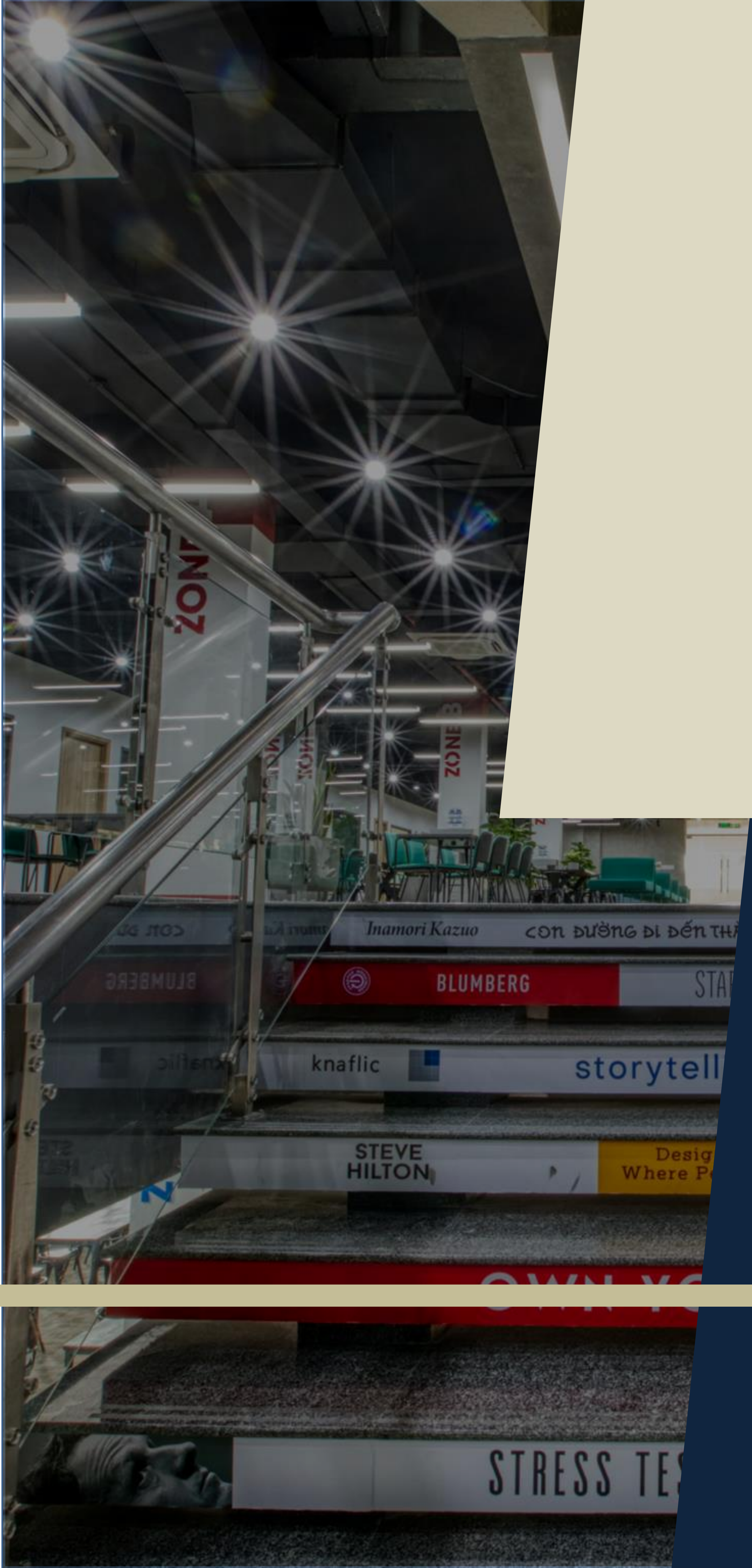


TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Software Engineering Course’s Code: CSE 305

Course Description

- ➢ Software Engineering course is aimed at helping you build up an understanding of how to develop a software system from scratch by guiding them through the development process.

- ➢You will learn different stages of Software Development Life Cycle (SDLC).

- ➢The course will introduce students to the different software development process models, software requirements engineering process, systems analysis and design as a problem-solving activity, and building of software.

Course Objectives

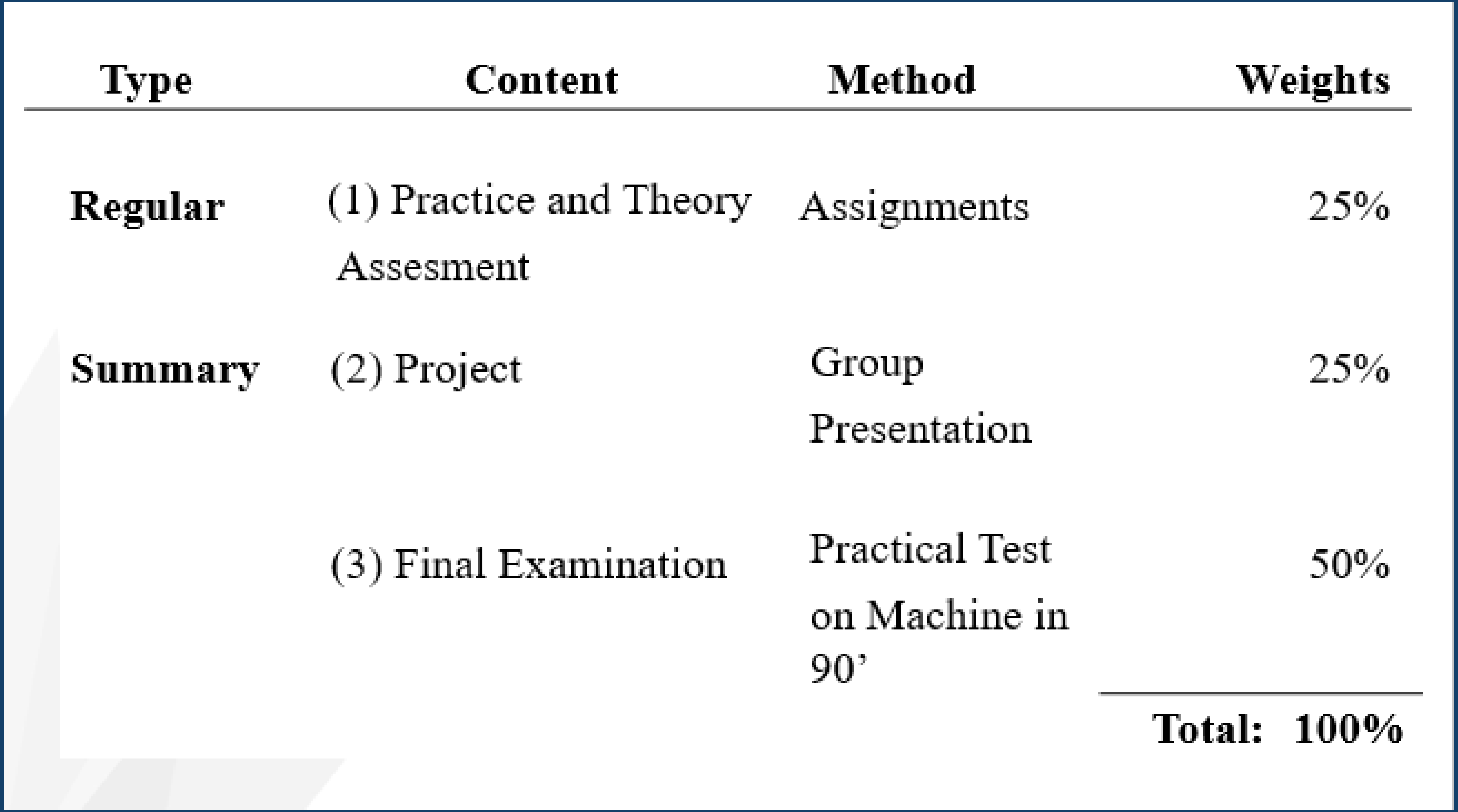
- After finishing the course, you will achieve the following goals:
  - ❑ To acquire a thorough understanding of Software Engineering.
  - ❑ To use the techniques, skills, processes and modern engineering tools necessary for software engineering practice.
  - ❑ To apply software engineering perspective through software design and construction, requirements analysis, verification, and validation.
  - ❑ To work as an individual and as part of a team to develop and deliver quality software.

Books and Teaching Materials

[1]. Roger S. Pressman, Bruce R. Maxim (2015). Software Engineering : A Practitioner's Approach. (Software Engineering). McGraw-Hill Education: New York, NY.

[2]. Ian Sommervill (2016). Software Engineering. (Software Engineering). Pearson: Boston

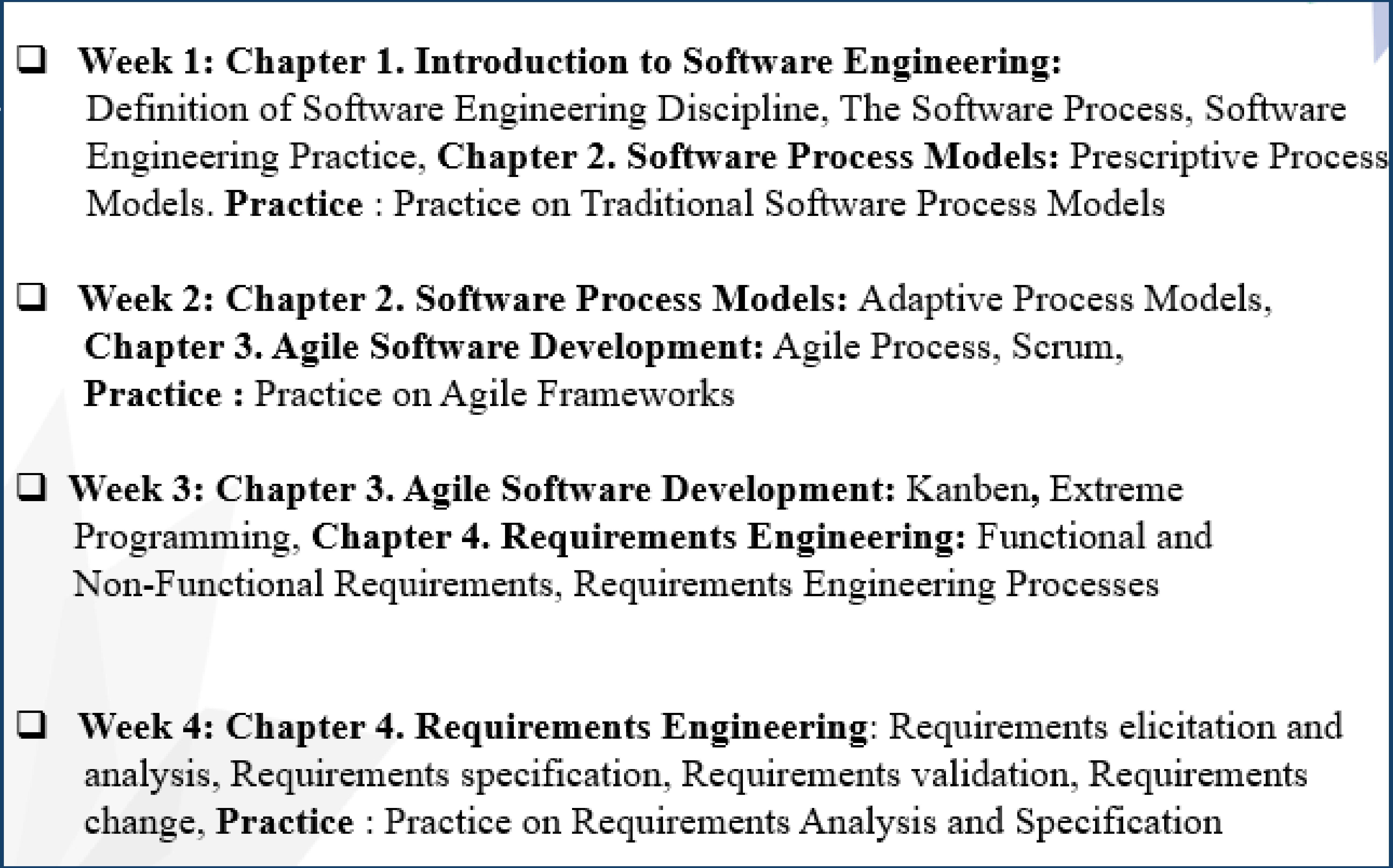
EVALUATION



Type	Content	Method	Weights
Regular	(1) Practice and Theory Assesment	Assignments	25%
Summary	(2) Project	Group Presentation	25%
	(3) Final Examination	Practical Test on Machine in 90’	50%
		Total:	**100%**

EIU

SCHEDULE



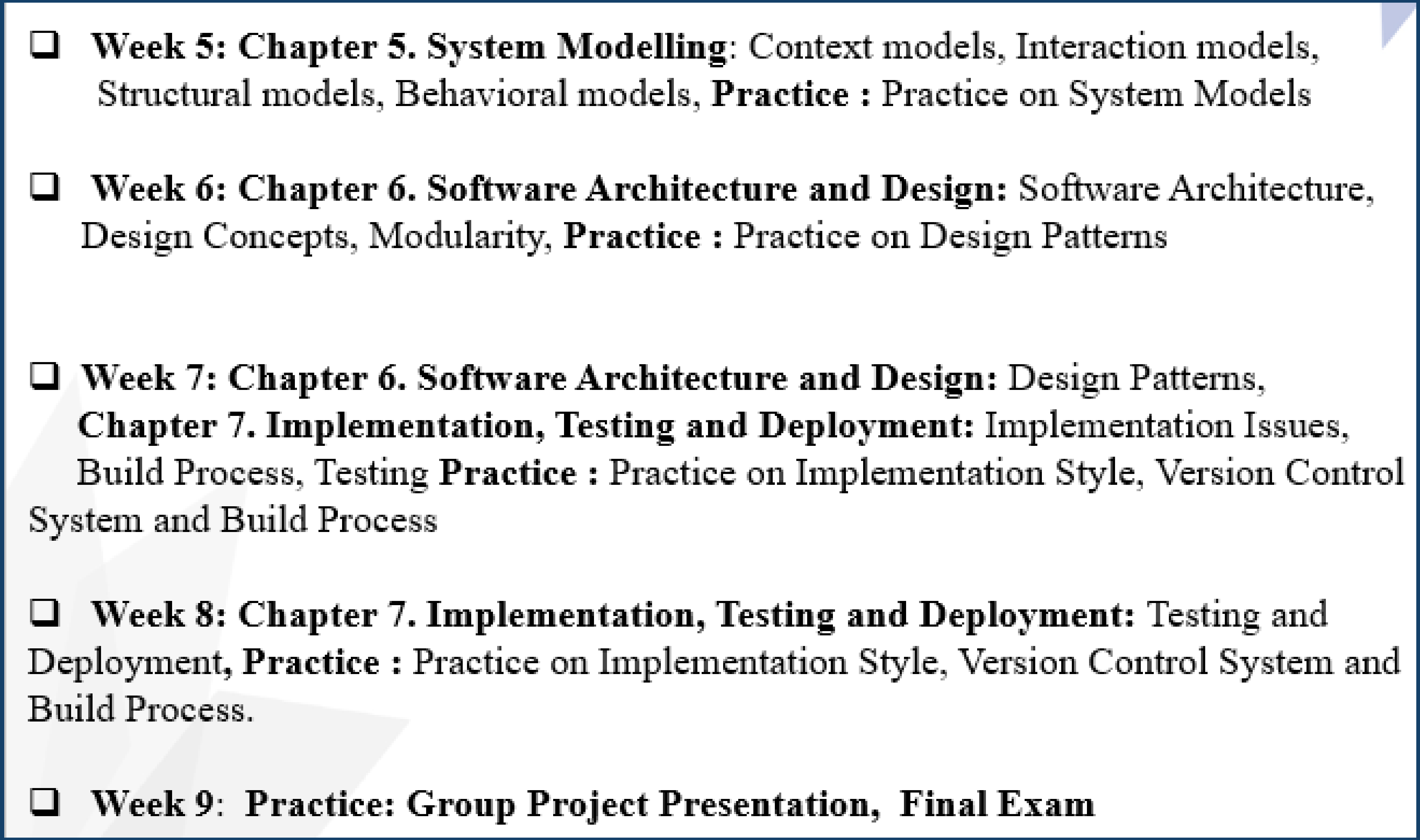
- **Week 1: Chapter 1. Introduction to Software Engineering:** Definition of Software Engineering Discipline, The Software Process, Software Engineering Practice, **Chapter 2. Software Process Models:** Prescriptive Process Models. **Practice** : Practice on Traditional Software Process Models

- **Week 2: Chapter 2. Software Process Models:** Adaptive Process Models, **Chapter 3. Agile Software Development:** Agile Process, Scrum, **Practice :** Practice on Agile Frameworks

- **Week 3: Chapter 3. Agile Software Development:** Kanben, Extreme Programming, **Chapter 4. Requirements Engineering:** Functional and Non-Functional Requirements, Requirements Engineering Processes

- **Week 4: Chapter 4. Requirements Engineering**: Requirements elicitation and analysis, Requirements specification, Requirements validation, Requirements change, **Practice** : Practice on Requirements Analysis and Specification

SCHEDULE



- ❑ **Week 5: Chapter 5. System Modelling**: Context models, Interaction models, Structural models, Behavioral models, **Practice :** Practice on System Models

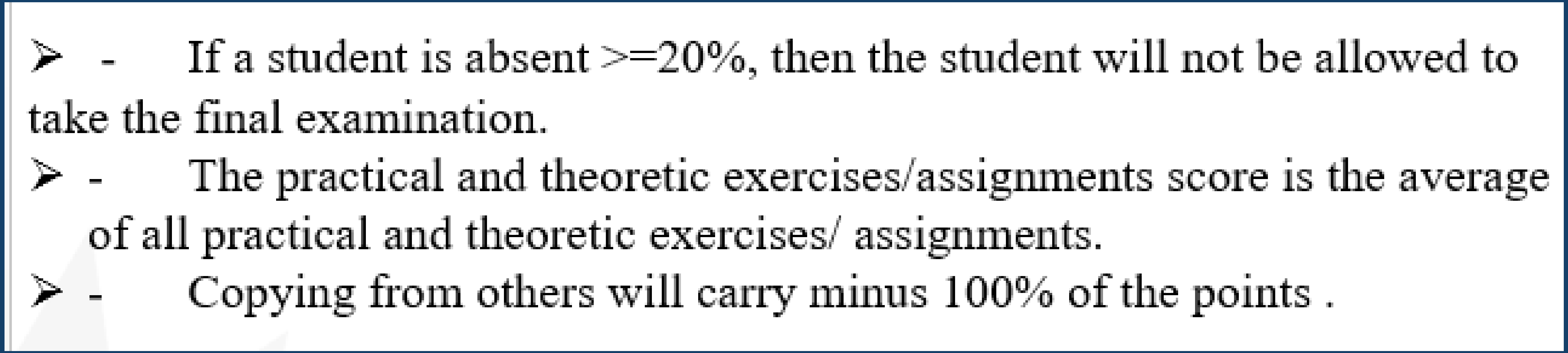
- ❑ **Week 6: Chapter 6. Software Architecture and Design:** Software Architecture, Design Concepts, Modularity, **Practice :** Practice on Design Patterns

- ❑ **Week 7: Chapter 6. Software Architecture and Design:** Design Patterns, **Chapter 7. Implementation, Testing and Deployment:** Implementation Issues, Build Process, Testing **Practice :** Practice on Implementation Style, Version Control System and Build Process

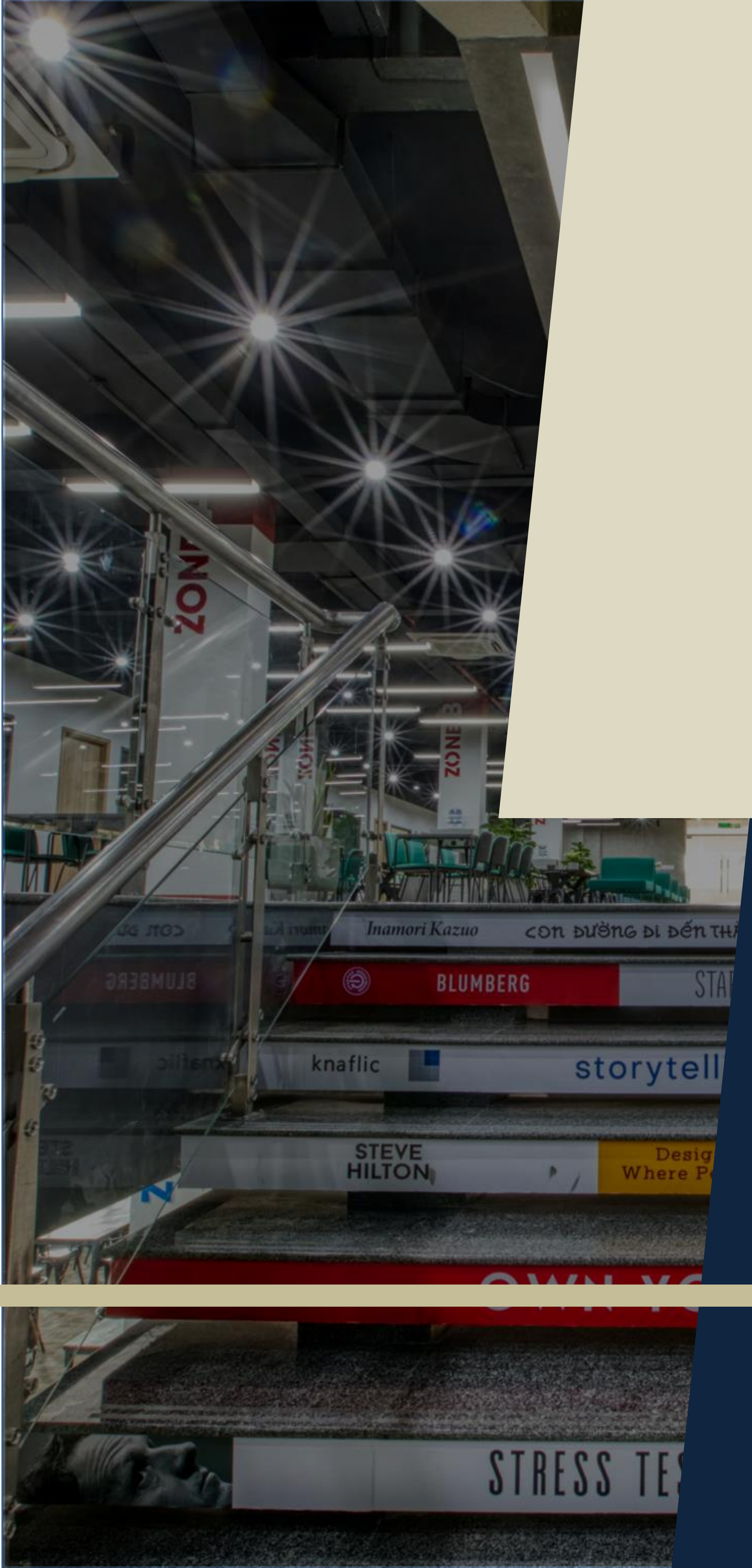
- ❑ **Week 8: Chapter 7. Implementation, Testing and Deployment:** Testing and Deployment**, Practice :** Practice on Implementation Style, Version Control System and Build Process.

- ❑ **Week 9**: **Practice: Group Project Presentation, Final Exam**

COURSE POLICIES



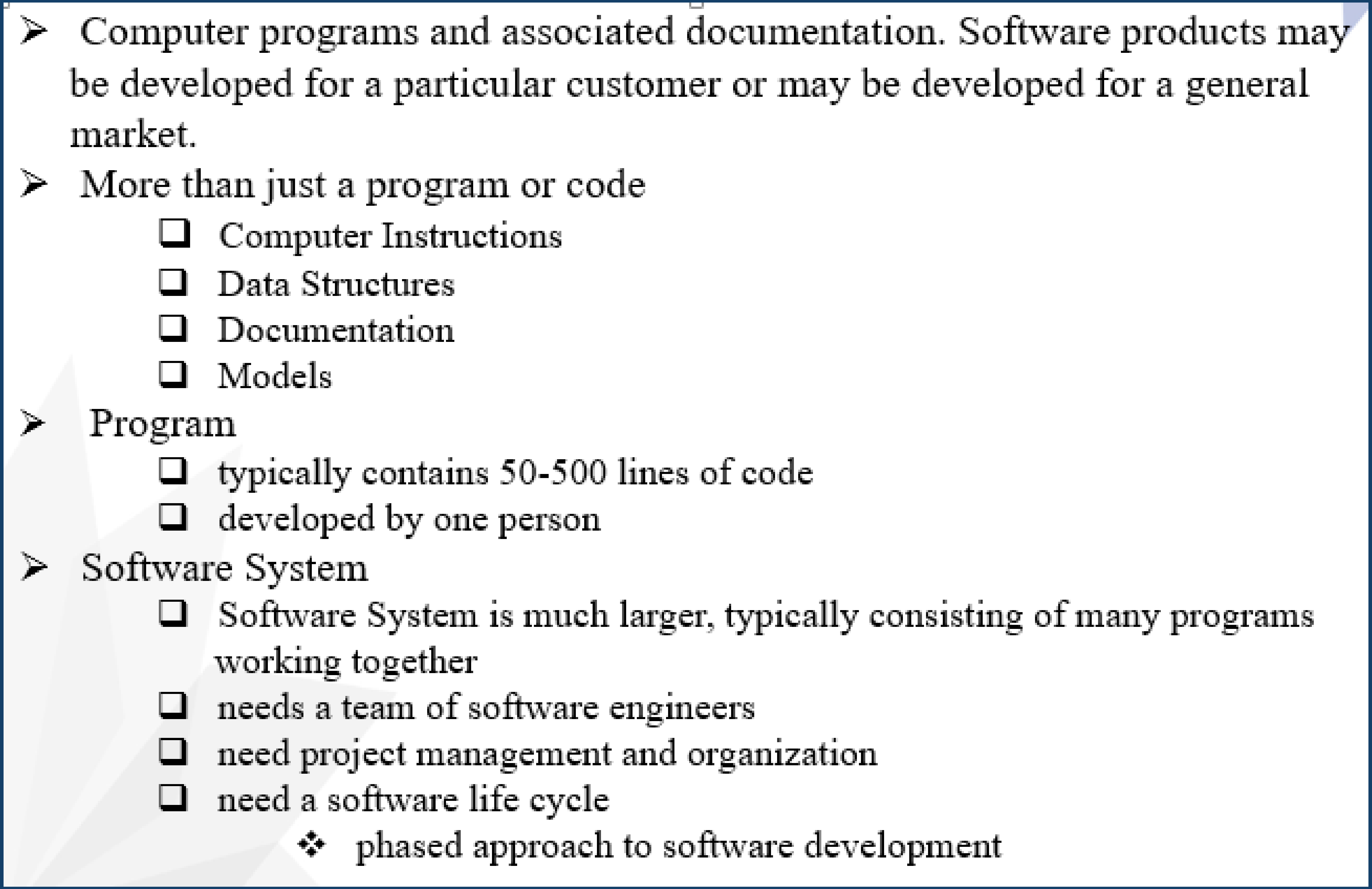
- If a student is absent >=20%, then the student will not be allowed to take the final examination.
- The practical and theoretic exercises/assignments score is the average of all practical and theoretic exercises/ assignments.
- Copying from others will carry minus 100% of the points .



Chapter 1. Introduction to Software Engineering

1.1. Definition of Software Engineering Discipline
1.2. The Software Process
1.3. Software Engineering Practice

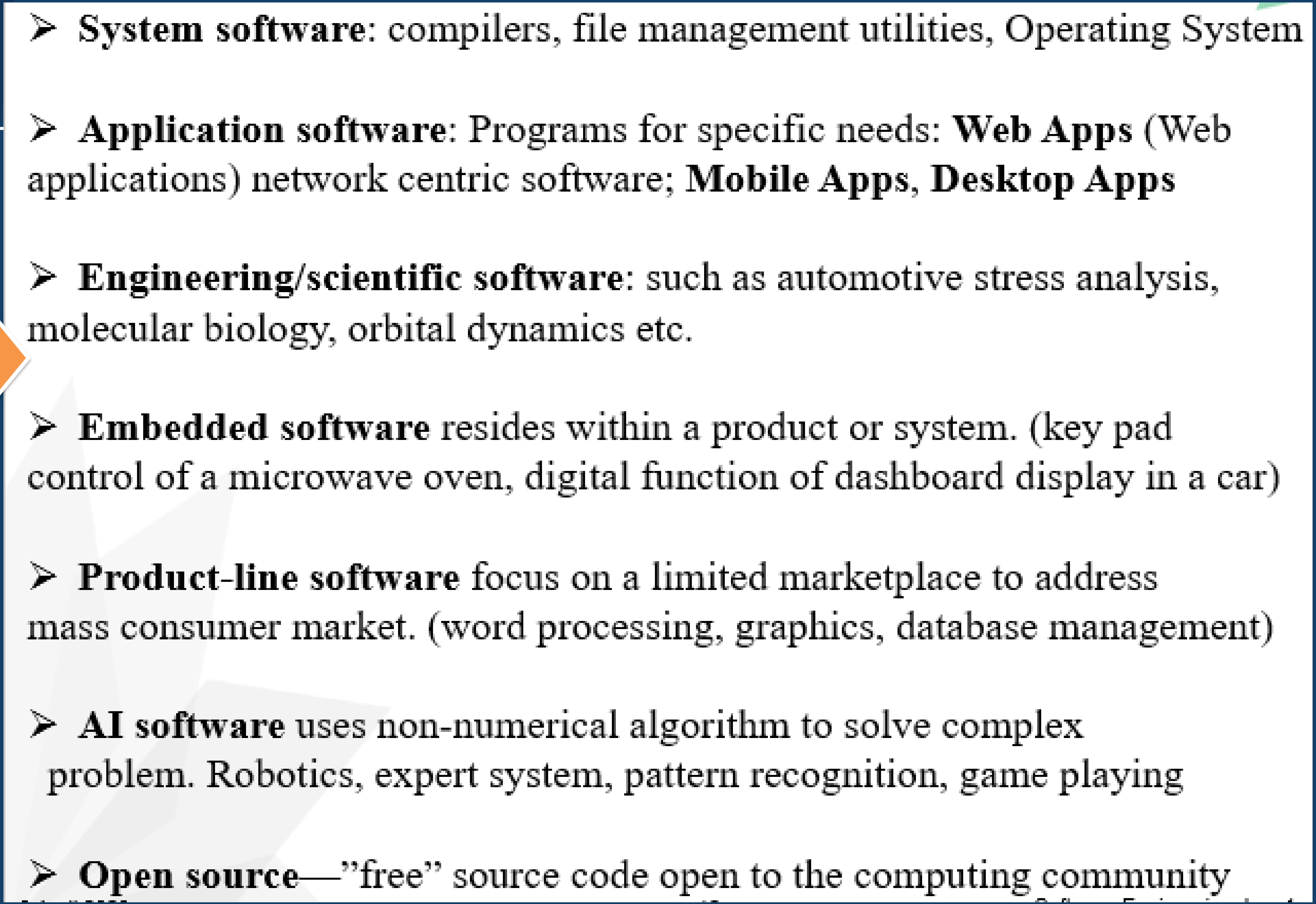
WHAT IS SOFTWARE?



- ➢ Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
- ➢ More than just a program or code
  - ❑ Computer Instructions
  - ❑ Data Structures
  - ❑ Documentation
  - ❑ Models
- ➢ Program
  - ❑ typically contains 50-500 lines of code
  - ❑ developed by one person
- ➢ Software System
  - ❑ Software System is much larger, typically consisting of many programs working together
  - ❑ needs a team of software engineers
  - ❑ need project management and organization
  - ❑ need a software life cycle
    - ❖ phased approach to software development

WHAT IS SOFTWARE?

- *Types of Software*



➢ **System software**: compilers, file management utilities, Operating System

➢ **Application software**: Programs for specific needs: **Web Apps** (Web applications) network centric software; **Mobile Apps**, **Desktop Apps**

➢ **Engineering/scientific software**: such as automotive stress analysis, molecular biology, orbital dynamics etc.

➢ **Embedded software** resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

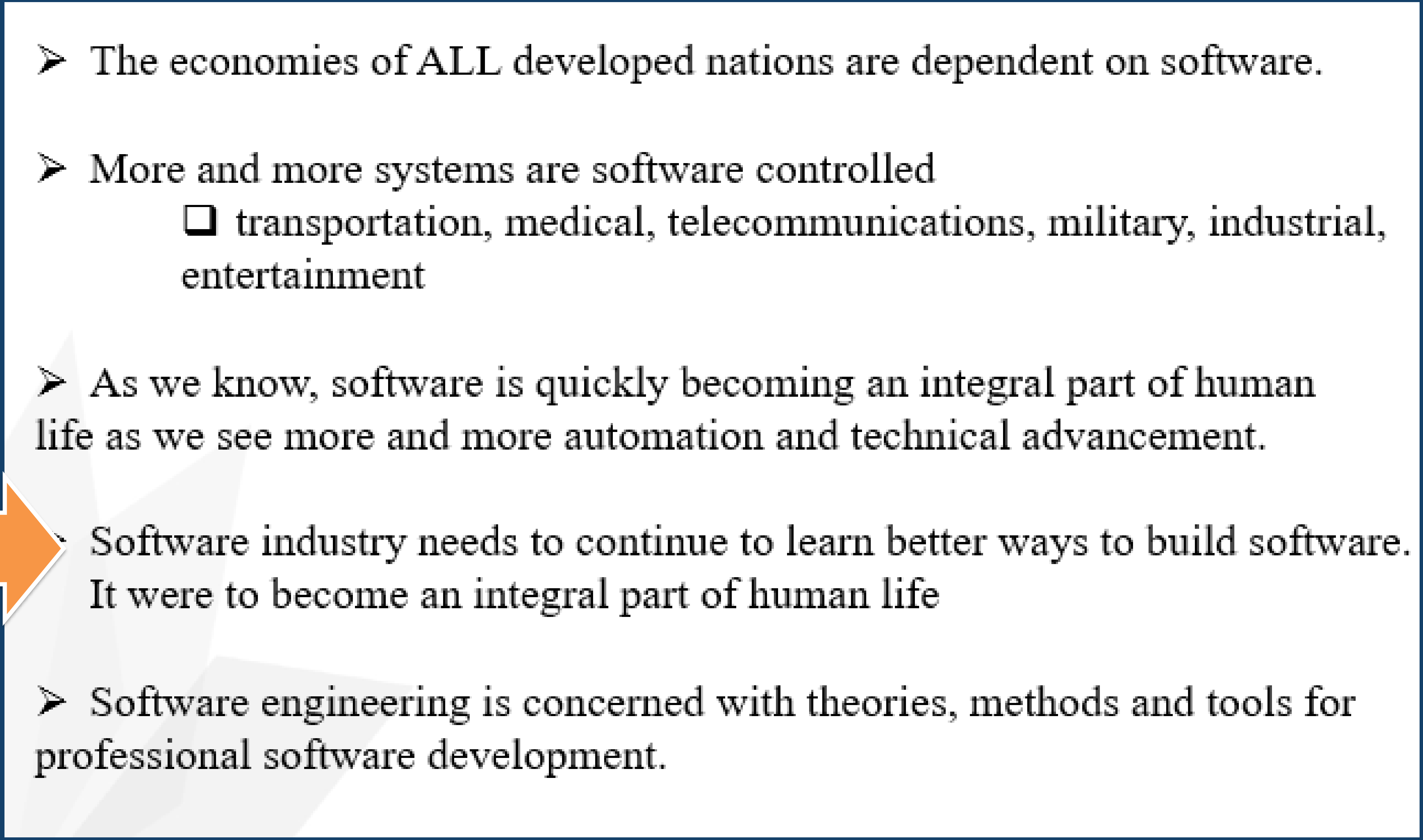
➢ **Product-line software** focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)

➢ **AI software** uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition, game playing

➢ **Open source**—"free" source code open to the computing community

WHAT IS SOFTWARE?

- *Types of Software*
- *Important of Software*



➢ The economies of ALL developed nations are dependent on software.

➢ More and more systems are software controlled

❑ transportation, medical, telecommunications, military, industrial, entertainment

➢ As we know, software is quickly becoming an integral part of human life as we see more and more automation and technical advancement.

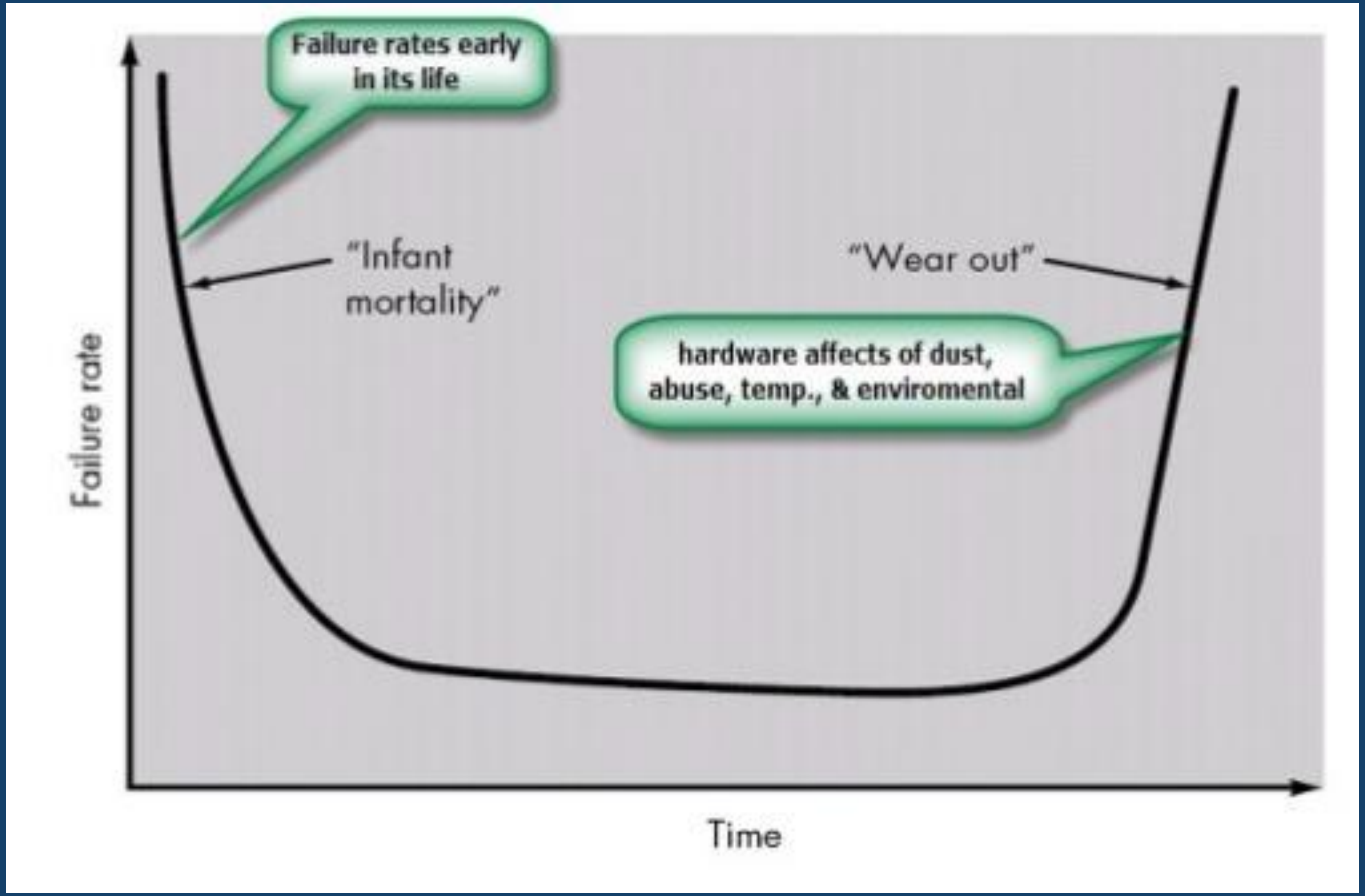
➢ Software industry needs to continue to learn better ways to build software. It were to become an integral part of human life

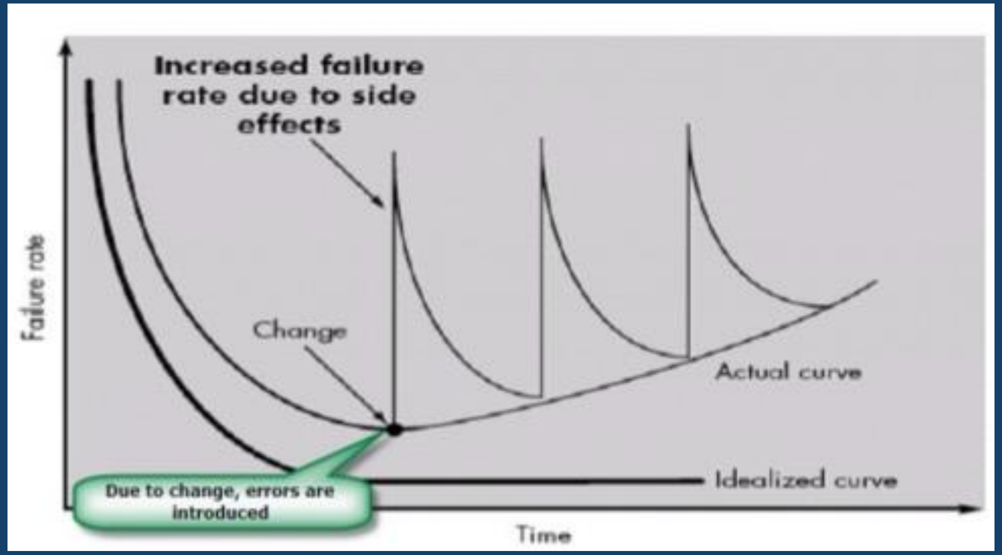
➢ Software engineering is concerned with theories, methods and tools for professional software development.

WHAT IS SOFTWARE?

- *Types of Software*
- *Important of Software*
- *Features of software*

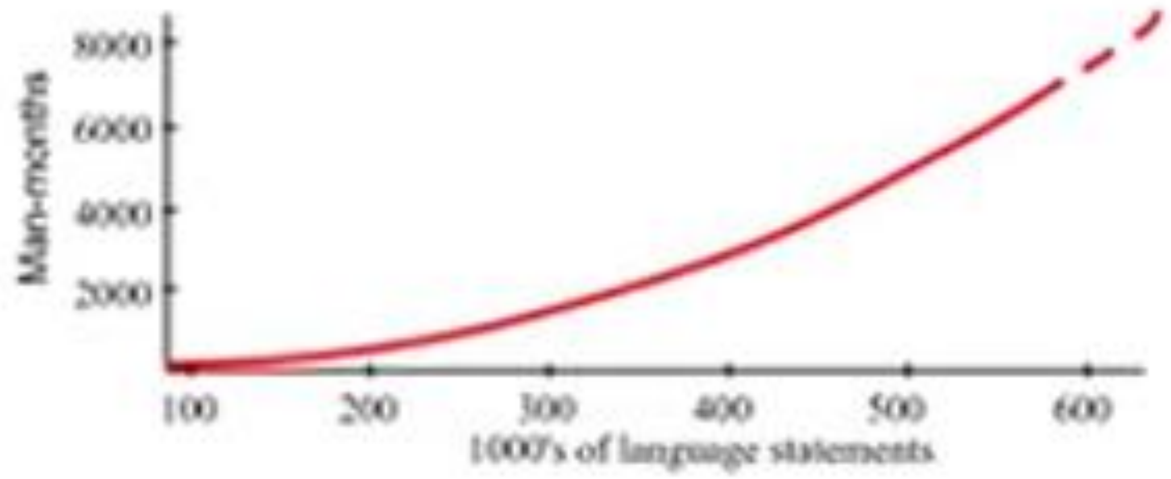
- Software is developed or engineered
  - Not manufactured in the classical sense
- Software doesn’t wear out but it deteriorates (due to change).
- Software can be custom-built, and/or component based software





SOFTWARE IS COMPLEX TO DEVELOP ?

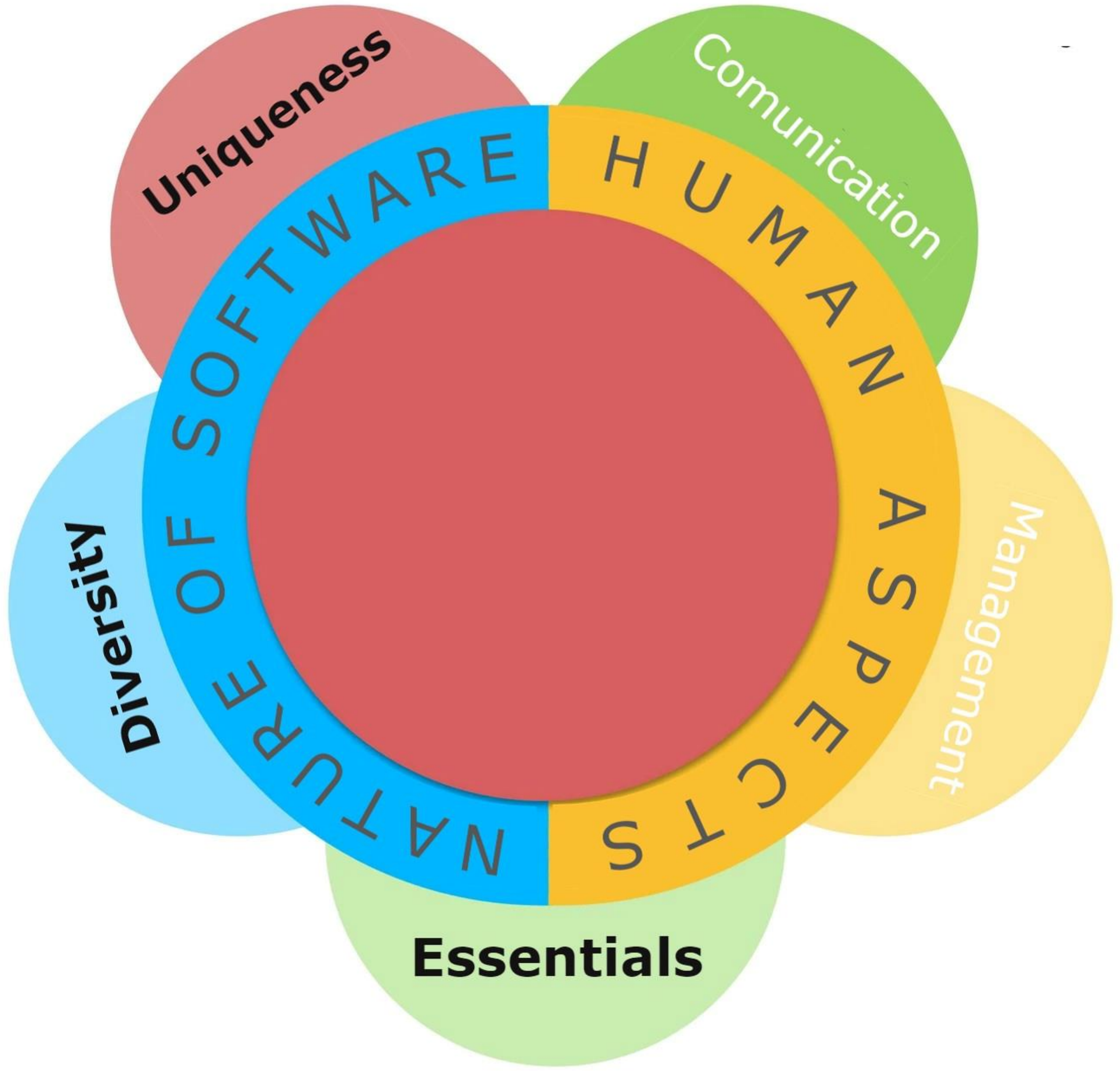
- Rome: Total War: 3 MLOC
- Boeing 787: 14 MLOC
- F-35 Fighter Jet: 24 MLOC
- Windows 7: ~40MLOC
- Windows 10: ~50 MLOC
- Facebook: 61 MLOC
- Mac OS X: ~90 MLOC
- Luxury passenger car: 100 MLOC



Development effort is not linear with respect to amount of code!

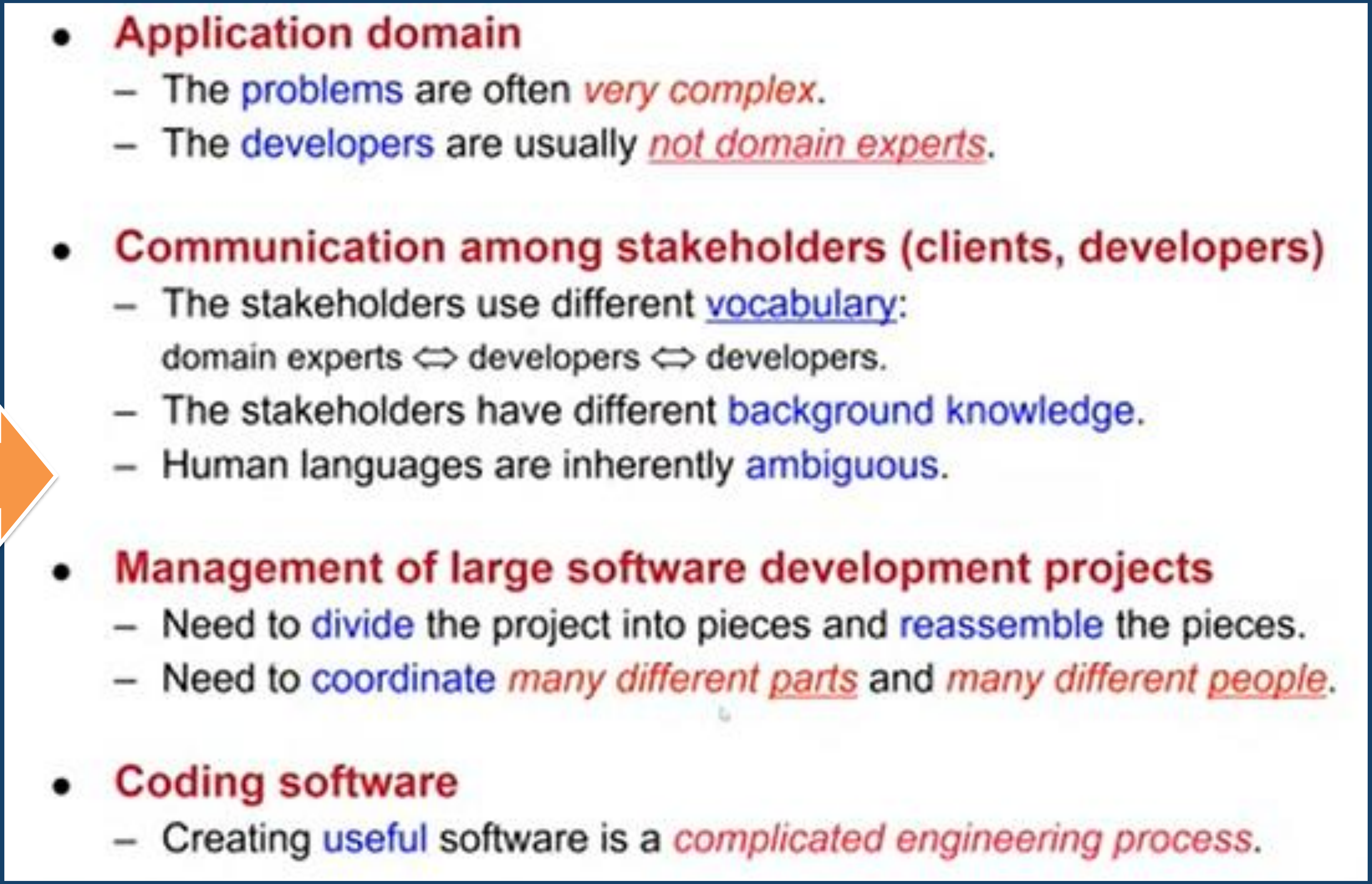
➢ LOC : Line of Code
➢ MLOC: Million Lines of Code

SOFTWARE IS COMPLEX TO DEVELOP



SOFTWARE IS COMPLEX TO DEVELOP

- *Where does it come from ?*



- Application domain
  - The problems are often *very complex*.
  - The developers are usually *not domain experts*.
- Communication among stakeholders (clients, developers)
  - The stakeholders use different vocabulary:
    domain experts ⇔ developers ⇔ developers.
  - The stakeholders have different background knowledge.
  - Human languages are inherently ambiguous.
- Management of large software development projects
  - Need to divide the project into pieces and reassemble the pieces.
  - Need to coordinate *many different parts* and *many different people*.
- Coding software
  - Creating useful software is a *complicated engineering process*.

EIU

SOFTWARE IS COMPLEX TO DEVELOP

- ***Where does it come from ?***
- ***Software complexity leads to "Software Crisis"***

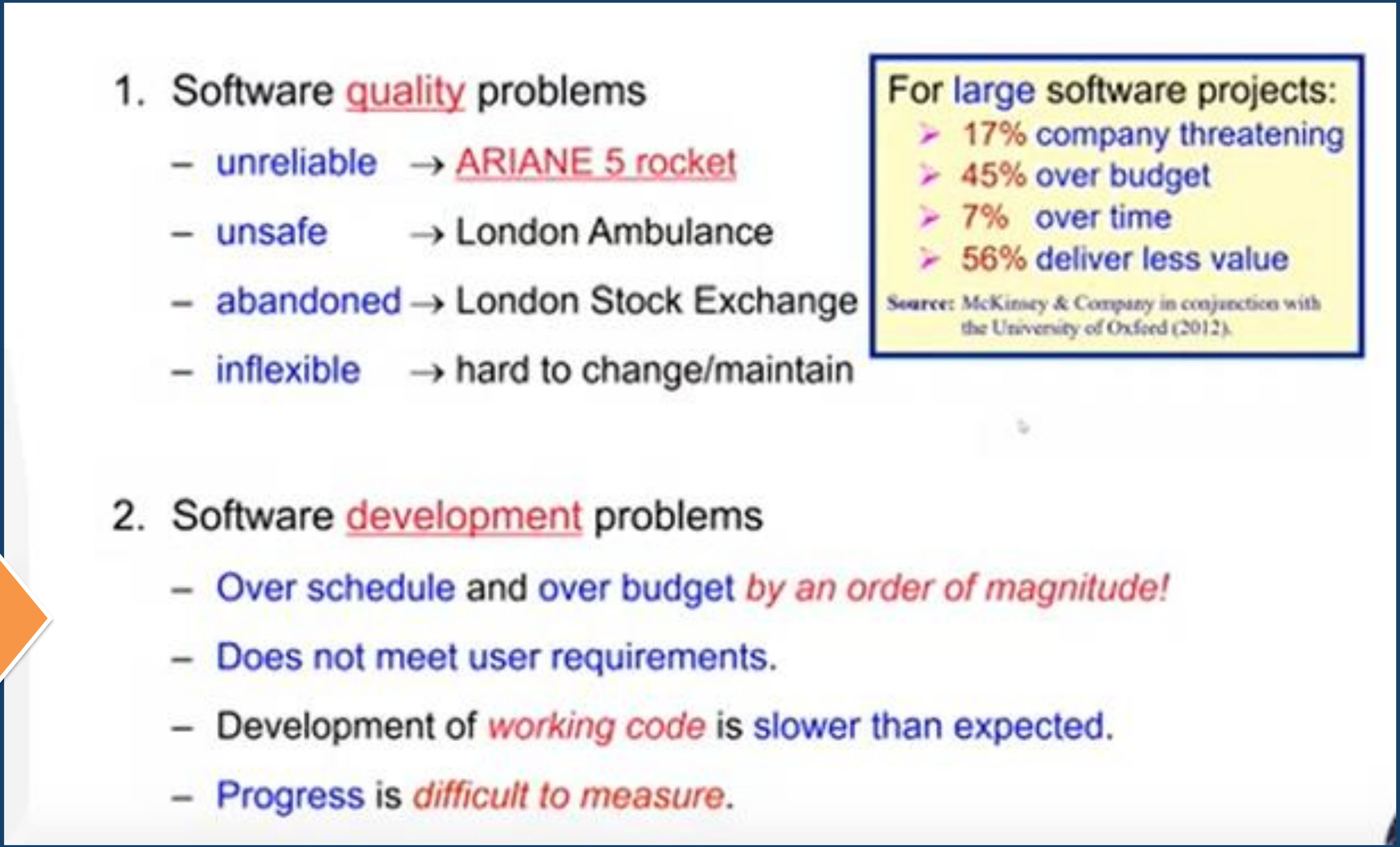
➢ Software Crisis
  - ❑ Software development projects were delivered late
  - ❑ Software were full of errors
  - ❑ Software did not satisfy requirements
  - ❑ Software was difficult to maintain
  - ❑ Software sizes and complexity have grown rapidly over time

➢ Some of these issues still exist today, although to a much lesser extent due to the consistent application of engineering principles to the software development process

➢ The solution to the "Software Crisis" involved transforming unorganized coding efforts into an established engineering discipline.

SOFTWARE IS COMPLEX TO DEVELOP

- *Where does it come from ?*
- *Software complexity leads to "Software Crisis"*



1. Software quality problems
   - unreliable → ARIANE 5 rocket
   - unsafe → London Ambulance
   - abandoned → London Stock Exchange
   - inflexible → hard to change/maintain

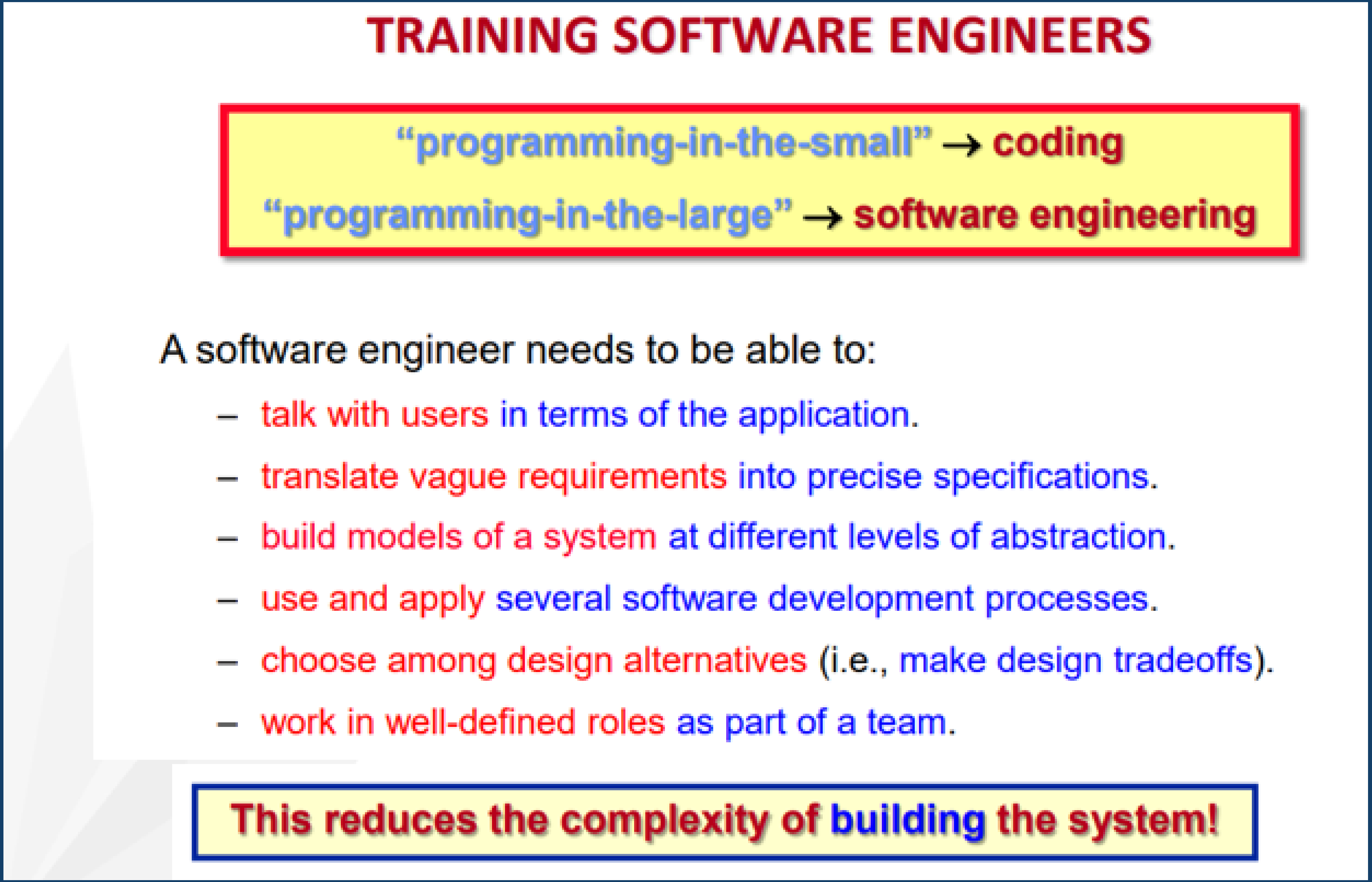
For large software projects:
- 17% company threatening
- 45% over budget
- 7% over time
- 56% deliver less value

Source: McKinsey & Company in conjunction with the University of Oxford (2012).

2. Software development problems
   - Over schedule and over budget *by an order of magnitude!*
   - Does not meet user requirements.
   - Development of *working code* is slower than expected.
   - Progress is *difficult to measure*.

EIU

DEALING WITH SOFTWARE COMPLEXITY



TRAINING SOFTWARE ENGINEERS

"programming-in-the-small" → coding

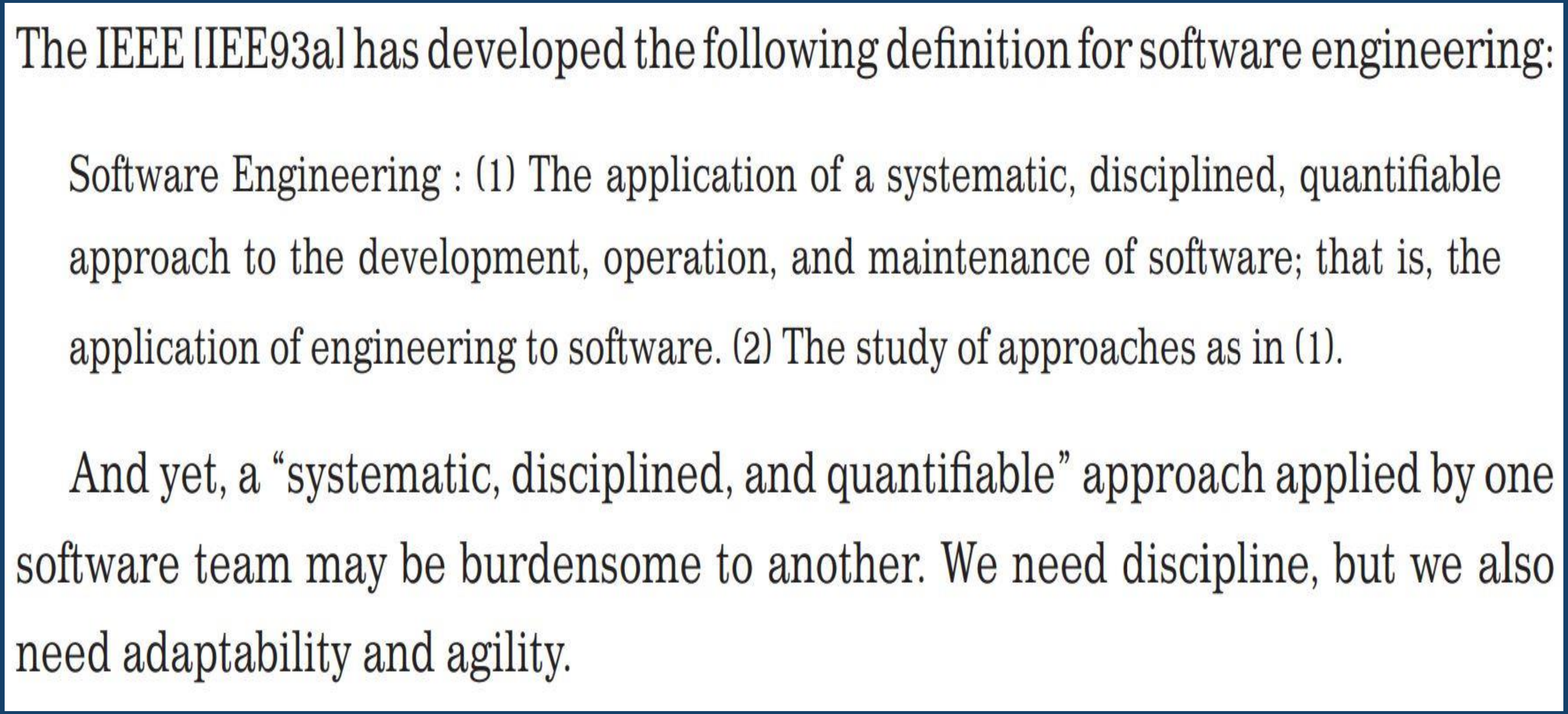
"programming-in-the-large" → software engineering

A software engineer needs to be able to:

- talk with users in terms of the application.
- translate vague requirements into precise specifications.
- build models of a system at different levels of abstraction.
- use and apply several software development processes.
- choose among design alternatives (i.e., make design tradeoffs).
- work in well-defined roles as part of a team.

This reduces the complexity of building the system!

WHAT IS SOFTWARE ENGINEERING?



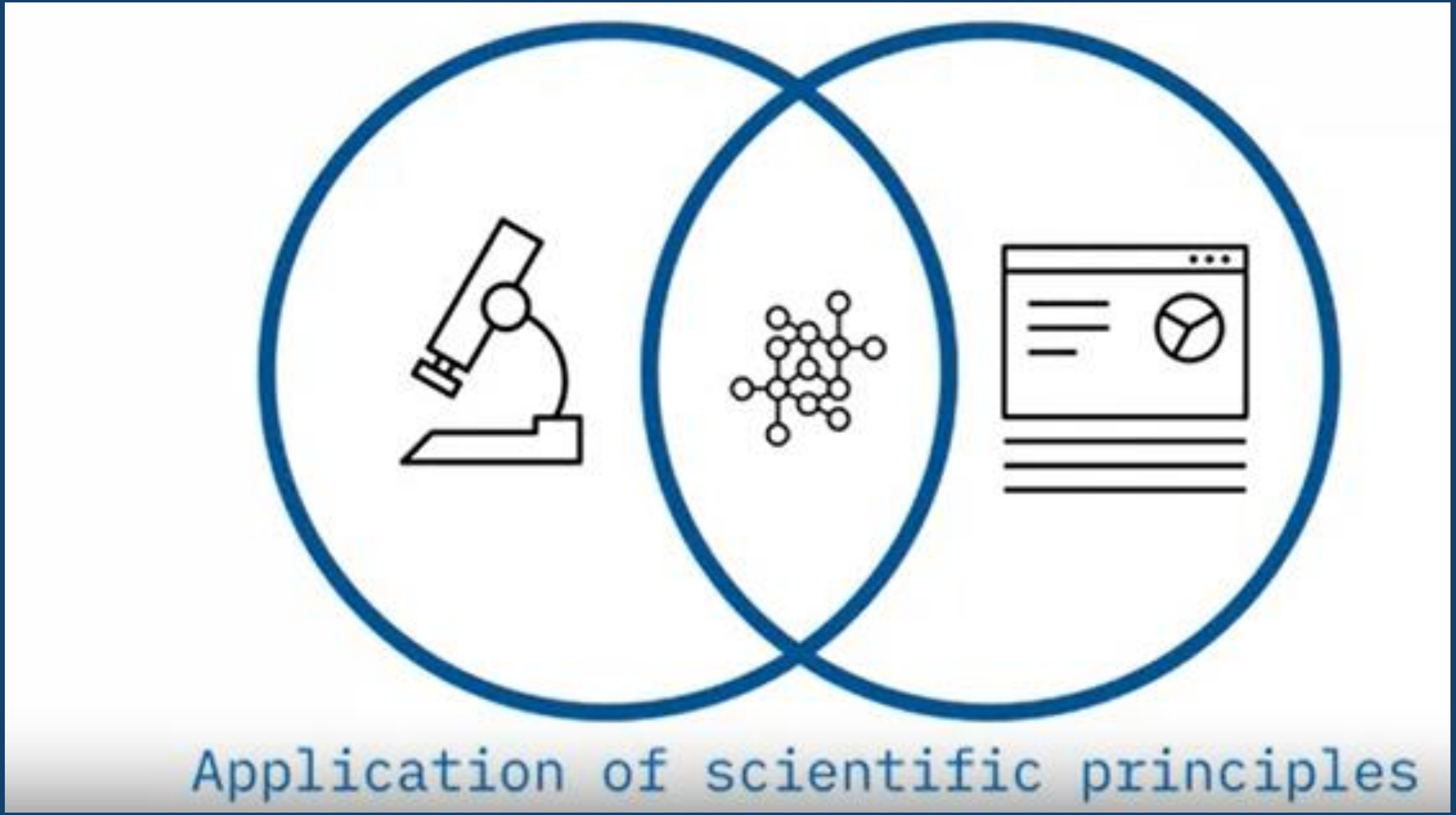
The IEEE [IEE93a] has developed the following definition for software engineering:

Software Engineering : (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

And yet, a "systematic, disciplined, and quantifiable" approach applied by one software team may be burdensome to another. We need discipline, but we also need adaptability and agility.

WHAT IS SOFTWARE ENGINEERING?

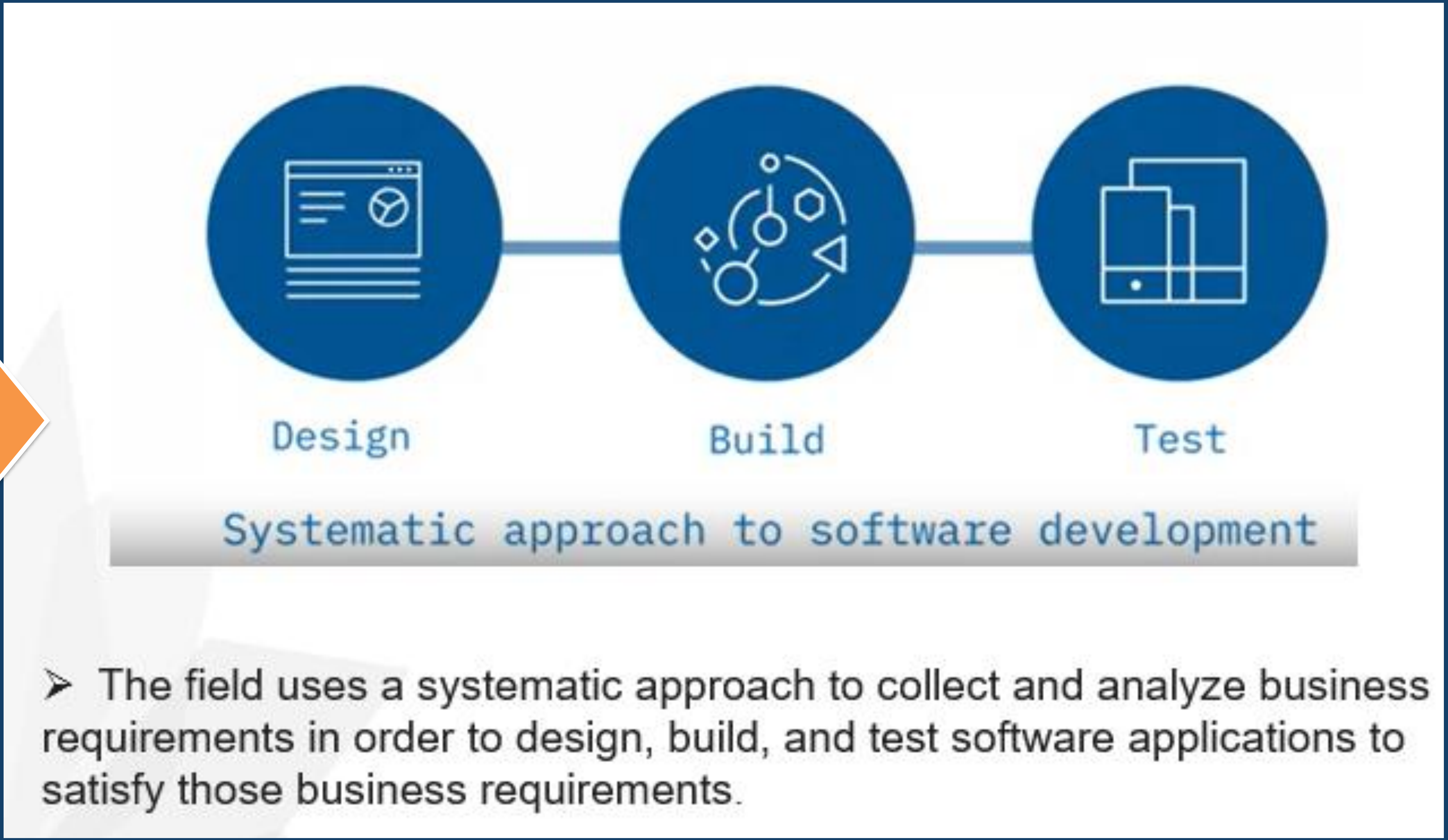
- *Quantifiable*



➢ Software engineering is the application of scientific principles to the design and creation of software.

WHAT IS SOFTWARE ENGINEERING?

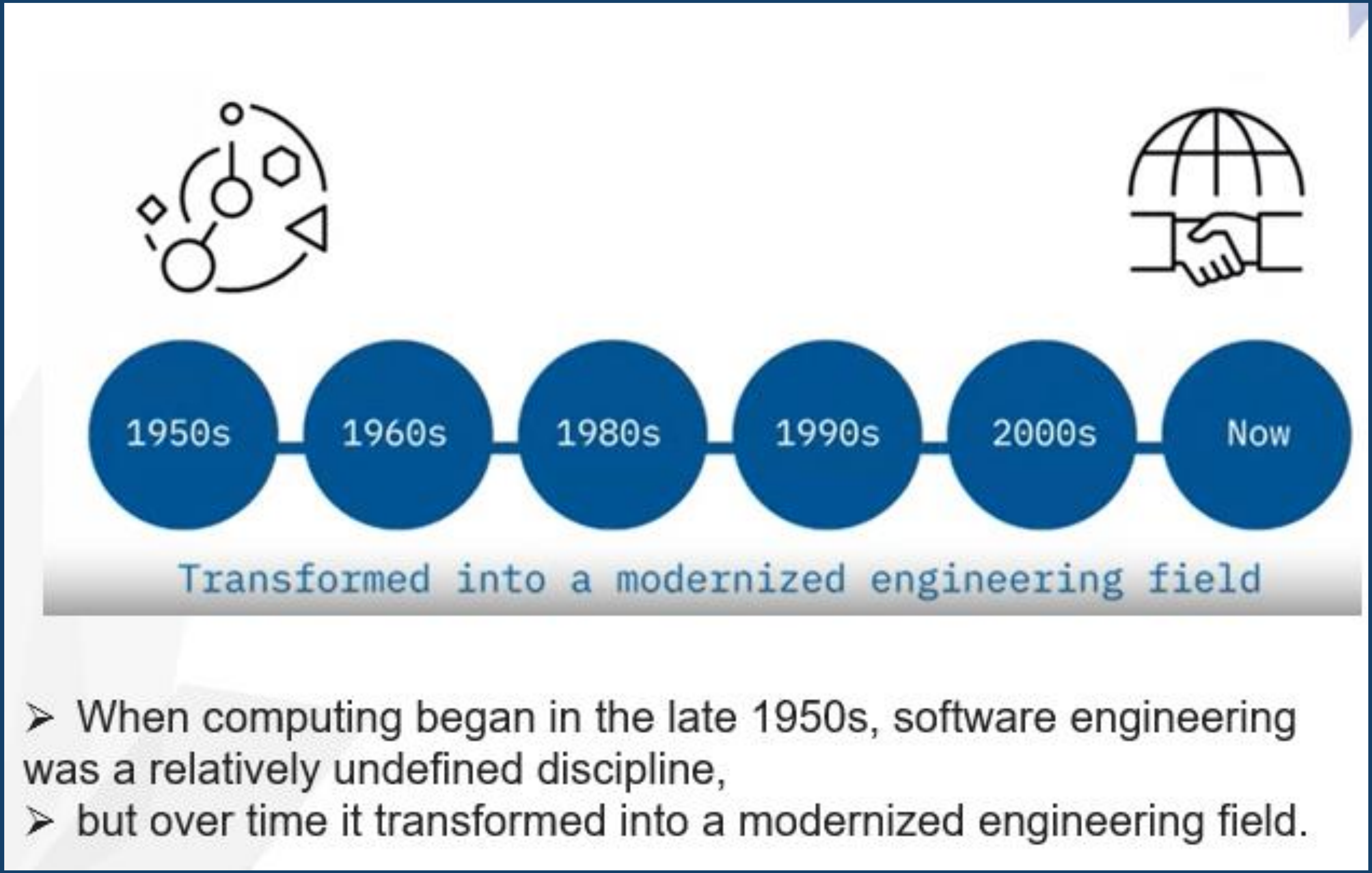
- *Quantifiable*
- *Systematic*



➢ The field uses a systematic approach to collect and analyze business requirements in order to design, build, and test software applications to satisfy those business requirements.

WHAT IS SOFTWARE ENGINEERING?

- *Quantifiable*
- *Systematic*
- *Discipline*

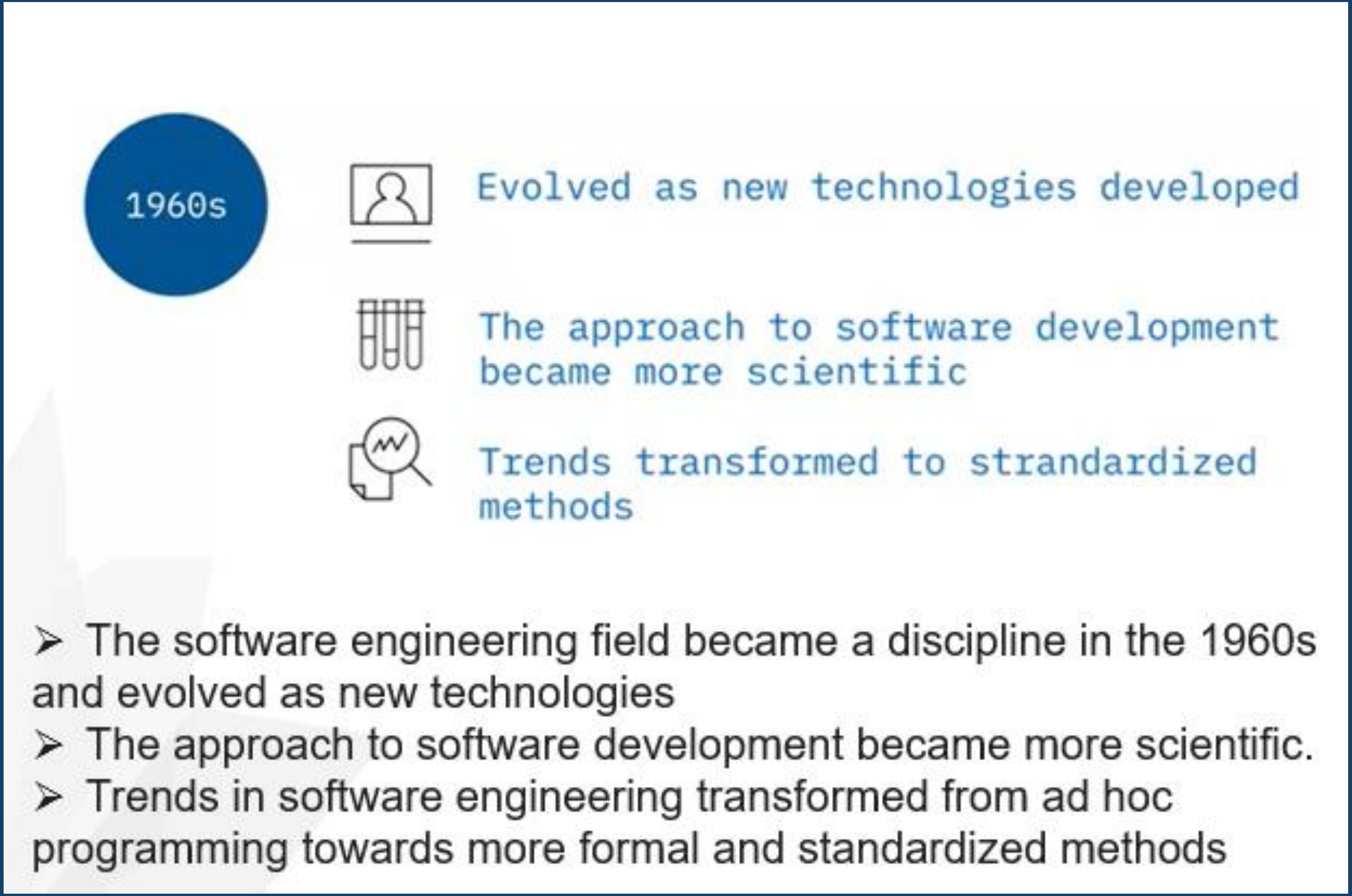


➢ When computing began in the late 1950s, software engineering was a relatively undefined discipline,

➢ but over time it transformed into a modernized engineering field.

WHAT IS SOFTWARE ENGINEERING?

- *Quantifiable*
- *Systematic*
- *Discipline*



1960s

Evolved as new technologies developed

The approach to software development became more scientific

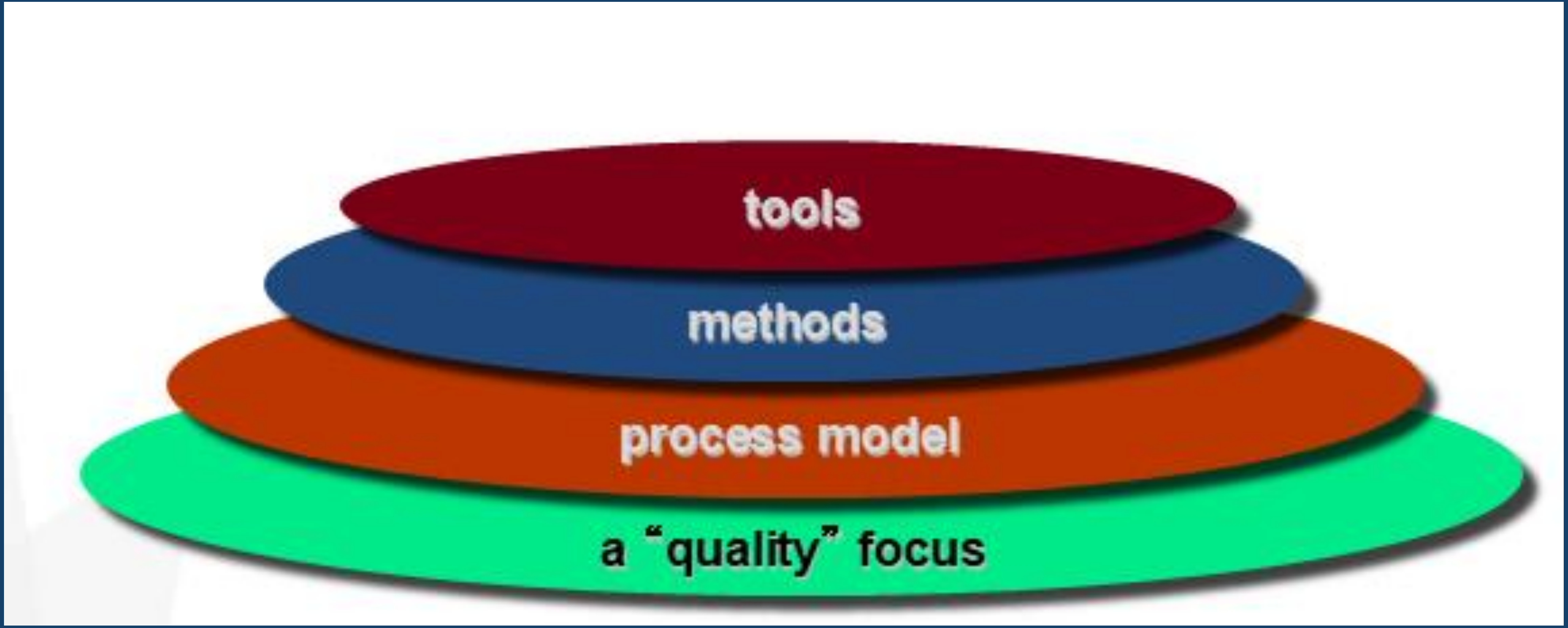
Trends transformed to strandardized methods

➢ The software engineering field became a discipline in the 1960s and evolved as new technologies

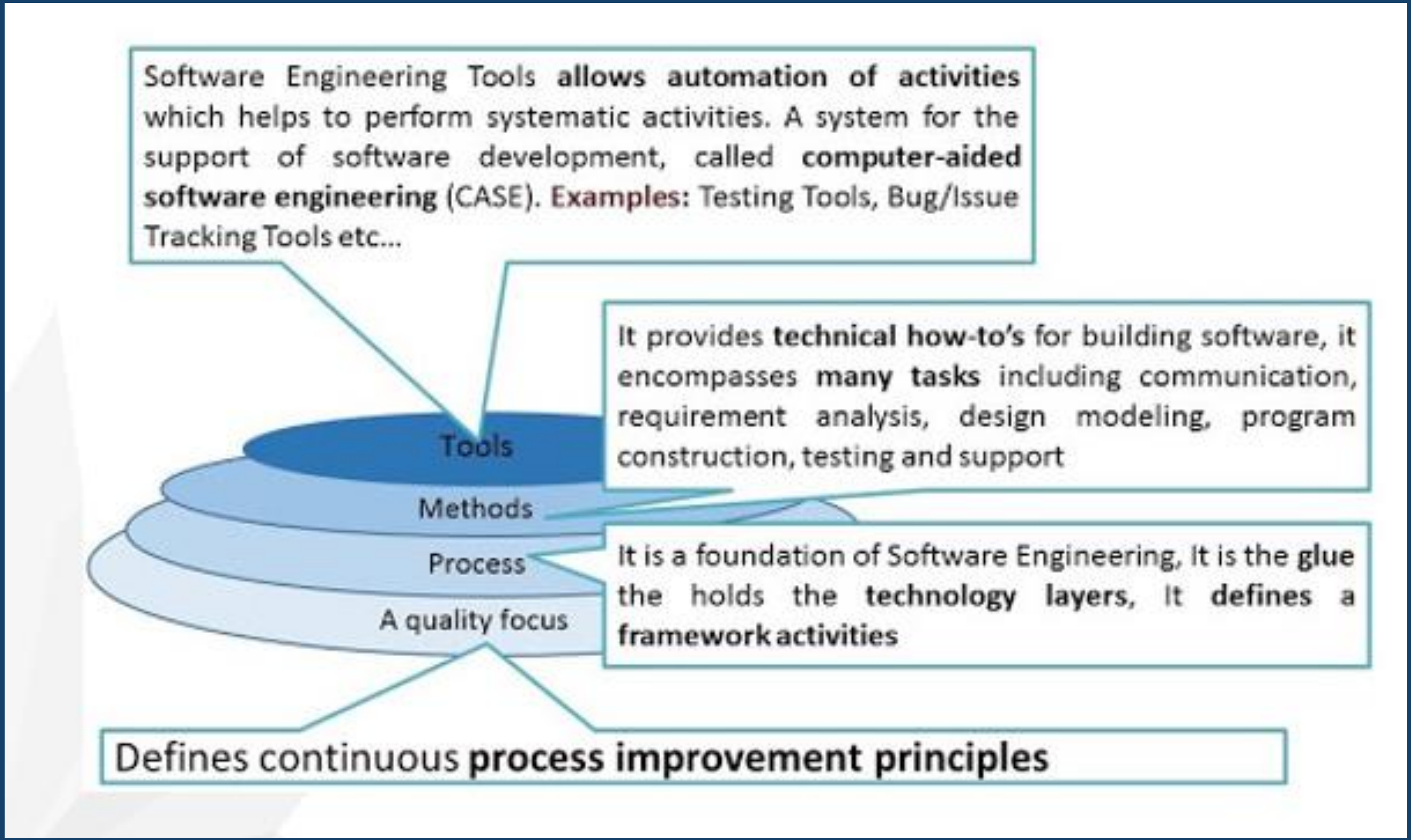
➢ The approach to software development became more scientific.

➢ Trends in software engineering transformed from ad hoc programming towards more formal and standardized methods

SOFTWARE ENGINEERING: A LAYERED TECHNOLOGY



SOFTWARE ENGINEERING: A LAYERED TECHNOLOGY



Software Engineering Tools **allows automation of activities** which helps to perform systematic activities. A system for the support of software development, called **computer-aided software engineering** (CASE). Examples: Testing Tools, Bug/Issue Tracking Tools etc...

It provides **technical how-to's** for building software, it encompasses **many tasks** including communication, requirement analysis, design modeling, program construction, testing and support

It is a foundation of Software Engineering, It is the **glue** the holds the **technology layers**, It **defines a framework activities**

Defines continuous **process improvement principles**

WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Before we get into a software development process, let's take a look at a process that you might be familiar with, like building a house.

- So, if you were to build a house,
- you'll probably reach out to a builder and tell about your needs, like, I need two bedrooms, I need a bigger great room, I need a room on the basement, and etc.

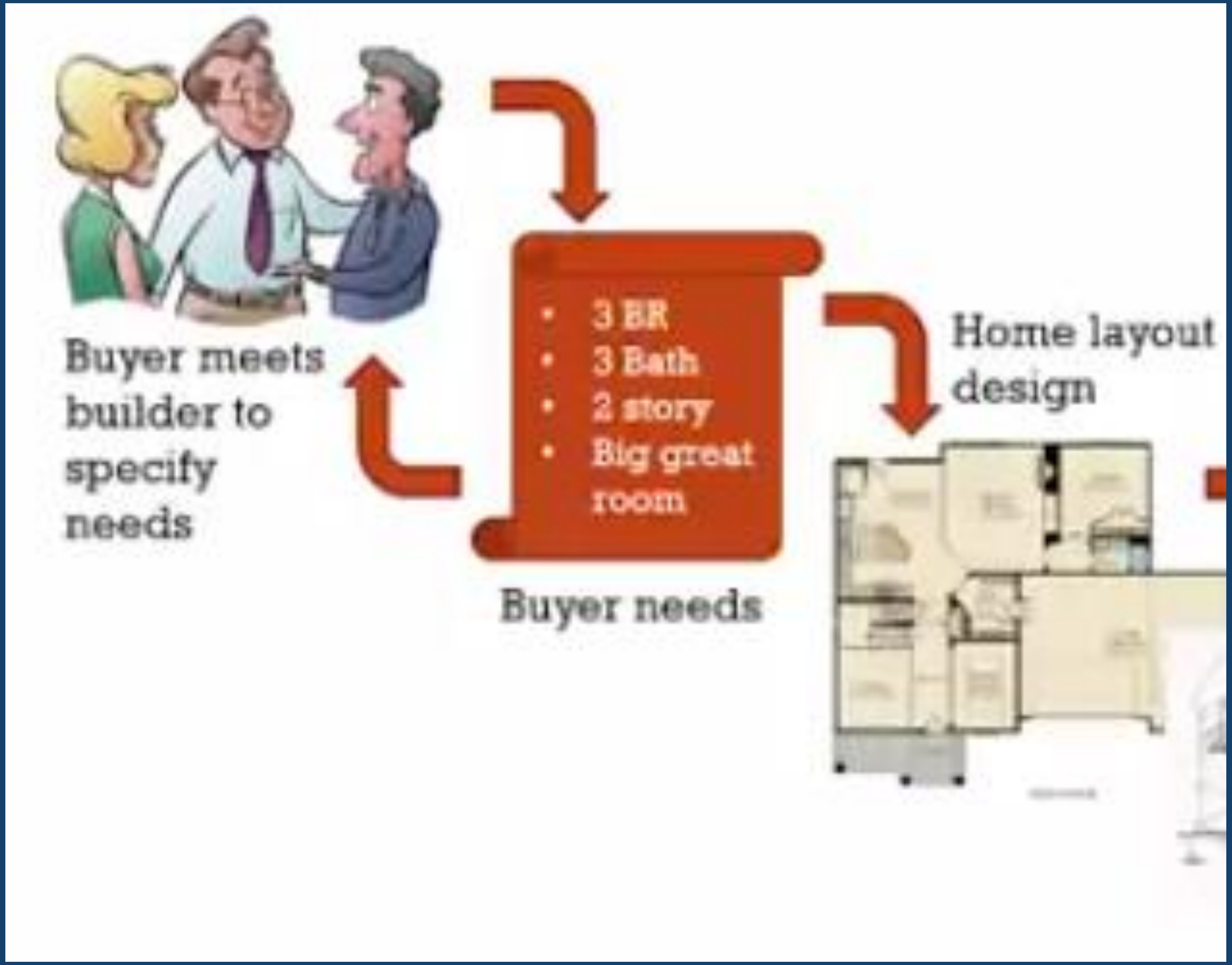
BUILDING A HOUSE?

Buyer meets builder to specify needs
3 BR
3 Bath
2 story
Big great room
Buyer needs

WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Then, builder is going to go back, and going to create a floor plan.
- Then, he's going to show it to you the floor plan, and you might make so me adjustment.

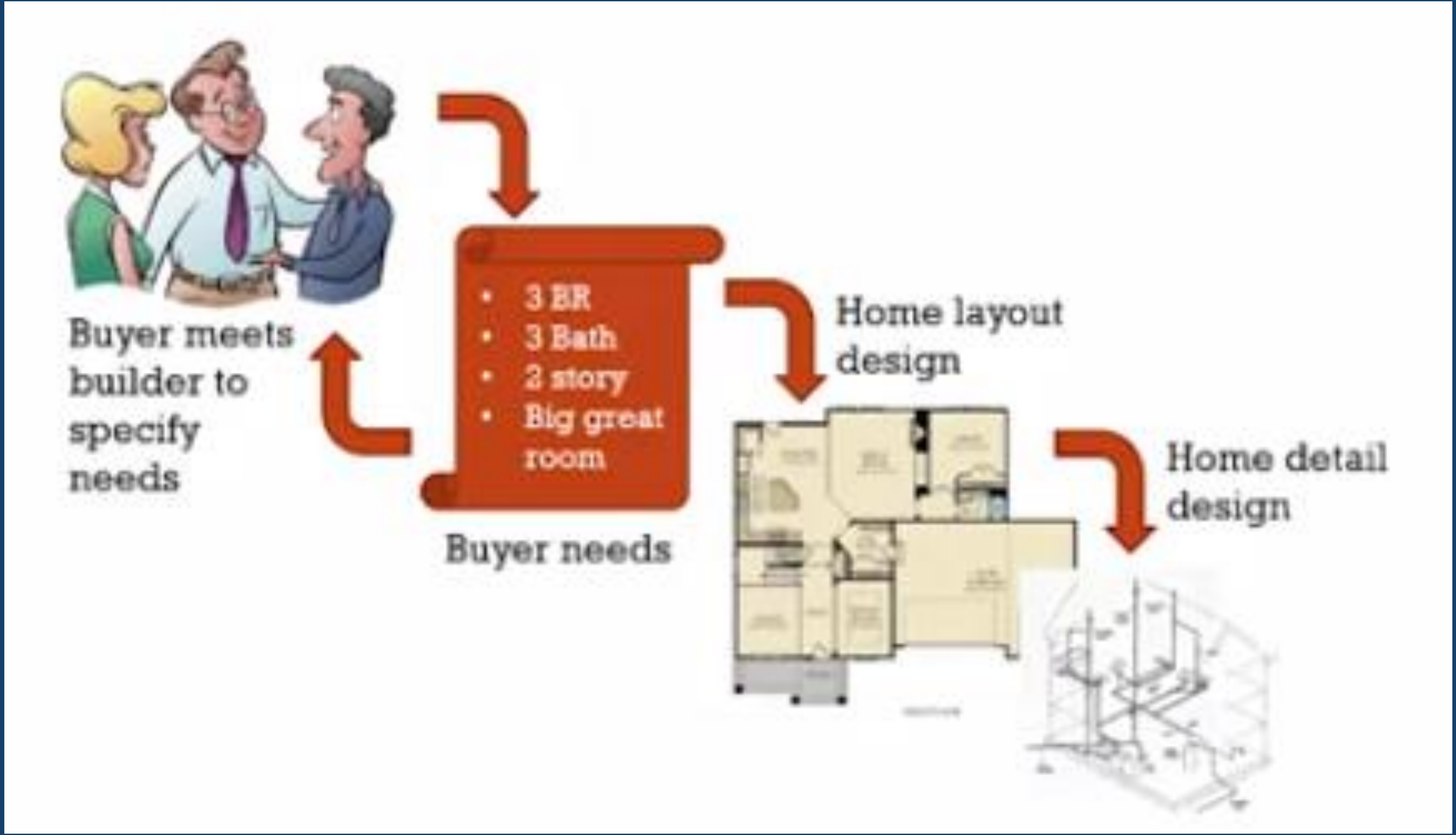
BUILDING A HOUSE?



WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

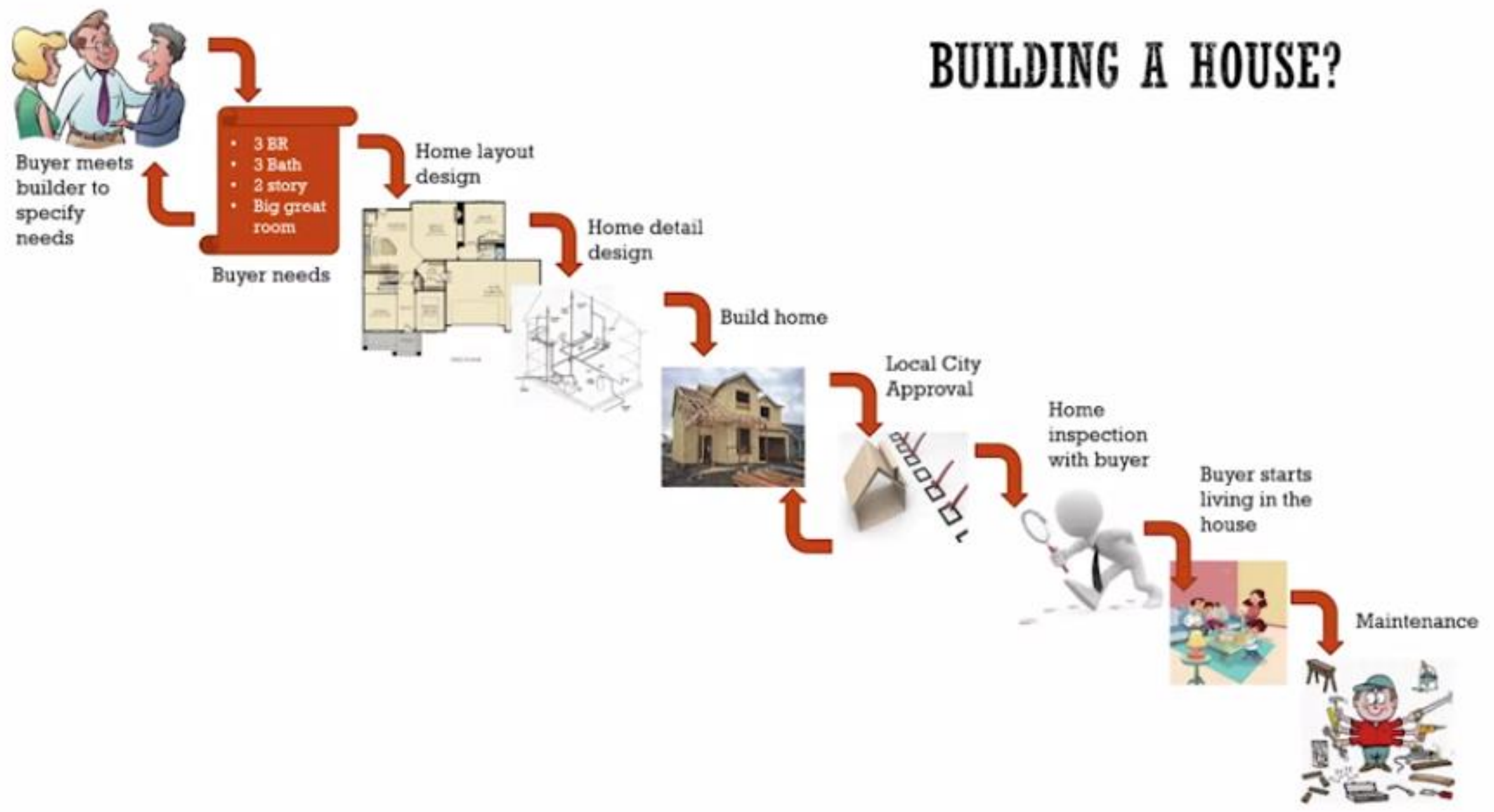
➢ Once that is done, builder is going to create a detailed map or detailed plan about where the foundation is going to be, where the electricity is going to be, where the plugs are going to be, where the plumbing is going to be, and then, they're going to start building the home

BUILDING A HOUSE?



WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

➢ Once the house is ready, the builder is going to call you to say whether, you like the house, Is it what you expected? And then, once you approve it, then you're going to start, staying in that house or living in that house and, of course, there comes the maintenance of the house.



WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Let's see how software development is similar or different than this.

- So, when you're building a software, similar to building a house, **you're looking at requirements** or you're looking at what exactly you need to build.

- During that time, you will do analysis of different options or you might create a prototype and show it to your customer, and just say, "Is this what you're looking for?" But at the end of this phase, you have exactly what you're looking to build.

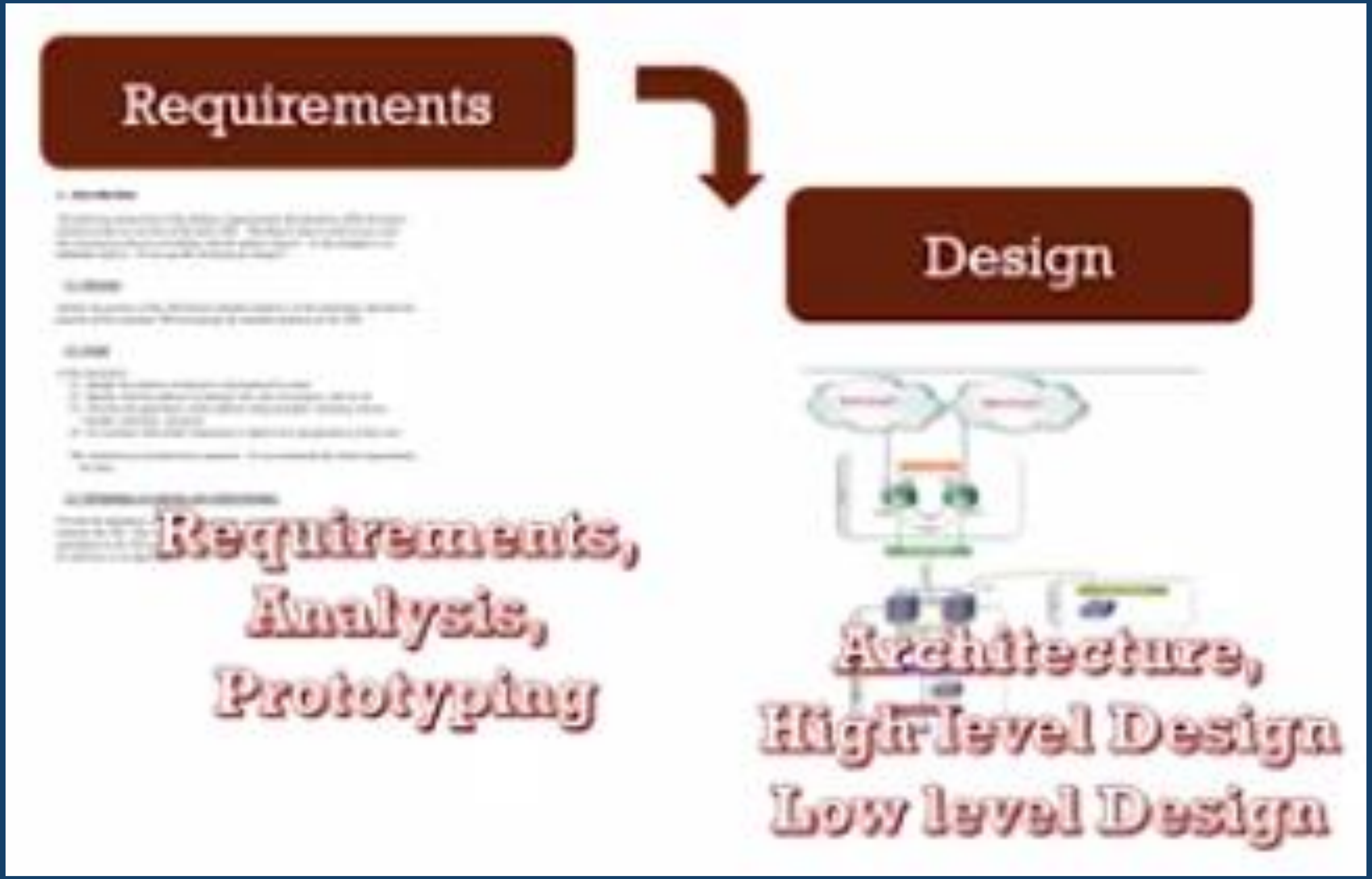
BUILDING SOFTWARE?

EIU

WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Then the architects and the software developers are going to **design the system**, they're going **to architect the system**, and say, **what are the different components** that they need to build? And how they're going to work together?

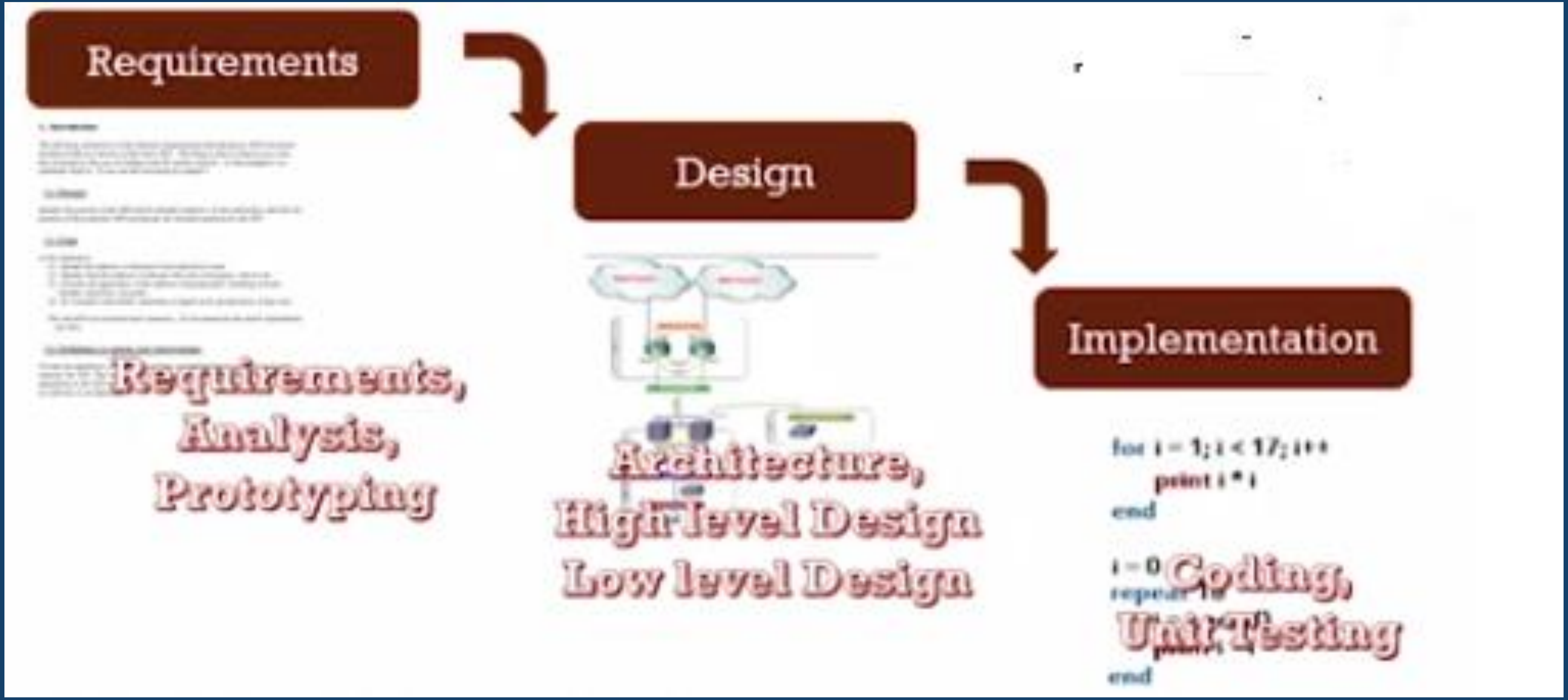
BUILDING SOFTWARE?


Requirements
Design
Requirements, Analysis, Prototyping
Architecture, High level Design Low level Design

WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Once the design is done, they're going **to start coding, and start doing unit testing**. So, each of the sub-teams are going to start building their components and test it

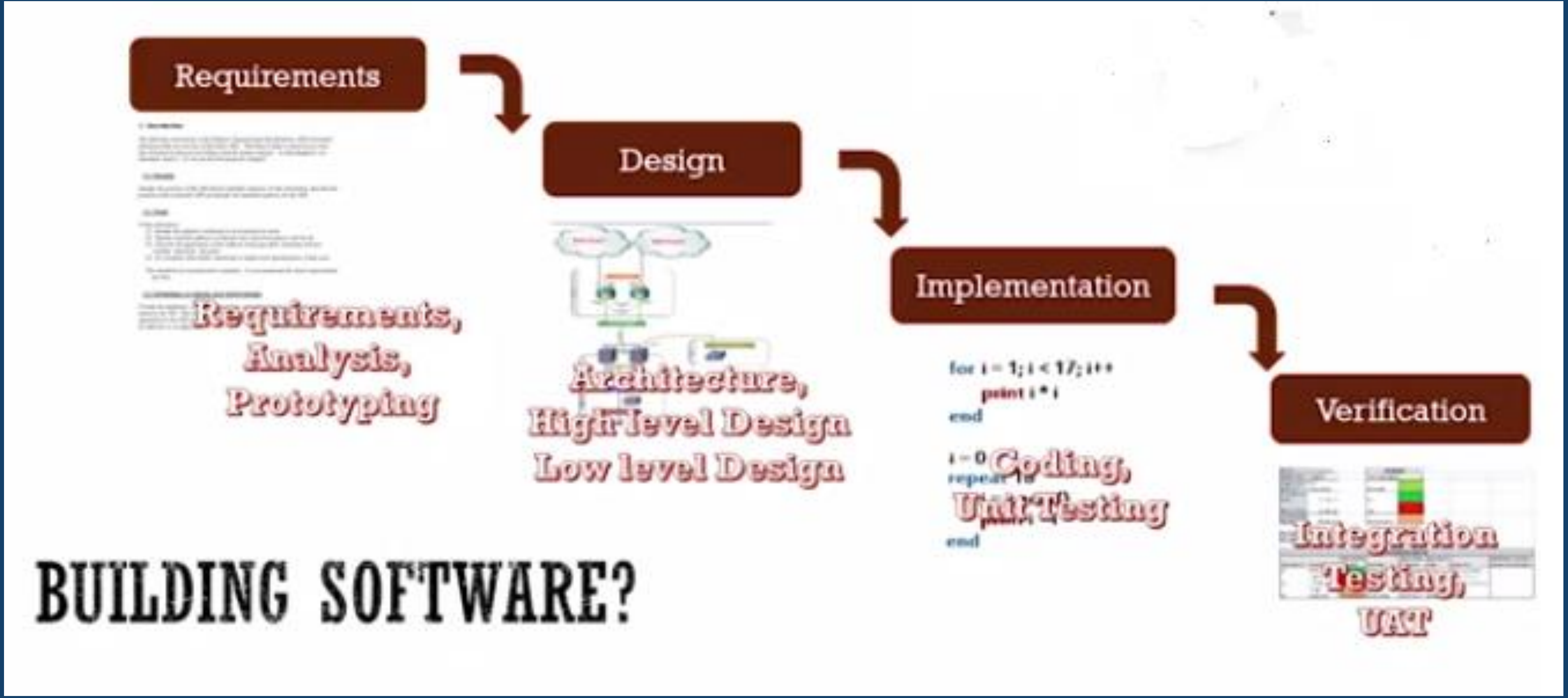
BUILDING SOFTWARE?


Requirements
Design
Implementation
Requirements, Analysis, Prototyping
Architecture, High level Design, Low level Design
Coding, Unit Testing

```
for i = 1; i < 17; i++
    print i * i
end
```

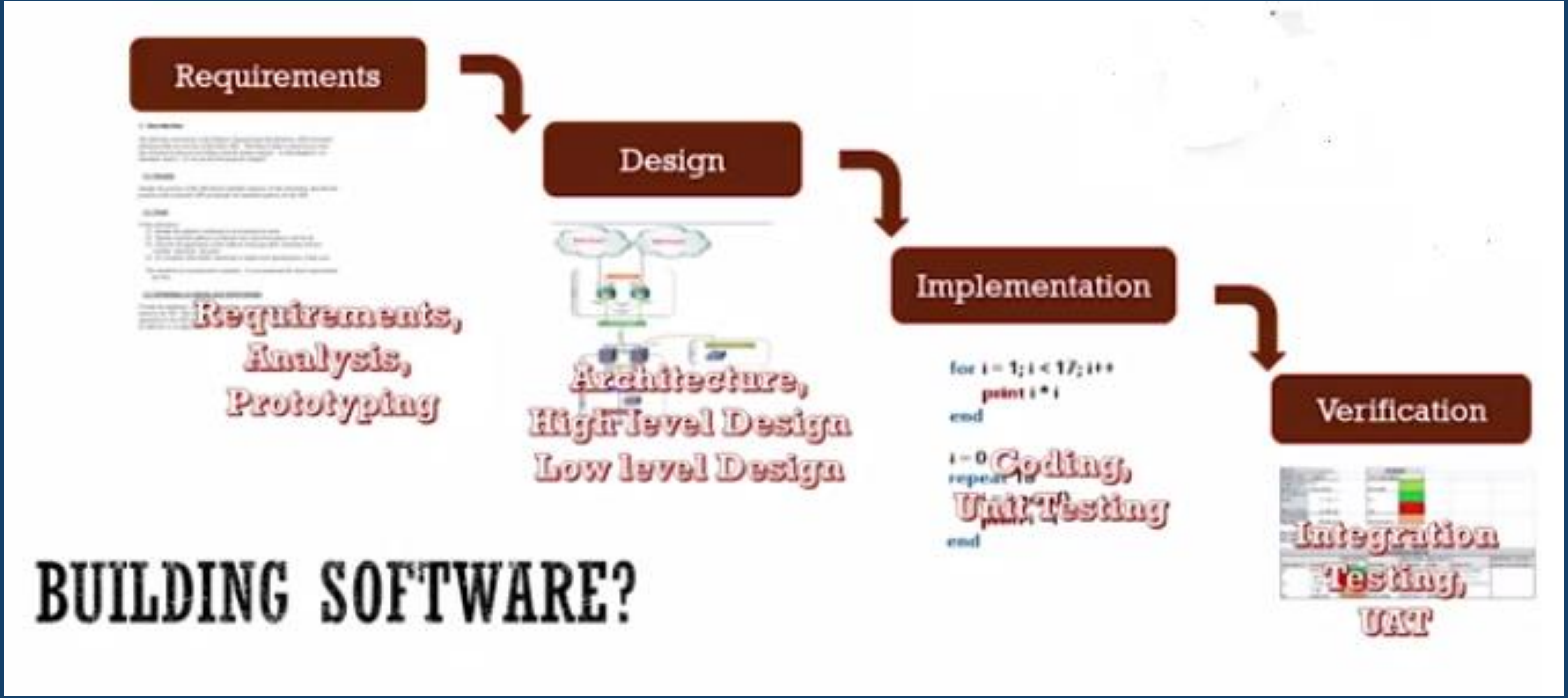
WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

- Once all the components are ready, they're going to **bring all of these components together**, and **do integration testing** or card verification. And, they will **do functional testing**.



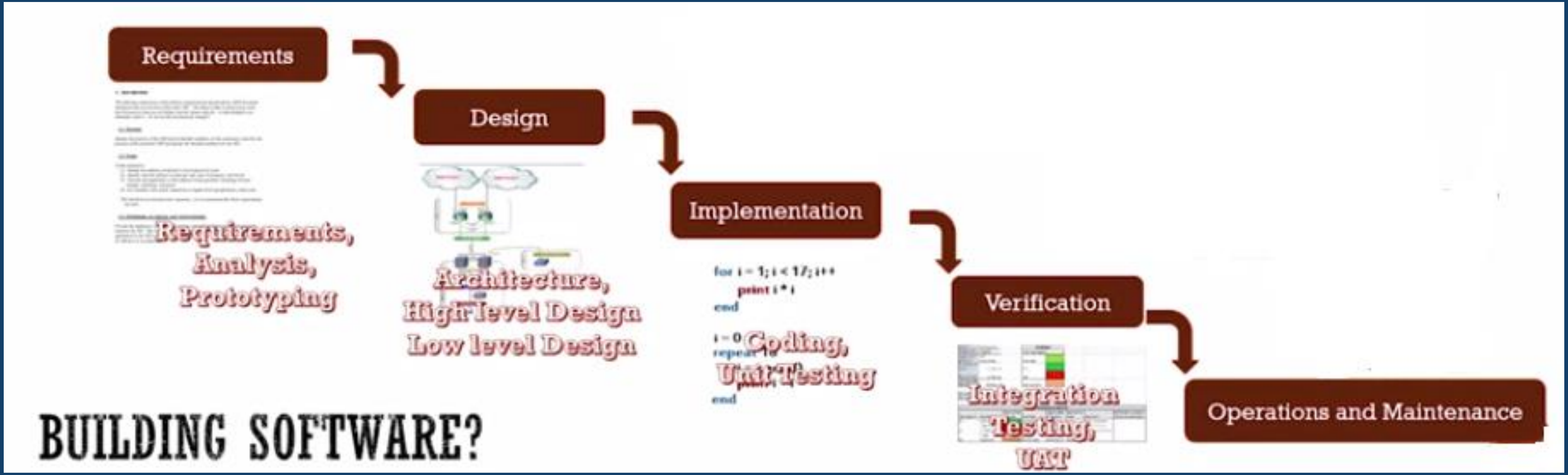
WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

➢ Next, when the software is ready from the developer point of view, then they're going **to invite the user, and they will do a testing called, UAT [User Acceptance Testing]** , in which a user says, "Yup, this is what I was looking for." or "No, this is not what I was looking for."



WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

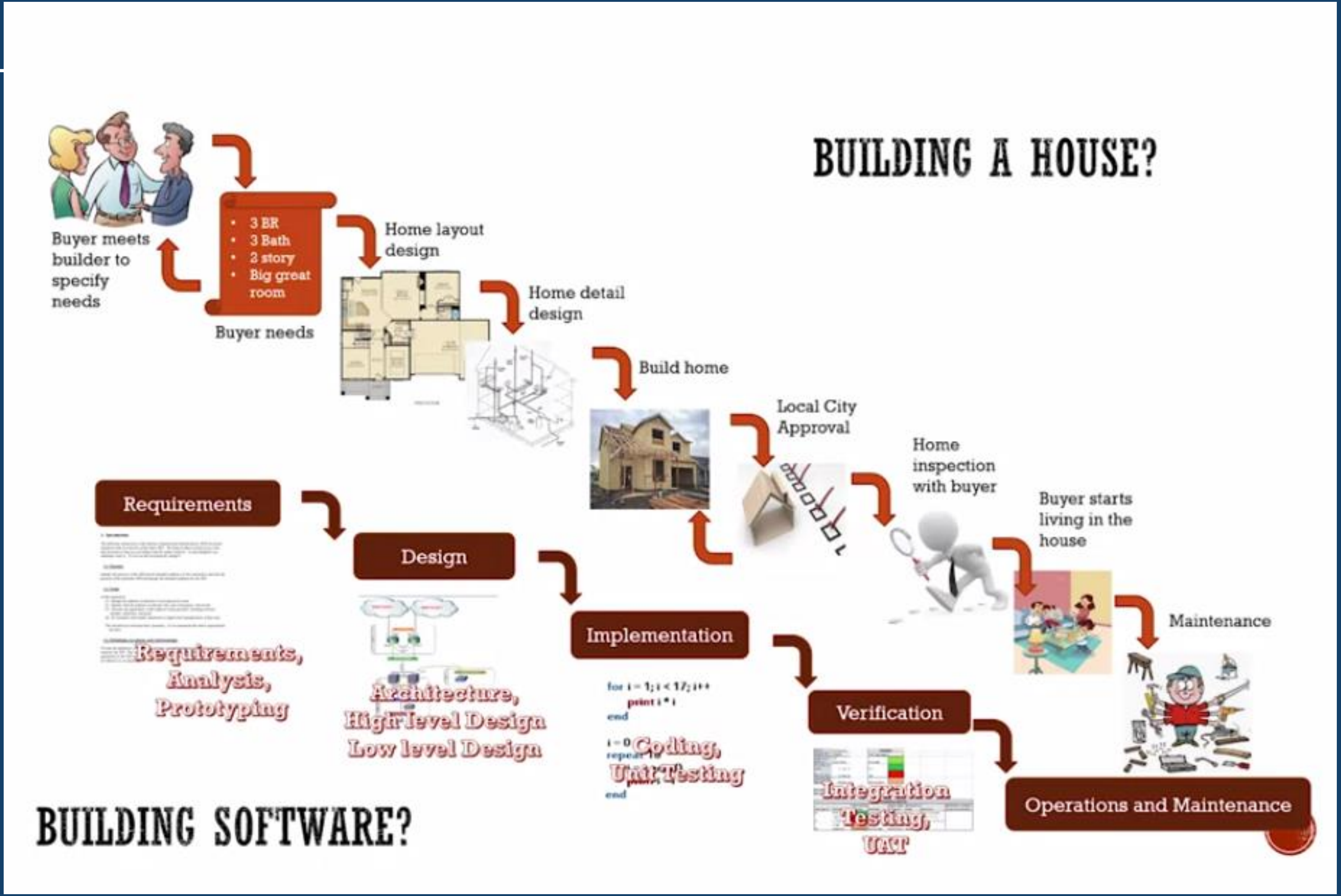
➢ Next, once everything is done, then **the software goes into production**, which means **the user is going to start using the software**, and they might **ask for some changes**, or **you may have to fix some defects**, and so on. So, **that's called the operation and maintenance**.

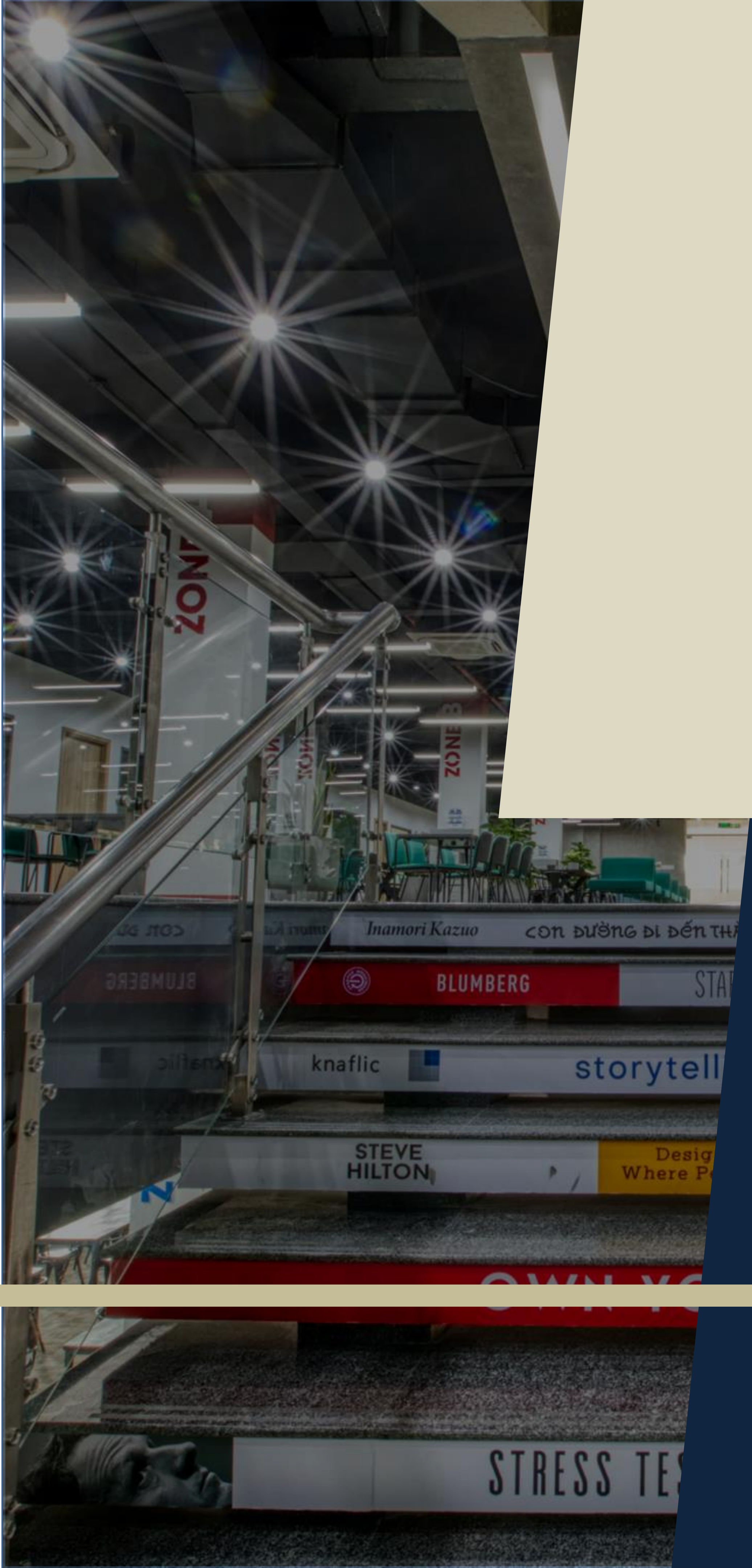


Requirements
Requirements, Analysis, Prototyping
Design
Architecture, High level Design Low level Design
Implementation
Coding, Unit Testing
Verification
Integration Testing, UAT
Operations and Maintenance
BUILDING SOFTWARE?

WHAT SOFTWARE DEVELOPMENT LOOKS LIKE?

➢ So, as you can see, the process is quite similar to building a house





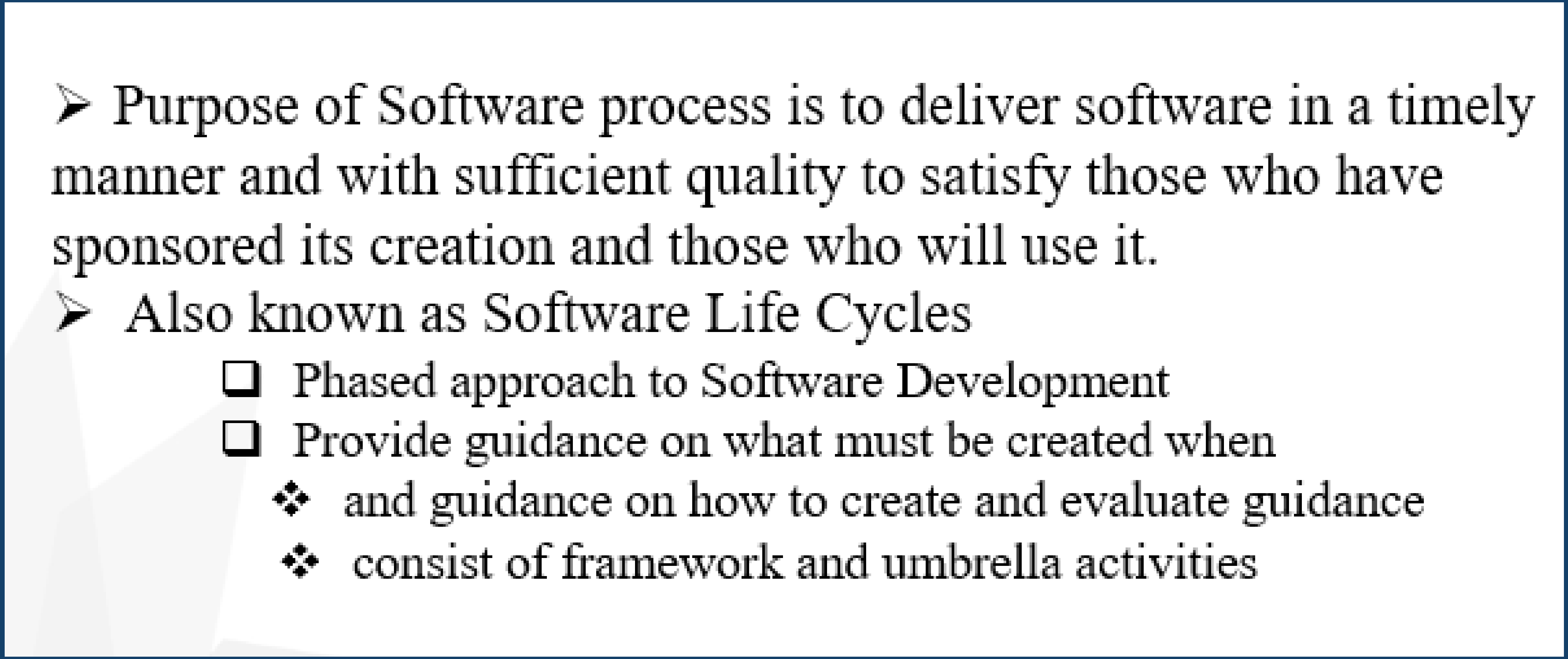
Chapter 1. Introduction to Software Engineering

1.1. Definition of Software Engineering Discipline

1.2. The Software Process

1.3. Software Engineering Practice

Software Process/ Software development Life Cycle

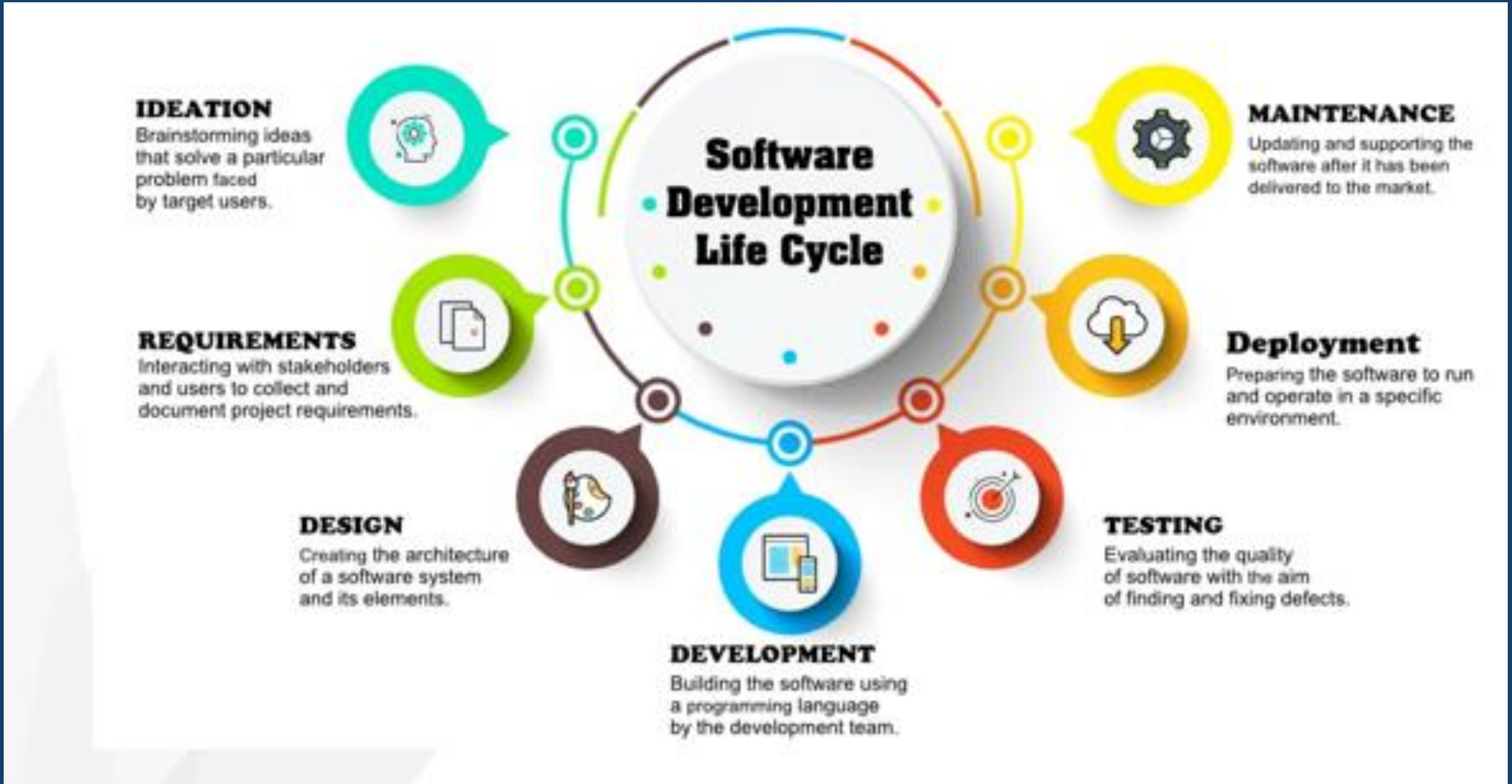


➢ Purpose of Software process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

➢ Also known as Software Life Cycles

- ❑ Phased approach to Software Development
- ❑ Provide guidance on what must be created when
  - ❖ and guidance on how to create and evaluate guidance
  - ❖ consist of framework and umbrella activities

Software Process/ Software development Life Cycle


Software Development Life Cycle
IDEATION
Brainstorming ideas that solve a particular problem faced by target users.
REQUIREMENTS
Interacting with stakeholders and users to collect and document project requirements.
DESIGN
Creating the architecture of a software system and its elements.
DEVELOPMENT
Building the software using a programming language by the development team.
TESTING
Evaluating the quality of software with the aim of finding and fixing defects.
Deployment
Preparing the software to run and operate in a specific environment.
MAINTENANCE
Updating and supporting the software after it has been delivered to the market.

- Software Development Life Cycle (SDLC) is a methodology that defines the steps of a software development project

Process Framework

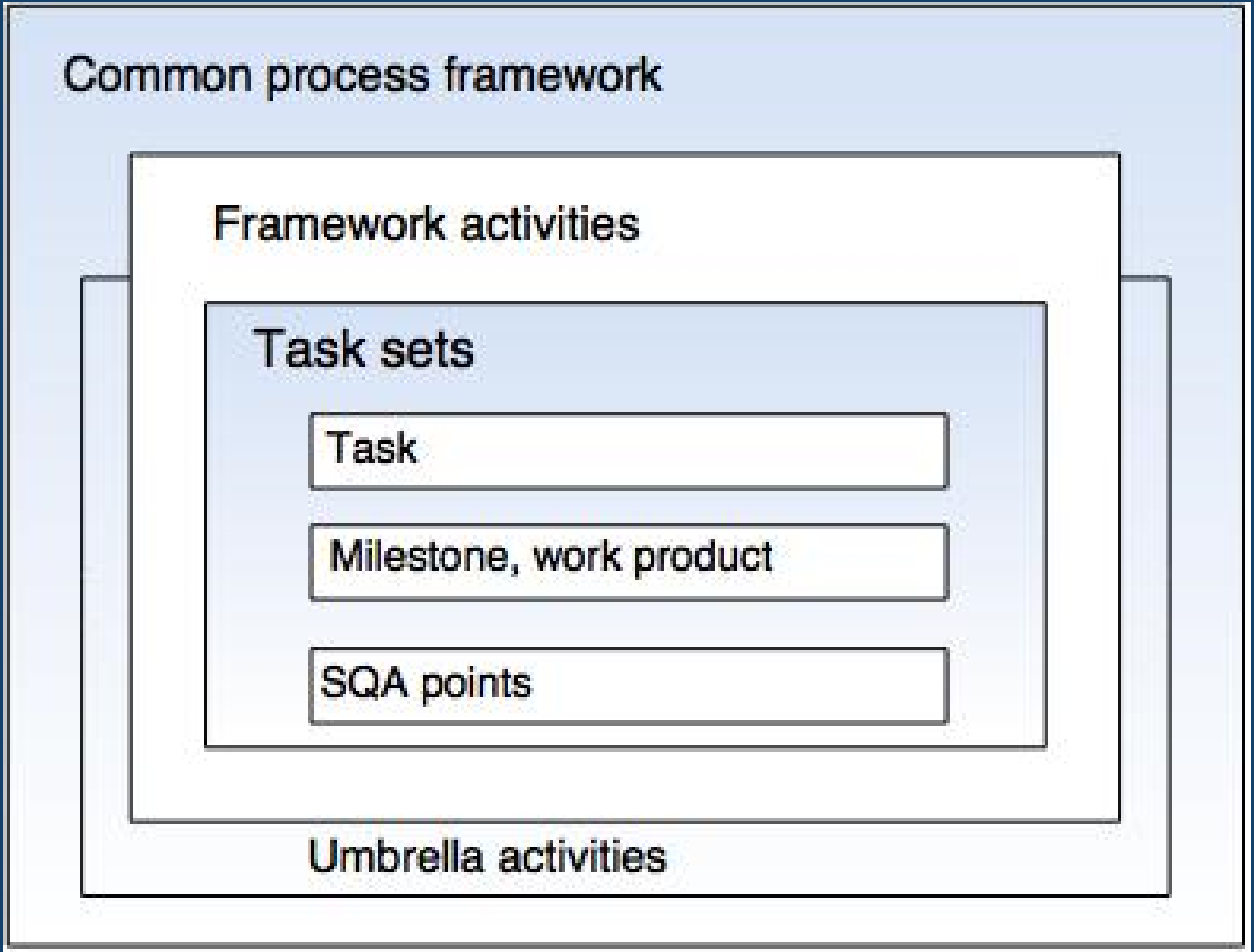
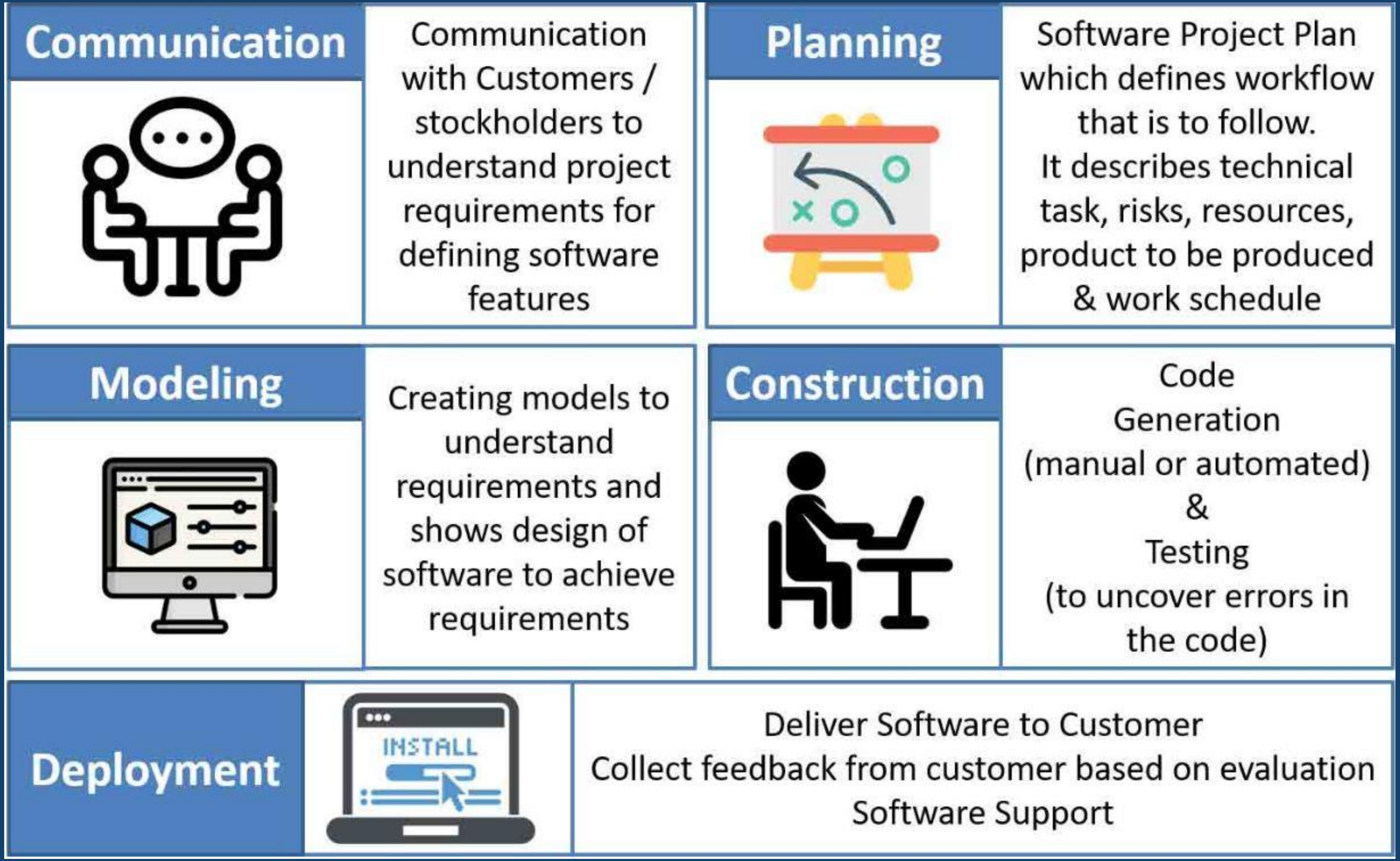


Fig.- A software process framework

- establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- encompasses a set of umbrella activities that are applicable across the

Process Framework

- **Software Phases/ Framework Activities**

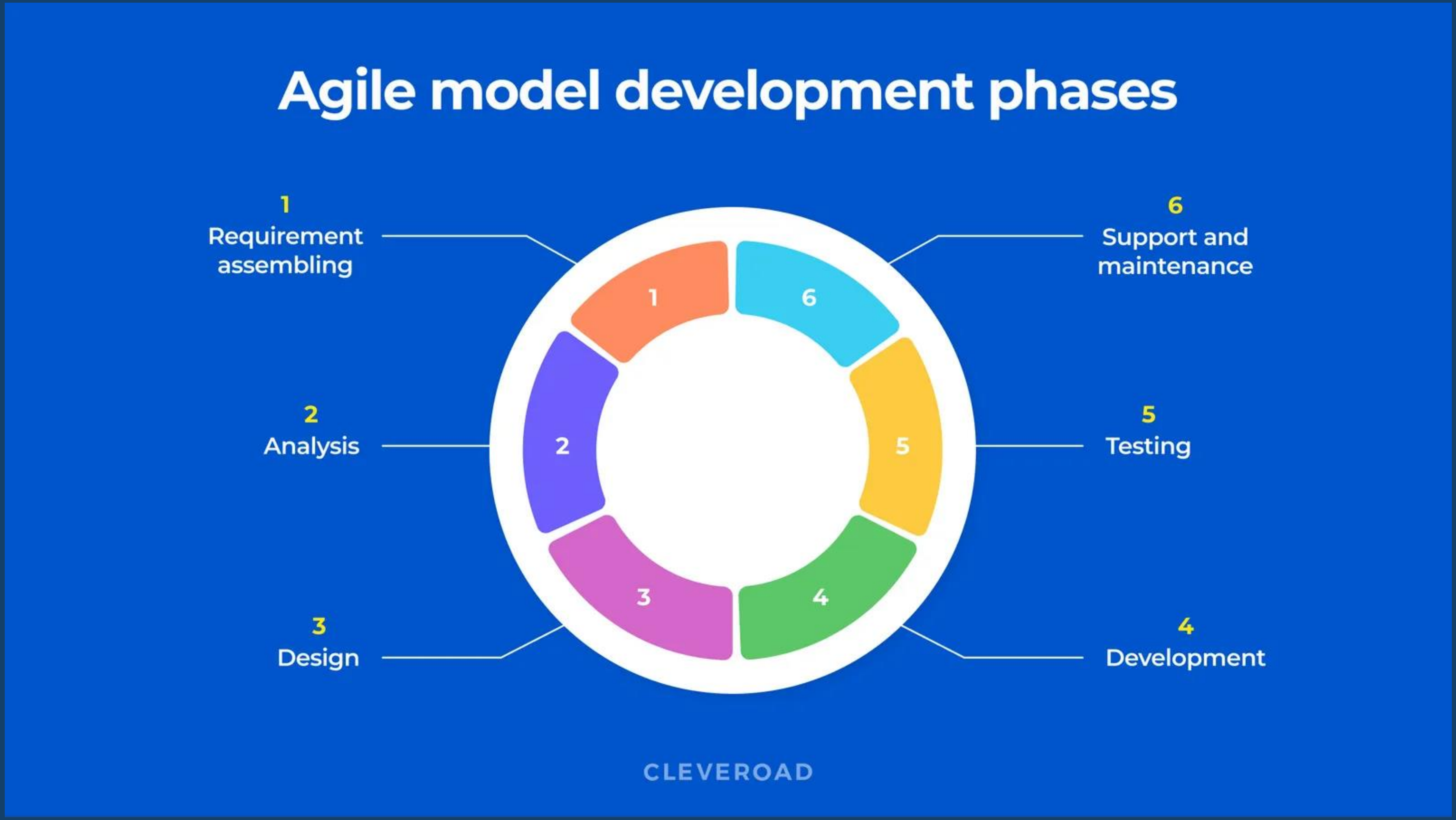


Communication	Communication with Customers / stockholders to understand project requirements for defining software features	Planning	Software Project Plan which defines workflow that is to follow. It describes technical task, risks, resources, product to be produced & work schedule
Modeling	Creating models to understand requirements and shows design of software to achieve requirements	Construction	Code Generation (manual or automated) & Testing (to uncover errors in the code)
Deployment	Deliver Software to Customer Collect feedback from customer based on evaluation Software Support		

- A generic process framework for software engineering encompasses five activities: Communication, Planning, Modeling, Construction and Deployment
- For many software projects, framework activities are applied iteratively as a project progresses

Process Framework

- **Software Phases/ Framework Activities**



- The number of phases in a software development process can vary depending on the specific model, framework, or methodology being used
- Software engineering process is not a rigid prescription that must be followed dogmatically by a software team. Rather, it should be **agile and adaptable**

Process Framework

- Software Phases/ Framework Activities
- Umbrella Activities

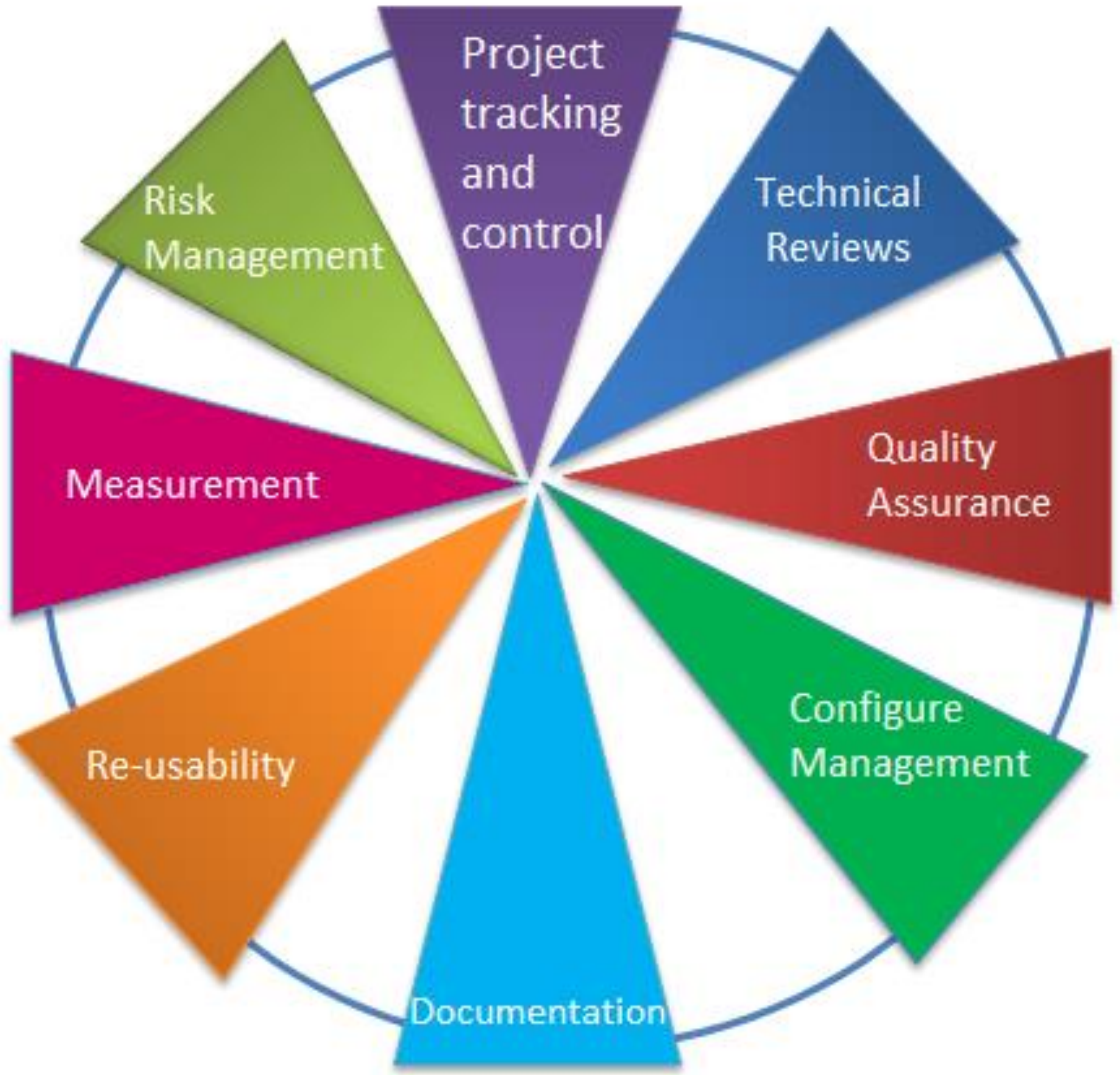
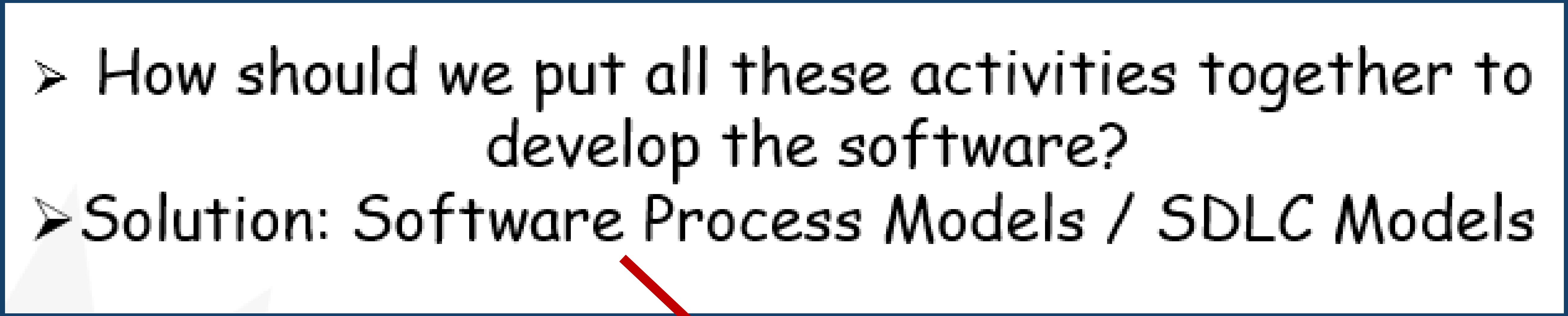


Fig: Umbrella Activities in Software Engineering

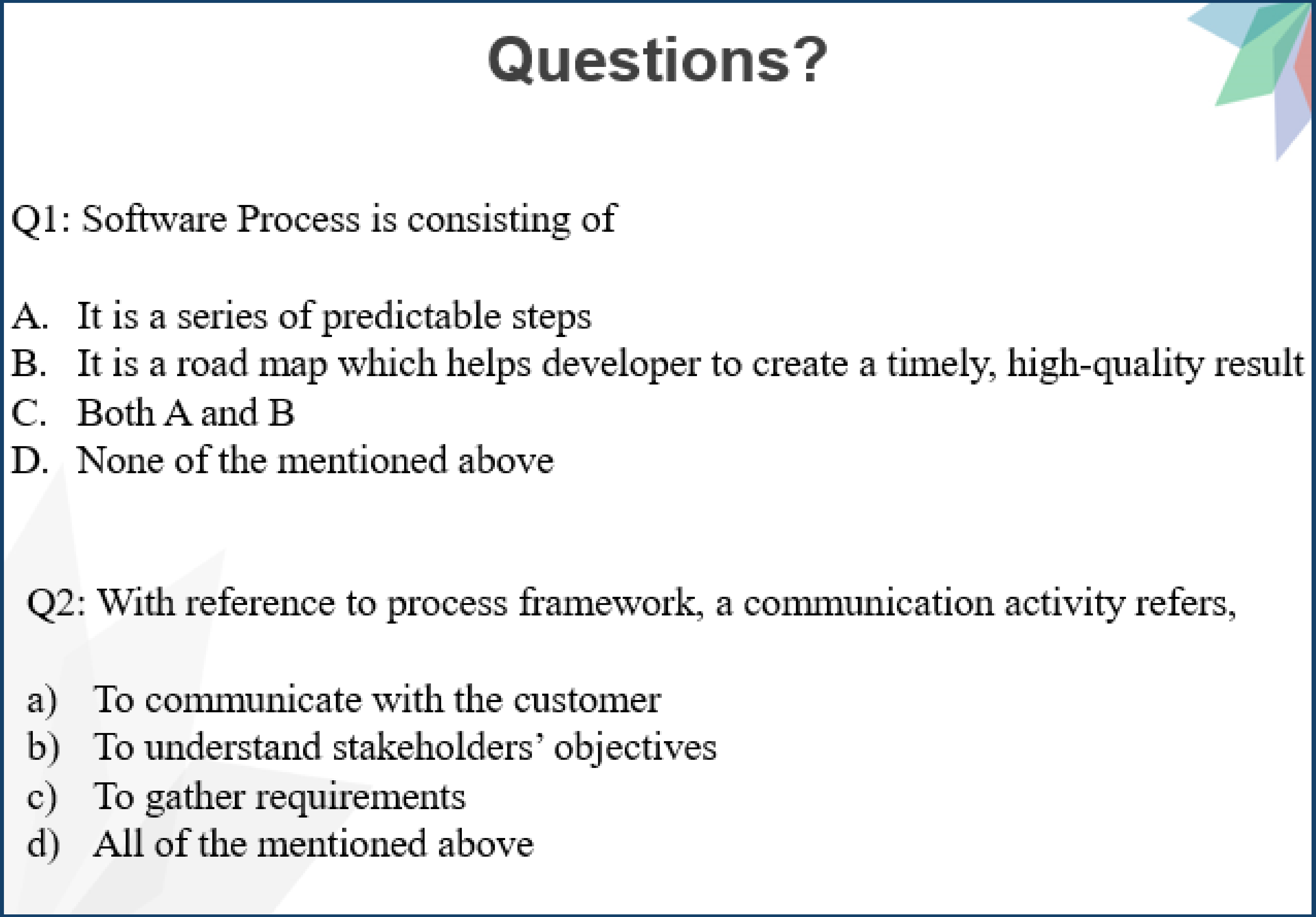
In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk



- How should we put all these activities together to develop the software?
- Solution: Software Process Models / SDLC Models

Topic of Chapter 2

Question?



Questions?

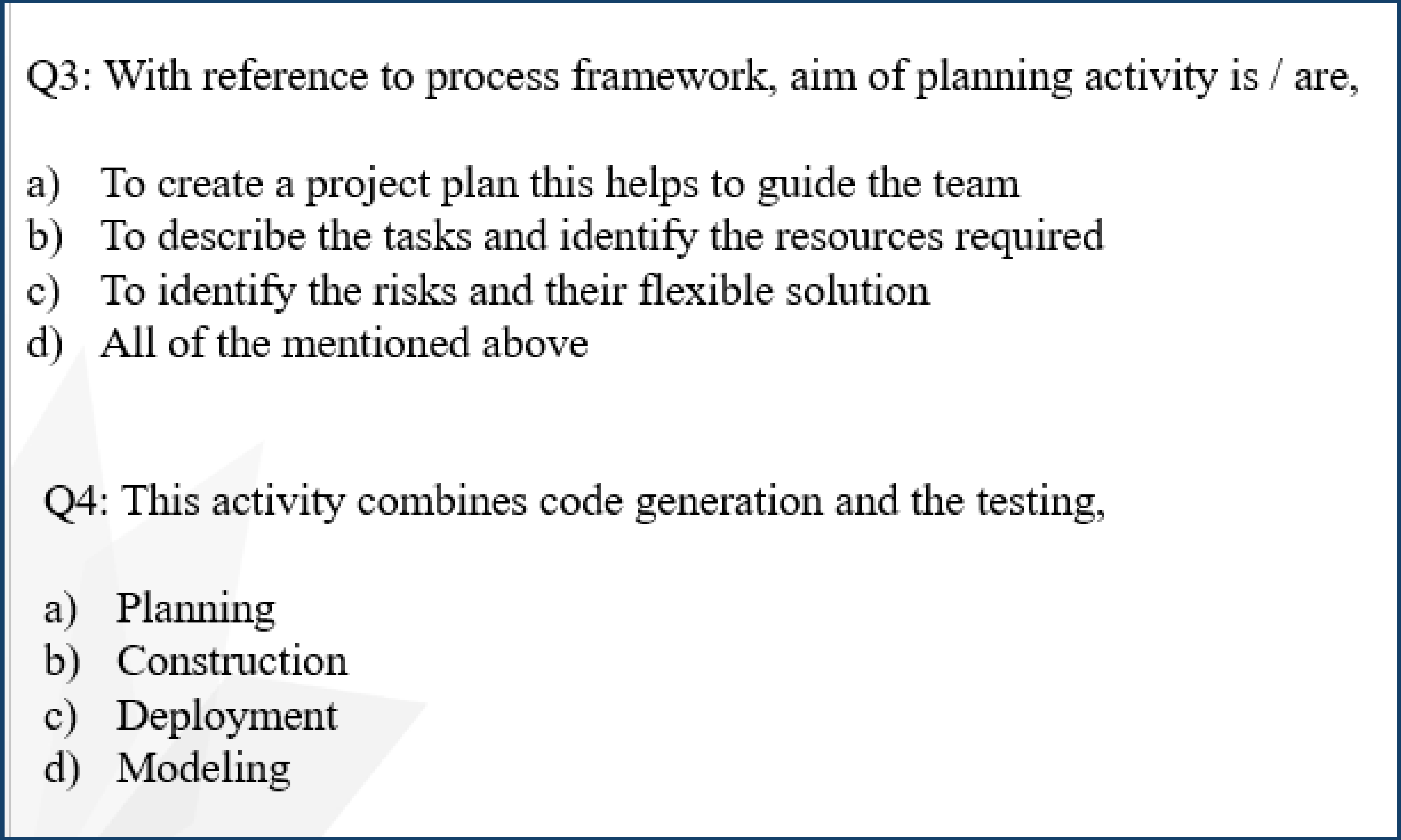
Q1: Software Process is consisting of

A. It is a series of predictable steps
B. It is a road map which helps developer to create a timely, high-quality result
C. Both A and B
D. None of the mentioned above

Q2: With reference to process framework, a communication activity refers,

a) To communicate with the customer
b) To understand stakeholders' objectives
c) To gather requirements
d) All of the mentioned above

Question?



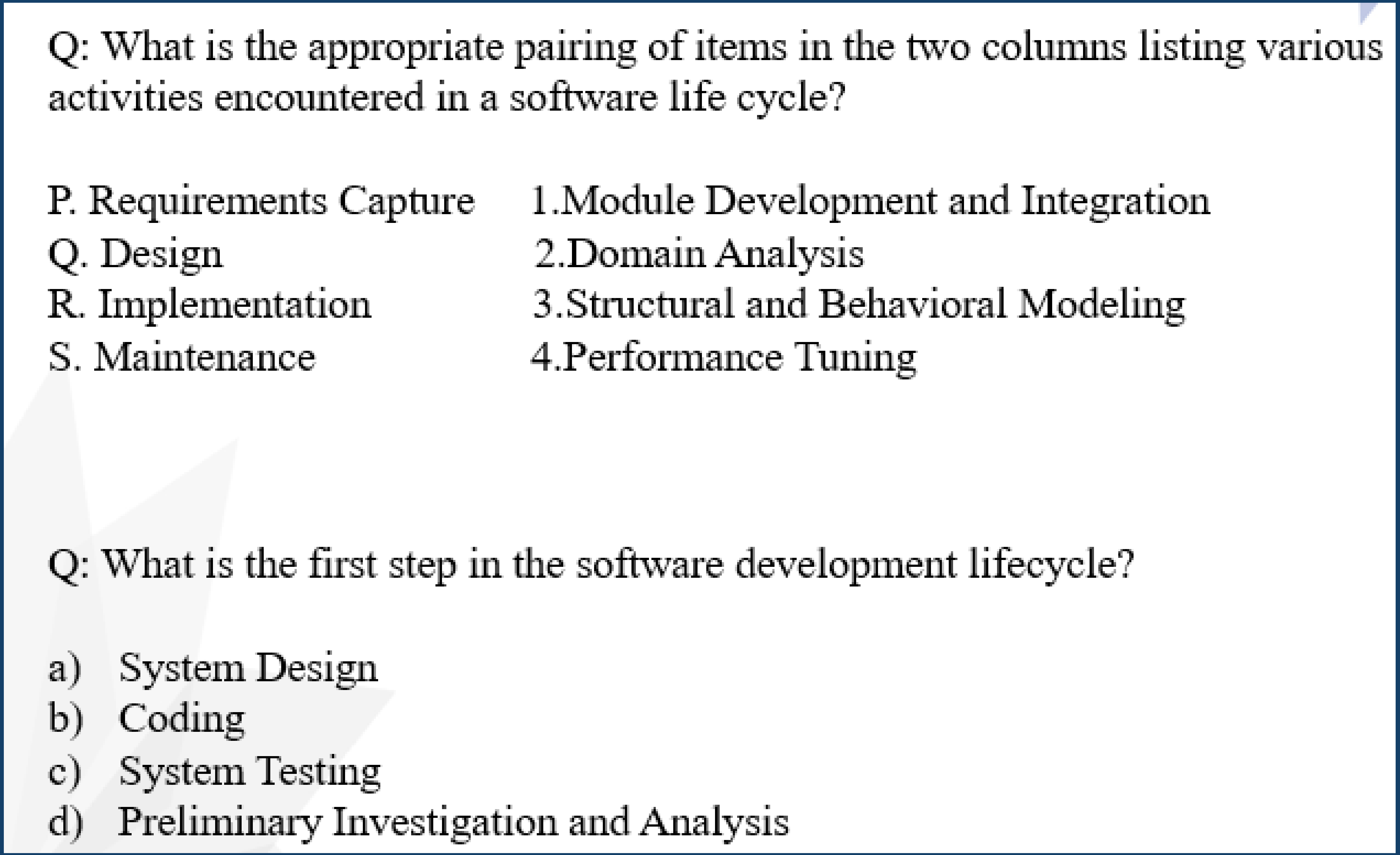
Q3: With reference to process framework, aim of planning activity is / are,

a) To create a project plan this helps to guide the team
b) To describe the tasks and identify the resources required
c) To identify the risks and their flexible solution
d) All of the mentioned above

Q4: This activity combines code generation and the testing,

a) Planning
b) Construction
c) Deployment
d) Modeling

Question?



Q: What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle?

P. Requirements Capture	1.Module Development and Integration
Q. Design	2.Domain Analysis
R. Implementation	3.Structural and Behavioral Modeling
S. Maintenance	4.Performance Tuning

Q: What is the first step in the software development lifecycle?

a) System Design
b) Coding
c) System Testing
d) Preliminary Investigation and Analysis