# Glossary

**Abstraction**

The process of generalising related things; in software design ‘abstraction’ means identifying an “is a type of” relationship, which is important in software design and data modelling.

Example: A cat *is a type of* Animal. A dog *is a type of* Animal. Hence, Animal is an abstraction of a dog and a cat.

**Agile Practices**

Flexible methods for developing software that prioritize teamwork, quick results, and customer feedback.

Example: A team releases a basic version of an app, then regularly updates it based on user feedback, rather than trying to perfect it before the first release.

**Algorithm Design**

Creating step-by-step instructions, usually involving a flow chart and/or pseudocode, to solve a problem or perform a task. See Flowchart and Pseudocode

Example: Binary Search Algorithm - Imagine you're searching for a word in a dictionary. Instead of starting from the beginning, you open the book in the middle. If your word comes before the middle, you search the first half; if after, you search the second half. You repeat this process, halving the search area each time until you find the word.

**Application Program**

Software designed for end-users to perform specific tasks.

Example: Microsoft Word, used for creating and editing documents.

**Compilers/Interpreters**

Tools that translate human-written code into a data format computers can understand (known as machine code).

Example: A Java compiler converts Java code into bytecode that can run on any device with a Java Virtual Machine.

**Database**

A database is an organized collection of data stored and accessed electronically. Databases are designed to offer an efficient way to store, retrieve and manage information.

Databases can be classified based on their organizational approach. The most prevalent type is the relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways.

Example: Consider a library system. It might have the following tables:

1. **Books**: Stores information about books such as title, author, genre, publication year, ISBN, etc.
2. **Members**: Stores member information like member ID, name, address, phone number, email, etc.
3. **Borrowed Books**: Stores information about which member has borrowed which book, the date of borrowing, due date for returning, etc.

**Data Collection Tools**

Software or devices used to gather and record information.

Example: A website form that collects customer feedback, a temperature meter on a weather station.

**Data Integrity**

Ensuring data remains accurate and consistent over time.

Example: A bank's system double-checking that an account balance is correct after every transaction.

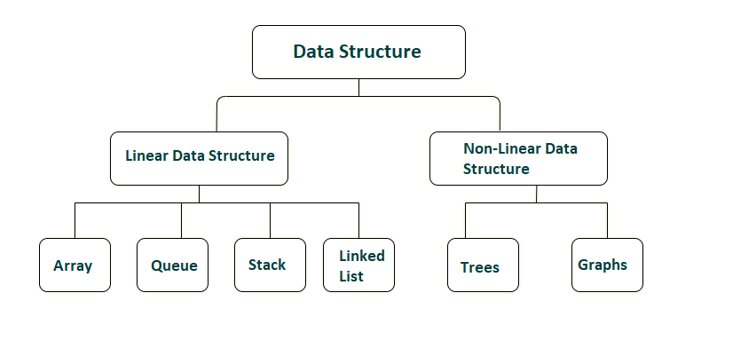
**Data Sets**

Collections of related information used for analysis or testing.

Example: A spreadsheet containing the test scores of all students in a school.

**Data Structures**

Ways of organizing data for efficient use in programs.



Example: A to-do list app using a list to store task items. A chess game using a tree to calculate all the possible outcomes over the next three moves.

**Debuggers**

Tools that help find and fix errors in computer programs.

Example: Using a debugger to pause a program and inspect variable values when it crashes.

**Decomposition**

Breaking a complex problem into smaller, more manageable parts. In programming, decomposition often involves breaking down a large program into smaller functions or classes. In project management, it might mean dividing a project into smaller, manageable tasks.

Example: Dividing the task of building a car into subtasks like designing the engine, body, and interior.

**Dynamic Website**

A website that can change its content based on user input or other factors. A dynamic website typically relies on a database and business logic which runs on the server whenever a webpage is viewed, in order to show user specific content.

Example: A news website that shows different articles based on a user's location or preferences.

**Encryption**

Scrambling information so only authorized parties can read it.

Example: WhatsApp encrypting messages so only the sender and receiver can read them.

**Feature Development**

The process of adding new capabilities to a product.

Example: Adding a dark mode option to a mobile app.

**File Management Techniques**

Methods for organizing and handling computer files efficiently.

Example: Using folders to organize documents by project or date.

**Flow Chart**

A diagram that shows the steps in a process using shapes and arrows. Flowcharts can be used to describe high-level processes such as whether or not to seek permission for use of an image in a photo (see figure 1) or they can describe an algorithm, such as providing output based on input (see figure 2).

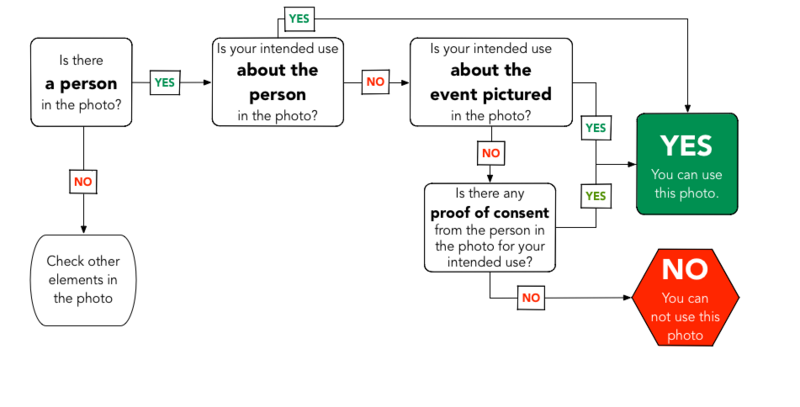


Figure 1: [File:Flow chart determining reusability of a portrait photo.png - Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Flow_chart_determining_reusability_of_a_portrait_photo.png)

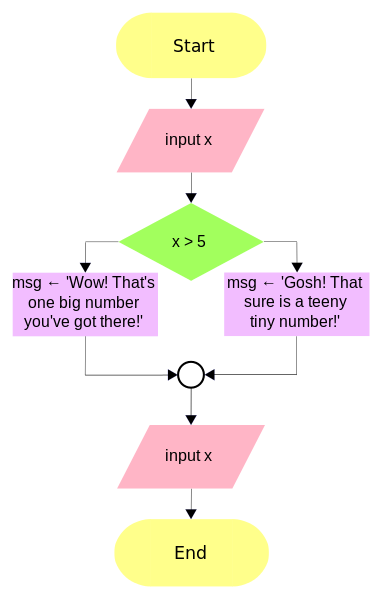


Figure 2: [File:Flowchart connector example.svg - Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Flowchart_connector_example.svg)

**General Purpose Programming Language (GPL)**

A programming language that can be used to create software across many different software domains (broad domains include Desktop Applications, Console/System Applications, Web Applications, Mobile Applications).

This is distinguished from a Domain Specific Programming Language (DSL), which is designed for use in a specific function within a software domain (e.g Database Querying (SQL), Video Game Scripting (LUA), Webpage Presentation (HTML & CSS)).

**Some Contemporary GPLs include:**

* **Python** - High-level, versatile language known for readability and extensive libraries
* **JavaScript** - Primary language for web development, both client and server-side
* **Java** - Object-oriented language, "write once, run anywhere" philosophy
* **C++** - Powerful language for system/application development and game programming
* **C#** - Microsoft's language for Windows and game development with Unity
* **PHP** - Server-side scripting language for web development
* **TypeScript** - Typed superset of JavaScript for large-scale applications
* **C** - Low-level language for system programming and embedded systems
* **Go** - Concurrent, compiled language designed for simplicity and efficiency
* **Kotlin** - Modern alternative to Java, officially supported for Android development
* **Swift** - Apple's language for iOS, macOS, and other Apple OS development
* **Rust** - Systems language focusing on safety and concurrency without sacrificing performance
* **Ruby** - Dynamic language known for its simplicity and productivity
* **Scala** - Combines object-oriented and functional programming on the JVM
* **Dart** - Google's language for web, mobile, and desktop application development
* **Lua** - Lightweight scripting language popular in game development
* **Objective-C** - Apple's older language for macOS and iOS development
* **Julia** - High-performance language for technical computing
* **Haskell** - Purely functional programming language

**The following are not GPLs, as they are Domain Specific Languages (DSLs) (Although they are commonly used in conjunction with a GPL, such as combining JavaScript (a GPL) with HTML and CSS (both DSLs))**

* **HTML** - Markup language for structuring web content
* **CSS** - Style sheet language for describing the presentation of web documents
* **SQL** - Language for managing and querying relational databases
* **Scratch** - Visual programming language for creating interactive stories and games
* **JSON** - Lightweight data interchange format, easy for humans to read/write
* **XML** - Markup language for storing and transporting data
* **GDScript** - Python-like scripting language for the Godot game engine
* **UnrealScript** - Object-oriented language for Unreal Engine game development
* **GameMaker** Language - Scripting language for GameMaker Studio
* **Regular Expressions** - Sequence of characters for pattern matching in strings
* **GraphQL** - Query language for APIs, allowing clients to request specific data
* **YAML** - Human-friendly data serialization standard for configuration files
* **Blockly** - Visual programming language for creating block-based coding interfaces
* **Twine** - Language for creating interactive, nonlinear stories
* **Logo** - Educational programming language known for turtle graphics

**Integrated Development Environments (IDEs)**

Software that provides tools for writing, testing, and debugging code in one place.

Example: Visual Studio Code, which offers features like syntax highlighting, code completion, and built-in terminal.

**Iterative Engineering Design Processes**

A cyclical approach to developing products, involving repeated rounds of planning, creating, and testing.

Example: Releasing a beta version of a game, gathering player feedback, and then updating the game based on that feedback.

**Market Research**

Gathering information about potential customers and competitors to inform business decisions.

Example: Surveying smartphone users to understand what features they want in a new app.

**Object-Oriented Design**

Object-Oriented Design (OOD) is a software design method that models a system as a collection of interacting objects. Each object, which is an instance of a class, contains its own state data and handles its operations.

Classes are abstract blueprints used to create more specific, concrete objects. They often represent broad categories, like Car or Dog, that share attributes. These classes define what attributes an instance of this type will have, like colour, but not the value of those attributes for a specific object.

Classes can also contain functions called methods that are available only to objects of that type. These functions are defined within the class and perform some action helpful to that specific object type.

Example: let’s consider a Library System as an example of Object-Oriented Design in Python. In a library system, we can have several classes like Library, Book, and Member. Each of these classes represents different entities with their own attributes and behaviours.

In the following example, **Member**, **Book**, and **Library** are classes. **Member** has attributes like *name* and *address*. **Book** has attributes like *title* and *author*. **Library** has an attribute: *books* which is a list of **Book** objects, and methods *add\_book* and *list\_books*.

An instance of **Member** is created with the name “John Doe” and address “123 Street, City”.

Instances of **Book** are created with different titles and authors. An instance of **Library** is created and the *add\_book* method is called to add books to the library.

The *list\_books* method is called to list all the books in the library.

class Member:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

class Book:

def \_\_init\_\_(self, title, author):

self.title = title

self.author = author

class Library:

def \_\_init\_\_(self):

self.books = []

def add\_book(self, book):

self.books.append(book)

print(f"Added {book.title} by {book.author}")

def list\_books(self):

for book in self.books:

print(f"{book.title} by {book.author}")

#Program Starts Here:

member = Member("John Doe", "123 Street, City")

book1 = Book("Book Title 1", "Author 1")

book2 = Book("Book Title 2", "Author 2")

library = Library()

library.add\_book(book1)

library.add\_book(book2)

library.list\_books()

**Pattern Recognition**

Identifying recurring themes or regularities in data or problems.

Example: Observing that users tend to spend more on their second purchase then their first. Noticing that successful e-commerce websites often have a similar checkout process, and applying that pattern to a new site.

**Programming Language**

A programming language is a formal system of communication designed for humans to instruct computers to perform specific tasks. Programming languages are either General Purpose or Domain Specific.(for more information, see General Purpose Programming Language)

**Project Time-Scale**

A timeline showing the planned duration and milestones of a project.

Example: A calendar showing that a new app will be designed in March, developed in April-May, tested in June, and launched in July.

**Proof of Concept**

A small project to test if an idea is feasible before fully developing it.

Example: Building a basic prototype of a flying car to show it's possible before investing in full production.

**Pseudocode**

A way of planning computer programs using simple, English-like statements.

Example: Designing a set of steps for a computer to find the cheapest and most expensive item in a list of groceries:

cheapest = groceries[0]

mostExpensive = groceries[0]

For each item in groceries

If item.price < cheapest.price Then

cheapest = item

End If

If item.price > mostExpensive.price Then

mostExpensive = item

End If

End For

Print "Cheapest item: " + cheapest.name

Print "Most expensive item: " + mostExpensive.name

**Risk Minimization Strategies**

Plans to reduce potential problems in a project.

Example: Regularly backing up code to prevent loss if a computer crashes.

**Stakeholder/User Representative**

Someone who speaks for the people who will use or be affected by a product.

Example: A teacher advising developers on an educational app to ensure it meets students' needs.

**Storyboard**

A series of drawings or images that show how a user might interact with a product.

Example: Sketches showing the screens a user would see when ordering food through a delivery app.

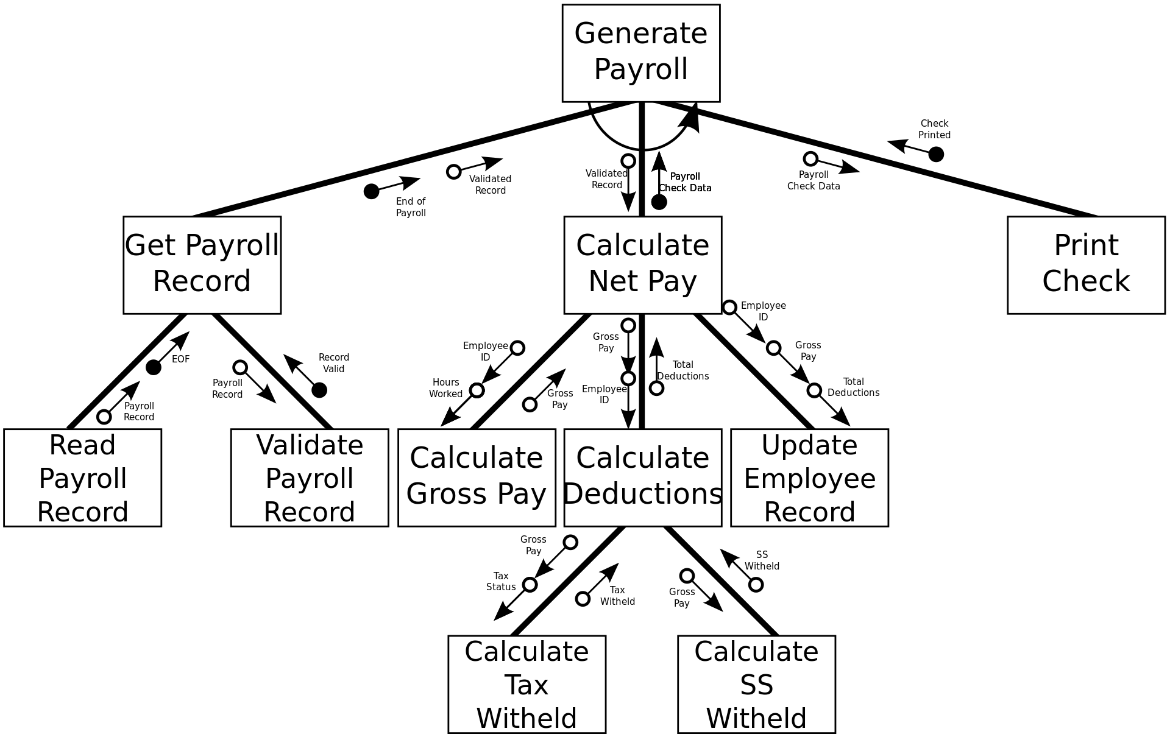
**Structure Chart**

A structure chart, also known as a hierarchical chart, is a diagram that shows the organization and relationships between different components or modules in a system, typically used in software engineering and project management. It represents a top-down design approach, breaking down a complex system into smaller, more manageable parts.

Key features:

1. Hierarchical layout
2. Main function or module at the top
3. Subordinate functions or modules branching downward
4. Connections showing relationships between modules

Example: A structure chart showing the process of performing a payroll cycle.



3 Sandia National Laboratories (1992). [Sandia Software Guidelines Volume 5 Tools, Techniques, and Methodologies](http://www.prod.sandia.gov/cgi-bin/techlib/access-control.pl/1985/852348.pdf) SANDIA REPORTS 85–2348qUC–32

**User Stories**

User stories are concise, informal descriptions of a software feature from the perspective of the end user. They are commonly used in agile software development to capture product requirements.

Example: "As a social media user, I want to be able to edit my posts so that I can fix typos."

**Version Control**

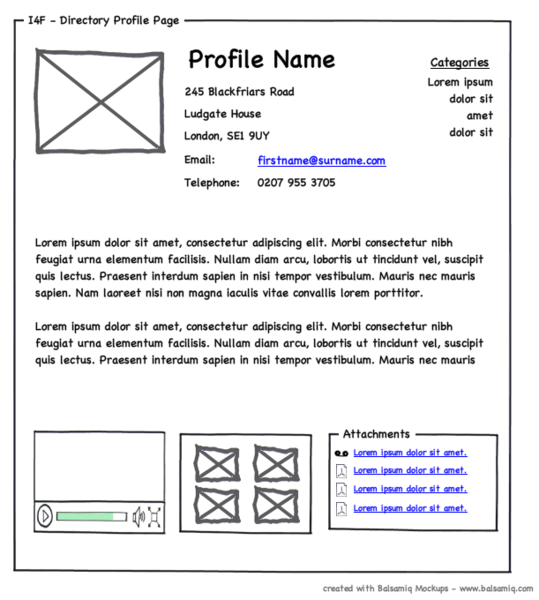
A system for tracking and managing changes to files over time. This is used frequently for text files containing project code.

Example: Using Git to save different versions of a website's code as it's developed.

**Wireframe Models**

Simple, visual representations of a product's layout and structure.

Example: A basic sketch showing where the menu, content, and buttons will go on a website, without details like colours or images.



4<https://www.flickr.com/photos/doos/3931846833/>