System Development life cycle

1. Project initiation
2. Project planning
3. Analysis
4. Design
5. Implementation
6. Deployment
7. Support

Traditional Development – moving onto the next stage after the current is complete, its better when requirements are clearly defined

Agile Development

1. Develop software as the primary goal
2. Enable the next effort as your secondary goal
3. Minimize your modelling activity
4. Embrace change and change incrementally
5. Model with a purpose
6. Build multiple models
7. Build high quality models and get feedback
8. Focus on content rather than representation
9. Learn from each other with open communication
10. Know your models and how to use them
11. Adapt to specific project needs
12. Maximize stakeholder ROI

Unified Process

1. Develop iteratively
2. Define and manage system requirements
3. Use component architectures
4. Create visual models
5. Verify quality
6. Control changes

Scrum – entire team focus on one thing for a short period of time

Extreme Programming Practices

1. Planning

2. Testing

3. Pair Programming

4. Simple Designs

5. Refactoring

6. Owning the code collectively

7. Continuous integration

8. On-site customer

9. System metaphor

10. Small releases

11. Forty-hour work week

12. Coding standards

Advantages

* Close contact with the customer
* No unnecessary programming work
* Stable software through continuous testing
* Error avoidance through pair programming
* No overtime, teams work at their own pace
* Changes can be made at short notice
* Code is clear and comprehensible at all times

Disadvantages

* Additional work
* Customer must participate in the process
* Relatively large time investment

Properties of good requirements

* Understandable by users
* Non-prescriptive
* Correct
* Complete
* Consistent
* Unambiguous
* Precious

Functional requirements

* Must have
* Should have
* Could have
* Won’t have

Non-functional requirements

* Usability
* Resilience
* User experience
* Performance
* Economic constraints
* Aesthetic concerns
* Comprehensibility
* Technology concerns
* Security

Internal stakeholders – people within the organisation

External stakeholders – people outside the organisation

Requirements testing – testing the information required from the requirements

Acceptance testing – testing the information system against the requirements

Predictive methodologies - model is designed, implemented, and tested incrementally until the product is finished.

Adaptive methodologies - freeform software design as it offers an incredibly flexible design model, promoting adaptive planning and evolutionary development when the end goal is not quite as concrete.

**Business processes and how they are modelled**

Anomalies are repetition of data in a database

Table

Description automatically generated

Normalisation splits a database up into smaller relations get rid of the anomalies

State – object at a certain point in time

Behaviour – things you can do on state

Encapsulation – code acting upon data

Inheritance – acquired properties from another class

Separation of public/ private API

- Public APIs have to maintain certain restrictions because of their generic target audience, Private APIs don't have to suffer this limitation

Monolithic Application – single tiered application with software and interface are combined

Client Server Application – runs on the client side and access remote server

Distributed Application – program runs on one or more application and communicates through a network

**Separation of responsibilities**

**Standard pattern of design**

**System templates**

S**oftware quality dimensions are accuracy and capability**

Verification – software meets requirements

Validation – how it addresses the business needs

Types of Testing

* Functional
* Structural
* Component
* Drivers
* Fixtures
* Best practices

File system – unstructured data store

Database – structured data base

Database API – lets application connect to data bases

HTTP – protocol the browser and web server use to communicate

HTML – structure of the document

CSS – style to the document

Cookies – web application can instruct the web browser to store small key value pairs

Sessions – temporary data store inside web applications

Client side – action takes place on the client computer

Server side – action takes place on the web server

Resources – hardware or software that can access another computer

Filter chains - object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource

NPV - sum of costs and benefits in the future

A picture containing clock

Description automatically generated

ROI - what is the discounted benefit relative to the cost

Example of NPV:

Table

Description automatically generated

Reasons to ignore

* Legal risk
* Reputational risk
* Too high risk
* Not aligned with organisation

Business function – specific responsibilities to perform activities on a regular basis

Business mechanism – activates which draw on functions to help with the mission

Direct development – when a child is a smaller version of the parent

Parallel development – simultaneous development of more than one version

Phased development – long term development with flexible parameters

Rich domain models - Sends object native instructions beyond basic getter and setter methods

Anemic Domain Models – mainly just state behaviour

Swim lanes – show the interactions with activities

Transitions – represent activity

Decision points – reflect the previous activity

Synchronisation bars - forks

Activity Diagram

Diagram

Description automatically generated

State Diagrams

Diagram

Description automatically generated

Use case diagrams

Actors – people interacting with system

Use cases – unit of system function

Associations – relationship between actors and use cases

Dependencies – between use cases

Diagram

Description automatically generated

ERD concepts

Subtypes – model mutual exclusivity

Parallel relationships – separate relationships between the same relationships

Recursive relationships – relationships call themselves

Historical data – break up relationship with many to many

Diagram

Description automatically generated

ERD vs class diagram

ERD

* separate cardinality and participation
* roles names are revered
* unique identifiers built in
* value require (\*)

Class Diagrams

* more detailed multiplicity
* richer relationship semantics

Class Diagrams

Diagram

Description automatically generated with medium confidenceText

Description automatically generated

Sequence diagrams

Diagram

Description automatically generatedText

Description automatically generated