Assignment

On

Software Process Models

DEPARTMENT OF COMPUTER SCIENCE
BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN PAKISTAN

Contents

Software process model

Software process model is an abstract representation of a software process. Each process model represents from a particular perspective and provides only partial information about that process.

There are three generic process models are defined for software engineering. These are:

- Waterfall Model
- Evolutionary Development
- Component-based Software Engineering

Three Other models are also important for the software engineering point of view. These are listed below.

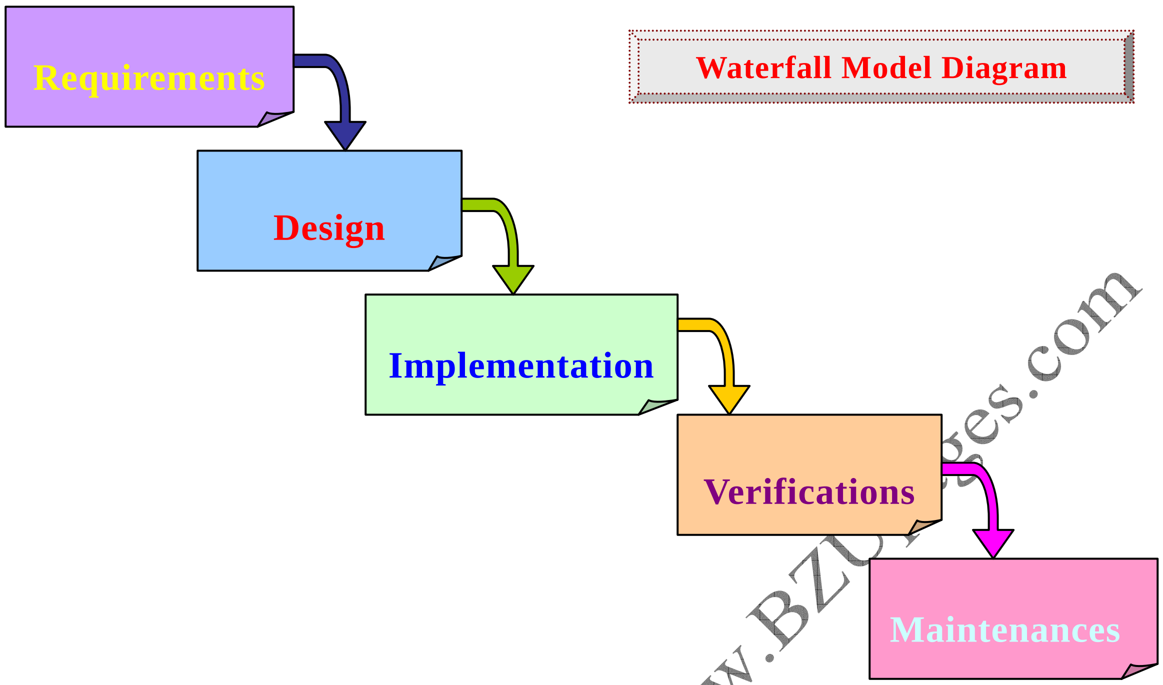
- Incremental Model
- Spiral Model
- Rational Unified Process Model

Now we explain all Software Process Models one by one.

Waterfall Model

It is the first published model of software development process in 1970 by Winston W. Royce (1929–1995) The waterfall model is a sequential development process, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design (validation), Construction, Testing and maintenance.

The unmodified "waterfall model" Progress flows from the top to the bottom, like a waterfall.



The Steps of Waterfall model are:

Requirements analysis and definitions: The system's services, constrains and goals are established by consultation with system users. They are then defined in details and serve as a system specification.

System and Software Design: The system design process partitions the requirements to either hardware or software systems. It establishes overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

Implementations and Unit testing: During this stage of the software development the software design is realized as a set if programs or program units. Unit testing involves verifying that each unit meets its specification.

Integrations and system testing: The individual program units or programs are integrated and tested as a complete system to insure that the software requirements have been met, after testing; the software system is delivered to the customers.

Operation and maintenance: Normally this is the longest life-cycle phase. The system is installing and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system services as new requirements are discovered.

Advantages of Waterfall Model: Fans of the Waterfall software development model will argue that the amount of pre-planning that goes into the requirements and design phases makes it the most economical and risk free way to develop software as it identifies and weeds out any potential problems at the outset. If these problems arose later in a project they could be very costly.

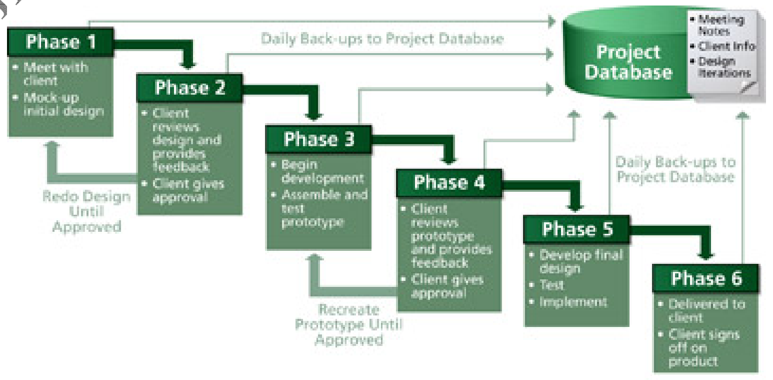
The Waterfall model also puts an emphasis on documentation and structure. This is an advantage when someone leaves the development team as the necessary documentation is there to help a new person take over.

Disadvantages of Waterfall Model: The Waterfall model certainly isn't to everyone's taste. Those who argue against it are usually opposed to its rigid structure and the inability to backtrack. It also isn't very client-focused as it makes any requests to change the software during the development process almost impossible to agree to. And while each phase of development should be 100% perfect before it is completed, it can become very complicated if they are not.

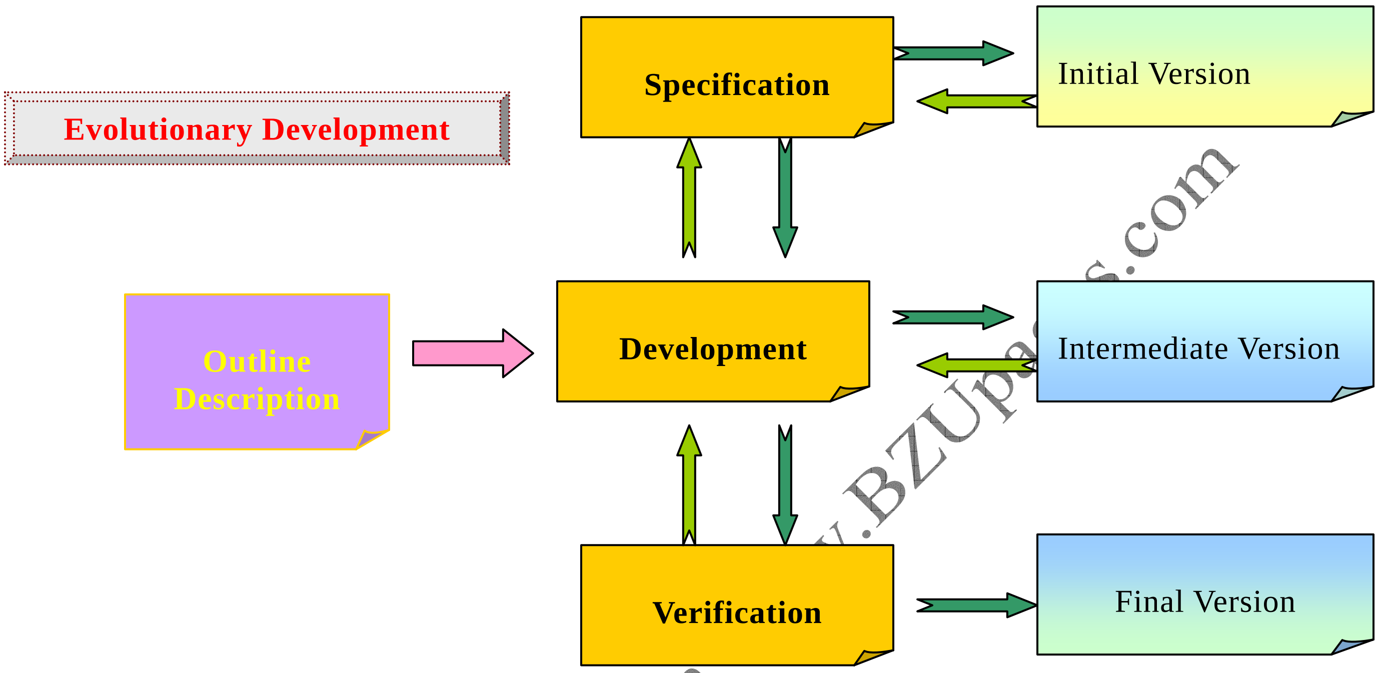
Some important Usages of waterfall model: The waterfall model is widely used by such large software development houses as those employed by the U.S. Department of Defense and NASA, and for many large government projects (see "the standard waterfall model" on the Internet Archive). Those who use such methods do not always formally distinguish between the pure waterfall model and the various modified waterfall models, so it can be difficult to discern exactly which models are being used and to what extent.

The U.S. Department of Defense has moved away from the waterfall model since 1994 with MIL-STD-498 and in 1998 with IEEE 12207.

The sashimi model (so called because it features overlapping phases, like the overlapping fish of Japanese sashimi) was originated by Peter DeGrace. It is sometimes referred to as the "waterfall model with overlapping phases" or "the waterfall model with feedback". Since phases in the sashimi model overlap, information of problem spots can be acted upon during phases that would typically, in the pure waterfall model, precede others.


Phase 1
• Meet with client
• Mock-up initial design
Redo Design Until Approved
Phase 2
• Client reviews design and provides feedback
• Client gives approval
Daily Back-ups to Project Database
Phase 3
• Begin development
• Assemble and test prototype
Recreate Prototype Until Approved
Phase 4
• Client reviews prototype and provides feedback
• Client gives approval
Phase 5
• Develop final design
• Test
• Implement
Phase 6
• Delivered to client
• Client signs off on product
Project Database
• Meeting Notes
• Client Info
• Design Iterations
Daily Back-ups to Project Database

Evolutionary Development



This type of Development is based on the idea of development an initial implementation exposing this to user comment and refining it through many versions until an adequate system has been developed. There are two fundamental types of evolutionary.

Exploratory development

Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.

Throw-away prototyping

Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

Importance of evolutionary approach:

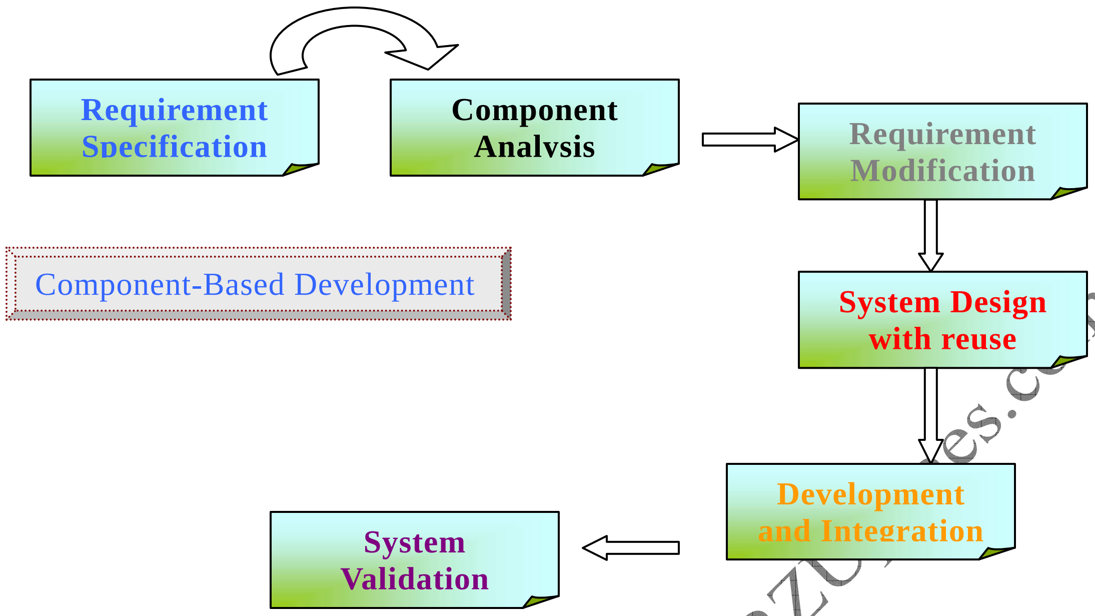
All evolutionary approach to software development is often more effective than the waterfalls approach in producing system that meet the immediate needs of customers. The advantage of a software process that is based on an evolutionary approach is that the specification can be developed incrementally. As user develop a better understanding of there requirements,

Problems of evolutionary approach: *The process is not visible* Manager need regular deliverable to measure progress. If system is developed quickly, It is not cost effective to produce documents that reflect every version of the system. *Systems are often poorly structured* Evolutionary changes may corrupt the software structure. It is used for the small and medium sized systems.

Component Based Software Engineering

Component-based software engineering (CBSE) (also known as Component-Based Development (CBD) or Software Component) We may say it is Reuse-oriented development.. To understand this Approach we first we discuss what is

Software component: A software component is a system element offering a predefined service or event, and able to communicate with other components. In the Approach Developer use a systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.



There are following important Process stages of Component based Software development...

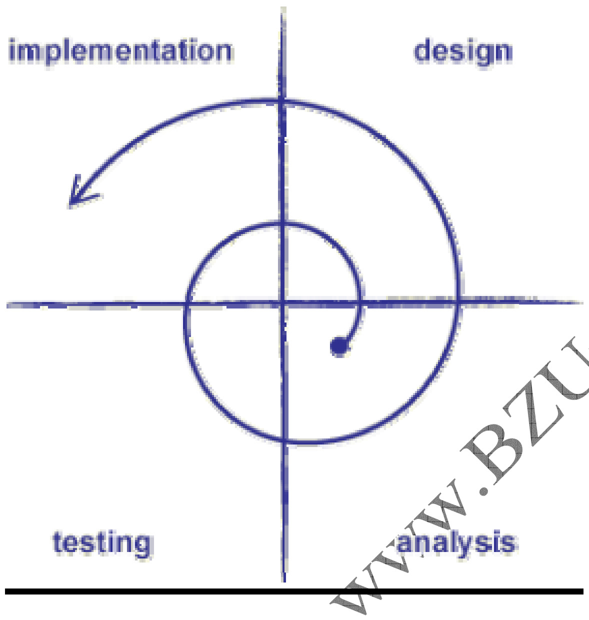
- ***Component analysis***: Given the requirement specification a search is made for components to implement that specification.
- ***Requirements modification***: In this stage the all requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components.
- System design with reuse: This phase of Component based Software development the frame work of system is designed or an existing frame work is reused by the developers. The designers take into account the components that re reused and organize the frame work to cater to this. Some new software's may have to be designed if reusable components are not available.
- Development and integration: Software that cannot be externally procured is developed and the components and COTS systems are integrated to create the new system. System integrations, in this model may be part of the

This approach is becoming increasingly used as component standards have emerged.

It is usually leads to the faster delivery if the software.

The Spiral Model

Four Repeated approaches in the Model are as following.

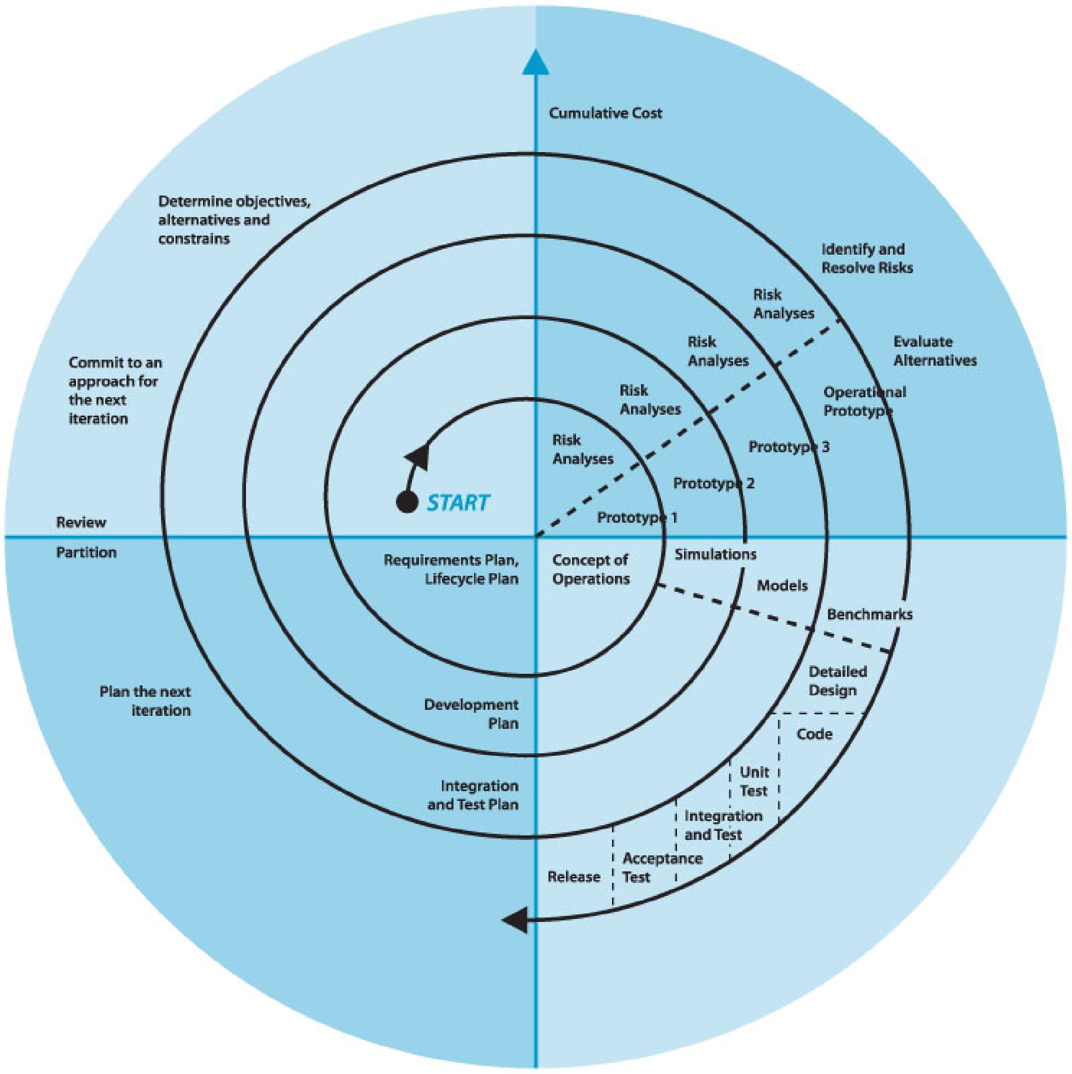


The spiral model, originally proposed by Boehm [BOE88], is evolutionary software Process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model . It provides the potential for rapid development of incremental versions of the software. Using the spiral model, Software is developed in a series of incremental releases. During early iterations, the Incremental release might be a paper model or prototype. During later iterations, Increasingly more complete versions of the engineered system are produced.

A spiral model is divided into a number of framework activities, also called task Regions.6 typically, there are between three and six task regions.

Spiral model that contains six task regions:

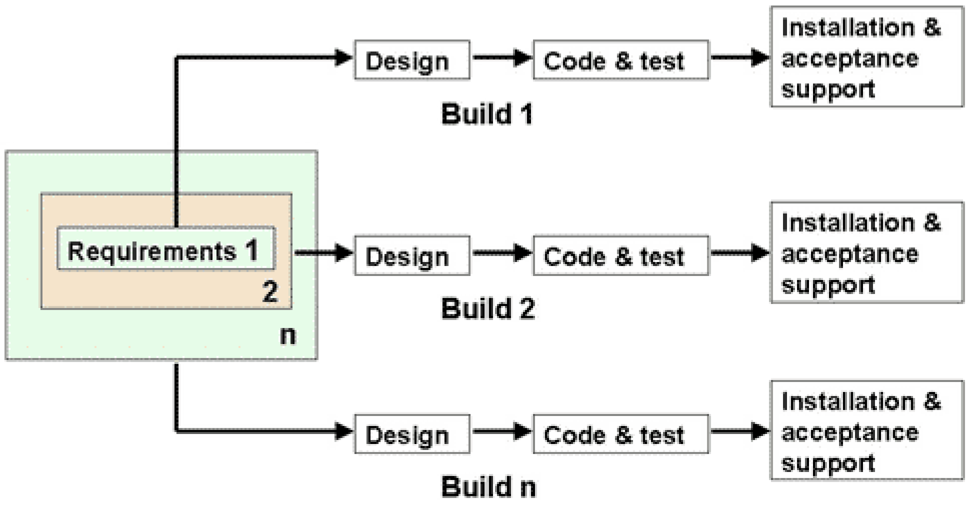
- **Customer communication** tasks required to establish effective communication between developer and customer.



- **Planning tasks** required to define resources, timelines, and other project related information.
- **Risk analysis** tasks required to assess both technical and management risks.
- **Engineering tasks** required to build one or more representations of the application.
- **Construction and release tasks** required to construct, test, install, and provide user support (e.g., documentation and training)

- **Customer evaluation tasks** required to obtain customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.

Incremental Model



The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

The software project is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

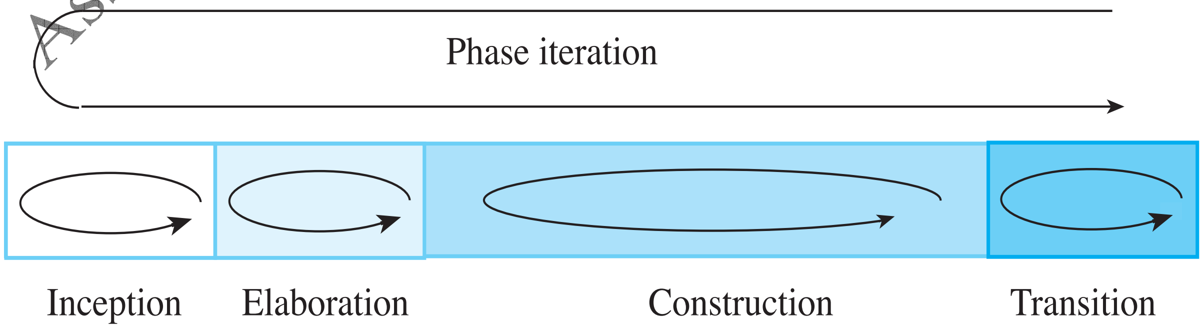
Some Problems: There are some problems with this model. One is that each new build must be integrated with previous builds and any existing systems. The task of decomposing product into builds not trivial either. If there are too few builds and each build degenerates this turns into Build-And-Fix model. However if there are too many builds then there is little added utility from each build.

Some Advantages of Incremental Model:

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

RUP (Rational Unified Process Model)

Stands for "Rational Unified Process. " RUP is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases that each involves business modeling, analysis and design, implementation, testing, and deployment.



There are four phases of Rational Unified Model of software Engineering process:

Inception - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.

Elaboration - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.

***Construction* -** The project is developed and completed. The software is designed, written, and tested.

Transition - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

The RUP development methodology provides a structured way for companies to envision create software programs. Since it provides a specific plan for each step of the development process, it helps prevent resources from being wasted and reduces unexpected development costs.

Some Advantages of **RUP** are:

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

The IBM Rational Method Composer product: The IBM Rational Method Composer product is a tool for authoring, configuring, viewing, and publishing processes.

RUP is similar in concept to Extreme Programming in that only what is useful and required is produced and the development plan is updated throughout the process. Both methods seek to develop a system of best practices in software development.