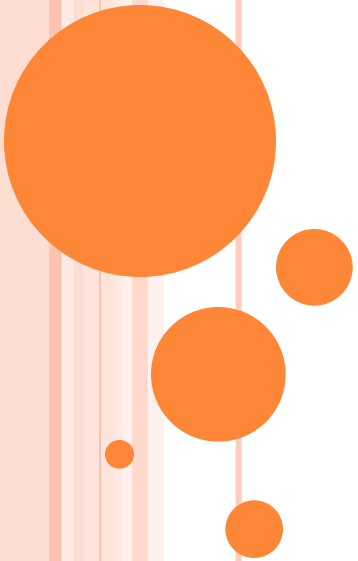
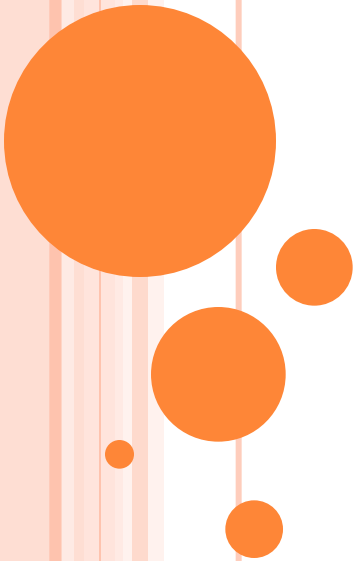
Software Engineering



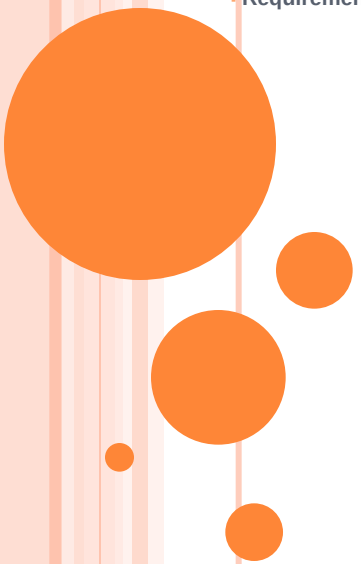
LECTURE-27



Requirements Engineering

Topics Covered

- Problems with requirements practices
- Requirements engineering tasks
- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Requirements management



The Problems with our Requirements Practices

- We have trouble understanding the requirements that we do acquire from the customer
- We often record requirements in a disorganized manner
- We spend far too little time verifying what we do record
- We allow change to control us, rather than establishing mechanisms to control change
- Most importantly, we fail to establish a solid foundation for the system or software that the user wants built

(more on next slide)

The Problems with our Requirements Practices (continued)

- Many software developers argue that
  - Building software is so compelling that we want to jump right in (before having a clear understanding of what is needed)
  - Things will become clear as we build the software
  - Project stakeholders will be able to better understand what they need only after examining early iterations of the software
  - Things change so rapidly that requirements engineering is a waste of time
  - The bottom line is producing a working program and that all else is secondary
- All of these arguments contain some truth, especially for small projects that take less than one month to complete
- However, as software grows in size and complexity, these arguments begin to break down and can lead to a failed software project

A Solution: Requirements Engineering

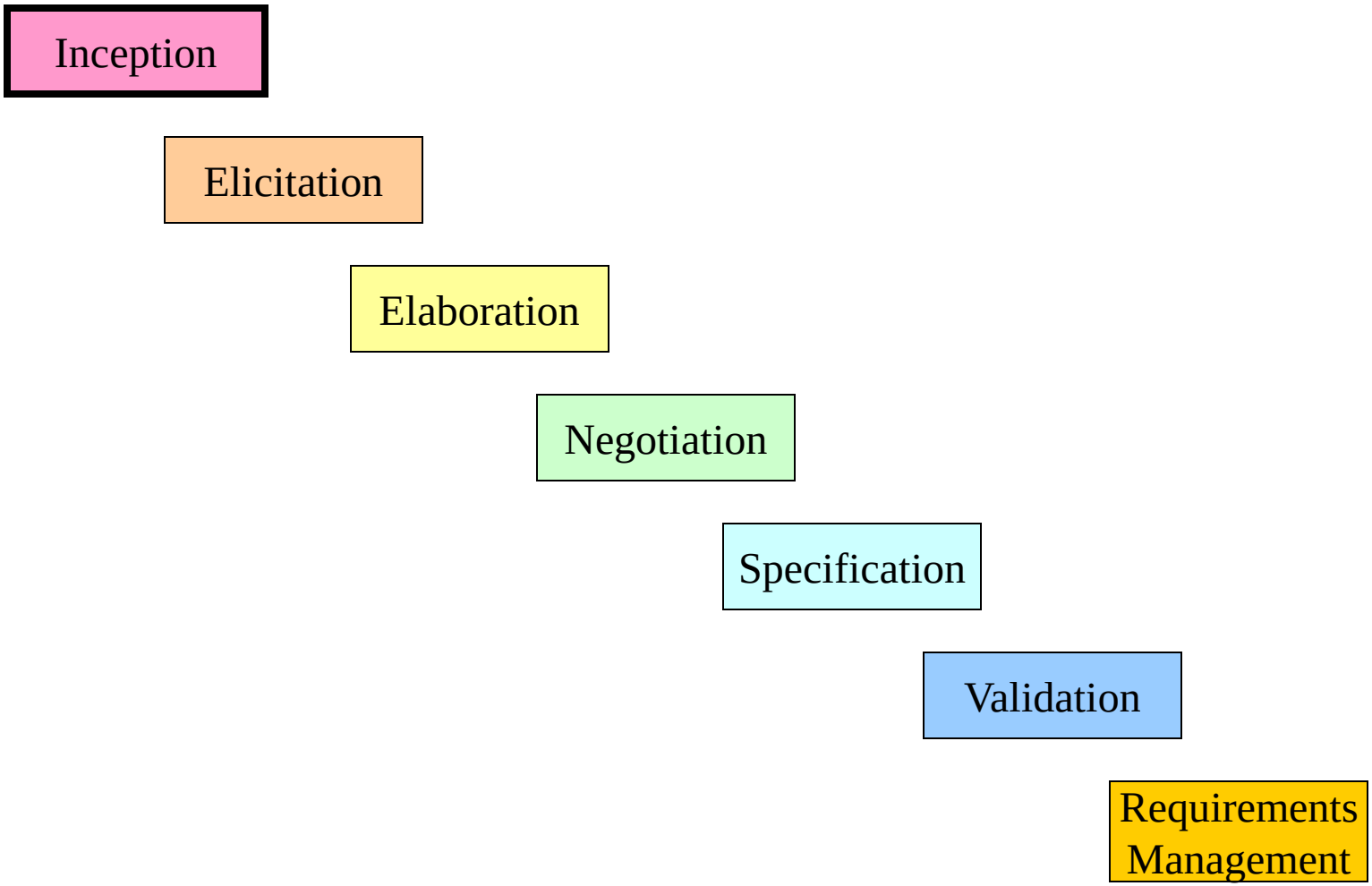
- Begins during the communication activity and continues into the modeling activity
- Builds a bridge from the system requirements into software design and construction
- Allows the requirements engineer to examine
  - the context of the software work to be performed
  - the specific needs that design and construction must address
  - the priorities that guide the order in which work is to be completed
  - the information, function, and behavior that will have a profound impact on the resultant design

Requirements Engineering Tasks

- Seven distinct tasks
  - Inception
  - Elicitation
  - Elaboration
  - Negotiation
  - Specification
  - Validation
  - Requirements Management
- Some of these tasks may occur in parallel and all are adapted to the needs of the project
- All strive to define what the customer wants
- All serve to establish a solid foundation for the design and construction of the software

Example Project: Campus Information Access Kiosk

- Both podium-high and desk-high terminals located throughout the campus in all classroom buildings, admin buildings, labs, and dormitories
- Hand/Palm-login and logout (seamlessly)
- Voice input
- Optional audio/visual or just visual output
- Immediate access to all campus information plus
  - E-mail
  - Cell phone voice messaging


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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?

The Next Set of Questions

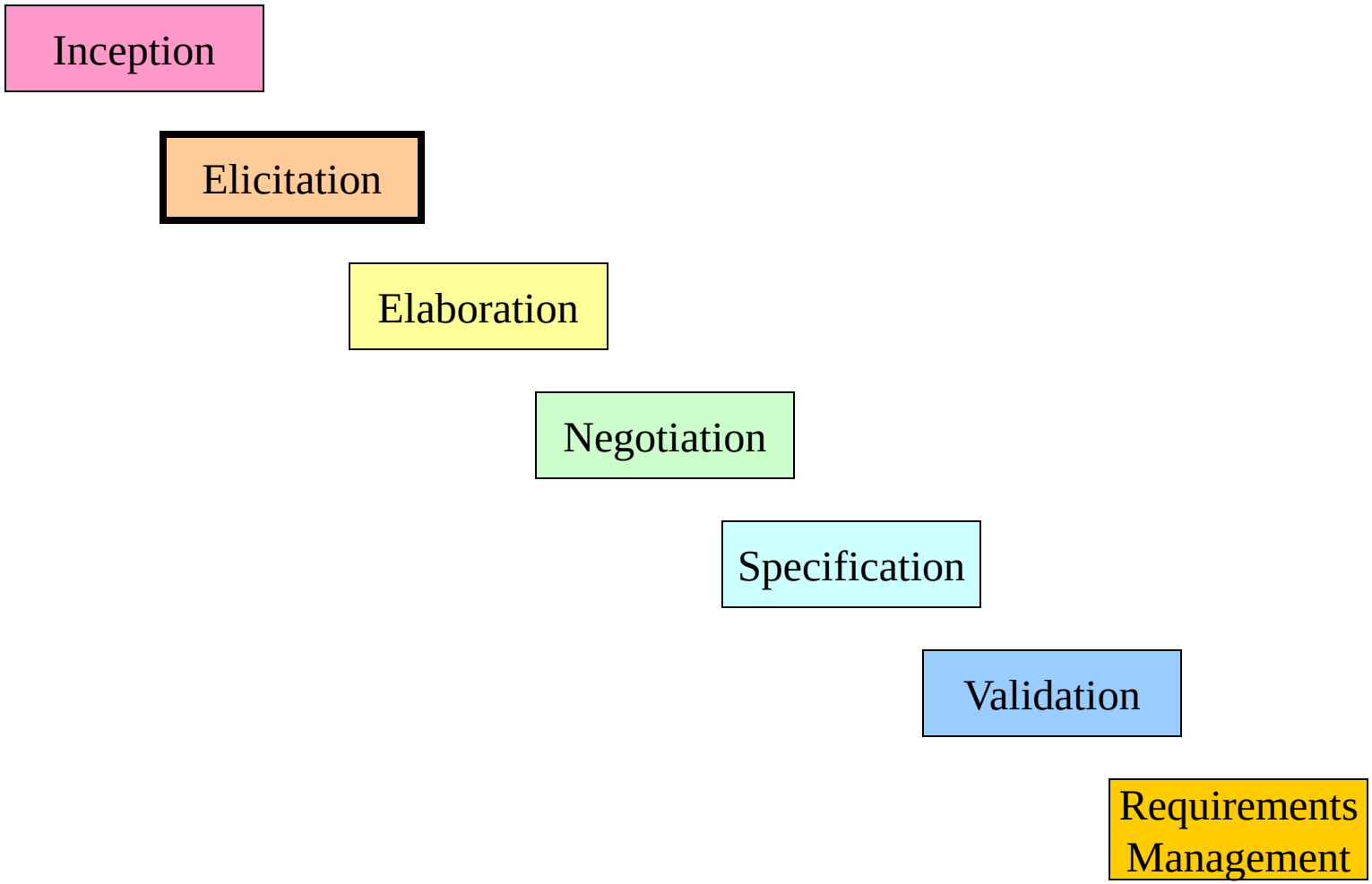
These questions enable the requirements engineer to gain a better understanding of the problem and allow the customer to voice his or her perceptions about a solution

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

The Final Set of Questions

These questions focus on the effectiveness of the communication activity itself

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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

Basic Guidelines of Collaborative Requirements Gathering

- Meetings are conducted and attended by both software engineers, customers, and other interested stakeholders
- Rules for preparation and participation are established
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas
- A "facilitator" (customer, developer, or outsider) controls the meeting
- A "definition mechanism" is used such as work sheets, flip charts, wall stickers, electronic bulletin board, chat room, or some other virtual forum
- The goal is to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements

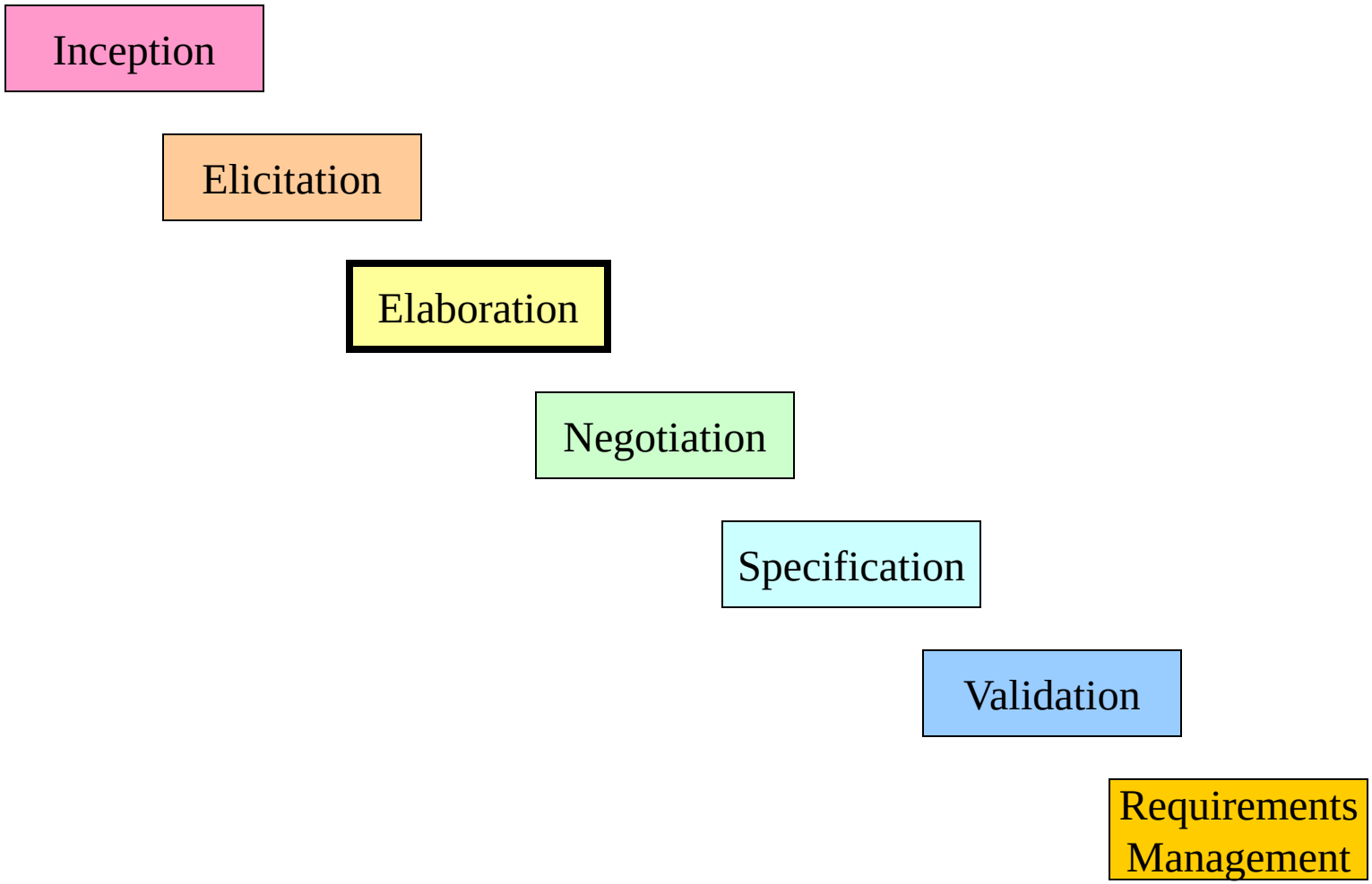
Quality Function Deployment

- This is a technique that translates the needs of the customer into technical requirements for software
- It emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process through functions, information, and tasks
- It identifies three types of requirements
  - Normal requirements: These requirements are the objectives and goals stated for a product or system during meetings with the customer
  - Expected requirements: These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them
  - Exciting requirements: These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present

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


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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

Developing Use Cases

- Step One – Define the set of actors that will be involved in the story
  - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
  - Actors are anything that communicate with the system or product and that are external to the system itself
- Step Two – Develop use cases, where each one answers a set of questions

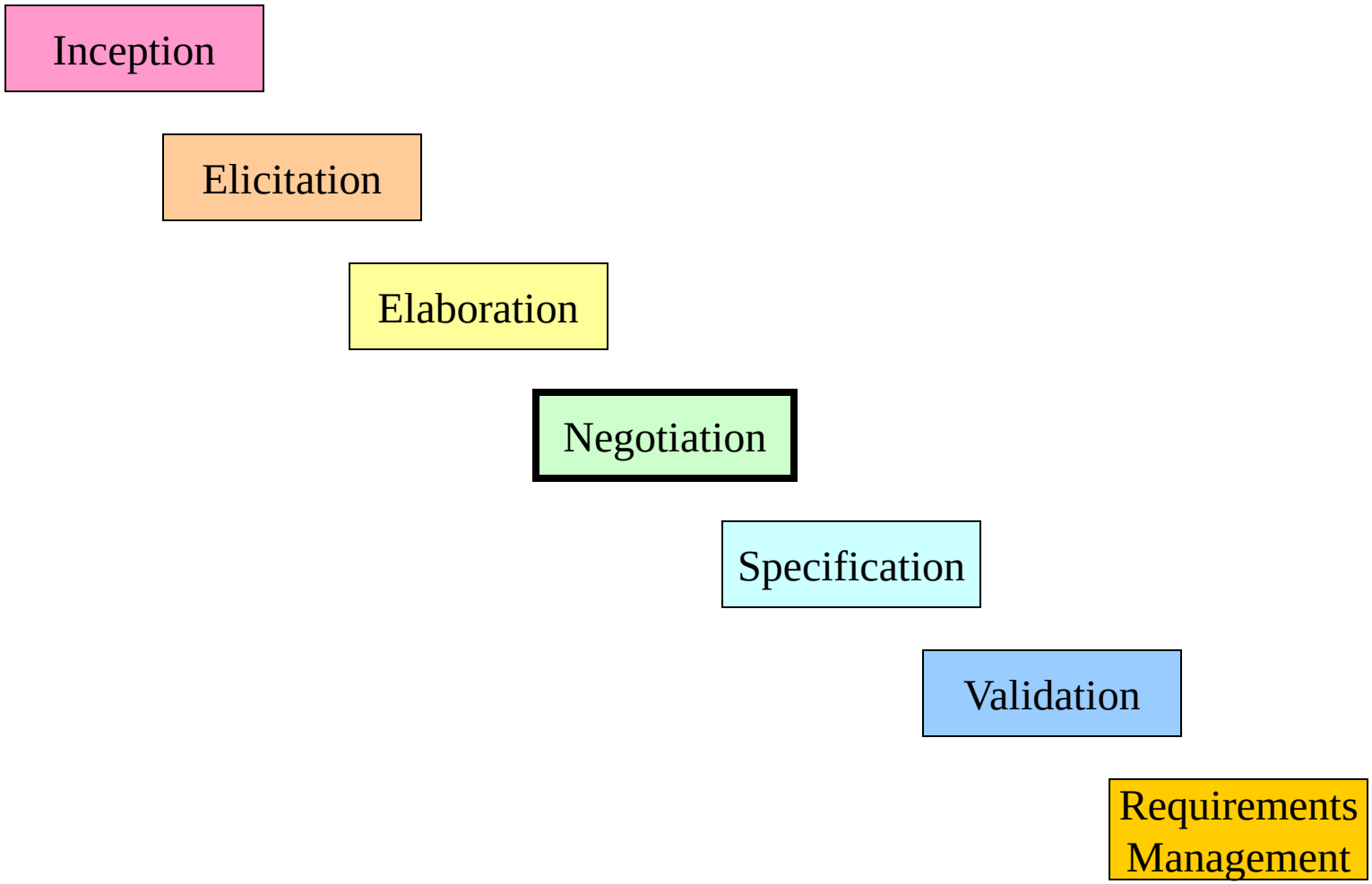
(More on next slide)

Questions Commonly Answered by a Use Case

- Who is the primary actor(s), the secondary actor(s)?
- What are the actor's goals?
- What preconditions should exist before the scenario begins?
- What main tasks or functions are performed by the actor?
- What exceptions might be considered as the scenario 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?

Elements of the Analysis Model

- Scenario-based elements
  - Describe the system from the user's point of view using scenarios that are depicted in use cases and activity diagrams
- Class-based elements
  - Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; they utilize class diagrams to do this
- Behavioral elements
  - Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event; can also be applied to each class in the system
- Flow-oriented elements
  - Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced

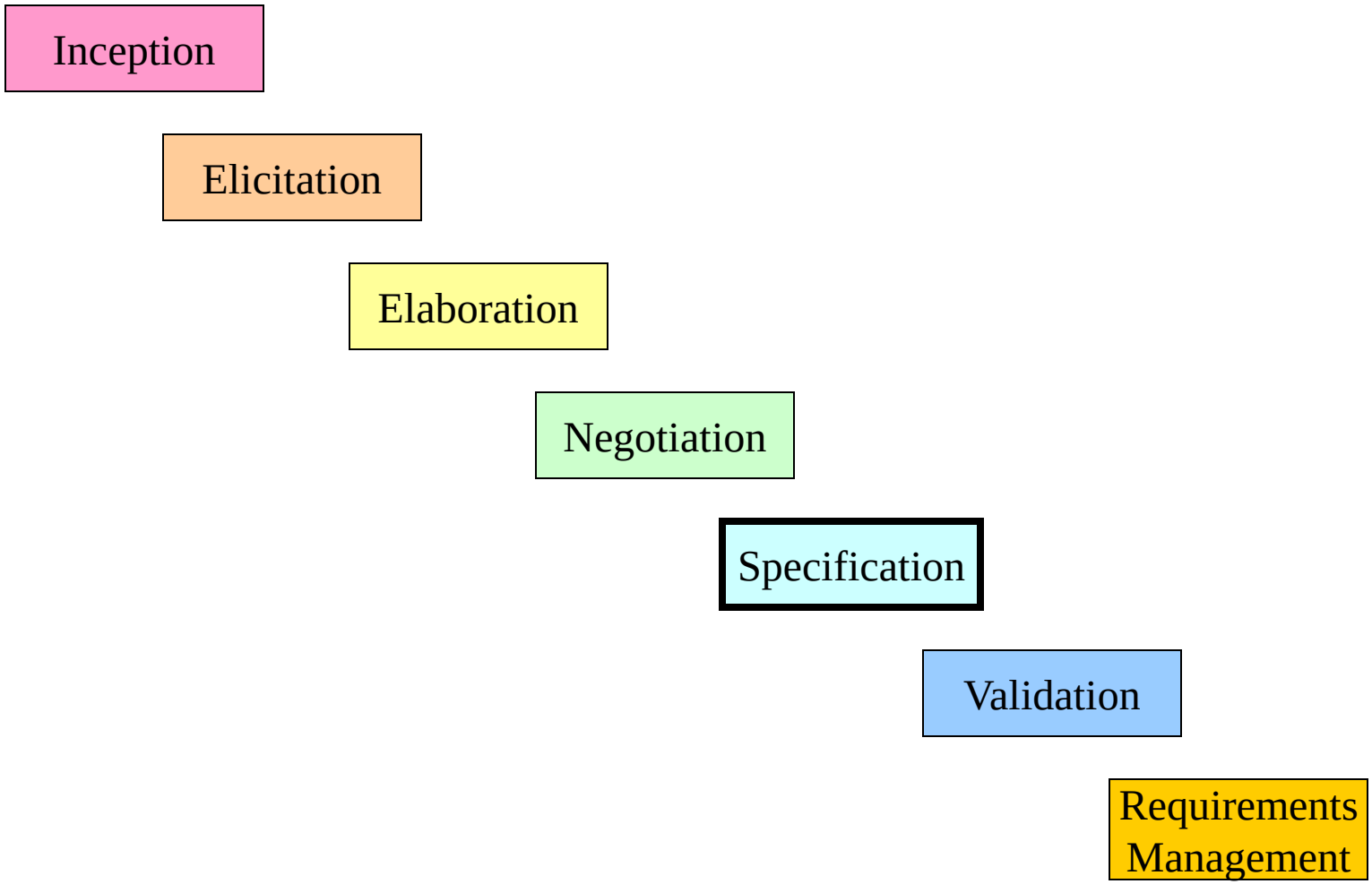

Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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

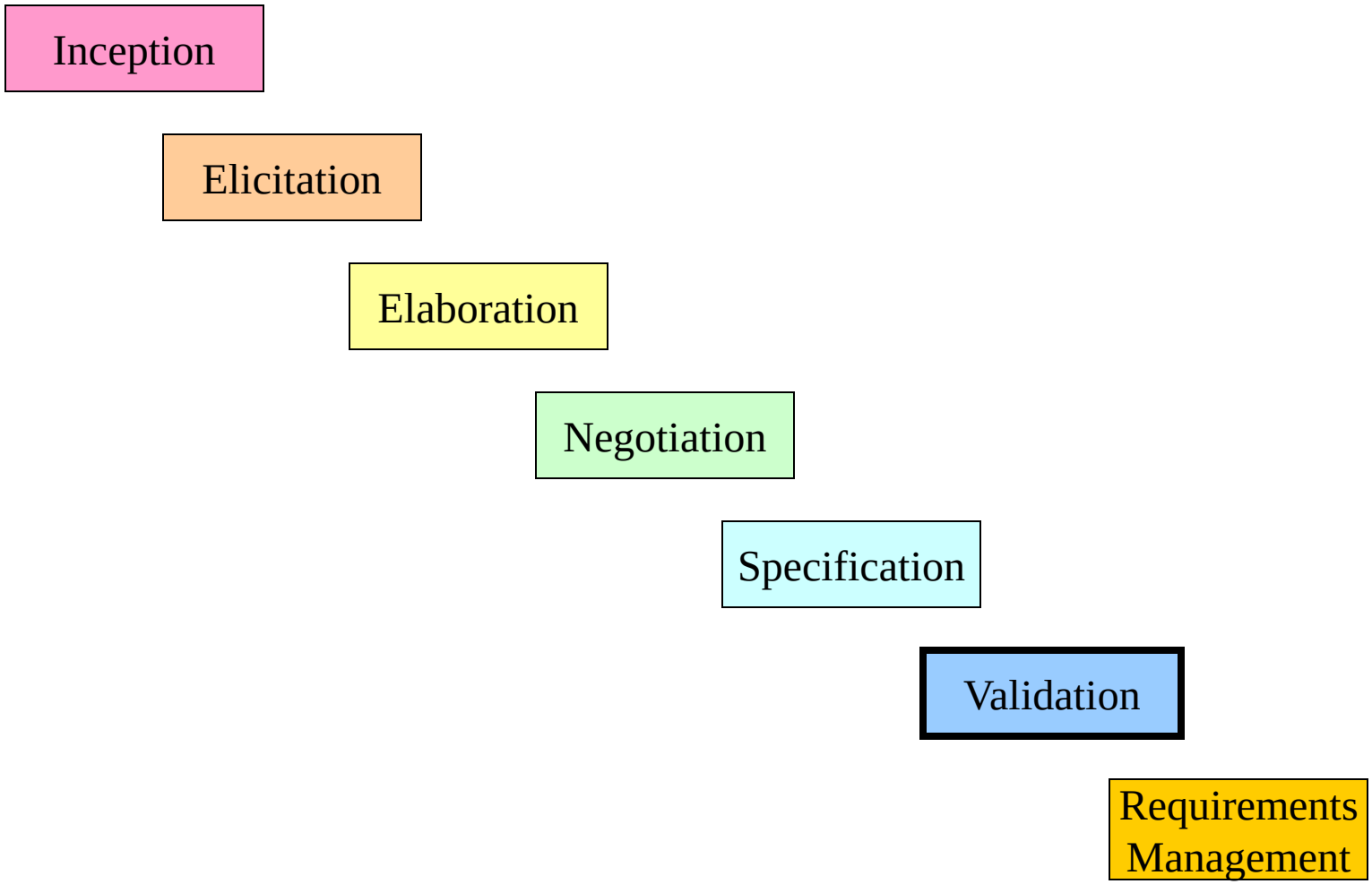

Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

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

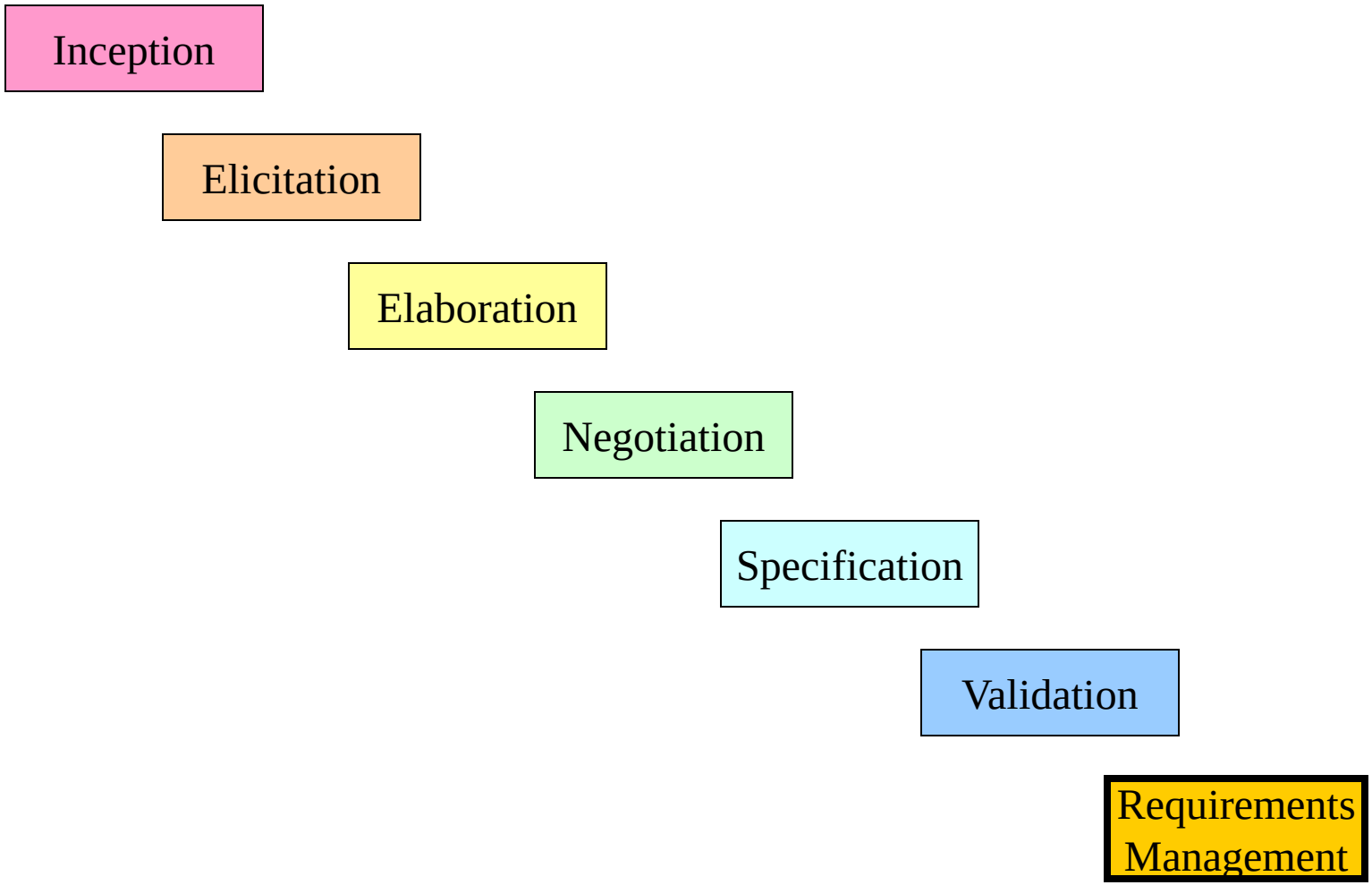
Questions to ask when Validating Requirements

- 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?

(more on next slide)

Questions to ask when Validating Requirements (continued)

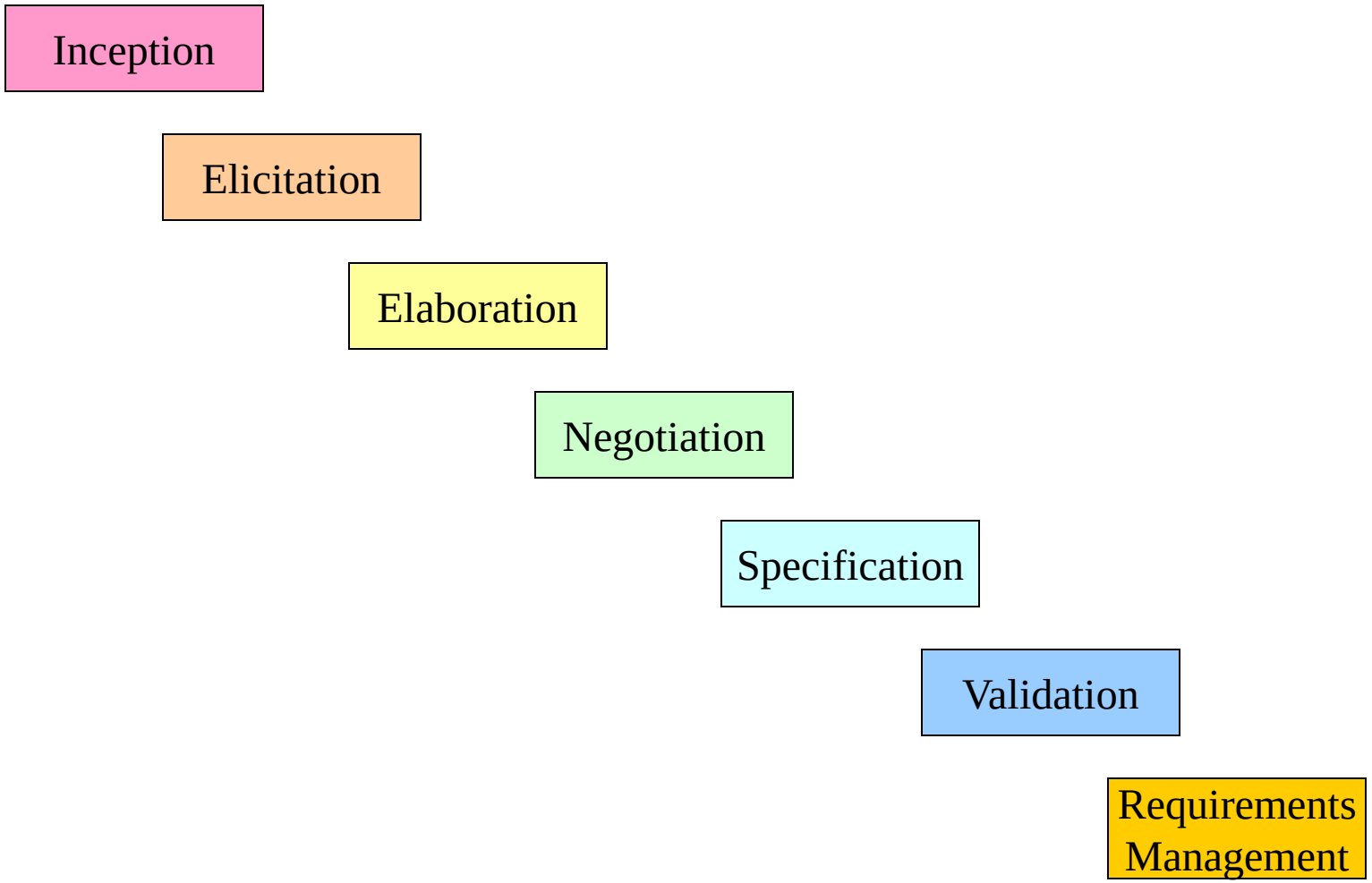
- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
  - Approaches: Demonstration, actual test, analysis, or inspection
- Does the requirements model 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?


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements 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

Summary


Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

☺