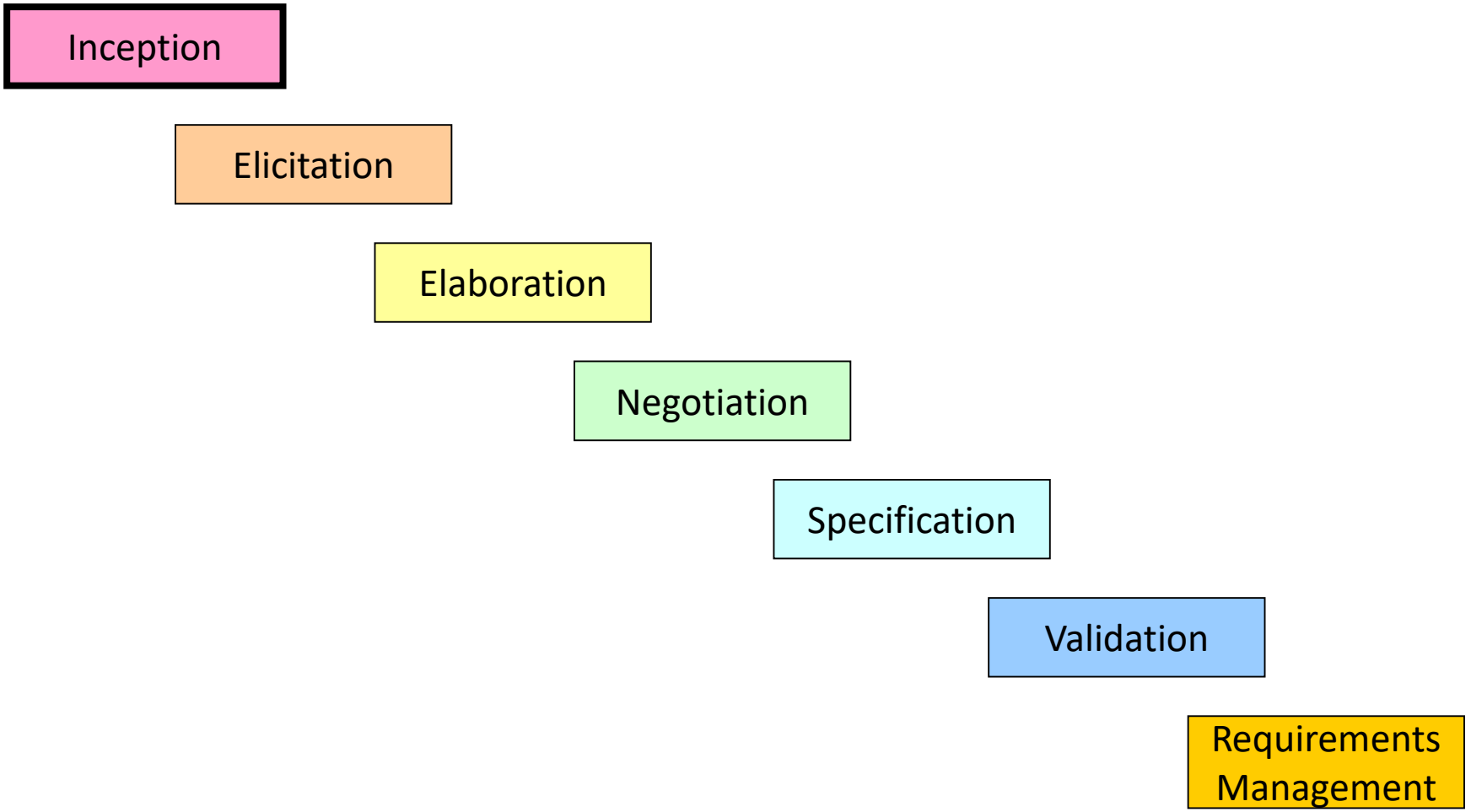
THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Requirement Gathering

Slide Set - 5

Organized & Presented By:
Software Engineering Team CSED
TIET, Patiala

Requirements Gathering
(7 Step Model)



Inception Task

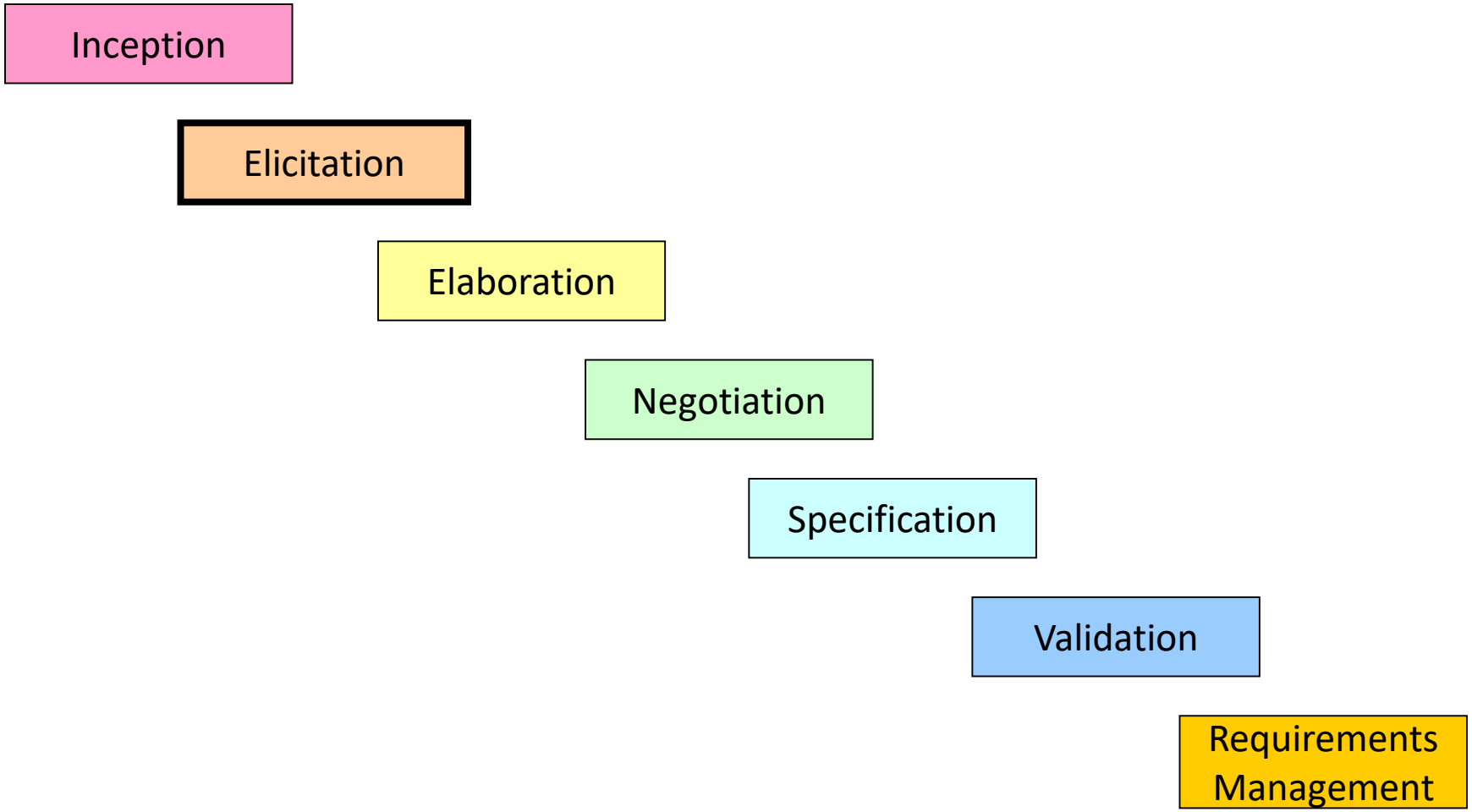
- During inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the problem
  - The people who want a solution
  - The nature of the solution that is desired
  - The effectiveness of preliminary communication and collaboration between the customer and the developer
- Through these questions, the requirements engineer needs to…
  - Identify the stakeholders
  - Recognize multiple viewpoints
  - Work toward collaboration
  - Break the ice and initiate the communication

The First Set of Questions

These questions focus on the customer, other stakeholders, the overall goals, and the benefits

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

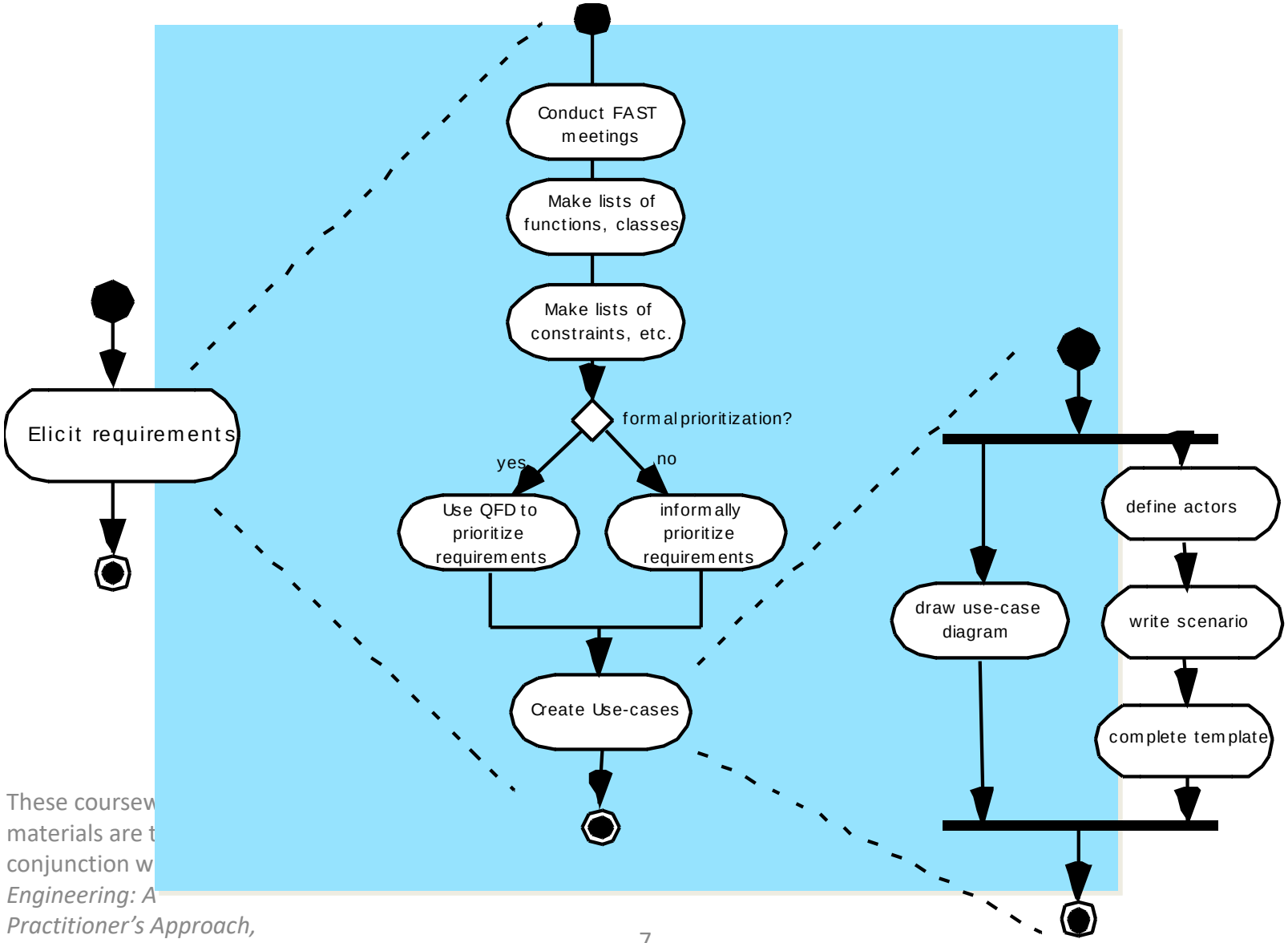
Requirements Gathering (Elicitation)



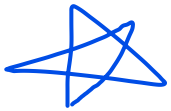
Elicitation Task

- Eliciting requirements is difficult because of
  - Problems of scope in identifying the boundaries of the system or specifying too much technical detail rather than overall system objectives
  - Problems of understanding **what is wanted, what the problem domain is, and what the computing environment can handle** (Information that is believed to be "obvious" is often omitted)
  - Problems of volatility because the **requirements change over time**
- Elicitation may be accomplished through two activities
  - Collaborative requirements gathering
  - Quality function deployment

Eliciting Requirements



These coursew
materials are t
conjunction w
Engineering: A
Practitioner's Approach,
6/e and are provided with
permission by R.S.



Quality Function Deployment (QFD)

Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software.

- **Function deployment** determines the "value" (as perceived by the customer) of each function required of the system
- **Information deployment** identifies data objects and events
- **Task deployment** examines the behavior of the system
- **Value analysis** determines the relative priority of requirements

QFD identifies three types of requirements [Zul92]:

Normal requirements. The objectives and goals that are stated for a product or system during meetings with the customer. If these requirements are present, the customer is satisfied. Examples of normal requirements might be requested types of graphical displays, specific system functions, and defined levels of performance.

Expected requirements. These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause for significant dissatisfaction. Examples of expected requirements are: ease of human/machine interaction, overall operational correctness and reliability, and ease of software installation.

Exciting requirements. These features go beyond the customer's expectations and prove to be very satisfying when present. For example, software for a new mobile phone comes with standard features, but is coupled with a set of unexpected capabilities (e.g., multitouch screen, visual voice mail) that delight every user of the product.

Elicitation Work Products

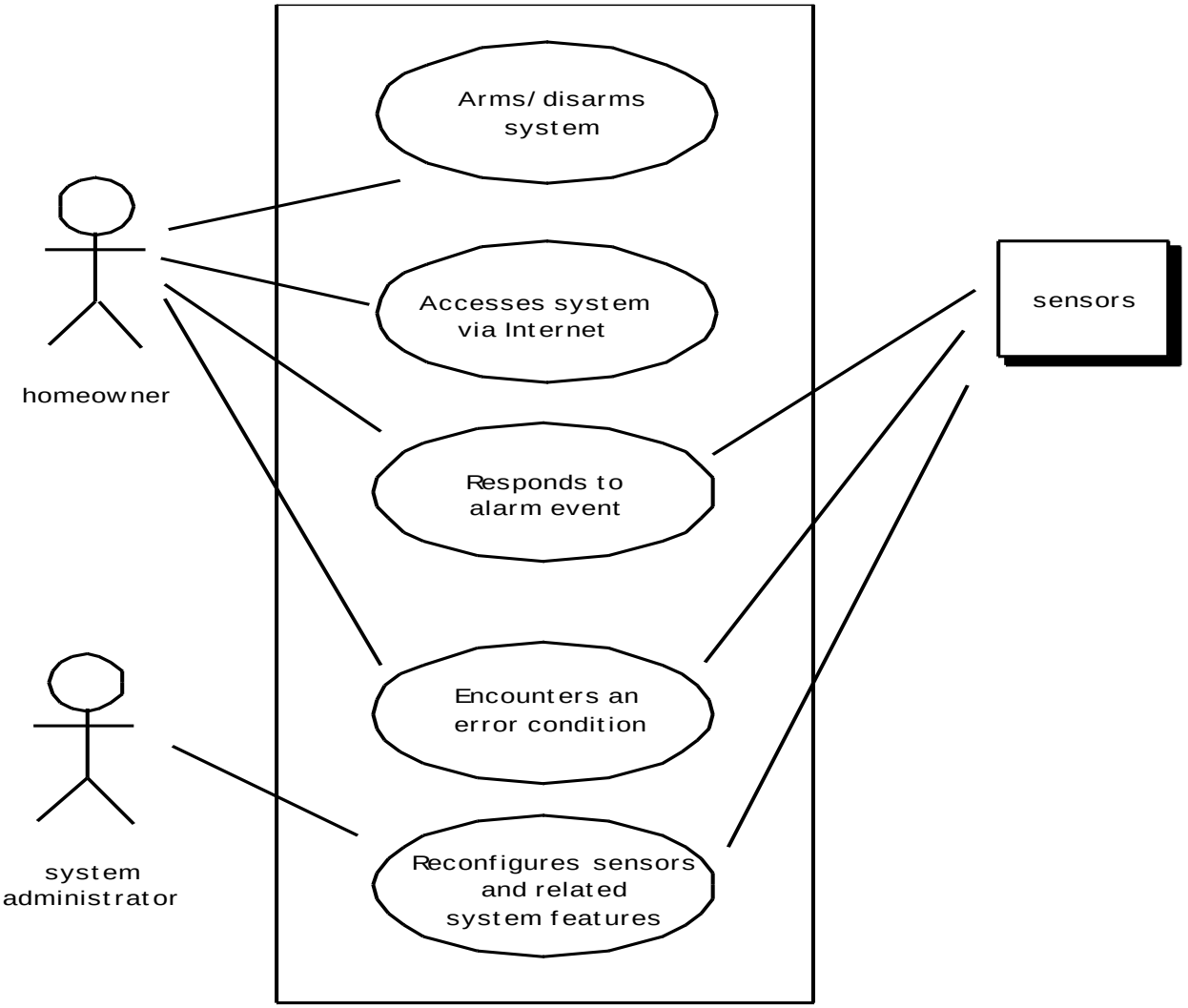
The work products will vary depending on the system, but should include one or more of the following items

- A statement of need and feasibility
- A bounded statement of scope for the system or product
- A list of customers, users, and other stakeholders who participated in requirements elicitation
- A description of the system's technical environment
- A list of requirements (organized by function) and the domain constraints that apply to each
- A set of preliminary usage scenarios (in the form of use cases) that provide insight into the use of the system or product under different operating conditions
- Any prototypes developed to better define requirements

Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor’s goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?
  - What variations in the actor’s interaction are possible?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

Use-Case Diagram

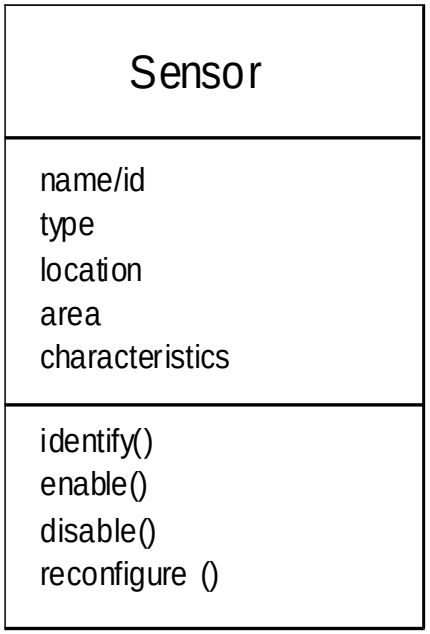


Building the Analysis Model

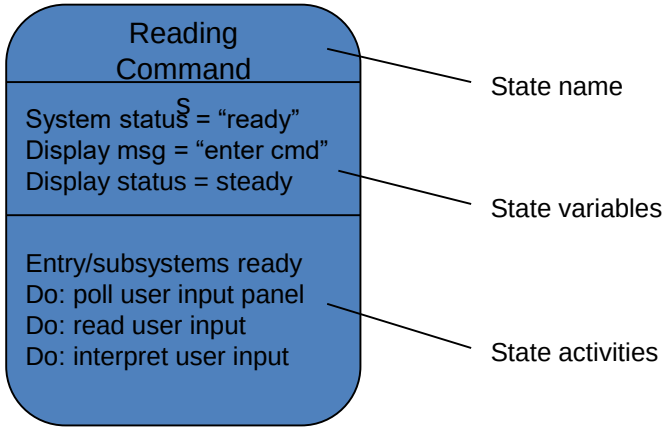
- Elements of the analysis model
  - **Scenario-based elements**
    - Functional—processing narratives for software functions
    - Use-case—descriptions of the interaction between an "actor" and the system
  - **Class-based elements**
    - Implied by scenarios
  - **Behavioral elements**
    - State diagram
  - **Flow-oriented elements**
    - Data flow diagram

Class Diagram

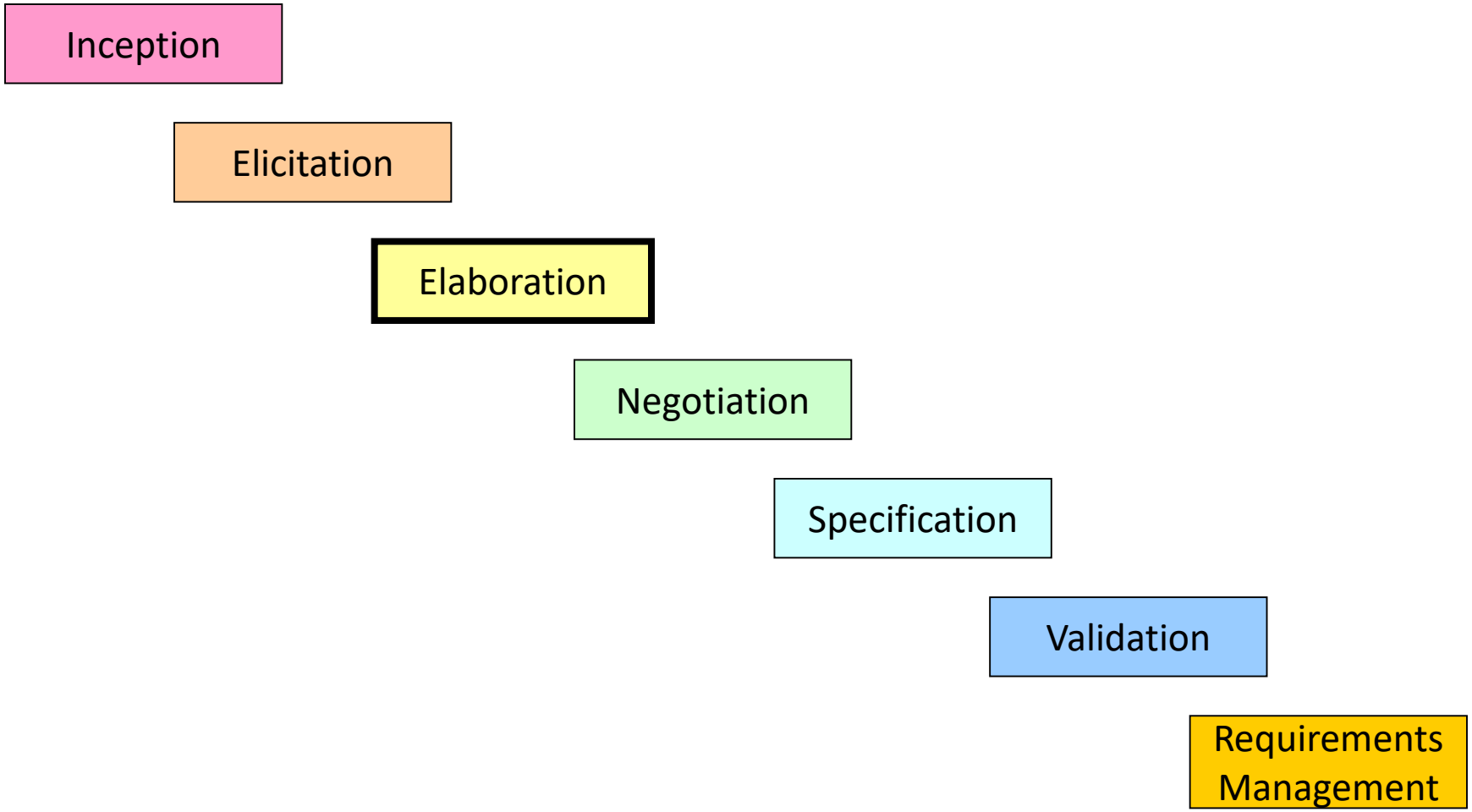
From the *SafeHome* system …



State Diagram



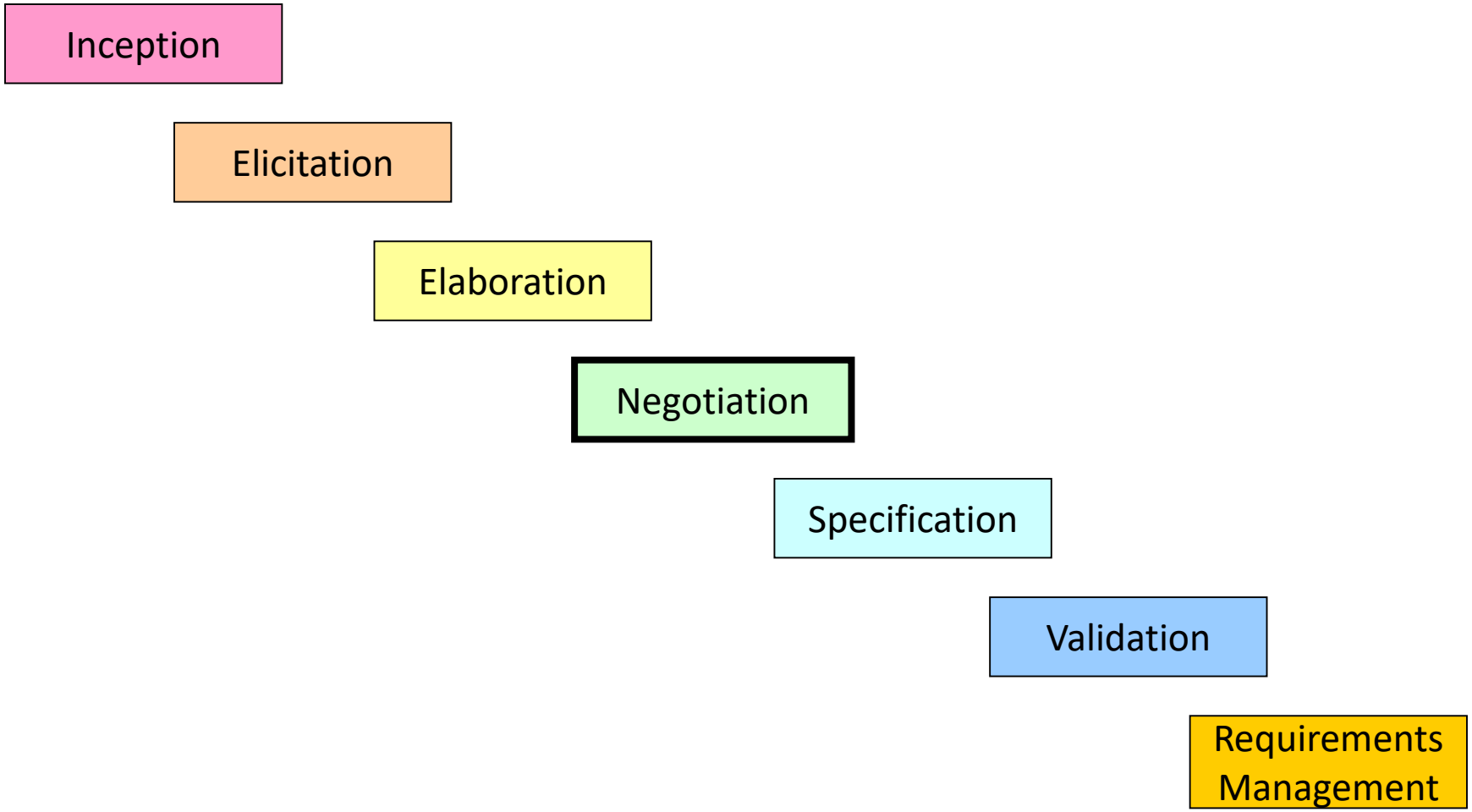
Requirements Gathering (Elaboration)



Elaboration Task

- During elaboration, the **software engineer takes the information obtained during inception and elicitation and begins to expand and refine it**
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- **It is an analysis modeling task**
  - Use cases are developed
  - Domain classes are identified along with their attributes and relationships
  - State machine diagrams are used to capture the life on an object
- **The end result is an analysis model that defines the functional, informational, and behavioral domains of the problem**

Requirements Gathering (Negotiation)



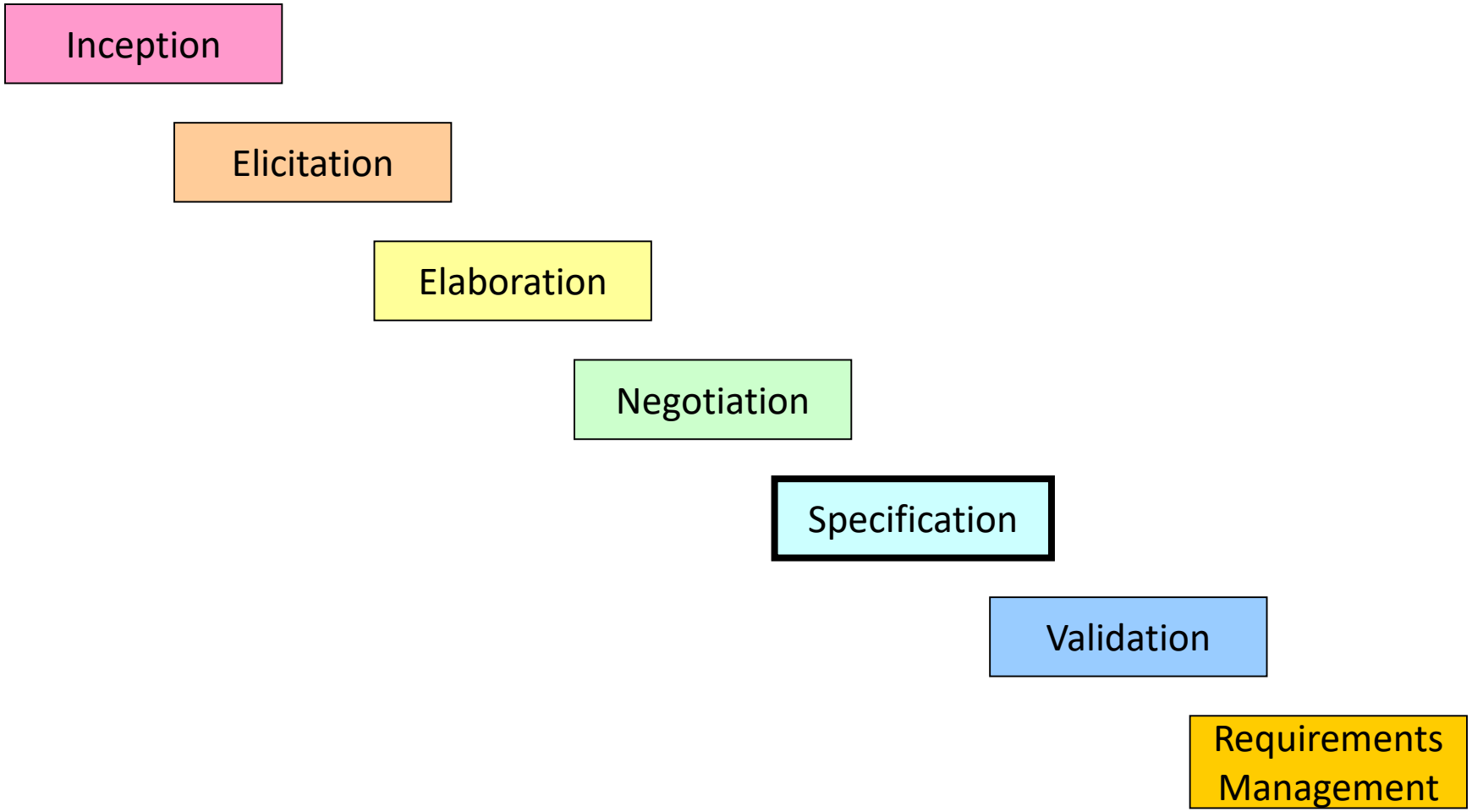
Negotiation Task

- During negotiation, the **software engineer reconciles the conflicts between what the customer wants and what can be achieved given limited business resources**
- **Requirements are ranked (i.e., prioritized)** by the customers, users, and other stakeholders
- **Risks** associated with each requirement are **identified and analyzed**
- Rough guesses of development effort are made and used to assess the **impact of each requirement** on **project cost and delivery time**
- Using an iterative approach, requirements are eliminated, combined and/or modified so that each party achieves some measure of satisfaction

The Art of Negotiation

- Recognize that it is not competition
- Map out a strategy
- Listen actively
- Focus on the other party's interests
- Don't let it get personal
- Be creative
- Be ready to commit

Requirements Gathering (Specification)



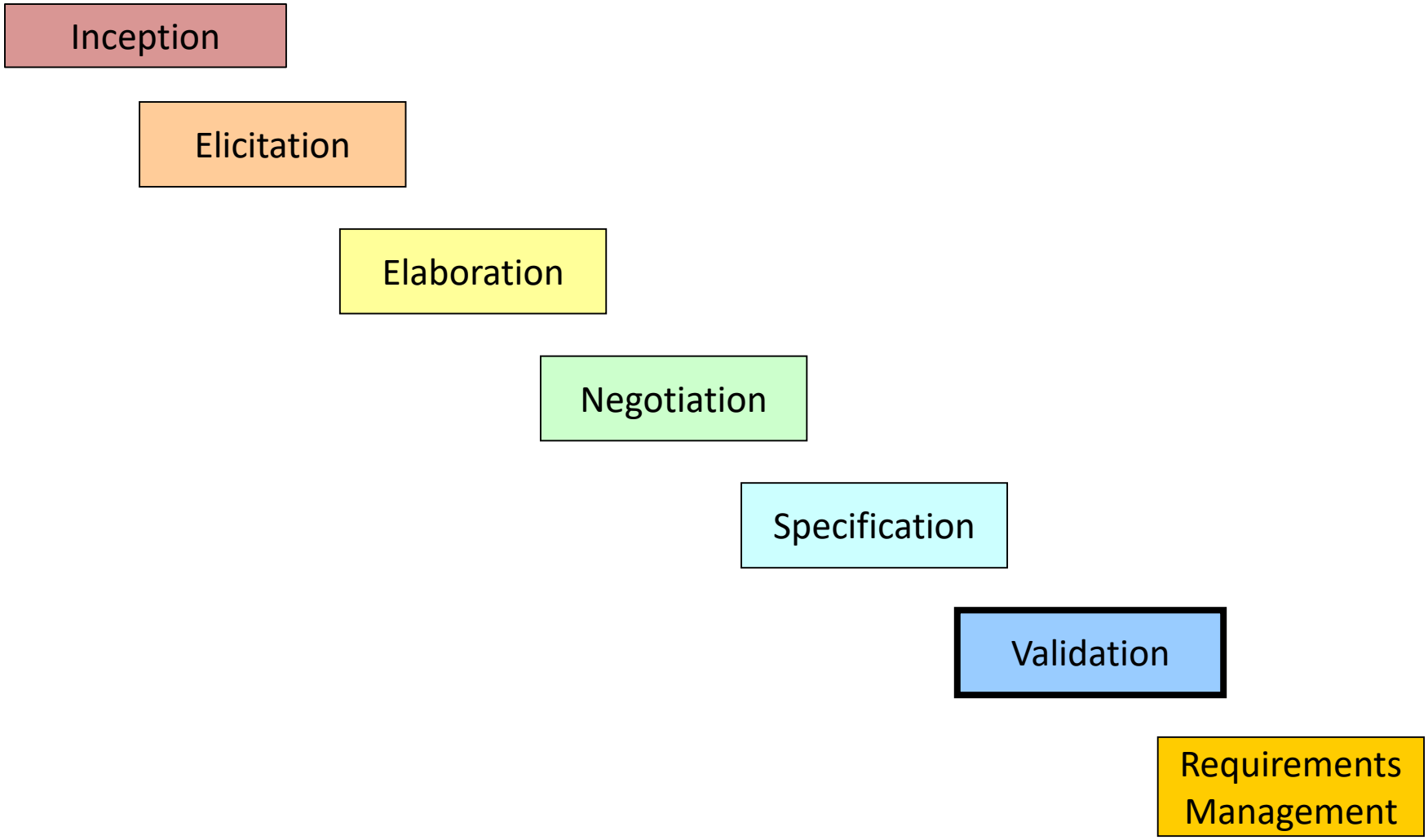
Specification Task

- A specification is **the final work product** produced by the requirements engineer
- It is normally in the form of a **software requirements specification**
- It serves as the foundation for subsequent software engineering activities
- It describes the function and performance of a computer-based system and the constraints that will govern its development
- It **formalizes the informational, functional, and behavioral requirements of the proposed software in both a graphical and textual format**

Typical Contents of a Software Requirements Specification

- Requirements
  - Required states and modes
  - Software requirements grouped by capabilities (i.e., functions, objects)
  - Software external interface requirements
  - Software internal interface requirements
  - Software internal data requirements
  - Other software requirements (safety, security, privacy, environment, hardware, software, communications, quality, personnel, training, logistics, etc.)
  - Design and implementation constraints
- Qualification provisions to ensure each requirement has been met
  - Demonstration, test, analysis, inspection, etc.
- Requirements traceability
  - Trace back to the system or subsystem where each requirement applies

Requirements Gathering (Validation**)**



Validation Task

- During validation, the work products produced as a result of requirements engineering are assessed for quality
- The specification is examined **to ensure** that
  - all software requirements have been stated unambiguously
  - inconsistencies, omissions, and errors have been detected and corrected
  - the work products conform to the standards established for the process, the project, and the product
- The formal technical review serves as the primary requirements validation mechanism
  - Members include software engineers, customers, users, and other stakeholders

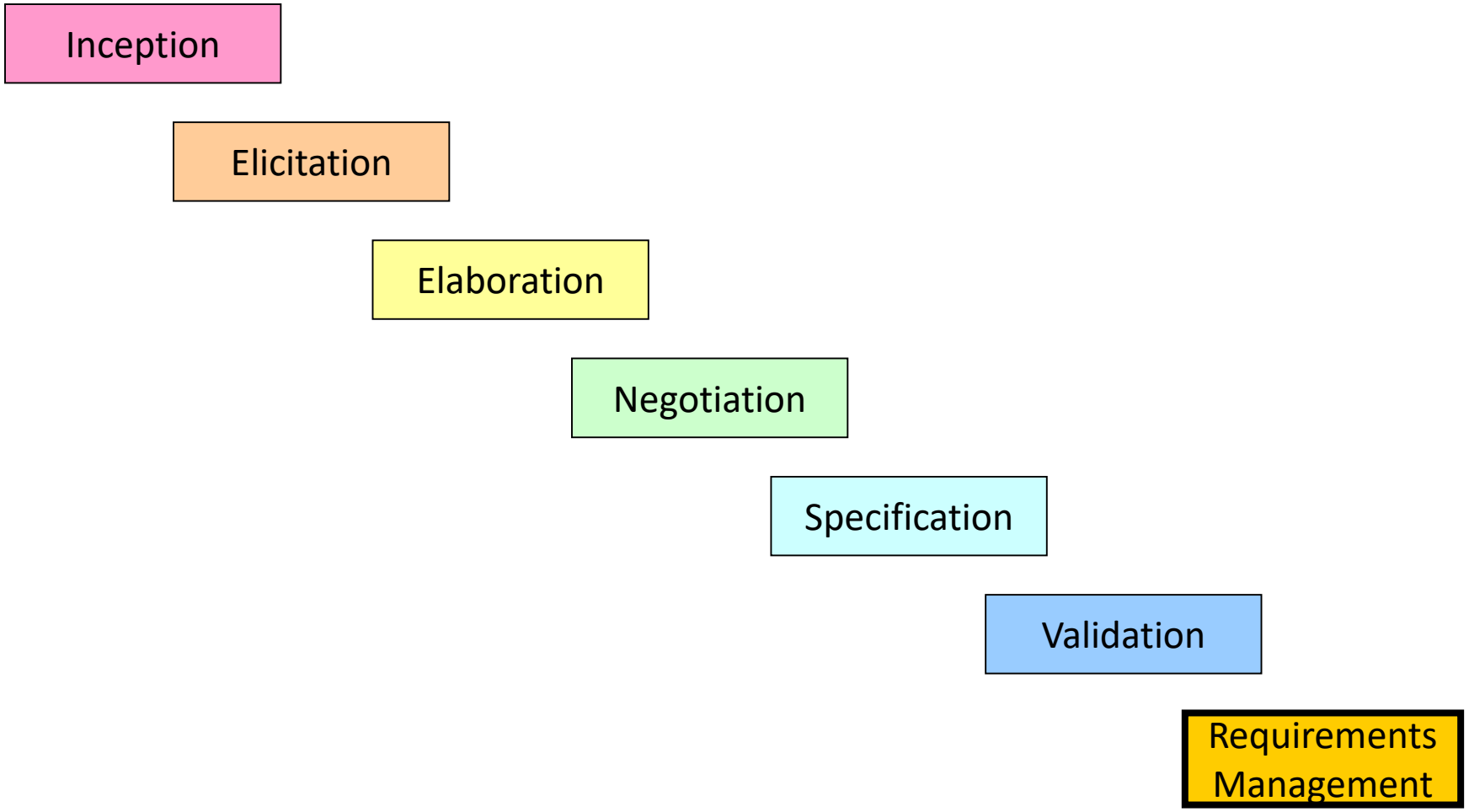
Questions to ask when Validating Requirements

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?

Questions to ask when Validating Requirements (continued)

- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
  - Approaches: Demonstration, actual test, analysis, or inspection
- Does the requirements model properly reflect the information, function, and behavior of the system to be built?
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system?

Requirements Gathering (Requirement Management)



Requirements Management Task

- During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds
- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables
- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements
- A requirements traceability table is also placed at the end of the software requirements specification