**Chapter 1: INTRODUCTION**

**Topic – 1: Definition**

**Introduction**

* It is different from **traditional approach** to develop a software.
* Basic **intuition fails** when it comes to writing **large software**, as it lacks planning.
* We use some **methodologies**, **techniques**, **guidelines** & **tools** in software engineering.

**IEEE Definition**

***“Software engineering is the application of systematic, disciplined, quantifiable approach to development, operation and maintenance of software i.e. the application of engineering software.”***

**Topic – 2: Software Crisis**

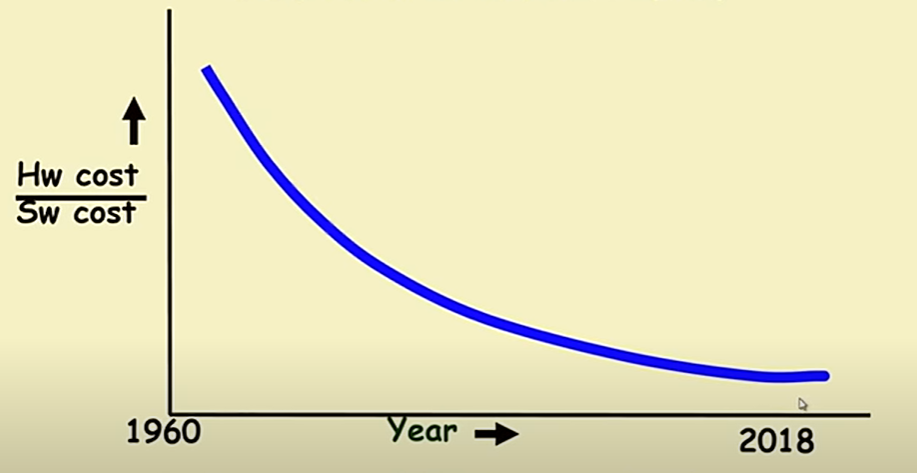
**Introduction**

* Anything that **worsens customer experience** with software can be termed as ***software crisis***.
* Software **fails to meet** user requirement.
* Software is **expensive**.
* **Difficult to modify** or debug the software.
* **Late delivery** of software.
* **Badly optimised** software.

**Factors Affecting Software Crisis**

* Large software.
* Poor project management.
* Lack of quality training to software engineers.
* Skill shortage.

**Topic – 3: Hardware to Software Relative Cost**



* Notice how software at starting used to be **very inexpensive**, but later became almost as **expensive** as the hardware.
* Also, the price of hardware has been dropped.

**Topic – 4: Fundamental Questions**

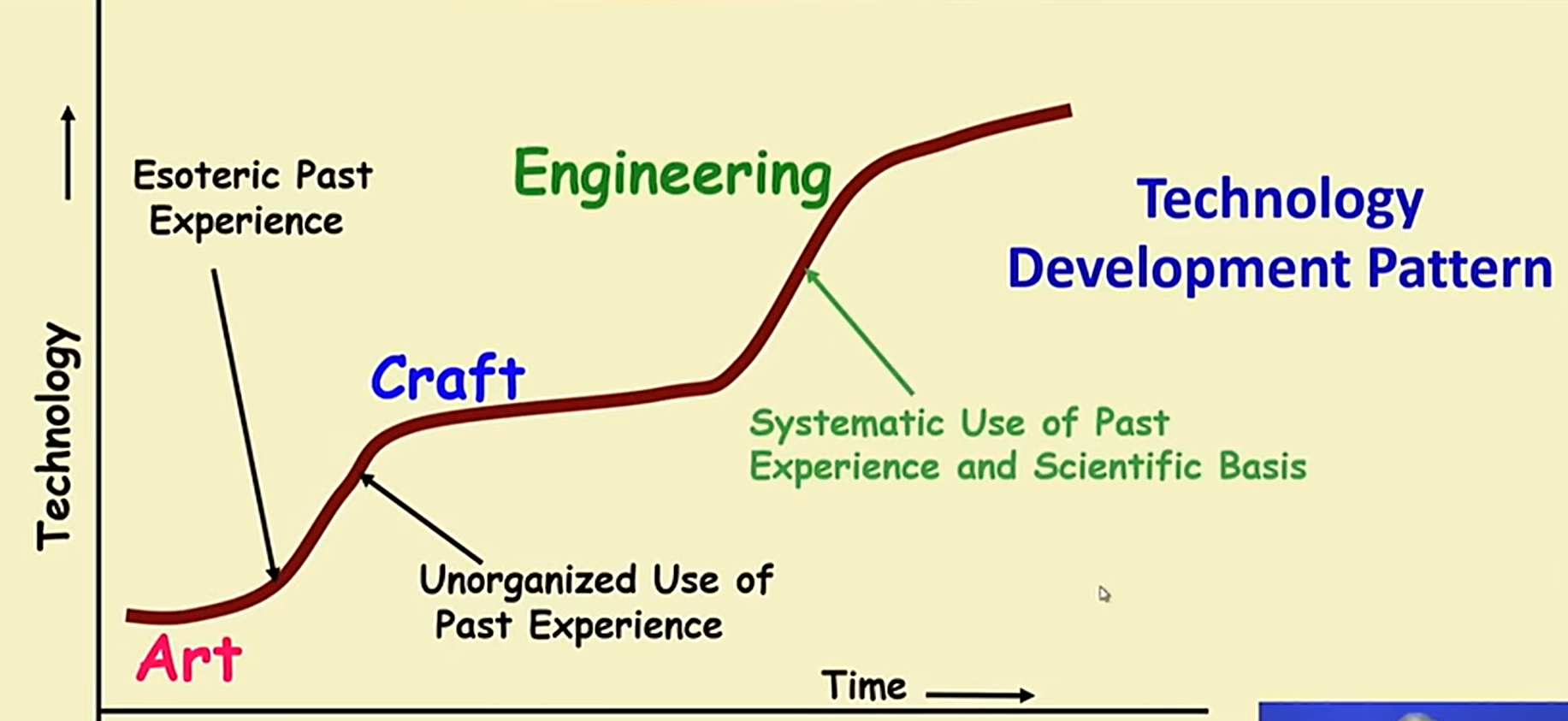
**Why preference to software?**

* **Easier and faster** to develop than making everything on hardware directly.
* Consumes **no space** or **weight**.

**Why software is difficult to modify if its complex?**

* If we do make changes, the chances are that the complexity will **only increase**.
* Modifying even the smallest changes, makes changes to the **very underlying design**.

**Programming is art or engineering?**



* Programming used to be an **art** when people made software in an **unorganized manner**, but soon the past experiences turned it into an **organized way** to engineer software.
* Art is more **secretive** while engineering is more **openly sharable**.
* At early inception of programming, there were good & bad programmers. Good programmers wrote better code, which were later identified as **techniques** & **systematic approaches** were formed.

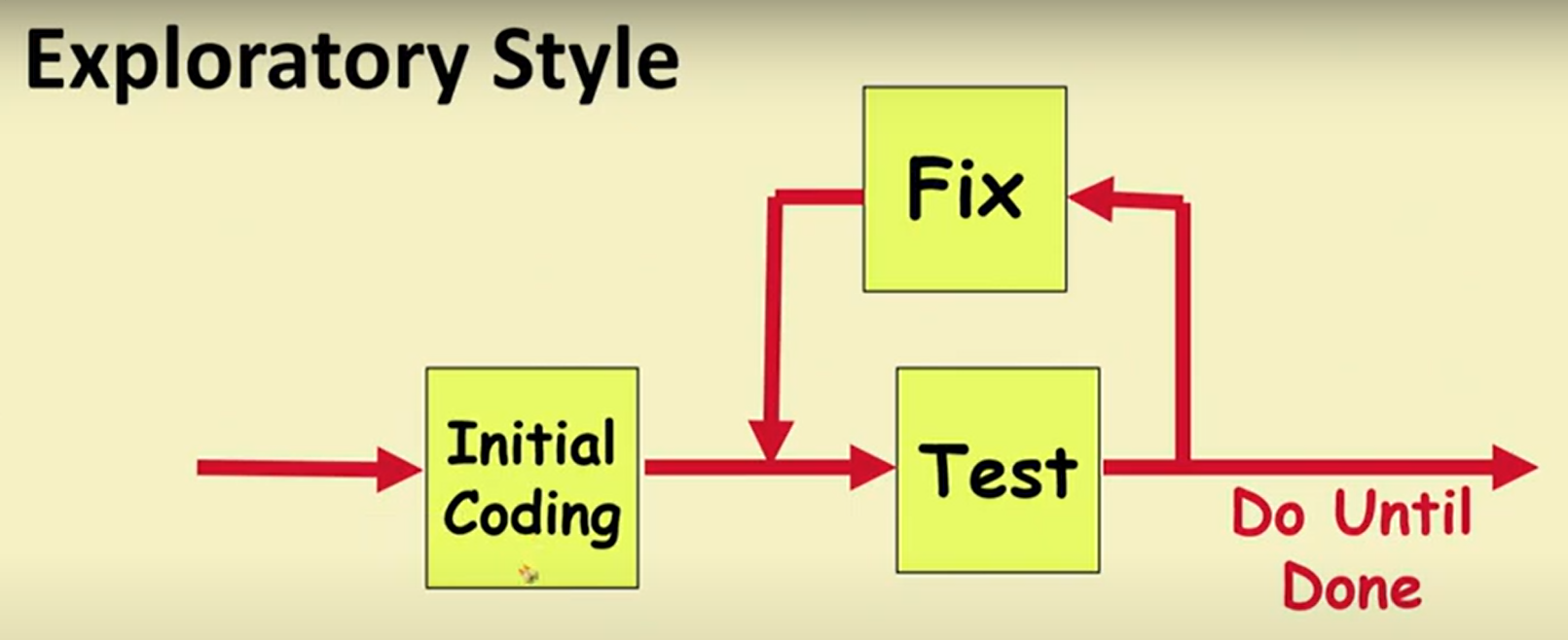
**Topic – 5: Standish Group Report**

* Only **28%** projects are **successful**.
* And **23%** projects are **cancelled**.
* And **49%** projects are **delayed & their price increases**.

**Topic – 6: Exploratory Style**

**Introduction**

* ***Exploratory style development*** refers to style of programming **without** planning.



**Why SE Over Traditional Programming?**

* Human cognitive functions allow to store data only in **short-term memory**.
* When analysing or creating small codes, small number of variables **can be** remembered by the programmer. But **not** so in large programs.

**Topic – 7: Techniques to Overcome Complexity**

* **Abstraction:** Also called model building.
* **Decomposition:** Decomposing problems in smaller parts.

**Topic – 8: Techniques We Learn in SE**

* Specification
* Design
* User interface development
* Testing
* Project management
* Maintenance

**Topic – 9: Software Projects**

**Introduction**

* Comes **in between** predictable & unpredictable.

**Types Of Software Projects**

* **Products/packaged software:** Generic software, **readily available** for purchase.
* **Services software:** Custom software, packaged software **tailored** to meet requirements of a **specific** organization.

**Types Of Packaged Software**

* **Horizontal market software:** Meets need of many companies, like a Windows & MS Office components etc.
* **Vertical market software:** Targets specific type of industry, like banking systems etc.

**Note!**

🡪 Software business is in several **trillions** of **US dollars**.

🡪 Services segment is growing **faster** than products segment.

**Topic – 10: Service Based Software Industries**

**Services Provided**

* Software customization
* Software maintenance
* Software testing
* Contract programmers

**Facts!**

🡪 Reason behind large **Indian** service-based software industry is **aversion to risk** of running product-based companies.

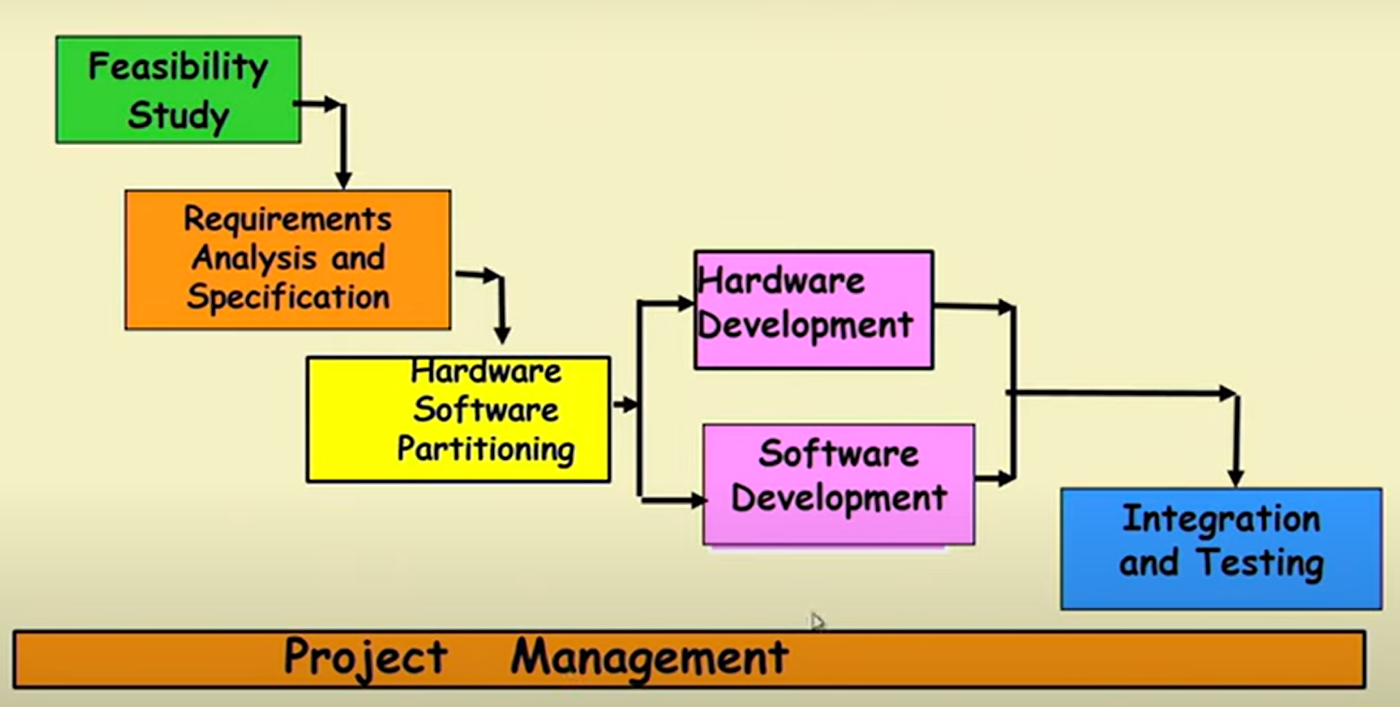
🡪 Programming languages like **FORTRAN**, **COBOL**, **PASCAL**, **BASIC**, **ALGOL** etc were used earlier without reusability feature & were all **command based**.

**Topic – 11: Computer Systems Engineering**

**Introduction**

* Many software are developed for **specific** hardware.
* This process is known as ***system engineering***.
* Here, software and hardware are developed **parallelly** & thus software **can’t** be tested on that particular hardware as it is too under development.
* So, these software are tested on a **hardware simulator**.
* The software is **again tested** on actual hardware after it is developed.

**Project Management**



**Topic – 12: Emergence Of Software Engineering**

**Introduction**

* Careful assessment of ***control flow structure*** started to be done.
* **Control flow:** Sequence of instructions in which they are executed.
* **Flow charts** are used to represent **control flow diagram**.
* It makes easier to trace the path of program.

**Complexity Of Programs**

* Earlier programming languages like **COBOL**, **PASCAL**, **FORTRAN** etc used **GO TO** statements heavily.
* Even **assembly** programmers used it heavily.
* **Dijkstra** once commented on **GO TO** statement saying they are harmful, which made programmers unhappy.

**Structured Programming**

* Writing program without using **GO TO** statements.
* It uses **sequence**, **selection** & **iteration** type of constructs.

**History Of Programming**

* **1950s –** **Assembly programs** of **few 100 lines** were written.
* **Early 1960s –** **High-level languages** like FORTRAN, ALGOL and COBOL were introduced & programs used to be of **few 1000 lines** of code.
* **Late 1960s –** **Control flow-based** designs were developed.
* **Early 1970s –** **Data structure-oriented** designs.
* **Late 1970s –** **Data flow-oriented** designs.
* **1980s –** **Object-oriented** designs.

**Topic – 13: Data Structure-Oriented Programming**

**Introduction**

* Came into focus at **1970s** when the **program size increased** too much.
* It focused on **improving** data structures.

**JSP Method**

* Stands for ***Jackson’s structured programming*** method.
* Focused on representing data structures in **diagrammatic form** first & then **convert** it into code.
* Used **notations** for sequence, selection & iteration.

**Honorary Mention**

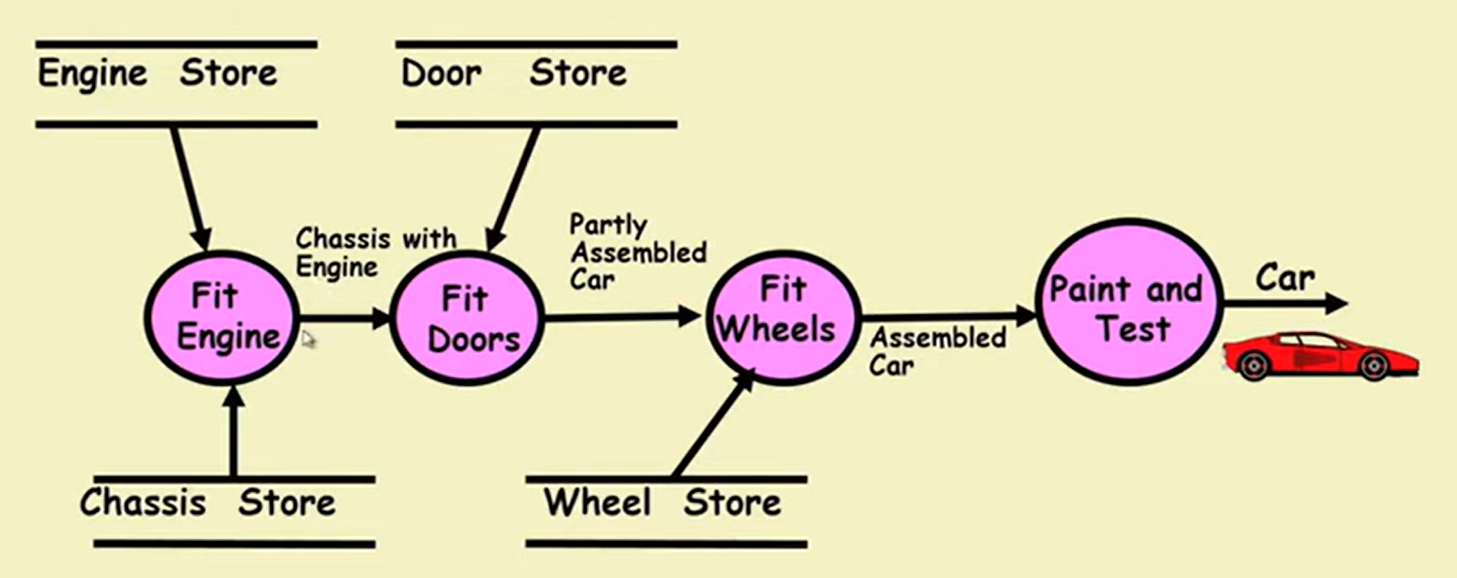
* ***Warnier-Orr Methodology***

**Topic – 14: Data Flow-Oriented Design**

**Introduction**

* It is about tracking the **flow of data** like **inputs** throughout the program.
* **Processing stations:** Parts of program where processing is done on input.
* Also known as ***processing functions***.
* **Data flow** can be used in any system, **not** only software system.

**Car Assembly Example**



* **Chassis:** Base frame of a motor vehicle.

**Topic – 15: Evolution Of Other Techniques**

* Life cycle models
* Specification techniques
* Project management techniques
* Testing techniques
* Debugging techniques
* Quality assurance techniques
* Metrics
* CASE tools etc.