The **Prototyping Model** was developed on the assumption that it is often difficult to know all of your requirements at the beginning of a project. Typically, users know many of the objectives that they wish to address with a system, but they do not know all the nuances of the data, nor do they know the details of the system features and capabilities. The **Prototyping Model** allows for these conditions, and offers a development approach that yields results without first requiring all information up-front .

When using the **Prototyping Model**, the developer builds a simplified version of the proposed system and presents it to the customer for consideration as part of the development process. The customer in turn provides feedback to the developer, who goes back to refine the system requirements to incorporate the additional information. Often, the prototype code is thrown away and entirely new programs are developed once requirements are identified.

There are a few different approaches that may be followed when using the **Prototyping Model**:

- creation of the major user interfaces without any substantive coding in the background in order to give the users a "feel" for what the system will look like,
- development of an abbreviated version of the system that performs a limited subset of functions; development of a paper system (depicting proposed screens, reports, relationships etc.), or
- use of an existing system or system components to demonstrate some functions that will be included in the developed system.

Prototyping is comprised of the following steps:

- **Requirements Definition/Collection.** Similar to the Conceptualization phase of the **Waterfall Model**, but not as comprehensive. The information collected is usually limited to a subset of the complete system requirements.
- **Design.** Once the initial layer of requirements information is collected, or new information is gathered, it is rapidly integrated into a new or existing design so that it may be folded into the prototype.
- **Prototype Creation/Modification.** The information from the design is rapidly rolled into a prototype. This may mean the creation/modification of paper information, new coding, or modifications to existing coding.
- **Assessment.** The prototype is presented to the customer for review. Comments and suggestions are collected from the customer.
- **Prototype Refinement.** Information collected from the customer is digested and the prototype is refined. The developer revises the prototype to make it more effective and efficient.
- **System Implementation.** In most cases, the system is rewritten once requirements are understood. Sometimes, the **Iterative** process eventually produces a working system that can be the cornserstone for the fully functional system.

Problems/Challenges associated with the Prototyping Model

Criticisms of the **Prototyping Model** generally fall into the following categories:

- **Prototyping can lead to false expectations. Prototyping** often creates a situation where the customer mistakenly believes that the system is "finished" when in fact it is not. More specifically, when using the **Prototyping Model**, the pre-implementation versions of a system are really nothing more than one-dimensional structures. The necessary, behind-the-scenes work such as database normalization, documentation, testing, and reviews for efficiency have not been done. Thus the necessary underpinnings for the system are not in place.
- **Prototyping can lead to poorly designed systems.** Because the primary goal of **Prototyping** is rapid development, the design of the system can sometimes suffer because the system is built in a series of "layers" without a global consideration of the integration of all other components. While

initial software development is often built to be a "throwaway, " attempting to retroactively produce a solid system design can sometimes be problematic.

Variation of the Prototyping Model

A popular variation of the **Prototyping Model** is called **Rapid Application Development (RAD)**. RAD introduces strict time limits on each development phase and relies heavily on rapid application tools which allow for quick development.