The **SW Product Life Cycle (PLC)** is a framework by which an organization manages the development of its products from inception to EOL. The 4 key phases of PLC are:

- **Exploration:** Analyze market, business, technology trends & opportunities which identify product solutions.
- **Planning:** Formalize next level of detail, including market requirements, decision times, product scope, usage, features.
- **Development:** Implement requirements defined in planning phase. Milestones are synchronization points to quantify progress.
- **Refresh:** Constitute further product revisions (or EOL) after initial product launch. May include updates to SW or HW and other.

Five business & technical goals for large SW projects & managers are as follows:

- **Cost:** Must be within the bounds of estimated space, money, and time.
- **Portability:** Must use widely available tools, and languages. SW involving a single language is best.
- Robustness: No bugs, managing and repair strategy.
- **Legality:** Have rights to all tools and establish clear rights to your created SW
- **Backup:** Define time between 2 complete backups if recovery is needed. Cost is known in case of total loss.
- **Legal and Ownership Issues**

Typical SWE Challenges

- **Legacy Systems:** Old valuable system must be maintained and updated.
- **Heterogeneity:** Systems are distributed and include a mix of HW and SW.
- **Delivery Time:** Increasing pressure for SW delivery time.
- **Other issues:** Meeting requirements, staying within budget, documentation, on-time delivery, and inefficient use of resources.

Common Architecture Attributes:

- Main memory separate from the CPU
- Program instructions stored on main memory
- Data stored in memory, AKA Von Neuman's Architecture
- Instruction pointer (instruction counter, program counter)
- Von Neuman's memory bottleneck (everything on the same bus)
- Accumulator (1 or many) holds logical or arithmetic results.

SW Processes and **SWE models** are a structured set of activities that are used to develop a SW system. They involve:

1. Specification 2. Design 3. Validation 4. Evolution

A **SW Process Model** is an abstract representation of a SW process

The Waterfall Model separates distinct phases of specification and development. It is used to help organize and define SW development. Its main drawback being its difficulty accommodating change after the process is underway. The steps/phases are as follows:

- Requirements Analysis and Definition
- System and Software Design
- Implementation and Unit Testing
- Integration and System Testing
- Operation and Maintenance

Incremental Development process is a SW development module. Development and delivery is broken into increments with each component delivering a piece of the required functionality. Each delivery of an increment generates early feedback which allows for correction and refinement of the product. This allows the SW to be more accurate to the customers needs. Cons, more manpower and resources expended to produce the numerous increments.

Data-Stream, Instruction Stream Classification
SISD - Single Instruction, Single Data Stream
SIMD - Single Instruction, Multiple Data Stream
MISD - Multiple Instruction, Single Data Stream
MIMD - Multiple Instruction, Multiple Data Stream

Reuse Oriented Development or **Component Based SWE** can be extremely wise. Chances are there is something similar that already exists. You are responsible for finding it. If so, can you acquire it legally? It it cost effective? Is it well documented? The idea is to cheapen and shorten the process of making software. However, be careful to verify that it does what you expect and that modifying it won't be more time consuming than developing from scratch.

Software Engineering (SWE) is the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software

Key Goals and Deliverables during a SWE project include:

- Correct Function: That which works, must work correctly.
- Complete Function: All promised features must work.
- Acceptable, define speed must be defined.
- Low cost of development and maintenance.
- Portability to other computer environments.
- Complete documentation and high stability and reliability.

Testing and validation includes:

- **Component Testing**: Individual components tested independently. May be functions, objects, modules, and/or coherent groups.
- **Systems Testing:** Testing of functioning system as a while. Validating emergent properties is of particular importance.
- **Acceptance Testing:** Testing customer to verify that system meets needs. Should be as large as necessary and as small as possible.

Requirements Engineering is the SWE process of establishing the services the customer requires from SW, and constraints under which it operates.
Requirements Engineering Process:

1. Feasibility Study 2. Requirement Elicitations & Analysis
   3. Requirements Specifications 4. Requirements Validation.

Programming & Debugging involves 1. Locate Error 2. Design Error Repair 3. Repair Error 4. Re-Test Program.

Validating - Process of running SW to demonstrate a requirement.
Verification - Verify the solution works.
Verification and validation (V&V) is intended to show that a system conforms to specification and meets the requirements of the customer (end-user)

Rational Unified Process (RUP) is a generic SW development process practiced in (some mature) SW industry.

Unified Modeling Language (UML) is a modelling tool used to formally describe a product that consists of SW solutions. Sometimes can be turned into actual code.

CASE (Computer Aided SWE) is a compute environment to help develop project, display milestones and models in a more disciplined way.

Activity Automation includes:

- Graphical Editor: For system model development
- Data Dictionary: To manage design entities
- Graphical UI Builder: For UI construction
- Debugger: To support program fault finding
- Automated Translator: To generate program version

Capability Maturity Model (CMM) used to evaluate the maturity level of SW projects. Imposed to reduce cost & time, increase quality, and produce a better world.

- **Initial:** Characterized as 'ad hoc' and sometimes chaotic. Few processes are defined and success depends on individual effort.
- **Repeatable:** Basic management processes are established. Process discipline is in place to repeat earlier successes.
- **Defined:** SW process for management and SWE is documented, standardized, & integrated into a standard SW process
- **Managed:** Detailed measures of SW process & products are quantitatively understood and controlled.
- **Optimized:** Continuous process improvement enabled by feedback and piloting innovative ideas & technologies.

Except for Lvl 1 each can be broken down into different **key process areas (KPA).** KPA identifies related activities that, when performed collectively, are used to achieve goals.

1. Commitment 2. Ability 3. Activity 4. Measurement 5. Verification

General SWE activities are:
• Specification • Design • Implementation • Validation and Debug • Evolution

Some key issues that affect computer architecture is as follows:

- **Memory Access is Slow** - processor operates at a much faster speed than the CPU resulting in slowdowns as the CPU waits for memory.
- **Events Tend to Cluster** - unrelated items concentrate into one small domain
- **Heat is Bad** - Temp must stay within a certain range, or bad.
- **Resource Replication** - Will not always solve the problem due to data dependencies.

Component-Based SWE is based on systematic reuse where systems are integrated from existing components or **COTS (Commercial off the shelf)** systems. Process Stages: 1. Component Analysis 2. Requirements Modification 3. System Design with Reuse 4. Development & Integration

A **requirements document** is the official statement required to be crafted by system developers. It should include user requirements and system requirements.
Natural language is sometimes used in requirements documents. The **benefit**, is accessibility. Allowing a broader spectrum of users to understand the document. **Cons** include: Lack of Clarity, Requirements confusion, requirements amalgamation, ambiguity.

Non technical requirements required for commercial SW:

1. Cost 2. Legality 3. Ethical 4. Privacy 5. Safety

```
for( int row = 0; row < SZ; row++ ) {
   for( int col = 0; col < SZ; col++ ) {
      for( int k = 0; k < SZ; k++ ) {
         c[ row ][ col ] +=
            a[ row ][ k ] * b[ k ][ col ];
```

Multiply all values in the corresponding row & column and add together into the current location of the destination matrix. For a full multiplication of an NxN matrix **N^3** multiplications. Accesses = 3*n^3 overall O(N^3).

Some real world **non-technical** but **quantifiable parameters** of a major piece of SWE is the schedule, # of people, lines of code written, ease of use, and cost, number of customer interactions, quality of SW products, and added value for SW vendors & customers.

Requirements documents can be created using **form-based specification** (describes the function or entity by listing a description of the inputs/where they come from and where they go). **Mathematical specifications** (notations based on mathematical concepts). **Graphical** (tables, pictures, graphs). **Tabular** (concisely defines condition and action). The point of having these other methods is to more concisely, efficiently, and accurately describe the specifications of a project. Often used to supplement NL specification.

Six Phase PLC

- Requirements Gathering
- Requirements Specification
- Architecture Design
- Detailed Design
- Implementation
- Validation and Verification

ASCII CODES
'0' = 48 '9' = 57
'A' = 65 'Z' = 90
'a' = 97 'z' = 122

Functional requirements define what SW should do. **Non-Functional requirements** define requirements not related to SW itself but necessary to consider. (timing constraints, development process, standards). **Domain requirements** simply consider who should have access to it. **User requirements** describe functional and nonfunctional requirements in a way that they are understandable by a user who lacks detailed technical knowledge.

Sequence diagrams show the sequence of events that take place during the interaction with a system

Non-Functional Classifications:
Product Requirements - (Efficiency, reliability, portability, usability)
Organisational Requirements - (Delivery, Implementation, Standards)
External Requirements - (Ethical, Legislative, Safety, Privacy, Interoperability)

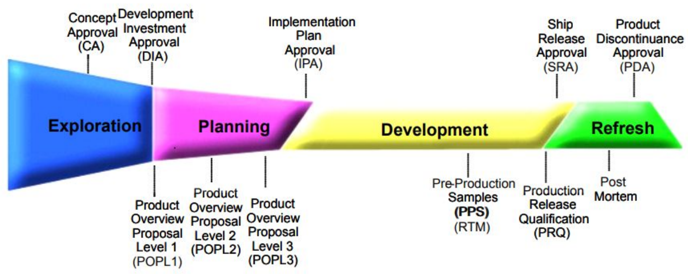
Production SW can be measured with Cost, Schedule, manpower. The total cost is a result of these.

V-Model is a variation of the waterfall model. You work your way down the left side and go up the right. No difference between the two but this model better illustrates the testing process.

Evolutionary Development works with customers to evolve a final system from an initial outline spec. Starts with well understood requirements, adding new features as they are proposed.

Easy way to convert from ASCII or back is to subtract '0' to go from char to int. Use vice versa to go from int to char. Note, you can use printf format specifiers to denote how a variable is printed.

```
for (int row = 0; row < rowSize; row++) {
    for (int column = 0; column < columnSize; column++) {
        cout << matrix[row][column] << " ";
    }
    cout << endl;
}
```


Concept Approval (CA)
Development Investment Approval (DIA)
Implementation Plan Approval (IPA)
Ship Release Approval (SRA)
Product Discontinuance Approval (PDA)
Exploration
Planning
Development
Refresh
Product Overview Proposal Level 1 (POPL1)
Product Overview Proposal Level 2 (POPL2)
Product Overview Proposal Level 3 (POPL3)
Pre-Production Samples (PPS) (RTM)
Production Release Qualification (PRQ)
Post Mortem