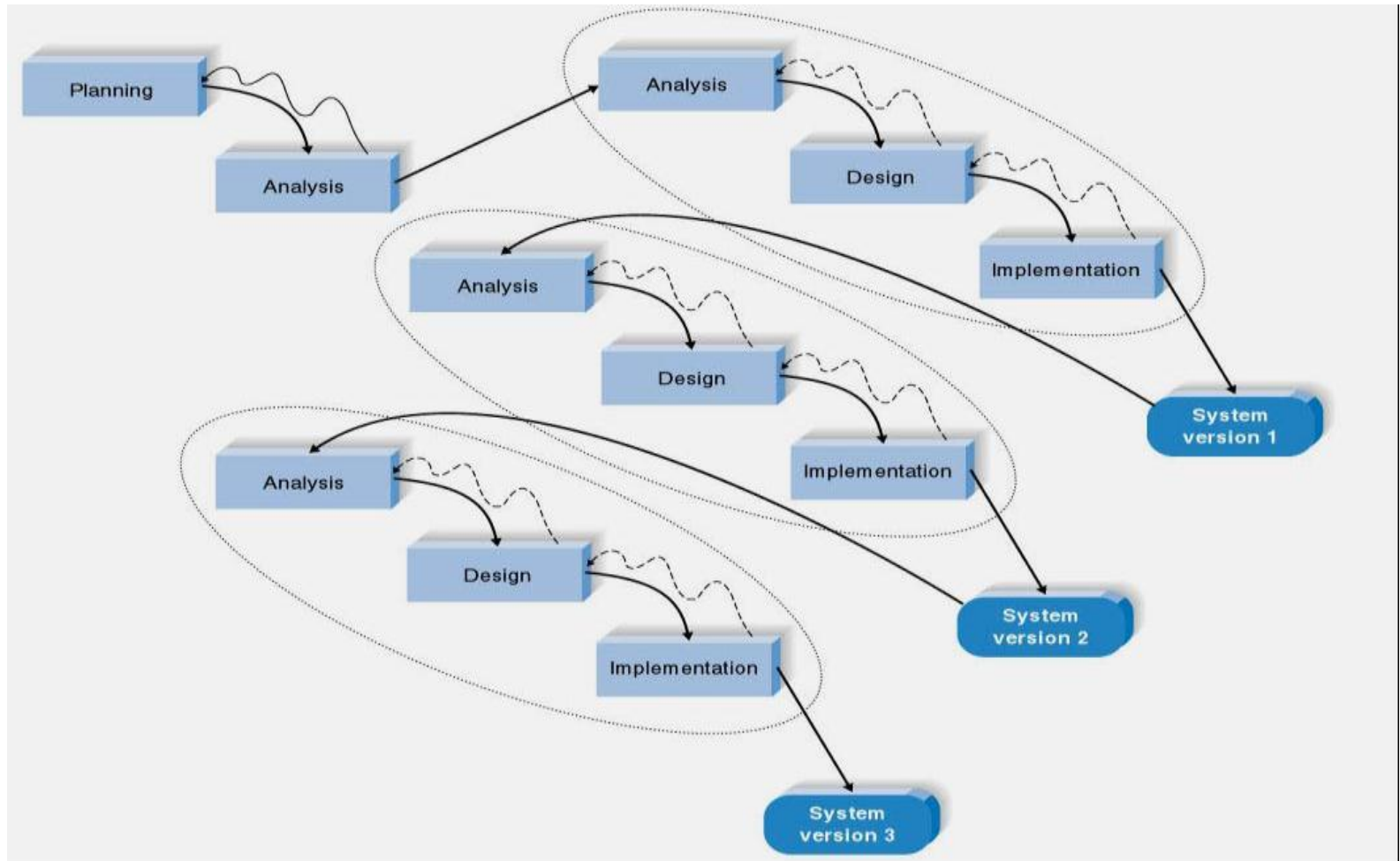
A ***phased development***-based methodology breaks an overall system into a series of *versions*, which are developed sequentially.

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


- 
- Once version 1 is implemented, work begins on version 2.
 - Additional analysis is performed 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.

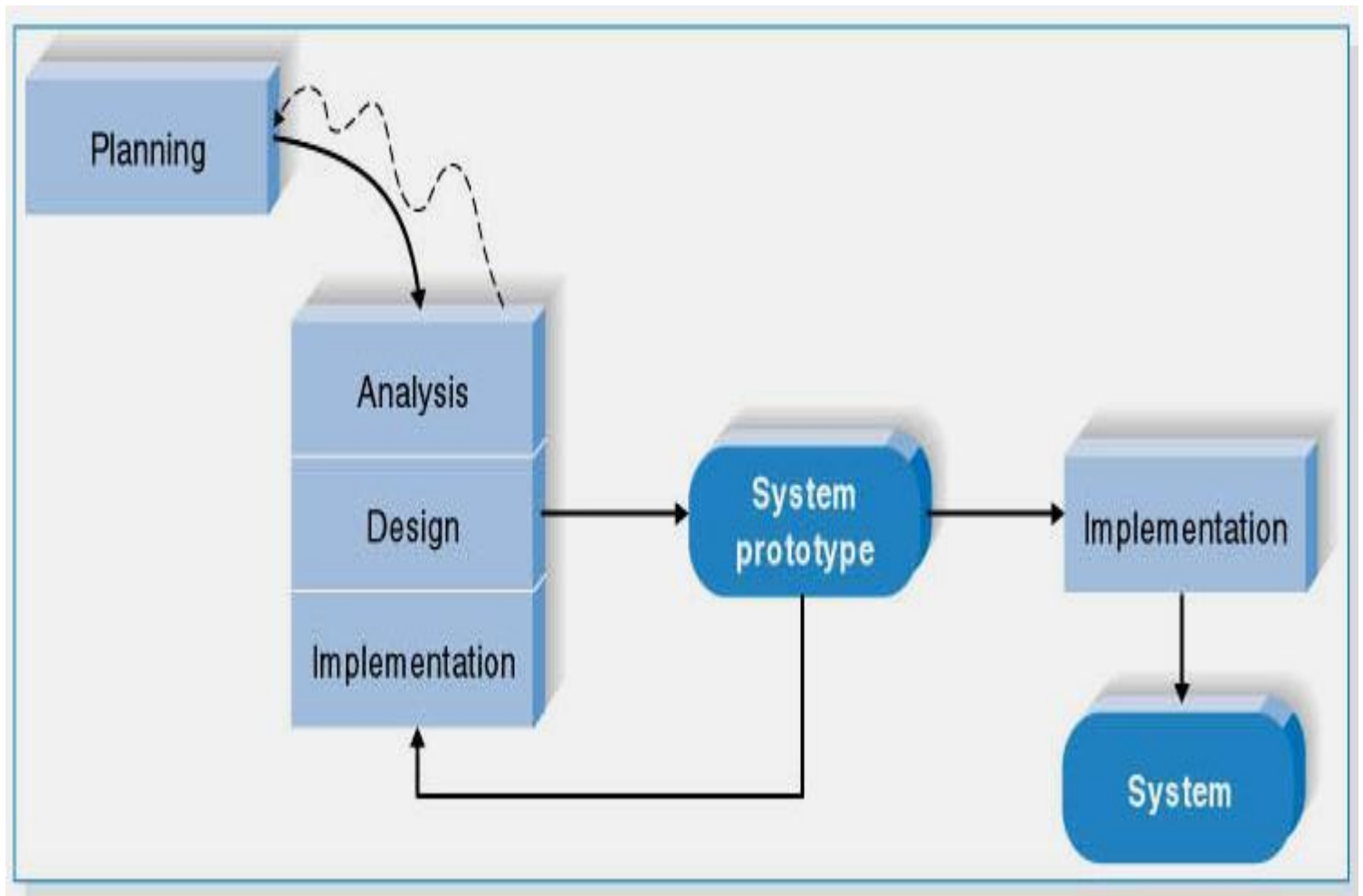



b). Prototyping

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 amount 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




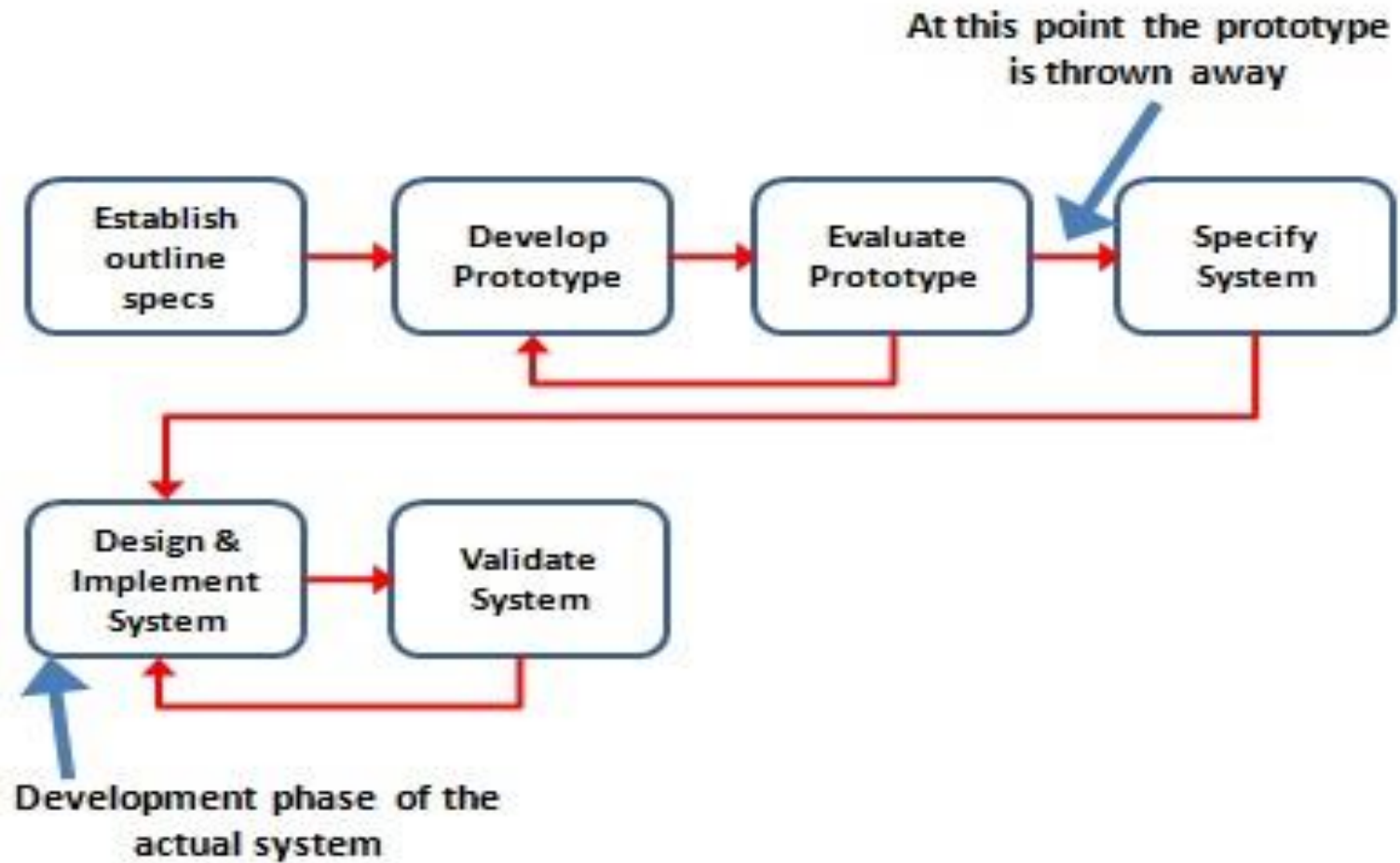
- 
- **Advantage:** Provides a system for the users to interact with, even if it is not initially ready for use.
 - **Disadvantage:** Often the prototype undergoes such significant changes that many initial design decisions prove to be poor ones.

- **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


- The throw-away prototype sounds like what it is. You make a prototype, then when you are done with it, you abandon it.
- For example, you and your colleagues go to lunch, you come up with an idea of the new product. When you discuss the idea, write down the idea and pull out the original design of the paper napkin.
- When you return to the office, take out the prototype of the paper napkin and transfer it to your computer. You throw out the napkin.
- In that case, the napkin is considered a thrown away prototype.

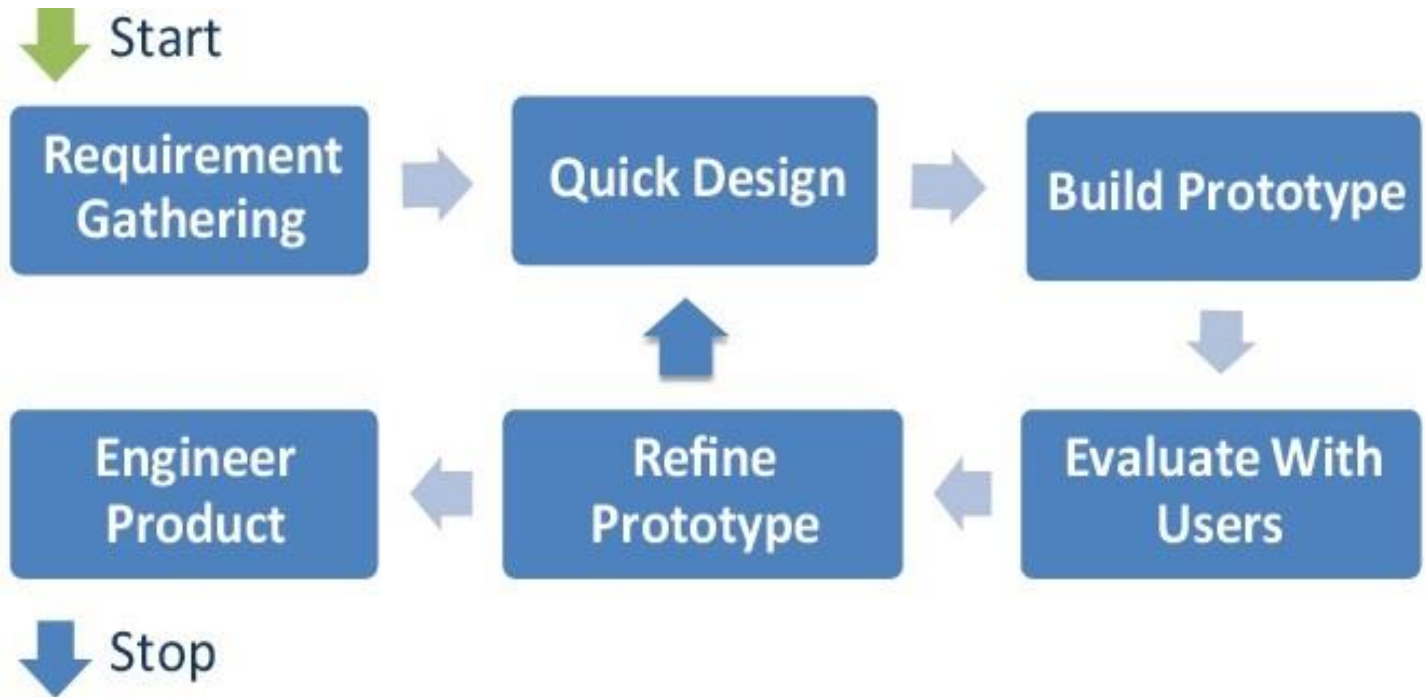
- 
- The throw-away approach is most appropriate in the project acquisition stage, where prototypes demonstrate the feasibility of new concepts and convince potential sponsors to fund the proposed development projects.
 - In this situation, available resources are limited and the ability to convey the benefits of a new approach with a very low-cost demonstration is essential to create a new project.

- 
- The throw-away approach can be a drawback for insufficient level of technology and is best suited for rough system mockups used very early in the project.



- **Evolutionary Prototyping**
- In evolutionary prototyping, the concept of the system will be developed as the project progresses.
- First of all, we will develop the most visual aspect of the system.
- You present a portion of the system to the customer and continue to develop prototypes based on the feedback received.
- At some point, you and the customer agree that the prototype is “good enough” and release the prototype as the final product.

- 
- If evolved prototyping does not provide more control than is necessary or if you already know what the system should do, you can use evolutionary delivery or gradual delivery instead.



Prototyping Model

3. Agile Development

A third category of systems development methodologies is still emerging today: *agile development*.

These programming-centric methodologies have few rules and practices, all of which are fairly easy to follow.

They focus on streamlining the SDLC by eliminating much of the modeling and documentation overhead and the time spent on those tasks.

Instead, projects emphasize simple, iterative application development. Examples of agile development methodologies include extreme programming, Scrum, and the Dynamic Systems Development Method (DSDM). The agile development approach, as described next, typically is used in conjunction with object-oriented methodologies.

a). Extreme Programming


Extreme programming (XP) is founded on four core values: communication, simplicity, feedback, and courage.

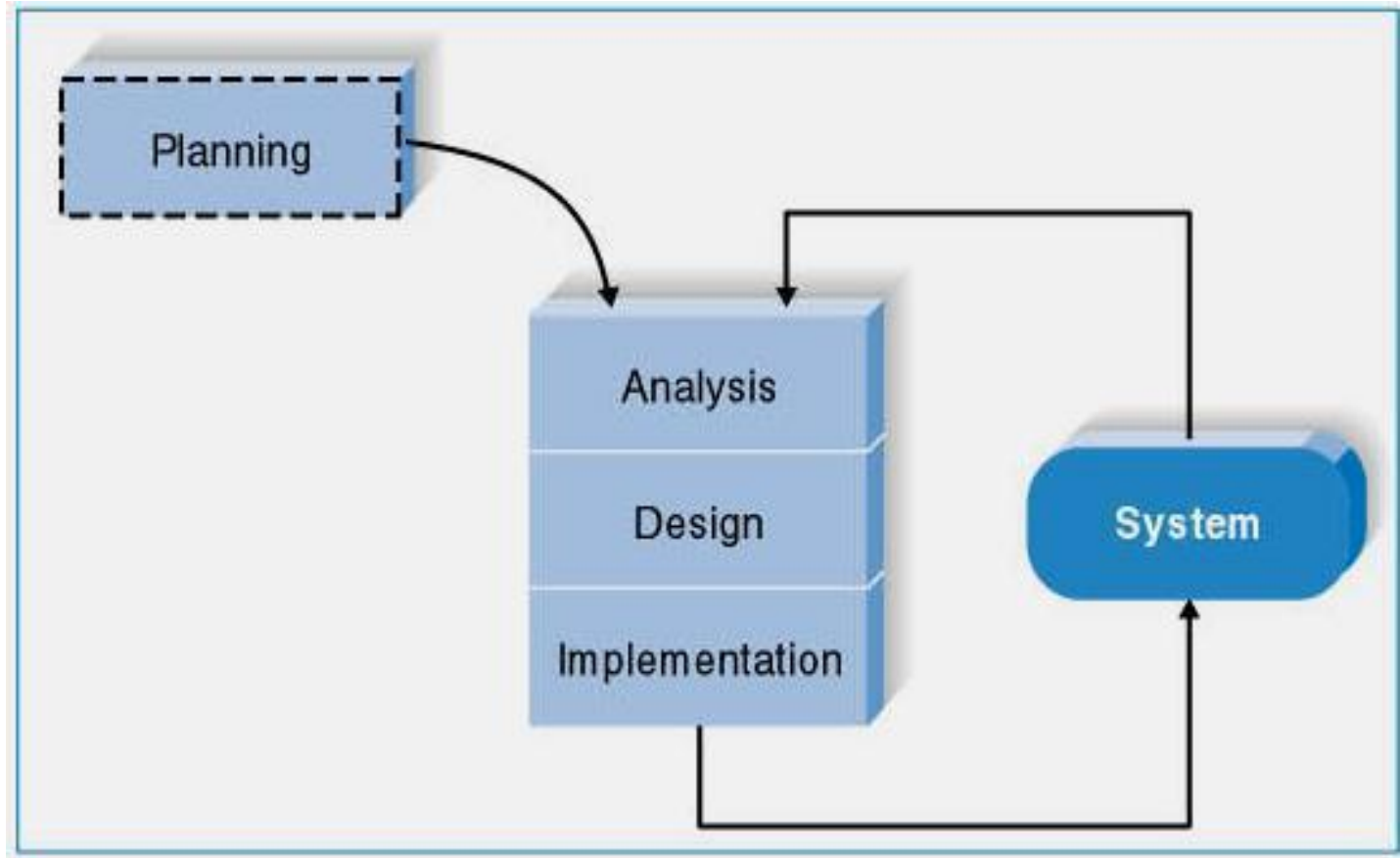
These four values provide a foundation that XP developers use to create any system. First, the developers must provide rapid feedback to the end users on a continuous basis.


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.
 - The next figure summarizes some important methodology selection criteria.

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