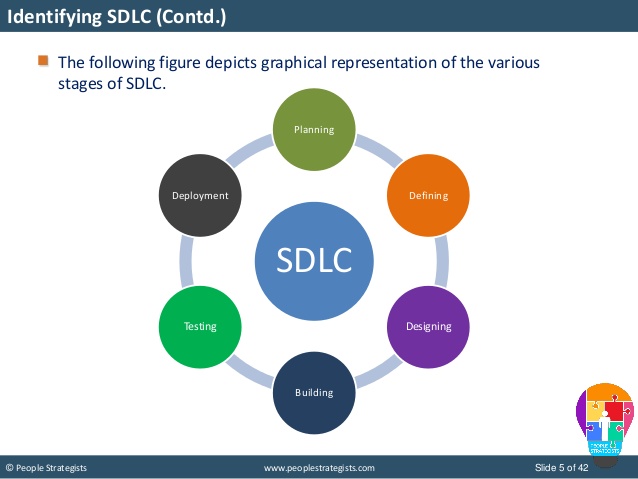
**Software Development Life Cycle (SDLC)**

It is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. It is also known as Software Development Process.

It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The figure below represents stages of a typical SDLC.



SDLC Models

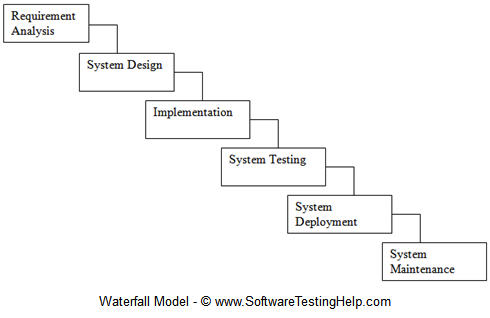
There are various software development life cycle models defined and designed. These Software Development Process Models follow a series of steps unique to its own type to make sure it is successful in the process of software development.

The more popular SDLC models are;

* Waterfall Model
* Iterative Model
* Spiral Model
* V-Model
* Big Bang Model
* As well as Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

**Waterfall Model**

Also referred to as a linear-sequential life cycle model is the waterfall model. This type of [software development model](http://istqbexamcertification.com/what-are-the-software-development-models/) is used for a project that is small and there are no unknown requirements. In a waterfall model, each phase must be completed before the next phase can begin and the phases do not overlap. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. Usually, the outcome of one phase becomes the input for the next phase sequentially. All phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases, so the name "Waterfall Model".



The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Advantages of Waterfall Model

* Easy to manage due to the rigidity of the model
* Phases are processed and completed one at a time.
* Requirements and results are very well documented, clear and fixed.
* Technology is understood and is not dynamic.
* Ample resources with required expertise are available to support the product.
* Divide complex tasks into smaller and hence more manageable tasks

Disadvantages of Waterfall Model

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty as project cannot accommodate changing requirements.
* It is difficult to measure progress within stages.
* Integration is done as a "big-bang” at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

**Software Prototyping**

Prototype is a working model of software with some limited functionality. Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design. The goal is to provide a system with overall functionality.



Software Prototyping- Types

### *Throwaway/Rapid Prototyping*

Throwaway prototyping is also called as rapid or close ended prototyping. This type of prototyping uses very little efforts with minimum requirement analysis to build a prototype. Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.

### *Evolutionary Prototyping*

Evolutionary prototyping also called as breadboard prototyping is based on building actual functional prototypes with minimal functionality in the beginning. The prototype developed forms the heart of the future prototypes on top of which the entire system is built. By using evolutionary prototyping, the well-understood requirements are included in the prototype and the requirements are added as and when they are understood.

### *Incremental Prototyping*

Incremental prototyping refers to building multiple functional prototypes of the various sub-systems and then integrating all the available prototypes to form a complete system.

### *Extreme Prototyping*

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

Advantages of Prototype Model

* Increased user involvement in the product even before its implementation.
* Since a working model of the system is displayed, the users get a better understanding of the system being developed.
* User requirements can be clarified.
* Reduces time and cost as the defects can be detected much earlier.
* Quicker user feedback is available leading to better solutions.

Disadvantages of Prototype Model

* Risk of insufficient requirement analysis owing to too much dependency on the prototype.
* This methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
* Developers may try to reuse the existing prototypes to build the actual system, even when it is not technically feasible.
* The effort invested in building prototypes may be too much if it is not monitored properly.
* Continual changes tend to corrupt software structure. Changes become more costly and difficult.
* Incomplete or inadequate problem analysis.

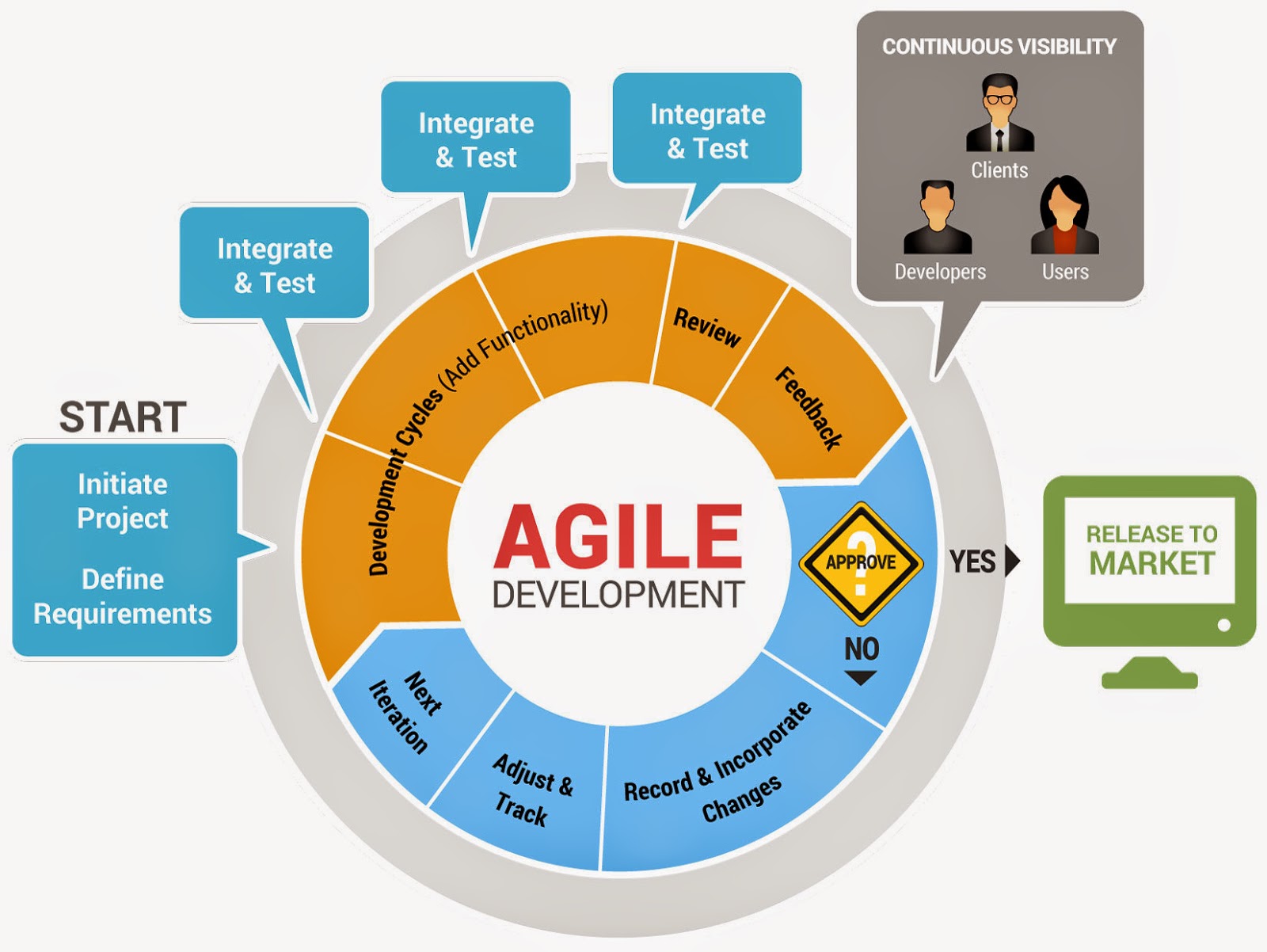
**Agile SDLC Model**

**Agile development model**is also a type of [Incremental model](http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/). Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly [tested](http://istqbexamcertification.com/why-is-testing-necessary/) to ensure [software quality](http://istqbexamcertification.com/what-is-software-quality/) is maintained. It is used for time critical applications. Cross functional teams work simultaneously on various areas like;

* Planning
* Requirements Analysis
* Design
* Coding
* Unit Testing and
* Acceptance Testing

At the end of the cycle, a working product is displayed to the customer and important stakeholders.

Graphical illustration of the Agile Model



Advantages of Agile Model

* Functionality can be developed rapidly and demonstrate partial working solutions.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions.
* Enables concurrent development and delivery within an overall planned context.
* Easy to manage as there is little or no planning required.
* Gives flexibility to developers.
* Customers, developers and testers constantly interact with each other.

Disadvantages of Agile Model

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is a very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.
* Lack of emphasis on necessary designing and documentation.

<https://www.tutorialspoint.com/sdlc/sdlc_overview.htm>

<http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/>