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.

Instruction Set Architecture (ISA) is a very low-level piece of software that handles exceptions, interrupts, and HW specific services. A collection of all the operations the HW is capable of performing. It is the boundary between running SW and hosting HW.

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.

Moore's Law is an observation made by Gordon Moore (co-founder of Intel) that the number of transistors per inch squared on an integrated circuit has doubled every year since its invention. Now roughly 18 months.

The key message behind **Amdahl's law** is no matter how many CPUs you have, execution speed will be limited by software that is unable to be parallelized. Sequential software dominates execution speed.

Single accumulator architecture (stores results) was used in the earliest systems (1940s) and is sometimes called John Von Neuman's Computer or John Vincent Antasnoff's

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.

General Purpose Register accumulates ALU results in X number of registers. It allows for register to register operations and is essentially a multi register extension of the Single Accumulator (SA) architecture.

Stack Machine Architecture (SMA) (Uniprocessor) A stack machine has no registers and all operations are performed on memory. However, since memory is slow, stack machines are slow. The only advantage here is bits need not be wasted on memory locations for operations.

Pipelined Architecture (PA) (Uniprocessor) Pipelined machine breaks instructions into sub operations and executes one sub operation of multiple sequential instructions in one step

RAW - Read after write dependence. **WAR** - Write after read dependence.
WAW - Write after write dependence.

Vector Architecture (VA) (Uniprocessor) uses registers that are split into indexes that can load and store blocks of contiguous data. VA registers are essentially arrays. Note, may also have scalar registers.

Multiprocessor (MP) Architecture allows for multiple processors (cores) on a single chip. However, memory can only serve one chip at a time. Meaning other cores must wait until memory is free to get data.

Distributed Memory Architecture is a MP architecture where each processor has local memories. However, if the data is needed from other processors local memory the core must wait until the other cpu returns the data. Yet, if the data is on local memory this is very fast.

Systolic Array (SA) (MULTI) Architecture each processor has its own private memory with a network defined by a systolic (pulse) pathway. Pathways can be bi-directional and can have a 2d or 3d topology. These pathways are high performance networks that send and receive data between CPU's.

Super Scalar Architecture (SSA) (Uniprocessor) replicates some operations in HW. May have multiple ALU's, FPU's but still is a uniprocessor architecture. Arithmetic operations can take place simultaneously.

VLIW machines have the potential to be extremely fast because they can perform multiple operations in a single instruction. However, a key issue with their operation is data dependencies. For example, the result of a Floating Point ADD cannot be used as an operand for another operation in the same instructions. A VLIW machine is still limited by data dependencies. Sequentially executed code cannot be packed onto a single instruction.

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.

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.

Throw-away prototyping (used to understand system requirements, starts with poorly understood requirements with a goal to clarify). It is good for small/medium systems or parts of larger ones and/or short lifetime software. Bad at long range process visibility and often requires special skills for fast prototyping.

Throw-away prototyping can sometimes be thrown away in the end. This is okay as the prototypes help you understand the project requirements better. It is still useful. However, often times the prototype moves into the actual product with many or few modifications.

Spiral Model can be used if there is a strong need to get early software working. Works well with a mature/organized team. It is a non-linear process where phases can be repeated in different stages of completeness. Risks are explicitly assessed and resolved in this process.

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

History
Early computers 1940s, had hard-wired instructions they were easy to use, but inflexible! Single use! Instead "stored program" architecture was needed, resulting in Von Neumann computer design. As a result, programming languages appeared in the 1950s (assembly, Fortran, Cobol). Trend continued with many additional languages into 1960s; luckily fewer languages nowadays! David Parnas argued 1960s for modularity, for information hiding 1970s. The Term "Software Engineering" evolved in the 1960s, popularly documented by Bauer.

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.

During SWE one should keep records such as past code with revision numbers, documentation, testing results, and milestones. The advantages are: allowing you to more easily adapt to changes, prevents the loss of crucial code and keeps costs and maintenance down.

Extreme programming is a special case of incremental development. It is based on the creation and delivery of very small increments of functionality and relies on constant improvement. This requires the user to be involved in the development team. Often, the user is added as a consultant.

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 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 practicing in (some mature) SW industry.

Unified Modeling Language (UML) is a modelling tool used to formally describe a product that consists of SW solutions.

Often times SWE is **behind schedule,** we can help mitigate this by developing plans that include necessary team, chosen SWE model. Defining periods for testing and bug fixing, and setting milestones.

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

Satisficing is the acceptance of an available SW option as satisfactory, though known to be not the complete or perfect solution. This is due to insufficient time to create a complete solution. However, it must solve the current problem domain.

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