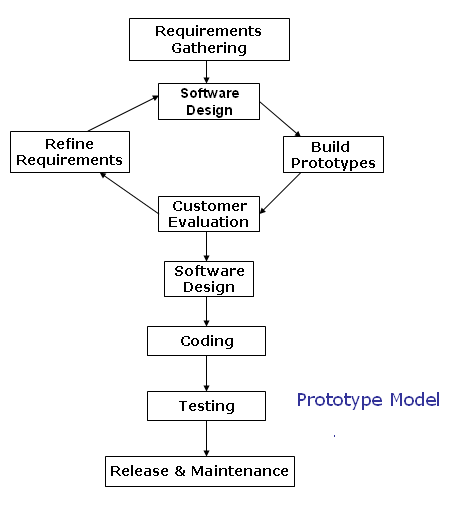
**Overview on Prototype Model:**

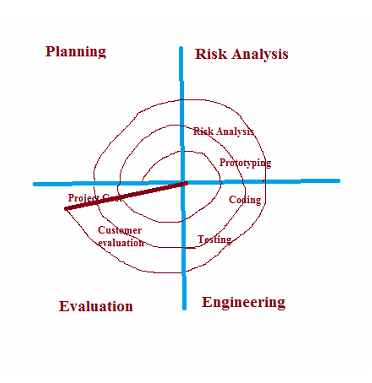
It begins with requirements gathering. Developers and Customers meet and define the overall objectives of the software.  
  
Developers prepare design documents using all available requirements then build the prototypes, prototypes are sent to Customer, and Customer evaluates Prototypes and gives feedback. Developers redefine Requirements, modify software design and produce modified Prototypes. Process will be continued like this, after getting Customer’s confirmation Developers Start Regular process; Software Design, Coding (Implementation), Testing and Release & Maintenance.  
  
The Objective of this approach is getting clear (Complete and Correct)  
Requirements from Customers in order to avoid Defects multiplication.



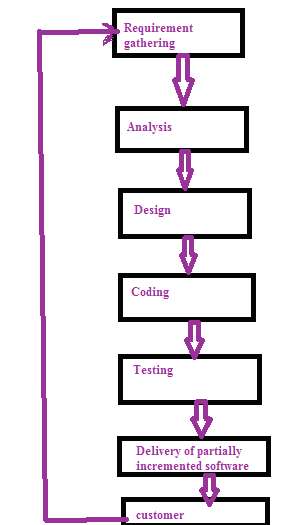
**Prototype Model Diagram**

**Advantages of Prototype Model:**  
  
o    The customer doesn’t need to wait long as in the Linear Model or Waterfall Model.   
  
o    Feedback from customer is received periodically and the changes don’t come as a last minute surprise.   
  
o    Customer’s interaction improves the Quality as well success rate.  
  
  
**Disadvantages of Prototype Model:**  
  
o    It is an expensive approach and time taking process when its compare to sequential models like Waterfall model.   
  
o    Customer could believe the prototype as the working version.   
  
o    Developer also could make the implementation compromises.   
  
o    Once Requirements finalized then adopting changes in Requirements and adding New Requirements are difficult.

**Spiral Model**

Spiral model is also known as the spiral lifecycle model. This development of model combines the features of the prototyping model and the waterfall model. In spiral model, we can easily adjust the software development with the required changes. The prototypes which are created in every stage enable us to roll back only a few steps. This model is favored for large, expensive, and complicated projects.  It is a circular view of the software lifecycle.  
  
  
Overview  
   
Spiral model was first defined in an article by Barry Boehm in 1986. This spiral model is based on experience with large government software projects. The goal was to provide a model with greater flexibility that could better serve these types of projects.  
  
Spiral-Model Software Development Life Cycle  
   
In this cycle, each passes through the spiral model consists of Planning, Risk analysis, Engineering, and Evaluation phase.   
  
  
  
  
**Planning:** In this phase, the new system requirements are gathered and defined after a comprehensive system study of the various business processes. This usually involves interviewing internal and external users, preparation of detailed flow diagrams showing the process or processes for which going to be developed, the inputs and outputs in terms of how the data is to be recorded/ entered and the form in which the results are to be presented.  
  
**Risk Analysis Phase:**  In this phase to identify risk and alternate solutions, a process is followed. The process includes addressing the factors that may risk the successful completion of the entire project development including alternative strategies and constraints. The issues pertaining to the possibility of the development if not meeting, for example, user requirements, reporting requirements or the capability of the  development team or the compatibility and functionality of the hardware with software. To undertake development, the Risk analysis and suggested solutions to mitigate and eliminate the Risks would thus become a part of the finalized strategy. A prototype is produced at the end of the risk analysis phase.  
  
**Engineering Phase:** In this phase, software is produced along with coding and testing at the end of the phase. After preparation of prototype tested against benchmarks based on customer expectations and evaluated risks to verify the various aspects of the development. Until customer satisfaction is achieved before development of the next phase of the product, refinements and rectifications of the prototype are undertaken.  
  
**Evaluation phase:** In this phase, the final system is thoroughly evaluated and tested based on the refined prototype. Evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. Routine maintenance is carried out on a continuing basis to prevent large scale failures and to minimize downtime.  
  
Advantages of Spiral Model  
  
•    It is a good model for large and complex projects.  
•    It reduces problems at all levels with direct consideration of risks.  
•    In this phase, software is produced early in the software life cycle.  
  
Disadvantage of Spiral Model  
  
•    It can be a costly model to use.  
•    When the risk analysis is carried out, it needs highly specific expertise which may not be available all the time.  
•    The project success is highly dependent on the risk analysis phase.  
•    It is applicable only to large scale systems and doesn’t work well for smaller projects because of the cost of evaluating risk.

**Agile Development Model**  
  
The term Agile means 'moving quickly.' Agile process itself is a software development process by small teams, in a short time frame, and involving system users as well as developers. This agile process is an iterative approach in which customer satisfaction is at highest priority as the customer has direct involvement in evaluating the software. An Agile approach focuses on adaptability and response time to changing requirements.  
  
Overview  
   
The definition for agile software development evolved in the mid 1990s. It helps in adaptive planning, iterative development and time boxing. It is a theoretical framework that promotes foreseen interactions throughout the development cycle.   
  
Agile Model Phases

The agile process follows the software development life cycle which includes requirements gathering, analysis, design, coding, testing and delivers partially implemented software and waits for the customer feedback. In the whole process, customer satisfaction is at highest priority with faster development time. The following diagram depicts the software development life cycle of Agile Process.  
  
  
   
The Agile process requires less planning and it divides the tasks into small increments. This process is meant for short term projects with an effort of team work that follows the software development life cycle. Software development life cycle includes the following phases   
  
1.Requirements gathering,   
2.Analysis,   
3.Design,   
4.Coding,   
5.Testing,   
6.Maintenance.   
The involvement of software team management with customers reduces the risks associated with the software. This agile process is an iterative process in which changes can be made according to the customer satisfaction.   
The Agile system development life cycle model begins with initial elicitation of requirements in order to conceptualize the system. Once the system concept is formed, requirements are estimated and prioritized in order to plan the development iterations. Within each iteration, requirements continue to evolve and are used in brief model storming sessions in order to design the functionality required to meet requirements allocated to the iteration. When the system model is developed/modified, the coding is initiated/continued using a test-driven approach. Once the functionality for the iteration is tested successfully, refactoring takes place. Refactoring is a disciplined technique for altering the structure of an existing code base without altering functionality. When refactoring is completed and functionality is confirmed, the code is deployed internally, completing the iteration.   
If plans include deploying the code beyond the development iteration, the code is then deployed externally. Eventually, all iterations are completed, resulting in a system with full functionality that is accepted by stakeholders because it meets allocated requirements.    
There are several methodologies through which we can implement Agile Projects. The agile methods are focused on different aspects of the software development life cycle. Some focus on the practices (extreme programming, pair programming), while others focus on managing the software projects (the scrum approach).  
The three methodologies which are most widely used in Industry are   
   
**1. Extreme Programming (XP).  
2. Scrum.  
3. Feature Driven Development (FDD).**  
  
**1. Extreme Programming (XP):** XP is the most successful method of developing agile software because of its focus on customer satisfaction. XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles. It welcomes and incorporates changes or requirements from the customers at any phase of the development life cycle.  
  
**2. Scrum:** Scrum is another popular method of agile development through which productivity becomes very high. It is basically based on incremental software development process. In scrum method the entire development cycle is divided into a series of iteration where each iteration is called as a sprint. Maximum duration of a sprint is 30 days.  
 **3. Feature Driven Development (FDD):** FDD is one of the agile development methods. The key advantage of this method is to design the domain of the software to be produced before development.  
  
Advantages of Agile Model  
  
•    Adaptive to the changing environment.  
•    Agile helps to speed up the SDLC phases and bypasses process steps that add little value to the project.  
•    Engages the stakeholders continuously so that the new requirements are gathered faster and there is no scope for guess work by the teams.  
•    Saves cost, time and efforts by following iterative incremental work delivery and thereby identifying deviations early.  
•    Least documentation.  
•    Provides the end result of higher quality of the software delivered and a highly satisfied customer.  
  
Disadvantage of Agile Model  
  
•    Time consuming and wastage of resources because of constant change of requirements.  
•    More helpful for management than developer.  
•    Only senior developers are in a better position to take the decisions necessary for the agile type of development.  
•    Once teams become bigger, agile methodologies begin to fail, as they don’t scale to large teams, or teams spread across geographies.  
•    If the projects are large then it becomes difficult to judge the efforts and the time required for the project in the software development life cycle.  
--------------------------------