

SDLC - RAD MODEL

http://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm

Copyright © tutorialspoint.com

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor 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.

Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

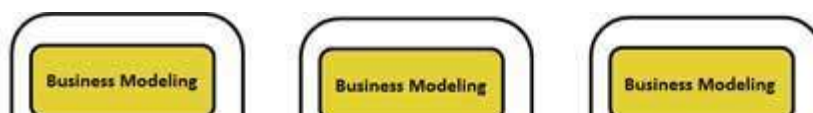
The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

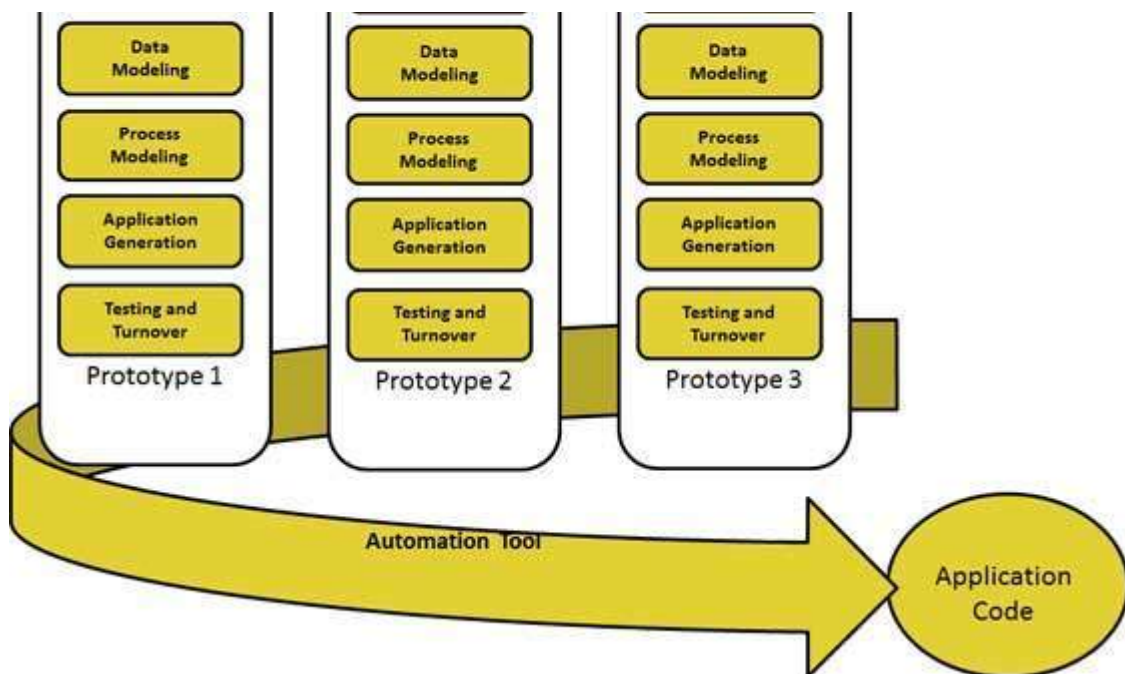
RAD Model Design

RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. Following are the phases of RAD Model:

- **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.

Following image illustrates the RAD Model:





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.

RAD model focuses on iterative and incremental delivery of working models to the customer. This results in rapid delivery to the customer and customer involvement during the complete development cycle of product reducing the risk of non conformance with the actual user requirements.

RAD Model Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail. Following are the typical scenarios where RAD can be used:

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
-
- It should be used if there.s high availability of designers for modeling.
-
- It should be used only if the budget permits use of automated code generating tools.
-
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
-
- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.
-

RAD Model Pros and Cons

RAD model enables rapid delivery as it reduces the overall development time due to reusability of the components and parallel development.

RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.

Following table lists out the pros and cons of RAD Model:

Pros	Cons
<ul style="list-style-type: none">• 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• Quick initial reviews occur• Encourages customer feedback• Integration from very beginning solves a lot of integration issues.	<ul style="list-style-type: none">• 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.• Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.• Management complexity is more.• Suitable for systems that are component based and scalable.• Requires user involvement throughout the life cycle.• Suitable for project requiring shorter development times.