Unit-1: Introduction of Software enginnering

Process and Project

A process is a sequence of steps performed for a given purpose. As mentioned earlier, while developing (industrial strength) software, the purpose is to develop software to satisfy the needs of some users or clients.

A software project is one instance of this problem, and the development process is what is used to achieve this purpose.

Program

Program is set of sequential logical instructions.

Software

Software is a set of items or objects that form a "configuration" that includes

- Programs
- Documents
- Data...

Software consists of

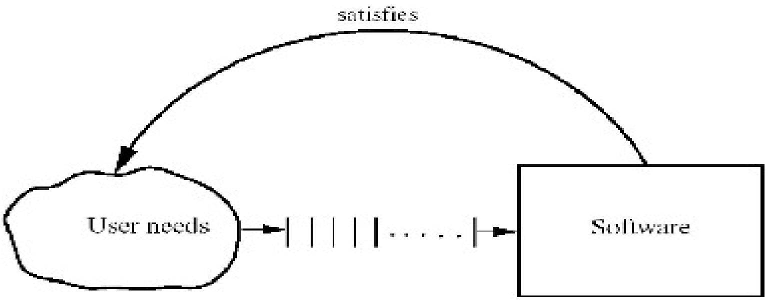
(1) Instructions (computer programs) that when executed provided desired function and performance,

(2) Data structures that enable the programs to adequately manipulate information, and

(3) Documents that describe the operation and use of the programs.

But these are only the concrete part of software that may be seen, there exists also invisible part which is more important:

- Software is the dynamic behavior of programs on real computers and auxiliary equipment.
- "… a software product is a model of the real world, and the real world is constantly changing."
- Software is a digital form of knowledge. ("Software Engineering," 6ed. Sommerville, Addison-Wesley, 2000)
- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.



Software products may be

- Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
- Bespoke (custom) - developed for a single customer according to their specification.

New software can be created by developing new programs, configuring generic software systems or reusing existing software.

Chronological evolution of software

1. 1950-1965 –Early Years
   General Purpose Hardware Batch
   Orientation
   Limited Distribution Custom Software
2. 1968-1973- Second Era
   Multi User Real Time
   Data Base
   Product Software
   Concept of sw Maintenance 3. 1974-
   1986-Third Era
   Distributed Systems Embedded
   Systems Low cost HW Consumer
   Impact
4. 1987-2000-Fourth Era
   Powerful Desktop
   Object Orient Technology Expert
   Systems
   Artificial Neural Network Parallel
   Computing Network Computers

Software Component

- **Lowest Level-** It mirrors the instruction set of the h/w. makes efficient use of memory and optimize programmed execution speed.
- **Machine language-** 1 Level
- **Assembly language**-2 Level
- **Mid Level - 3** Level. Machine independents used to create procedural description of the program. More efficient then machine language C, C++
- **Highest fourth level:-**These are non procedural moves the developer further from the h/w. it use geographical icons.

Software Characteristics

1) Software does not wear out.

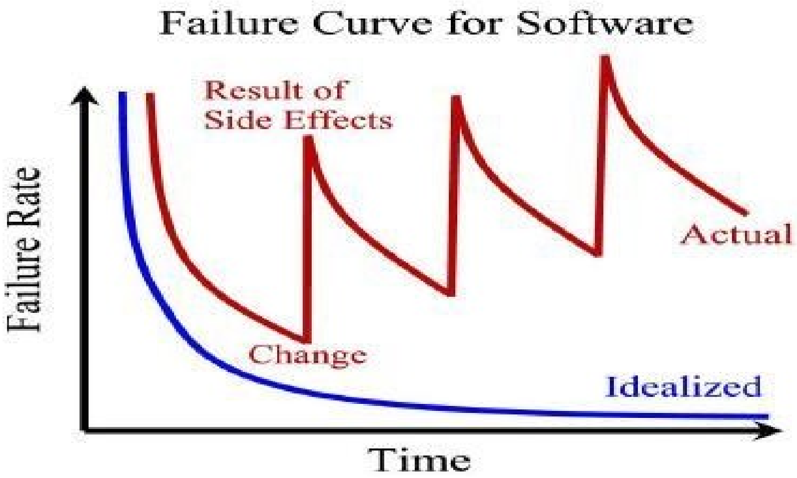
Software is not manufactured, but it is developed through the life cycle concept.

1) Reusability of components

2) Software is flexible and can be amended to meet new requirements

3) Cost of its change at if not carried out initially is very high at later stage .

4)



Difference in Program and Software

Program	Software
1. Usually small in size	1.Large
2. Author himself is sole user	2.Large number of users
3. Single developer	3.Team of developers
4. Lacks proper user interface	4.Well-designed interface
5. Lacks proper documentation	5.Well documented & user-manual prepared
6. Ad hoc development.	6.Systematic development

Software Crisis

Software fails because it

- crash frequently
- expensive
- difficult to alter, debug, enhance
- often delivered late
- use resources non-optimally
- Software professionals lack engineering training
- Programmers have skills for programming but without the engineering mindset about a process discipline

Standish Group Report OG States:

1. About US$250 billions spent per year in the US on application development
2. Out of this, about US$140 billions wasted due to the projects getting abandoned or reworked; this in turn because of not following best practices and standards
3. 10% of client/server apps are abandoned or restarted from scratch
4. 20% of apps are significantly altered to avoid disaster
5. 40% of apps are delivered significantly late
6. 30% are only successful

Software engineering Process

Concept of the Software Engineering has evolved in 1986 by BOHEM to reduce the effect of his software crisis because software engineering defined as scientific and systematic approach to develop , operate and maintain software project to meet a given specific object it beautifully envisages the concept of life cycle of any software project through following five phase:-

- Requirement analysis
- Design
- Coding
- Testing
- Maintenance

Software Engineering

Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products.

- "The systematic application of tools and techniques in the development of computer-based applications." (Sue Conger in The New Software Engineering)
- "Software Engineering is about designing and developing high-quality software." (Shari Lawrence Pfleeger in Software Engineering -- The Production of Quality Software)
- A systematic approach to the analysis, design, implementation and maintenance of software. (The Free On-Line Dictionary of Computing)The systematic application of tools and techniques in the development of computer-based applications. (Sue Conger in The New Software Engineering) Software Engineering is about designing and developing high-quality software. (Shari Lawrence Pfleeger in Software Engineering -- The Production of Quality Software)
- The technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost constraints (R. Fairley)
- A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers (Ghezzi, Jazayeri, Mandrioli)

Difference between software engineering and software programming

Software Engineering	Software Programming
1. Single developer	1.Teams of developers with multiple roles
1. "Toy" applications	2.Complex systems
2. Short lifespan	3.Indefinite lifespan
3. Single or few stakeholders Architect = Developer = Manager = Tester = Customer = User	4.Numerous stakeholders Architect ≠ Developer ≠ Manager ≠ Tester ≠ Customer ≠ User
5.One-of-a-kind systems	5.System families

Difference between software engineering and computer science

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

Difference between software engineering and system engineering

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.

System engineers are involved in system specification, architectural design, integration and deployment

- Software engineering is based on computer science, information science and discrete mathematics whereas traditional engineering is based on mathematics, science and empirical knowledge.
- Traditional engineers construct real artifacts and software engineers construct non- real (abstract) artifacts.
- In traditional engineering, two main concerns for a product are cost of production and reliability measured by time to failure whereas in software engineering two main concerns are cost of development and reliability measured by the no. of errors per thousand lines of source code.

Difference between software engineering and traditional software engineering

- Software engineers often apply new and untested elements in software projects while traditional software engineers generally try to apply known and tested principles and limit the use of untested innovations to only those necessary to create a product that meets its requirements.
- Software engineers emphasize projects that will live for years or decades whereas some traditional engineers solve long-ranged problems that ensure for centuries.

- Software engineering is about 50 years old whereas traditional engineering as a whole is thousands of years old.
- Software engineering is often busy with researching the unknown (eg. To drive an algorithm) right in the middle of a project whereas traditional engineering normally separates these activities. A project is supposed to apply research results in known or new clever ways to build the desired result.
- Software engineering has first recently started to codify and teach best practice in the form of design pattern whereas in traditional, some engineering discipline have thousands of years of best practice experience handed over from generation to generation via a field’s literature , standards, rules and regulations.

Software Development Life Cycle (SDLC)Models

Software-development life-cycle is used to facilitate the development of a large software product in a systematic, well-defined, and cost-effective way.An information system goes through a series of phases from conception to implementation. This process is called the Software-Development Life-Cycle.

Various reasons for using a life-cycle model include:

- Helps to understand the entire process
- Enforces a structured approach to development
- Enables planning of resources in advance
- Enables subsequent controls of them
- Aids management to track progress of the system

Activities undertaken during feasibility study: - The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product. **Activities undertaken during requirements analysis and specification: -** The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely Requirements gathering and analysis, this phase ends with the preparation of Software requirement Specification (SRS)

Activities undertaken during design: - The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language . Design specification Document is outcome of this phase.

Activities undertaken during coding and unit testing:-The purpose of the coding and unit testing phase (sometimes called the implementation phase) of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested. Code Listings are generated after this phase,.

Activities undertaken during integration and system testing: - Integration of different modules is undertaken once they have been coded and unit tested During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document.Test Rports are generatd after this phase.

Activities undertaken during maintenance: - Maintenance of a typical software product requires much more than the effort necessary to develop the product itself. Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratio. This phase continues till the software is in use.

Different software life cycle models

Many life cycle models have been proposed so far. Each of them has some advantages as well as some disadvantages. A few important and commonly used life cycle models are as follows:

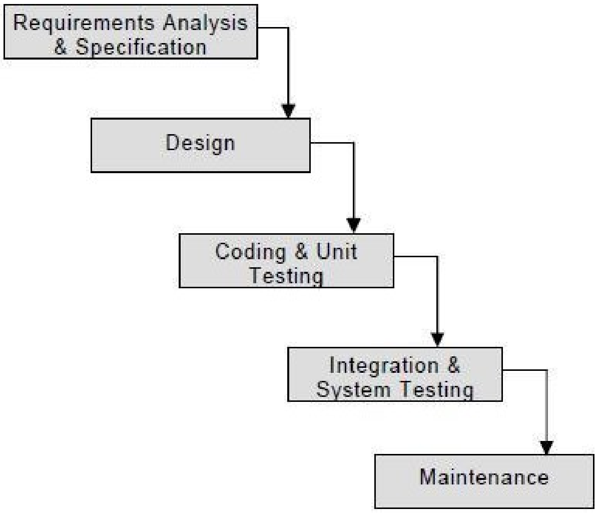
1. Classical Waterfall Model
2. Iterative Waterfall Model
3. Prototyping Model
4. Evolutionary Model
5. Spiral Model
6. Rapid Development Application model (RAD)

Classical Waterfall Model

The classical waterfall model is intuitively the most obvious way to develop software. Though the classical waterfall model is elegant and intuitively obvious, it is not a practical model in the sense t hat it cannot be used in actual software development projects. Thus, this model can be considered to be a theoretical way of developing software. But all other life cycle models are essentially derived from the classical waterfall model. So, in order to be able to appreciate other life cycle models it is necessary to learn the classical waterfall model.

Classical waterfall model divides the life cycle into the following phases as shown in fig:

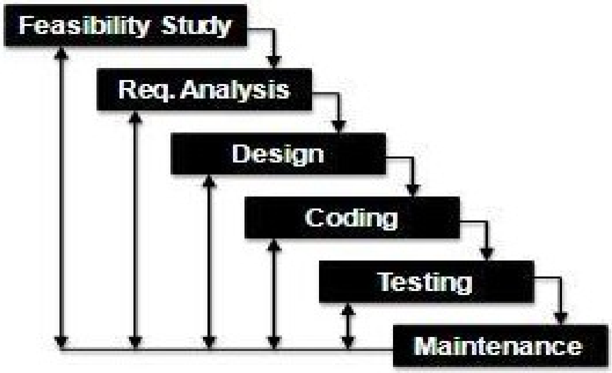
1. Feasibility Study
2. Requirements Analysis and Specification
3. Design
4. Coding and Unit Testing
5. Integration and System Testing
6. Maintenance


Requirements Analysis & Specification
Design
Coding & Unit Testing
Integration & System Testing
Maintenance

Classical Waterfall Model

Iterative Waterfall Model

- Waterfall model assumes in its design that no error will occur during the design phase
- Iterative waterfall model introduces feedback paths to the previous phases for each process phase
- It is still preferred to detect the errors in the same phase they occur
- Conduct reviews after each milestone



Iterative Waterfall Model

Advantages of Waterfall Model

- It is a linear model.
- It is a segmental model.
- It is systematic and sequential.
- It is a simple one.
- It has proper documentation

Disadvantages of Waterfall Model

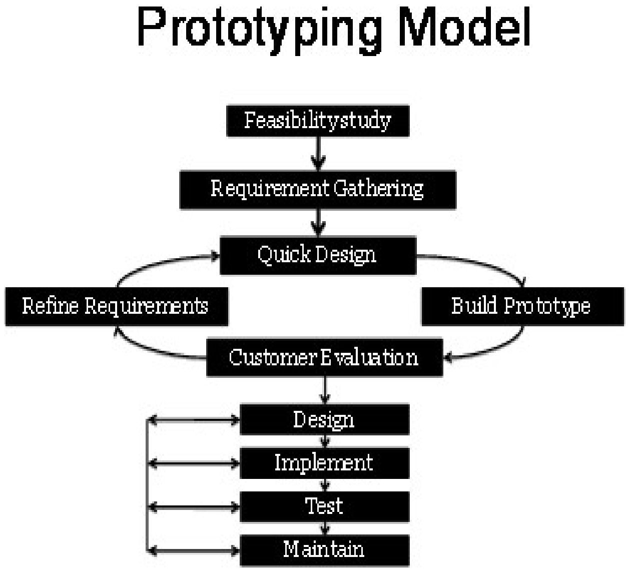
- It is difficult to define all requirements at the beginning of the project.
- Model is not suitable for accommodating any change
- It does not scale up well to large projects
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Therefore, this model is only appropriate when the requirements are well- understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Prototype model

The Prototyping Model is a systems development method (SDM) in which a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed prototyping paradigm begins with requirements gathering.

- Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.
- A "quick design" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats).

There are several steps in the Prototyping Model as shown in the diagram:-



1. The new system requirements are defined in as much detail as possible. This usually involves

interviewing a number of users representing all the departments or aspects of the existing system.

2. A preliminary design is created for the new system.

1. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
2. The users thoroughly evaluate the first prototype, noting its strengths and weaknesses, what needs to be added, and what should to be removed. The developer collects and analyzes the remarks from the users.
3. The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.
4. The second prototype is evaluated in the same manner as was the first prototype.
5. The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.
6. The final system is constructed, based on the final prototype.
7. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime. Customers could believe it's the working system. Developer could take "shortcuts" and only fill the prototype instead of redesigning it. Customers may think that the system is almost done, and only few fixes are needed

Advantage of Prototype model

- Suitable for large systems for which there is no manual process to define there requirements.
- User training to use the system.
- User services determination.
- System training.
- Quality of software is good.
- Requirements are not freezed.

Disadvantage of Prototype model

- It is difficult to find all the requirements of the software initially.
- It is very difficult to predict how the system will work after development.

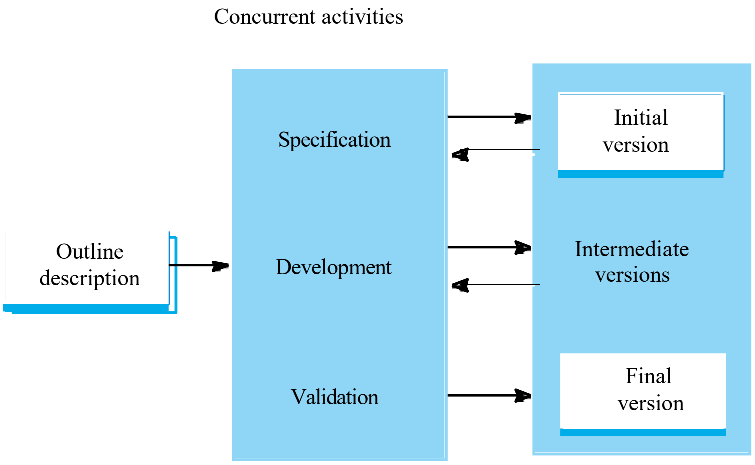
Evolutionary Process Model

In EP model development engineering effort is made first to establish correct, precise requirement definitions and system scope, as agreed by all the users across the organization. This is achieved through application of iterative processes to evolve a system most suited to the given circumstances. The process is iterative as the software engineer goes through a repetitive process of requirement until all users and stakeholders are satisfied. This model differs from the iterative enhancement model in the sense that this does not require a useable product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.

Main characteristics:

- – The phases of the software construction are interleaved
- – Feedback from the user is used throughout the entire process
- – The software product is refined through many versions

- Types of evolutionary development:
  - – Exploratory development
  - – Throw-away prototyping



Advantages:

- Deals constantly with changes
- Provides quickly an initial version of the system
- Involves all development teams

Disadvantages:

- Quick fixes may be involved
- “Invisible” process, not well-supported by documentation
- The system’s structure can be corrupted by continuous change

Spiral model

The Spiral model of software development is shown in fig. The diagrammatic representation of this model appears like a spiral with many loops. The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process. For ex ample, the innermost loop might be concerned with feasibility study. The next loop with requirements specification, the next one with design, and so on. Each phase in this model is split into four sectors (or quadrants) as shown in fig. 2. The following activities are carried out during each phase of a spiral model.

First quadrant (Objective Setting)

- During the first quadrant, it is n eeded to identify the objectives of the phase.
- Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk.
- Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

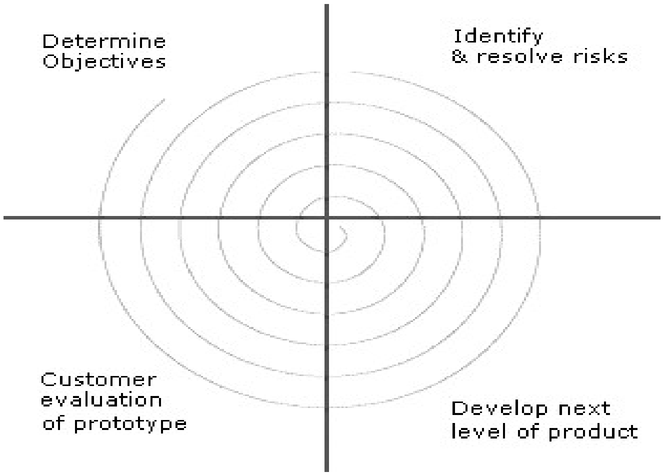
Third Quadrant (Development and Validation)

- Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.

Progressively more complete version of the software gets built with each iteration around the spiral


Determine Objectives
Identify & resolve risks
Customer evaluation of prototype
Develop next level of product

Advantages of Spiral Model

- It is risk-driven model.
- It is very flexible.
- Less documentation is needed.
- It uses prototyping

Disadvantages of Spiral Model

- No strict standards for software development.
- No particular beginning or end of a particular phase.

D**ifference between spiral and waterfall model**

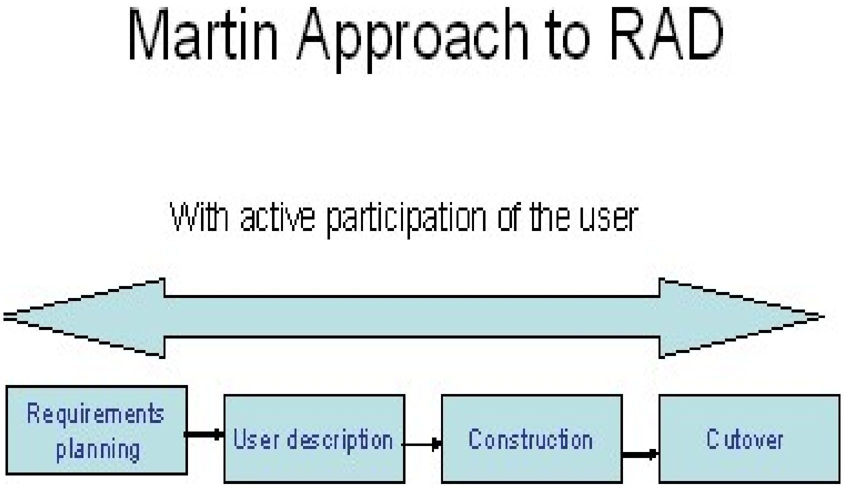
Waterfall model	Spiral model
Separate and distinct phases of specification and development.	Process is represented as a spiral rather than as a sequence of activities with backtracking
After every cycle a useable product is given to the customer.	Each loop in the spiral represents a phase in the process
Effective in the situations where requirements are defined precisely and there is no confusion about the functionality of the final product.	No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
Risks are never explicitly assessed and resolved throughout the process	Risks are explicitly assessed and resolved throughout the process

Rapid Development Application model (RAD)

RAD is proposed when requirements and solutions can be modularized as independent system or software components, each of which can be developed by different teams. User involvement is essential from requirement phase to deliver of the product. The process is stared with rapid prototype and is given to user for evolution. In that model user feedback is obtained and prototype is refined. SRS and design document are prepared with the association of users. RAD becomes faster if the software engineer uses the component's technology (CASE Tools) such that the components are really available for reuse. Since the development is distributed into component-development teams, the teams work in tandem and total development is completed in a short period (i.e., 60 to 90 days).

RAD Phases

- **Requirements planning phase** (a workshop utilizing structured discussion of business problems)
- **User description phase** – automated tools capture information from users
- **Construction phase** – productivity tools, such as code generators, screen generators, etc. inside a time-box. ("Do until done")
- **Cutover phase** -- installation of the system, user acceptance testing and user training



Martin Approach to RAD

Advantage of RAD

- Dramatic time savings the systems development effort
- Can save time, money and human effort
- Tighter fit between user requirements and system specifications
- Works especially well where speed of development is important

Disadvantage of RAD

- More speed and lower cost may lead to lower overall system quality
- Danger of misalignment of system developed via RAD with the business due to missing information
- May have inconsistent internal designs within and across systems
- Possible violation of programming standards related to inconsistent naming conventions and inconsistent documentation