# Rapid Application Development

## History

The RAD process is one of the first incremental software development processes that tries to deal with the inflexibility and cumbersome approach of the traditional models like Waterfall and V-Model.

The model was developed in the 1980s by IBM’s James Martin and was formally publicized in his 1991 book “Rapid Application Development”.

The approach implements many popular ideas and movements of the software industry 80s (most notably Barry Boehm’s Spiral model that was a risk-based development approach that used prototypes heavily).

## Methodology

The challenges facing software development organizations can be summarized as more, better, and faster. The RAD development path attacks these challenges head-on by providing a means for developing systems faster, while reducing cost and increasing quality.

* Using evolutionary prototypes that are eventually transformed into the final product
* Using workshops instead of interviews and negotiations with a group of users to gather requirements and review design
* Implementing time boxed development that allows the teams to quickly build the core of the system and implement refinements in subsequent releases
* Documenting and propagating guidelines for success and pitfalls to avoid

One of the most distinctive trait of the RAD is the active user involvement throughout the whole process, which ensures that the user’s expectations and requirements are well understood and implemented.

## The process

* Requirements planning. Combines elements of the system planning and analysis of the traditional SDLC. Users, managers and IT staff discuss and agree on business needs, projects scope, constraints and requirements.
* User Design. Users interact with the system analysis and develop models and prototypes that represent processes, inputs and outputs of the future system. This is a continuous interactive process that allows users to understand, modify and approve a working model of the system
* Construction. The system is developed and implemented into prototypes that are brought back to the user design phase and tested/modified/approved. The circle continues until there is a satisfactory working prototype of the final system
* Cutover. Similar to the final stages of the traditional SDLC, namely the testing, implementation and user training stages, but highly compressed, as a result of which the system is build, delivered and placed in operation much sooner.

## Advantages

* Successful when faced with unstable business requirements or when developing non-standard systems
* Provides higher value for the final users as the user participation allows the software to become more user-friendly and focused on the critical business problems that need to be addressed and solved
* Prototypes have the following advantages:
  + Risk reduction through providing valuable information as to the feasibility of a design
  + Users are better at using and reacting than at creating specification
  + Can be used as a stepping platform for the completed final product

## Disadvantages

* Process is time-consuming for the user’s business, which requires domain expert people that have important daily task in running the business itself to commit a significant amount of time to the development of the software
* Flexibility vs Control tradeoff. The flexibility that allows the project to adapt quickly to challenges and opportunities brings a natural tradeoff with the control both the users and the company have on the system, which is crucial in some projects (i.e. life-saving software)
* Unsuitable for large and safety- and mission-critical projects.
* Poor system design. The intense focus on the user design might sometimes lead to a poorly on system design, as the key here is to make key, but minor changes to the individual components, which may lead to ignoring system architecture issues.