When Defining Software: Look at 3 important things. **NB**

1. Instructions: What must your software do.

2. Structures: Data structures makes up your software. (Maybe: What is needed to run the software)

3. Information: Script of Information, is the manuals, source code… How to use the software

Software is Logical, not physical. It does not deteriorate.

**Heterogeneity**: Systems need to work/communicate with each other (**not in slide, in text book**). Eg Windows 10 not being able to use software from Vista. Therefore there is a deterioration in the software.

Business and Social Change: As society changes and evolves, software needs to change to meet that evolution. What you want to look at is the **need** of the consumer.

Security and Trust:

Change: nothing ever goes to plan.

Do not need the changing nature… in text book. (between above and below point in Txt book.)

**Definition** of **Software Engineering** will be asked in every test. **NB**

**-**The **application** of a **systematic, disciplined, quantifiable approach** to the **development, operation and maintenance** of **software;** that is the **application of engineering to software**

**The steps of a software engineer**

1. Understand the problem
2. Plan the solution
3. Carry out the plan
4. Examine the result for accuracy

**Core things when writing Software (NB Given a case study, list these things and how would you apply it) (In text book)**

1. Communication:
2. Identify who your stakeholders are and how to efficiently communicate with them with what they want.
3. Listen, as you must extract what they really need.
4. Negotiation is not about being a winner, but reaching a resolution to solve a problem. If you cannot agree, move on.
5. Planning: Understanding the scope of the project, realistic, involve the stakeholders in this. UML
6. Modelling:
7. Requirements modeling looks at the requirements of the system and what it must do, and then model a X for that system.
8. Design modeling: I am going to plan how the system is going to be built
9. A prototype, accepts change, a part of the system… there is more need to know in text book.
10. Construction:
11. Testing Code… No system is unbreakable.
12. Building the code
13. Validating the code
14. Deployment

**Core Principles**

1. Be **Agile:** be able/ready to change
2. Focus on **Quality** at every step: this is the biggest issue when change happens.
3. Be ready to **Adapt**: Issues or problems might come about later that need to be addressed. Eg Security protocols.
4. Build an **Effective** team:Everyone has their strengths and weaknesses
5. Establish mechanism for **Communication** and **Coordination**: Be able to take criticism, and establish guidelines on how it should take place.
6. Manage **Change:**
7. Assess **Risk:** With change brings risk. You must be able to manage these risks.
8. Create work products that provide **Value** for others: Don’t add unnecessary features.