**Software development life cycle**

Software development life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

There are following six phases in every Software development life cycle model:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

**Software Development lifecycle models**

Waterfall Model

Big Bang Model

Rad Model

**Waterfall Model**

The Waterfall Model referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model.

The sequential phases in Waterfall model are:

1. Gathering and analysis
2. System design
3. Implementation
4. Integration and testing
5. Deployment of system
6. Maintenance

**Requirement Gathering and analysis**: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

**System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

**Implementation**: With inputs from 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.

Pros:

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.

Cons:

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.

**Rad Model**

Rapid application development (RAD) is a software development methodology that uses minimal planning in favour of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

**Rad model design**

**Business Modeling**: The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

**Data Modeling**: The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

**Process Modeling**: The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding , deleting, retrieving or modifying a data object are given.

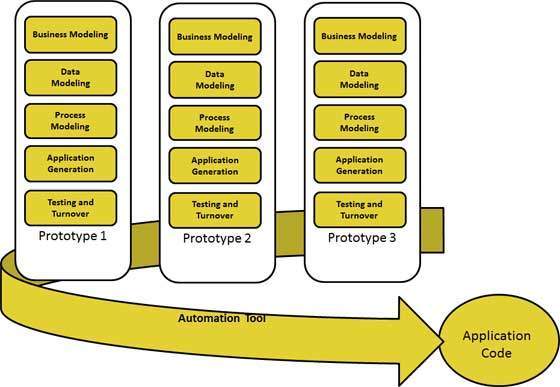
**Application Generation**: The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

**Testing and Turnover**: The overall testing time is reduced in RAD model as the prototypes are independently tested during every iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

**RAD Model Vs Traditional SDLC**

The traditional SDLC follows a rigid process models with high emphasis on requirement analysis and gathering before the coding starts. It puts a pressure on the customer to sign off the requirements before the project starts and the customer doesn.t get the feel of the product as there is no working build available for a long time.

The customer may need some changes after he actually gets to see the software, however the change process is quite rigid and it may not be feasible to incorporate major changes in the product in traditional SDLC.



Pros:

* Changing requirements can be accommodated.
* Progress can be measured.
* Iteration time can be short with use of powerful RAD tools.
* Productivity with fewer people in short time.
* Reduced development time.
* Increases reusability of components

Cons:

* Dependency on technically strong team members for identifying business requirements.
* Only system that can be modularized can be built using RAD.
* Requires highly skilled developers/designers.
* High dependency on modeling skills

**Big Bang Model**

The Big Bang model is SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement.

There is no formal development followed and very little planning is required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis.

Pros:

* This is a very simple model
* Little or no planning required
* Easy to manage
* Very few resources required

Cons:

* Very High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Can turn out to be very expensive if requirements are misunderstood

Reference

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

http://www.tutorialspoint.com/sdlc/sdlc\_overview.htm