Incremental build model

The **incremental build model** is a method of [software development](https://en.wikipedia.org/wiki/Software_development) where the product is [designed](https://en.wikipedia.org/wiki/Software_design), implemented and [tested](https://en.wikipedia.org/wiki/Software_testing) incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance

The incremental model can be used when developing web applications where:

The incremental model is used when the requirements are clearly defined and understood.

A new technology is being used

There are some high risk features and goals.

Evolutionary process models

Prototyping

**Software prototyping** is the activity of creating [prototypes](https://en.wikipedia.org/wiki/Prototype) of software applications, i.e., incomplete versions of the [software program](https://en.wikipedia.org/wiki/Software) being developed

It begins with communication, meet with the stakeholders and define the overall objectives of the software.

A prototyping iteration is planned quickly and modelling occurs

A quick design leads to the construction of the prototype.

The prototype is deployed and evaluated by stakeholders

Prototyping is used where the user is unsure of the efficiency of the algorithm. Or the adaptability of the operating system.

Spiral Model

Couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model, it uses prototyping as a risk reduction mechanism.

The spiral model is used in the development of software when:

* When project is large
* When releases are required to be frequent
* When creation of a prototype is applicable
* When risk and costs evaluation is important

Concurrent Models

The concurrent process model defines a series of events that will trigger transitions from state to state for each of the software engineering activities.

The concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project.

Specialized process models

Component based development

It is evolutionary in nature and incorporates many of the characteristics of the spiral model, it demands an iterative approach to the development of software.

It incorporates the following steps:

* Available component based products are researched for the application domain.
* Component integration 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.

Formal methods model

Formal methods are mathematical techniques for developing computer-based software and hardware systems.

They enable one to discover and correct errors that might otherwise go undetected.

* Concern about applicability in the business environment:
* The development of formal models is currently quite time consuming.
* Extensive training is required because few software developers have the necessary background to apply the formal methods model.
* It is difficult use the models as a communication mechanism for technically unsophisticated customers.

Aspect oriented software development

**A***spect-***o***riented* **s***oftware* **d***evelopment (***AOSD***)* is an approach to software development that addresses limitations inherent in other approaches, including object-oriented programming. AOSD aims to address crosscutting concerns by providing means for systematic identification, separation, representation and composition.

Crosscutting concerns are encapsulated in separate modules, known as aspects, so that localization can be promoted. This results in better support for modularization hence reducing development, maintenance and evolution costs.

Unified process model

This is an attempt to draw on the best features and characteristics of traditional software process model. It recognizes the important of customer communication and streamlined methods for describing the customer’s view of a system.

Phases of the unified process model:

Inception – encompasses both customer communication and planning activities, by collaborating with stakeholder’s business requirements for the software are identified.

Elaboration – encompasses the planning and modelling activities of the generic process model, it refines and expands the preliminary use cases of the inception phase.

Construction - develops the software components that will make each use case operational for the end user.

Transition – encompasses the later stages of the generic constriction activity, software is given to end users for beta testing and feedback is given do that necessary changes can be implemented.

Production – coincides with the deployment activity of the generic process

Specialized process models

The personal software process

The personal software process (PSP) emphasizes personal measurement of both the work product that is produced and the resultant quality of the work product.

It defines five framework activities:

Planning: Isolates requirements and develops both size and resource estimates.

High level design: external specifications for each component to be constructed are developed.

High-level design review: formal verification methods are applied to uncover errors in the design.

Development: Code is generated, reviewed, compiled and tested.

Post-mortem: The effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve effectiveness.

The team software process

The goal of TSP is to build a “self-directed” project team that organizes itself to produce high-quality software.

Objective for TSP:

Build self-directed teams that plan and track their work.

Show managers how to coach and motivate their teams.

Accelerate software process improvement

Facilitate university teaching of industrial-grade team skills.

What is an Agile Process?

An agile software process is a process with a more than effective response to change.

Extreme process

Uses an object oriented approach as its preferred development paradigm, and it encompasses a set of rules and practices that occur within the context of: planning, design, coding and testing.

The planning activity begins with listening, a requirements gathering activity that enables technical members of the XP team to understand the business context of the software.

Design follows the KIS principle. A simple design is best, CRC used as a mechanism for thinking about software, CRC encourages the immediate creation of an operational prototype, if a design problem is encountered.

Coding Unit test are done before moving on to code, once unit tests are done, the developer is better able to focus on what must be done to pass the test. Pair programming is used here.

Testing commences where the complete software is tested to see if it meets the set requirements.

Extreme process is used when the customers do not have a firm idea of what the system should do. It is also used when there is too much project risk involved in the project.

Industrial XP

This is an organic evolution of XP, it has a greater inclusion of management and

an expanded role for customers.

IXP incorporates six new practices that are designed to ensure that an XP project works successfully.

* Readiness assessment – ascertains whether all members of the project community are on board and understand the skill levels involved.
* Project community – determines where the right people with the right skills and training have been staged for the project.
* Project chartering – The IXP team assesses the project itself to determine whether an appropriate justification for the project exists.
* Test driven management – An IXP team establishes a series of measurable destinations that assess progress to date.
* Retrospectives – The IXP team conducts a specialized technical review after a software increment is delivered.
* Continuous learning – The IXP team is encouraged to learn new methods and techniques that can to higher quality product.

Industrial Extreme process is also used when the customers do not have a firm idea of what the system should do. It is also used when there is too much project risk involved in the project.

Dynamic systems development method

Is an agile development approach that provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping.

DSDM then defines three different iterative cycles:

Functional model iteration – provides a set of incremental prototypes that demonstrate functionality for the customer.

Design and build iteration – revisits prototypes built during the functional model iteration.

Implementation – places the latest software increment into the operational environment.

DSD process is also used when the customers do not have a firm idea of what the system should do. It is also used when there is too much project risk involved in the project.

Agile Modelling

|  |
| --- |
| Agile Modelling (AM) is a practice-based methodology for effective modelling and documentation of software-based systems.  Principles that make up AM are:  Model with purpose - a developer who uses AM should have a specific goal.  Use multiple models:  Travel light - As software engineering work proceeds keep only those models that will provide long term value and jettison the rest. |

Content is more important than representation – modelling should impart in formation to its intended audience.

Adapt locally – The modelling should be adapted to the needs of the agile team.

Agile Unified Process

It describes a simple, easy to understand approach to developing business application software using agile techniques.

Each AUP iteration addresses the following activities.

Modelling: UM - representations of the business and problem are created.

Implementation - Models are translated into source code.

Testing – A series of unit tests are designed and executed to uncover errors and ensure that source code meets requirements.

Deployment – this focuses on delivery of the software increment and acquisition of feedback from end users.

Configuration and project management: Configuration management addresses change management, risk management and control of any persistent work products.

Environment management – coordinates a process infrastructure that includes standards, tools, and other support technology available to the team.

The Agile unified process is also used when the customers do not have a firm idea of what the system should do. It is also used when there is too much project risk involved in the project.