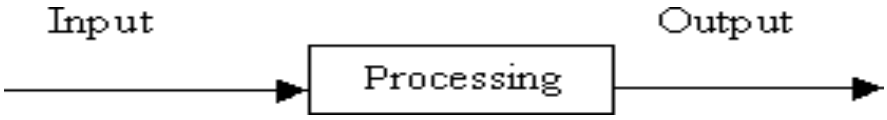
# SYSTEM DEVELOPMENT METHODOLOGIES

## DEFINITION OF A SYSTEM

A system is a collection of interrelated components that work together to realize some common objective. A system can be part of a large system and at the same time may have its own sub systems. Basically, there are three major components in every system, namely input, processing and output.




## DEFINITION OF AN INFORMATION SYSTEM

An information system is a system of interrelated components working together to collect, process, store, and disseminate information to support decision making, coordination, control, analysis, and visualization in an organization. Examples of IS in Organizations; Accounting IS, Human Resource IS, Finance IS, Management Information Systems etc.

## IS DEVELOPMENT METHODOLOGIES

Refers to the framework that is used to build, structure, plan, and control the process of developing an IS. System design methodologies are a discipline within the software development industry that seek to provide a framework for activity capture, storage, transformation and dissemination of information, so as to enable the development of computer systems that are fit for a purpose. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. However, one system development methodology is not necessarily suitable for use by all projects hence requires a thorough analysis.

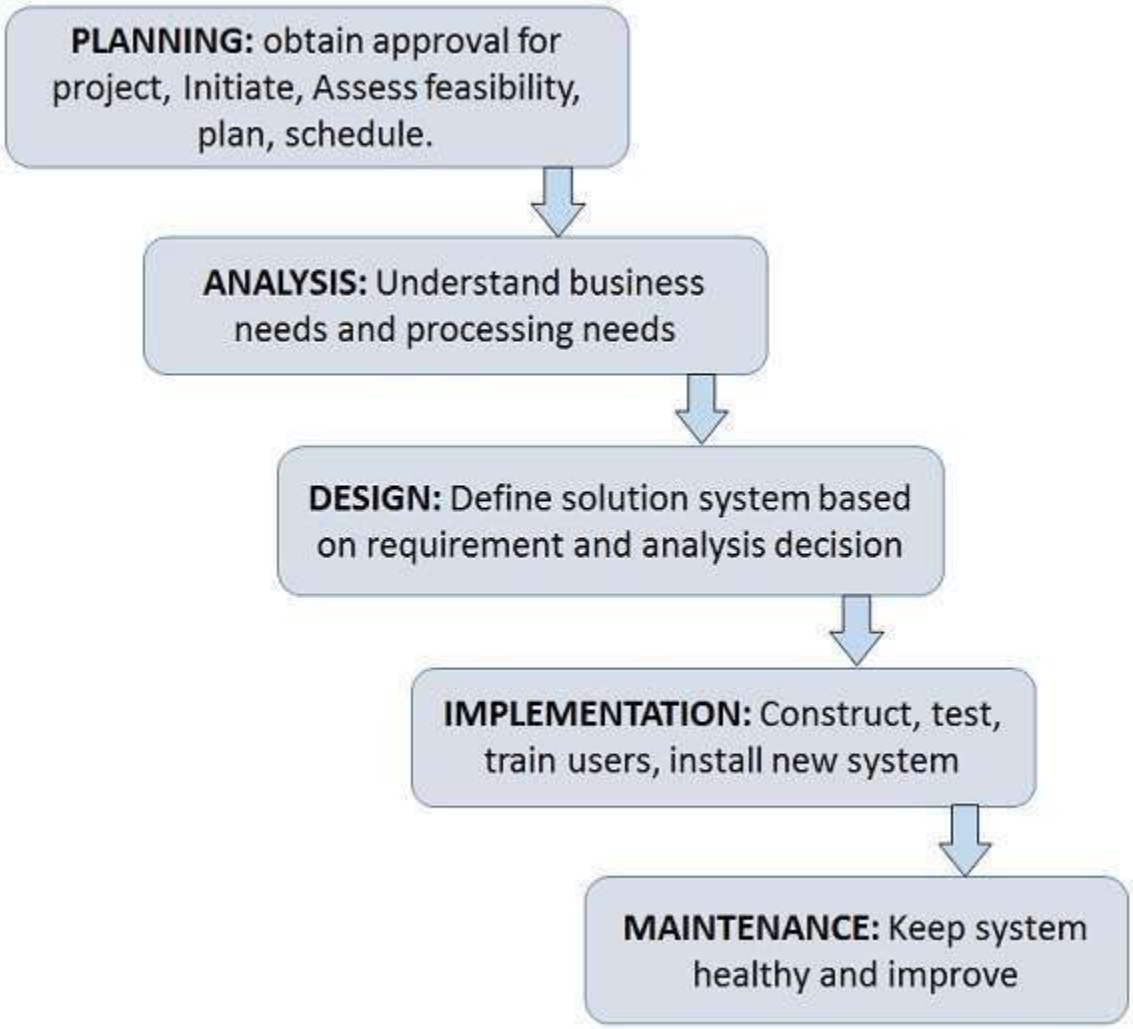
## SYSTEMS DEVELOPMENT LIFE CYCLE

The system-development life cycle enables users to transform a newly-developed project into an operational one. The System Development Life Cycle, "SDLC" for short, is a multistep, iterative process, structured in a methodical way. This process is used to model or provide a framework for technical and non-technical activities to deliver a quality system which meets or exceeds a business' expectations or manage decision-making progression.

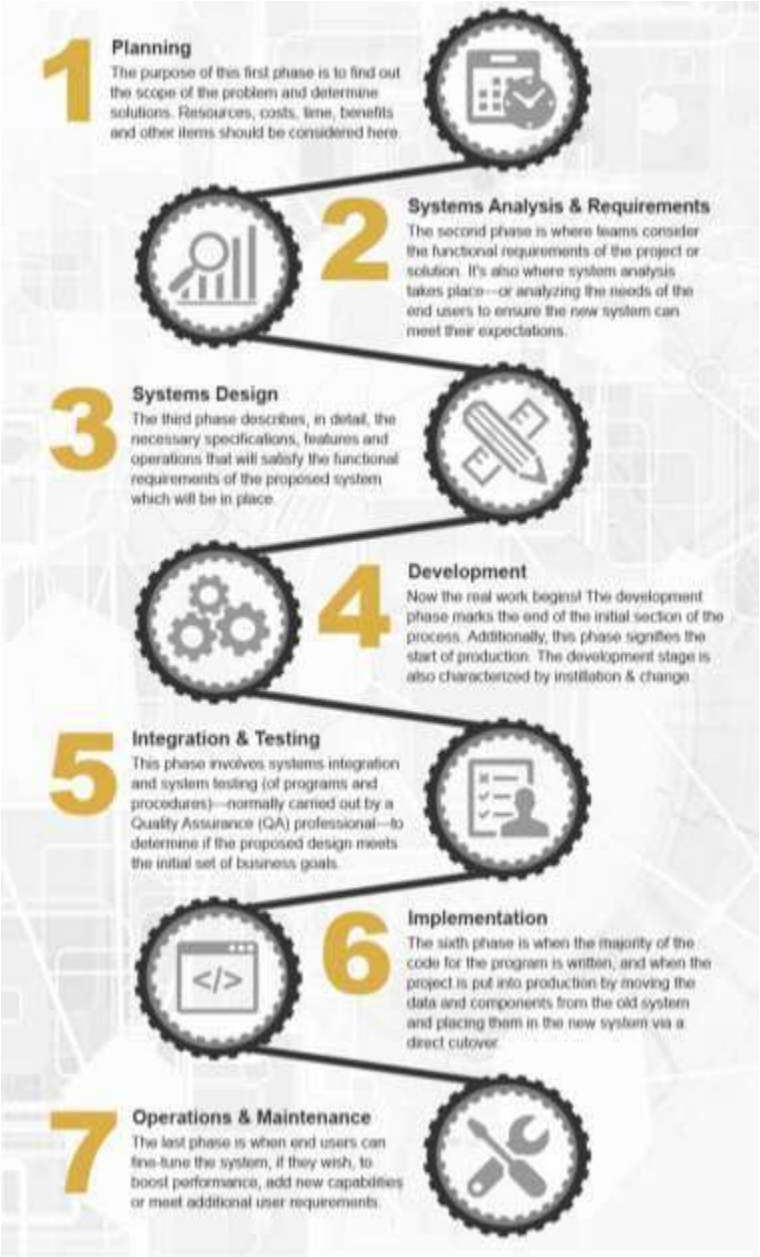
The SDLC highlights different stages (phrases or steps) of the development process. The life cycle approach is used so users can see and understand what activities are involved

within a given step. It is also used to let them know that at any time, steps can be repeated or a previous step can be reworked when needing to modify or improve

Traditionally, the systems-development life cycle consisted of five stages as shown in the figure below.




The five stages have been increased to seven phases over time. Increasing the number of steps helped systems analysts to define clearer actions to achieve specific goals.




The seven phases of the SDLC are described as follows:

### 1. Planning

This is the first phase in the systems development process. It identifies whether or not there is the need for a new system to achieve a business' strategic objectives. This is a preliminary plan (or a feasibility study) for a company's business initiative to acquire the resources to build on an infrastructure to modify or improve a service. The company might be trying to meet or exceed expectations for their employees, customers and

stakeholders too. The purpose of this step is to find out the scope of the problem and determine solutions. Resources, costs, time, benefits and other items should be considered at this stage. Specifically, in this stage the following activities are done:

- Define the problem and scope of existing system.
- Overview the new system and determine its objectives.
- Confirm project feasibility and produce the project Schedule.
- During this phase, threats, constraints, integration and security of system are also considered.
- A feasibility report for the entire project is created at the end of this phase.

**2. Systems Analysis**

The second phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goal(s) of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place—or analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business' needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected.

A feasibility analysis is also done at this stage to see whether the system development is worth undertaking. There are mainly five types of feasibilities checks:

- Economic: Can we complete the project within the budget or not?
- Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
- Operation feasibility: Can we create operations which is expected by the client?
- Technical: Need to check whether the current computer system can support the software
- Schedule: Decide that the project can be completed within the given schedule or not

During this stage the following is accomplished:

- Gather, analyze, and validate the information.
- Define the requirements and prototypes for new system.
- Evaluate the alternatives and prioritize the requirements.
- Examine the information needs of end-user and enhances the system goal.
- A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

There are several tools businesses can use that are specific to the second phase. They include:

- CASE (Computer Aided Systems/Software Engineering)
- Requirements gathering
- Structured analysis

**3. Systems Design.**

The third phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific business information needs for the proposed system. It's during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives.

The Design phase models the way a software application will work. Some aspects of the design include:

i. Architecture – Specifies programming language, industry practices, overall design, and use of any templates or boilerplate
ii. User Interface – Defines the ways customers interact with the software, and how the software responds to input
iii. Platforms – Defines the platforms on which the software will run, such as Apple, Android, Windows version, Linux, or even gaming consoles
iv. Programming – Not just the programming language, but including methods of solving problems and performing tasks in the application
v. Communications – Defines the methods that the application can communicate with other assets, such as a central server or other instances of the application
vi. Security – Defines the measures taken to secure the application, and may include SSL traffic encryption, password protection, and secure storage of user credentials

There are two kinds of design documents developed in this phase:

**Logical design or High-Level Design (HLD)**

- This is a brief description and name of each module.
- It is an outline about the functionality of every module.
- It describes the Interface relationship and dependencies between modules
- Identifies Database tables along with their key elements
- Complete architecture diagrams along with technology details

**Physical design or Low-Level Design (LLD)**

- Functional logic of the modules is defined

- Actual database tables, which include type and size
- Complete detail of the interface designs
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

**4. Development**

The fourth phase is when the real work begins—in particular, when a programmer, network engineer and/or database developer are brought on to do the major work on the project. This work includes using a flow chart to ensure that the process of the system is properly organized. The development phase marks the end of the initial section of the process. Additionally, this phase signifies the start of production. The development stage is also characterized by instillation and change. Focusing on training can be a huge benefit during this phase.

**5. Integration and Testing**

The fifth phase involves systems integration and system testing (of programs and procedures)—normally carried out by a Quality Assurance (QA) professional—to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program's successful completion.

**6. Implementation**

The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly-developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct cutover. While this can be a risky (and complicated) move, the cutover typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes.

**7. Operations and Maintenance**

The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements.

**Importance of the SDLC**

It is important to have an SDLC in place

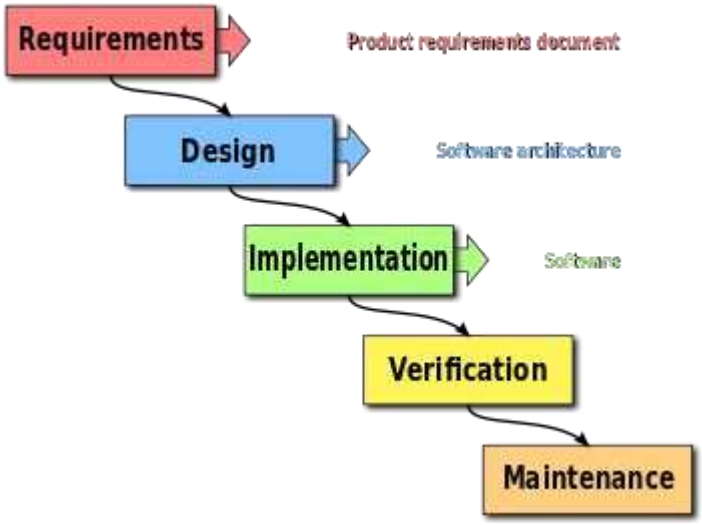
- It helps to transform the idea of a project into a functional and completely operational structure.
- In addition to covering the technical aspects of system development, SDLC helps with process development, change management, user experience, and policies.
- Another benefit of an SDLC is that it allows for planning ahead of time, determine costs and staffing decisions, define goals, measure performance, and validate points at each phase of the cycle to boost the quality of the final product.

**System Development Methodologies/Approaches**

There is flexibility within the SDLC. In recent decades a number of different methodologies have gained popularity. These include the following:

**1. Waterfall**

The waterfall approach is one of the oldest SDLC methodology, but it has fallen out of favor in recent years. This methodology involves a rigid structure that demands all system requirements be defined at the very start of a project. Only then can the design and development stages begin. Once development is complete, the product is tested against the initial requirements and rework is assigned. Companies in the software industry typically need more flexibility than what the waterfall methodology offers, but it still remains a strong solution for certain types of projects especially government contractors.




The waterfall method is a rigid linear model that consists of sequential phases (requirements, design, implementation, verification, maintenance) focusing on distinct goals. Each phase must be 100% complete before the next phase can start. There's usually no process for going back to modify the project or direction.

**Pros:**

- The linear nature of the waterfall development method makes it easy to understand and manage.
- Projects with clear objectives and stable requirements can best use the waterfall method.
- Less experienced project managers and project teams, as well as teams whose composition changes frequently, may benefit the most from using the waterfall development methodology.
- Ideal for supporting less experienced project teams and project managers, or project teams whose composition fluctuates.
- The orderly sequence of development steps and strict controls for ensuring the adequacy of documentation and design reviews helps ensure the quality, reliability, and maintainability of the developed software.
- Progress of system development is measurable

**Cons:**

- The waterfall development method is often slow and costly due to its rigid structure and tight controls.
- These drawbacks can lead waterfall method users to explore other software development methodologies.
- Inflexible, slow, costly and cumbersome due to significant structure and tight controls.
- Little room for use of iteration, which can reduce manageability if used.
- Depends upon early identification and specification of requirements, yet users may not be able to clearly define what they need early in the project.
- Problems are often not discovered until system testing.
- System performance cannot be tested until the system is almost fully coded
- Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.
- Produces excessive documentation and keeping it updated as the project progresses is time-consuming.
- Written specifications are often difficult for users to read and thoroughly appreciate.
- Promotes the gap between users and developers with clear division of responsibility.

**Situations where most appropriate**

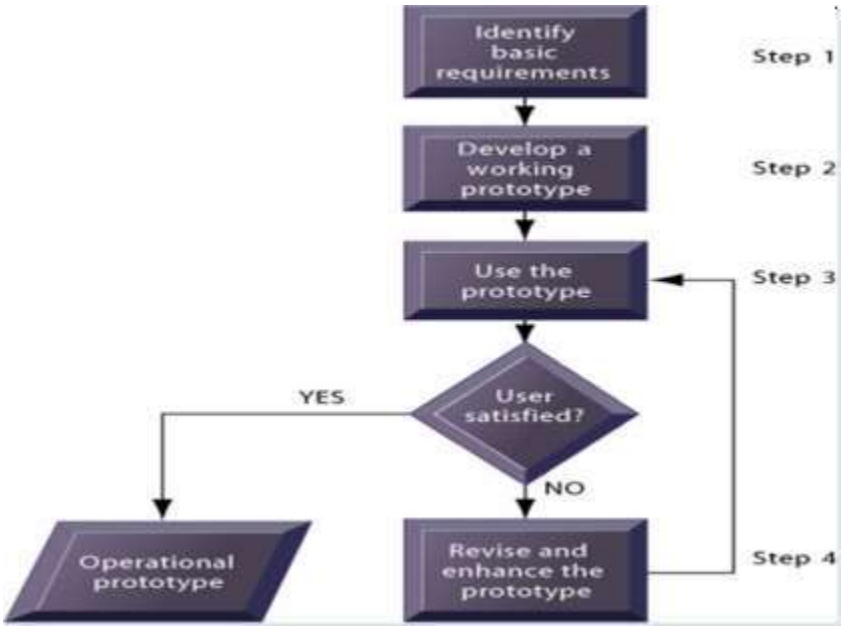
- System performance cannot be tested until the system is almost fully coded

- Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.
- Produces excessive documentation and keeping it updated as the project progresses is time-consuming.
- Written specifications are often difficult for users to read and thoroughly appreciate.
- Promotes the gap between users and developers with clear division of responsibility.

**2. Prototyping**

The prototyping methodology is a systems development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved from which the complete system or product can be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.

Steps of the prototyping model




In most cases, the steps of the prototyping model are as follows:

1. The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the departments or aspects of the existing system.

2. A preliminary, simple design is created for the new system. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
3. The users thoroughly evaluate the first prototype and note its strengths and weaknesses, what needs to be added and what should to be removed. The developer collects and analyzes the remarks from the users.
4. The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed. The second prototype is evaluated in the same manner as was the first prototype.

Steps 3 and 4 are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired. The final system is constructed, based on the final prototype. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

### Types of prototype models

There are a few types of prototype models that can be implemented by development teams based on their needs:

- Rapid throwaway- This method involves exploring ideas by quickly developing a prototype based on preliminary requirements that is then revised through customer feedback. The name rapid throwaway refers to the fact that each prototype is completely discarded and may not be a part of the final product.
- Evolutionary- This approach uses a continuous, working prototype that is refined after each iteration of customer feedback. Because each prototype is not started from scratch, this method saves time and effort.
- Incremental- This technique breaks the concept for the final product into smaller pieces, and prototypes are created for each one. In the end, these prototypes are merged into the final product.
- Extreme- This prototype model is used specifically for web development. All web prototypes are built in an HTML format with a services layer and are then integrated into the final product.

### Advantages of the prototyping model

Using a prototype model can bring multiple advantages, including:

- Customers get a say in the product early on, increasing customer satisfaction.
- Missing functionality and errors are detected easily.
- Prototypes can be reused in future, more complicated projects.

- It emphasizes team communication and flexible design practices.
- Users have a better understanding of how the product works.
- Quicker customer feedback provides a better idea of customer needs.

**Disadvantages of the prototyping model**

The main disadvantage of this methodology is that it is more costly in terms of time and money when compared to alternative development methods, such as the spiral or Waterfall model. Since in most cases the prototype is discarded, some companies may not see the value in taking this approach.

Additionally, inviting customer feedback so early on in the development lifecycle may cause problems. One problem is that there may be an excessive amount of change requests that may be hard to accommodate. Another issue could arise if after seeing the prototype, the customer demands a quicker final release or becomes uninterested in the product.

**Situations where most appropriate**

- Project is large with many users, interrelationships, and functions, where project risk relating to requirements definition needs to be reduced
- Project objectives are unclear.
- User is not fully knowledgeable.
- No need exists to absolutely minimize resource consumption
- Pressure exists for immediate implementation of something

**3. Rapid application development**

Rapid application development (RAD) is a condensed development process that produces a high-quality system with low investment costs.


**Steps in Rapid Application Development**

Step 1. Define and finalize project requirements

At the very beginning, rapid application development sets itself apart from traditional software development models. It doesn't require you to sit with end users and get a

detailed list of specifications; instead, it asks for a broad requirement. The broad nature of the requirements helps you give specific requirements at different points of the development cycle

During this step, stakeholders sit together to define and finalize project requirements such as project goals, expectations, timelines, and budget. When you have clearly defined and scoped out each aspect of the project's requirements, you can seek management approvals.

Step 2: Begin building prototypes

As soon as you finish scoping the project, you can begin development. Designers and developers will work closely with clients to create and improve upon working prototypes until the final product is ready.

This is where the actual development takes place. Instead of following a strict set of requirements, developers create prototypes with different features and functions as fast as they can. These prototypes are then shown to the clients who decide what they like and what they don't. More often than not, these prototypes are quickly made to work, just to show off certain features, without proper polish. This is normal, and the final product is only created during the finalization stage where the client and developer can both agree on the final product.

Step 3: Gather user feedback

In this step, prototypes and beta systems are converted into working models. Developers then gather feedback from users to tweak and improve prototypes and create the best possible product.

In this stage, feedback on what's good, what's not, what works, and what doesn't is shared. Feedback isn't limited to just pure functionality, but also visuals and interfaces. With this feedback in mind, prototyping continues. These two steps are repeated until a final product can be realized that fits both the developers' and client's requirements

Step 4: Test, test, test

This step requires you to test your software product and ensure that all its moving parts work together as per client expectations. Continue incorporating client feedback as the code is tested and retested for its smooth functioning.

Step 5: Present your system

This is the final step before the finished product goes to launch. Here, features, functions, aesthetics, and interface of the software are finalized with the client. Stability, usability,

and maintainability are of paramount importance before delivering to the client. It also involves data conversion and user training.

| Advantages of RAD | Disadvantages of RAD |
|---|---|
| Requirements can be changed at any time | Needs strong team collaboration |
| Encourages and priorities customer feedback | Cannot work with large teams |
| Reviews are quick | Needs highly skilled developers |
| Development time is drastically reduced | Needs user requirement throughout the life cycle of the product |
| More productivity with fewer people | Only suitable for projects which have a small development time |
| Time between prototypes and iterations is short | More complex to manage when compared to other models |
| Integration isn't a problem, since it integrates from project inception | Only systems which can be modularized can be developed using Rapid application development |

RAD is used when

- The team includes programmers and analysts who are experienced with it
- There are pressing reasons for speeding up application development
- The project involves a novel ecommerce application and needs quick results
- Users are sophisticated and highly engaged with the goals of the company
- Senior management commitment exists to ensure end-user involvement
- Project is of small to medium scale and of short duration
- Application is highly interactive, has a clearly defined user group.

### 4. Spiral

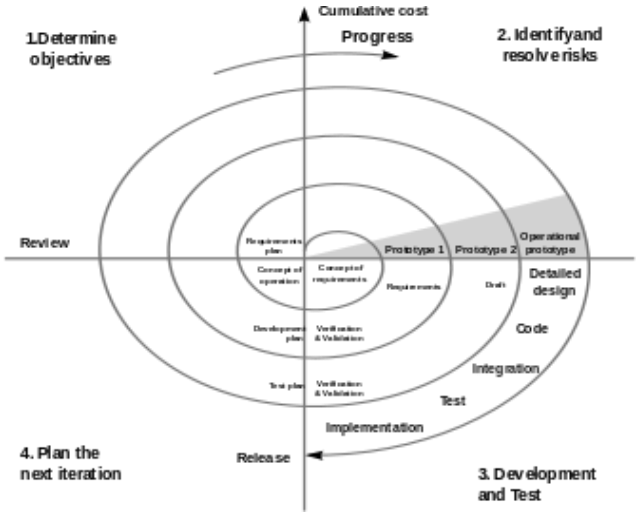
The spiral methodology allows teams to adopt multiple SDLC models based on the risk patterns of the given project. A blend of the iterative and waterfall approaches, the challenge with the spiral model is knowing when is the right moment to move onto the next phase. Business that aren't sure about their requirements or expect major edits during their mid to high-risk project can benefit from the scalability of this methodology.

### Steps in the spiral methodology

1. Determine the Objectives: Similar to the system conception phase of the Waterfall Model. Objectives are determined, possible obstacles are identified and alternative approaches are weighed.
2. Identify and resolve risks: Possible alternatives are examined by the developer, and associated risks/problems are identified. Resolutions of the risks are

evaluated and weighed in the consideration of project continuation. Sometimes prototyping is used to clarify needs.

3. Development and test: Detailed requirements are determined and the software piece is developed.
4. Plan the next iteration: The customer is given an opportunity to analyze the results of the version created in the Engineering step and to offer feedback to the developer




**Pros:**

- The risk assessment component of the Spiral Model provides both developers and customers with a measuring tool that earlier Process Models do not have. The practical nature of this tool helps to make the Spiral Model a more realistic Process Model than some of its predecessors.
- Focuses attention on reuse
- Accommodates changes, growth
- Eliminates errors and unattractive choices early
- Limits to how much is enough (not too much design, reqs, etc)
- Treats development, maintenance same way

**Cons:**

- It may be difficult to convince customers that the evolutionary approach is controllable
- It demands considerable risk assessment expertise and relies on the expertise for success
- Need for further elaboration of project steps (clearer milestones)

**Situations where Spiral is most suitable**

- Risk avoidance is a high priority
- If a project is to benefit from a mix of other development methodologies.
- A high degree of accuracy is essential
- Implementation has priority over functionality which can be added in the later versions

**5. End user development (EUD)**

EUD is "a set of methods, techniques and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact" (Lieberman et al 2006). In particular, EUD enables end users to design or customize the user interface and functionality of software. This is valuable because end users know their own context and needs better than anybody else, and they often have real-time awareness of shifts in their respective domains. Through EUD, end users can tune software to fit their requirements more closely than would be possible without EUD.

End users usually do not have training in professionals' programming languages, formal development processes, or modeling and diagramming notations. Moreover, end users often lack the time or motivation to learn these traditional techniques, since end users usually write code in order to achieve a short- or medium-term goal rather than to create a durable software asset that will produce a continuing revenue stream. Consequently, supporting EUD requires providing appropriate tools, social structures, and development processes that are highly usable, quickly learned, and easily integrated into domain practice. EUD overlaps with two similar concepts, end-user programming and end-user software engineering. End-user programming (EUP) enables end users to create their own programs (Ko et al 2011). This subset of EUD is the most mature from a research and practice perspective, so we focus a later section of this article on that portion of EUD. The difference between EUP and EUD is that EUD methods, techniques, and tools span the entire software development lifecycle, including modifying and extending software, not just the "create" phase.

End-user-developed systems can be completed more rapidly than those developed through the conventional systems life cycle. Allows users to specify their own business needs Improves requirements gathering. Leads to a higher level of user involvement and satisfaction with the system. Focus on the fundamental activities of any information system: input, processing, output, storage, and control.

**Advantages of EUD**

- Frees IS resources for higher priority projects.
- May help reduce the hidden backlog.

- Faster design/implementation cycle.
- More acceptable to users.
- Reduces communications problems between users and IS.
- Encourages innovation and creative solutions.

**Disadvantages of EUD**

- Duplication or effort and waste of resources
- Greatly increased costs
- Loss of control over data
- Loss of control of quality in both programs and data
- Incompatibles prevent sharing
- Can be used to circumvent control processes, such as the steering committee
- Generally, produces narrow, inflexible systems with short lives

**Choosing the Best SDLC Methodology**

When selecting the best SDLC approach for your organization or company, it's important to remember that one solution may not fit every scenario or business. Certain projects may run best with a waterfall approach, while others would benefit from the flexibility in the agile or iterative models.

Before deploying an SDLC approach for your teams and staff, consider contacting a knowledgeable IT consultant at Innovative Architects for advice. Our experts have seen how the different models function best in different industries and corporate environments. We are adept at finding a good fit for any situation.

Each one has its own strengths and weaknesses and works effectively in different situations. When choosing your development methodology, think about combining the elements of each method that work best for your team and your current project. In this way, you can create a hybrid development methodology that'll get you to production securely and efficiently.