Process models

Specifies roadmap for software engineering wait

Defines the flow of all activity actions and tasks, degree of iterations, work products and organizations of the work that must be done

This brings order to the chaos of software development

However, software engineering work and the product that are produced remain on the edge of chaos (a natural state between order and chaos, a grand compromise between structures and surprise)

This is unstable because it is constantly attracted to chaos or to absolute order (absence of variability)

Chaos of software development is, unpredictable, complex and often disorganized nature of the process.

Order of software development is, structured methodologies, process and practices designed to bring predictability discipline and efficiency to the development of software thus bringing up software development life cycle.

Prescriptive process models

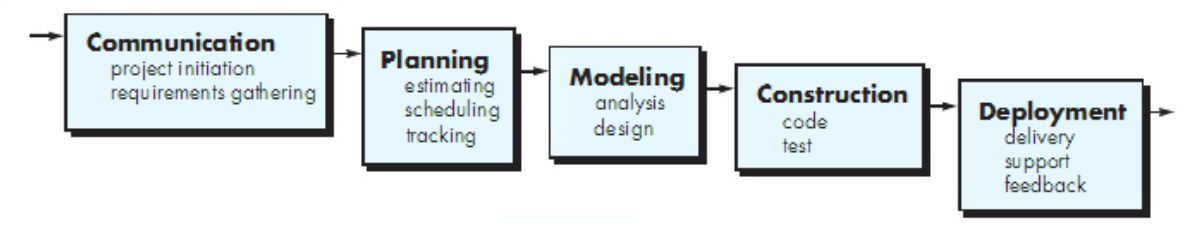
Prescriptive means the set of process models (framework activities, software engineering actions, tasks, work products, quality assurance and change control mechanisms)

Each process model also prescribes a process flow (work flow) the manner in which the elements are interrelated to one another

Prescriptive process models define a prescribed set of process elements and a predictable process workflow and are sometimes referred to as traditional process models

Types

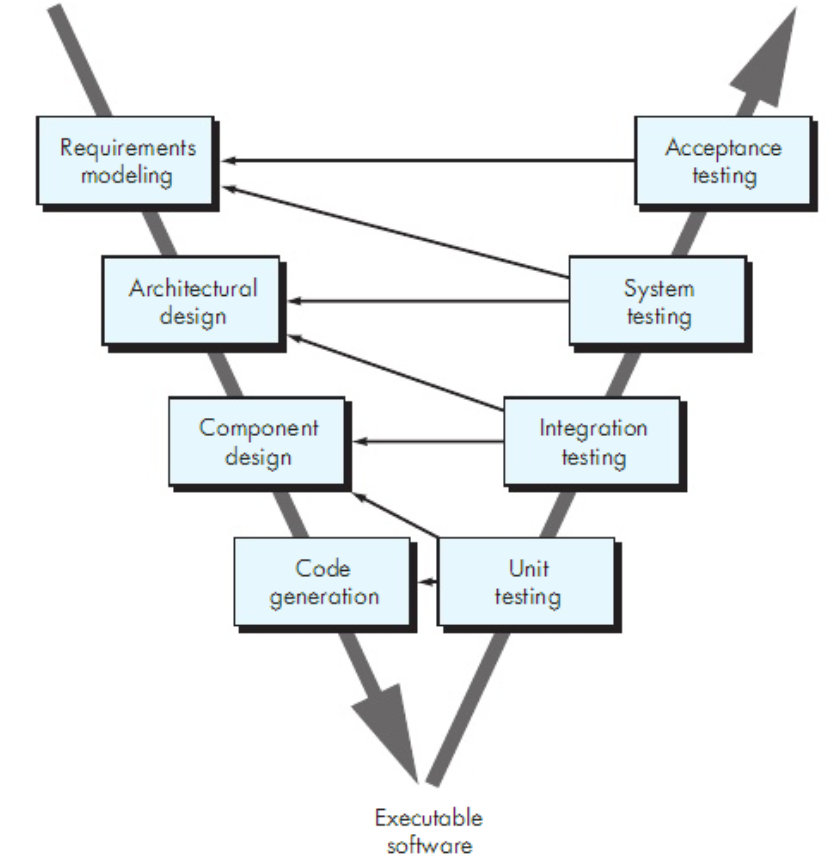
1. Waterfall model



It is also known as classic life cycle

It suggests that a systematic and sequential approach to software development that begins with customer’s specification of requirements and progresses through planning, modeling, construction and deployment.

A variation of the representation of the waterfall model is called a V-model



V-model

It depicts the relationship of quality assurance actions to the actions associated with communication, modelling and construction activities.

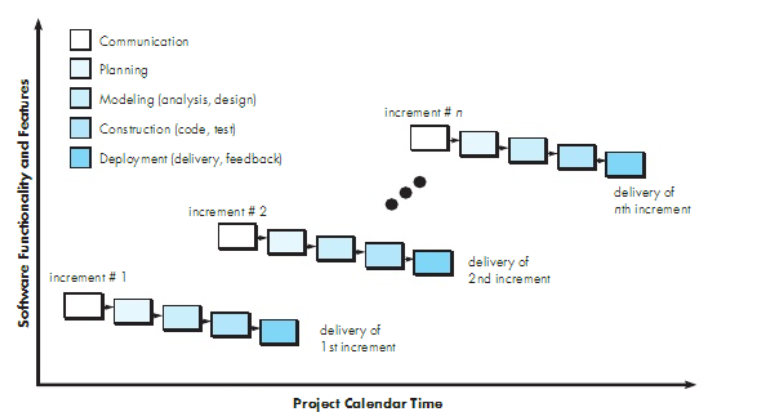
A software team moves down the left side of the V where basic requirements are refined into progressively more detailed technical representation of the problem and its solution. Once the code had been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moves down on the left side of the V.

This provides a way of visualizing how verification and validation actions are applied to the earlier engineering work

Problems encountered

* Real projects rarely follow the sequential flow that the model proposes thus can cause confusion as the project team proceeds
* It is often difficult for the customer to state all requirements explicitly
* The customer must have patience
* Leads to blocking state. Some project team members must wait for the other team members to complete dependent task

1. Incremental process models



This combines linear and parallel process flows. It applies linear sequences in a staggering fashion as calendar time progresses. Each linear sequences produces deliverable increments in the software

When this model is used, the first increment is often a core product, that is, basic requirements are addressed but many supplementary features, some known or unknown, remain undelivered.

The core product is used by the customer or undergoes detailed evaluation and as a result of use and or evaluation, a plan is developed for the next increment.

The plan addresses the modification of the core product to better the needs of the customer and the delivery of additional features and functionality.

This process is repeated following the delivery of each increment until the complete product is produced

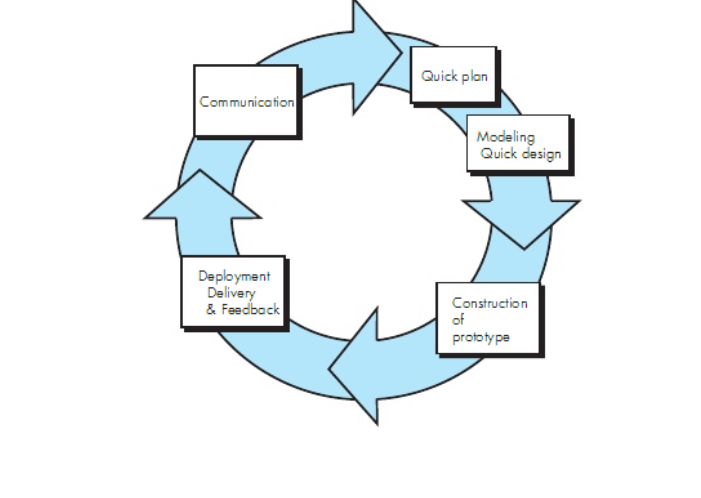
1. Evolutionary process models

Software evolves over a period of time and business and product requirements often change as development proceeds. In this situation, evolutionary process models are needed that has been explicitly designed to accommodate a product that grows and changes

These models are iterative, characterized in a manner that enables someone to develop increasingly more complete versions of the software

There are two common evolutionary process models

Prototyping



When a customer has a legitimate need but is clueless about the details, develop a prototype as a first step, prototyping paradigm

Can be used as a stand alone process model

This assists in understanding better what is to be built when the requirements are fuzzy

This begins with communication where the stakeholders define the overall objectives of the software, identify what requirements are known and outline areas where further definition is mandatory

A prototyping iteration is planned quickly and modelling in form of quick design occurs which focuses on the representation of the software that will be visible to the end user

This leads to the construction of a prototype

The prototype is deployed and evaluates by the stakeholders who provide feedback that is used to further refine the requirements

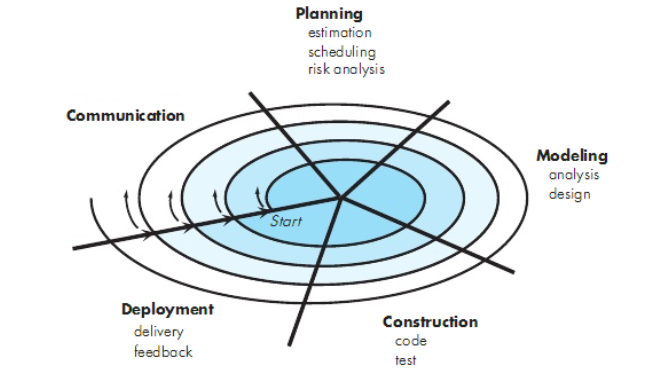
Iteration occurs as the prototype is tuned to satisfy the needs of various stakeholders and at the same time enabling you to better understand what needs to be done

Problems faced

Stakeholders see what appears to be a working version of a software unaware that the prototype is held together haphazardly

A software engineer makes implementation compromises in order to get the prototype to work quickly

The spiral model



It is an evolutionary process software that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model

It has 2 main distinctive features;

* cyclic approach for incrementally growing a systems degree of definition and implementation while decreasing its degree of risk
* set of anchor point milestones for ensuring stakeholders commitment to feasible and mutually satisfactory system solutions

it is divided into a set of framework activities defined by the software engineering team where each framework activity represents a segment of the spiral path

the first circuit around the spiral might result in development of the product specification, subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software

each pass through the planning region results in the adjustments of the project plan

cost and schedule are adjusted based on the feedback derived from the customer after delivery

the project manager adjusts the planned number of iterations required to complete the software

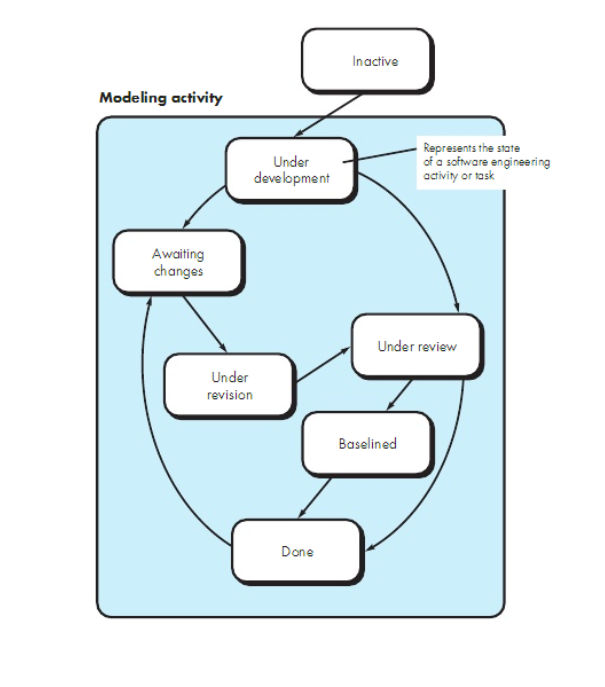
this model can adapt to apply throughout the life of a computer software, the first circuit around the spiral might represent a concept development project that starts at the core of the spiral and continues for multiple iterations until concept development is complete

problems faced

it may be to convince customers that the evolutionary approach is controllable

it demands considerable risk assessment expertise and relies on this expertise for success

concurrent models



sometimes called concurrent engineering

allows software team to represent iterative and concurrent elements of any of the process models

this model defines a series of events that will trigger transitions from state to state for each of the software engineering activities, actions or tasks

an inconsistency in the requirements is uncovered. This generates the event analysis model correction which will trigger the requirements analysis action from the done state into the awaiting changes state

this provides an accurate picture of the current state of a project

specialized process models

component based development

commercial off the shelf (COTS) software components developed by vendors who offer them as products, provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software that is to be built.

However, the component based development model comprises applications from the prepackaged software components

Steps

* available component based products are researched and evaluated for the application domain in question
* component integrated issues are considered
* a software architecture is designed to accommodate the components
* components are integrated into the architecture
* comprehensive testing is conducted to ensure proper functionality

the formal methods model

this encompasses a set of activities that leads to formal mathematical specification of computer software

formal methods enable you to specify, develop and verify a computer based system by applying a rigorous mathematical notation

a variation on this approach is called clean room software engineering

when formal methods are used during development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms like ambiguity, incompleteness and inconsistency through application of mathematical analysis

problems faced

* the development of formal models is currently quite time consuming and expensive
* because few software developers have the necessary background to apply formal methods, extensive training is required
* it is difficult to use the models as a communication mechanism for technically unsophisticated customers

aspect oriented software development

regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions and information content

the localized software characteristics are modelled as components and then structures within the context of a system architecture

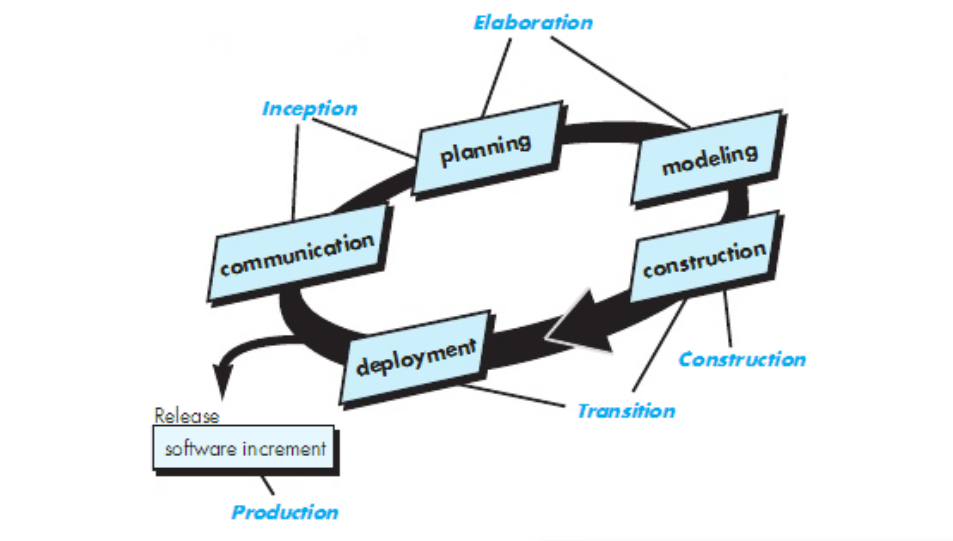
some concerns are high level properties of a system eg security other concerns affect functions eg application of business rules while others are systemic eg memory management

when concerns cut across multiple system functions, features and information are referred to as crosscutting concerns

aspect requirements define those crosscutting concerns that have an impact across the software architecture

aspect oriented software development is a software engineering paradigm that provides a process and methodological approach for defining, specifying, designing and constructing aspects

phases of unified process



Inception phase

This encompasses customer communication and planning activities

Business requirements for the software are defined, a rough architecture for the system is proposed and a plan for the iterative and incremental nature of the ensuing project is developed

Elaboration phase

This encompasses communication and modelling activities of the generic process models

It expands the architectural representation to include the use case model, the analysis model, the design model, the implementation model and the deployment model

Construction phase

Using architectural model as the input, this phase develops or acquires the software components that will make each use case operational for end users

Transition phase

The software is given to the end users for beta testing and user feedbacks reports both defects and necessary changes

The software team creates necessary support information eg user manuals that is required for the release

Production phase

The ongoing use of software is monitored

Support for the operating environment is provided

Defect reports and requests for changes are submitted and evaluated

Personal and team process models

Personal software process (PSP)

This emphasizes personal measurement of both the work product that is produced and the resultant quality of the work product

Five framework activities

* planning. This activity isolates requirements and develops both size and resource estimates. Development tasks are identified and project schedule is created
* high level design. External specifications for each component to be constructed are developed and a component design is created as well as the prototypes
* high level design review. Formal verifications applied to uncover errors in the design
* development. Component level design is refined and reviewed. Code is generated, reviewed, compiled and tested
* postmortem. Using the measures and the metrics collected, the effectiveness of the process is determined

team software process

goal is to build a self-directed project team that organizes itself to produce high quality software

objectives

* build self-directed team that plan and track their work, establish goals and own their processes and plans
* show managers how to coach and manage their teams and how to help them sustain peak performance
* accelerate software processes improvement
* provide improvement guidance to high maturity organizations
* facilitate university teaching of industrial grade team skills

defines the following framework; project launch, high level design, implementation, integration and test, postmortem

process technology

process technology tools are tools that have been developed to help software organizations analyze their current process, organize work tasks, control and monitor progress and manage technical quality. They allow software organizations to build an automated model of the process framework, tasks sets and umbrella activities thus reducing development time and cost