SOFTWARE REQUIREMENT ANALYSIS

CMMIDEV /3 SM

Exp. 2018-02-28 / Appraisal #23707

CONTENTS

01 SOFTWARE REQUIREMENTS 3

02 REQUIREMENT ENGINEERING 14

03 REQUIREMENT SPECIFICATION (TEMPLATE + CASE STUDY) 45

04 ANALYSIS MODEL 55

05 STRUCTURED ANALYSIS MODEL 66

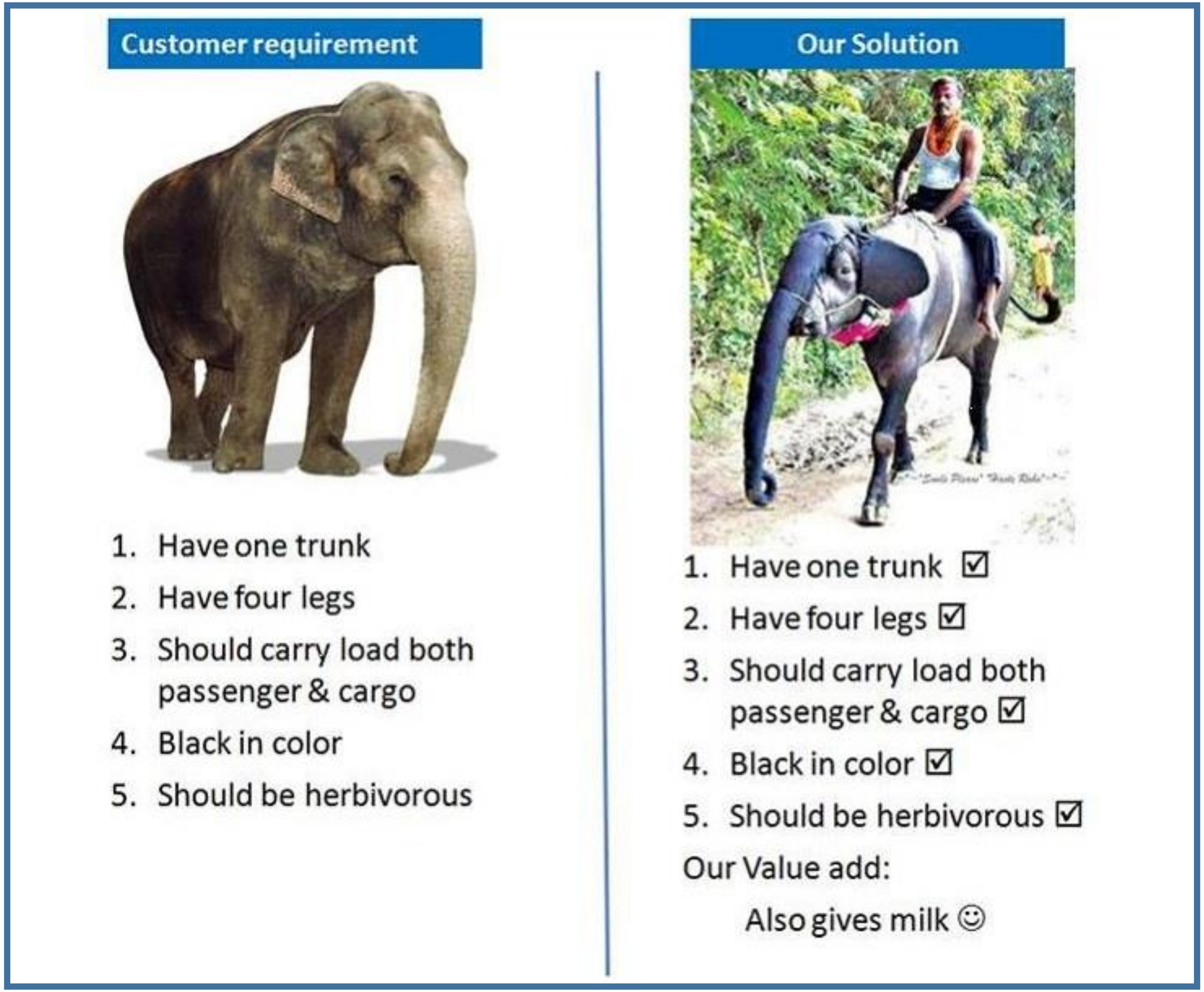
06 OBJECT ORIENTED ANALYSIS MODEL 87

Software Requirements

Why project fails?

- Many projects fail:
  - because we start implementing the system:
  - without determining whether we are building what the customer really wants.

An example



Customer requirement

1. Have one trunk
2. Have four legs
3. Should carry load both passenger & cargo
4. Black in color
5. Should be herbivorous

Our Solution

1. Have one trunk ☑
2. Have four legs ☑
3. Should carry load both passenger & cargo ☑
4. Black in color ☑
5. Should be herbivorous ☑

Our Value add:

Also gives milk ☺

Understand and specifying requirements

- A problem of scale
  - **For small scale:** understand and specifying requirements is easy
  - **For large scale:** very hard; probably the hardest, most problematic and error prone

- The requirements task:
  - **Input:** User needs in minds of people (hopefully)
  - **Output**: precise statement of what the future system will do

Challenges

- Identifying and specifying requirements
  - Necessarily involves people interaction
  - Cannot be automated

Background (1/2)

- What is a Requirement?
  - A condition or capability that must be possessed by a system (IEEE)
- What is the work product of the Req. phase ?
  - A software requirements specification (SRS) document
- What is an SRS ?
  - A complete specification of what the proposed system should do!

Background (2/2)

- Requirements understanding is hard
  - Visualizing a future system is difficult
  - Capability of the future system not clear, hence needs not clear
  - Requirements change with time
  - …
- Essential to do a proper analysis and specification of requirements

Purpose of SRS document

- SRS establishes basis of agreement between the user and the supplier.
  - Users needs have to be satisfied, but user may not understand software
  - Developers will develop the system, but may not know about problem domain

- SRS is
  - the medium to bridge the communications gap, and
  - specifies user needs in a manner both can understand

Need for SRS (1/3)

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - The requirement process helps clarify needs

- SRS provides a reference for validation of the final product
  - Clear understanding about what is expected.
  - Validation - “SW satisfies the SRS”

Need for SRS (2/3)

- High quality SRS essential for high Quality SW
  - Requirement errors get manifested in final SW
  - To satisfy the quality objective, must begin with high quality SRS

  - Requirements defects cause later problems
    - In one study, 25% of all defects found after user testing

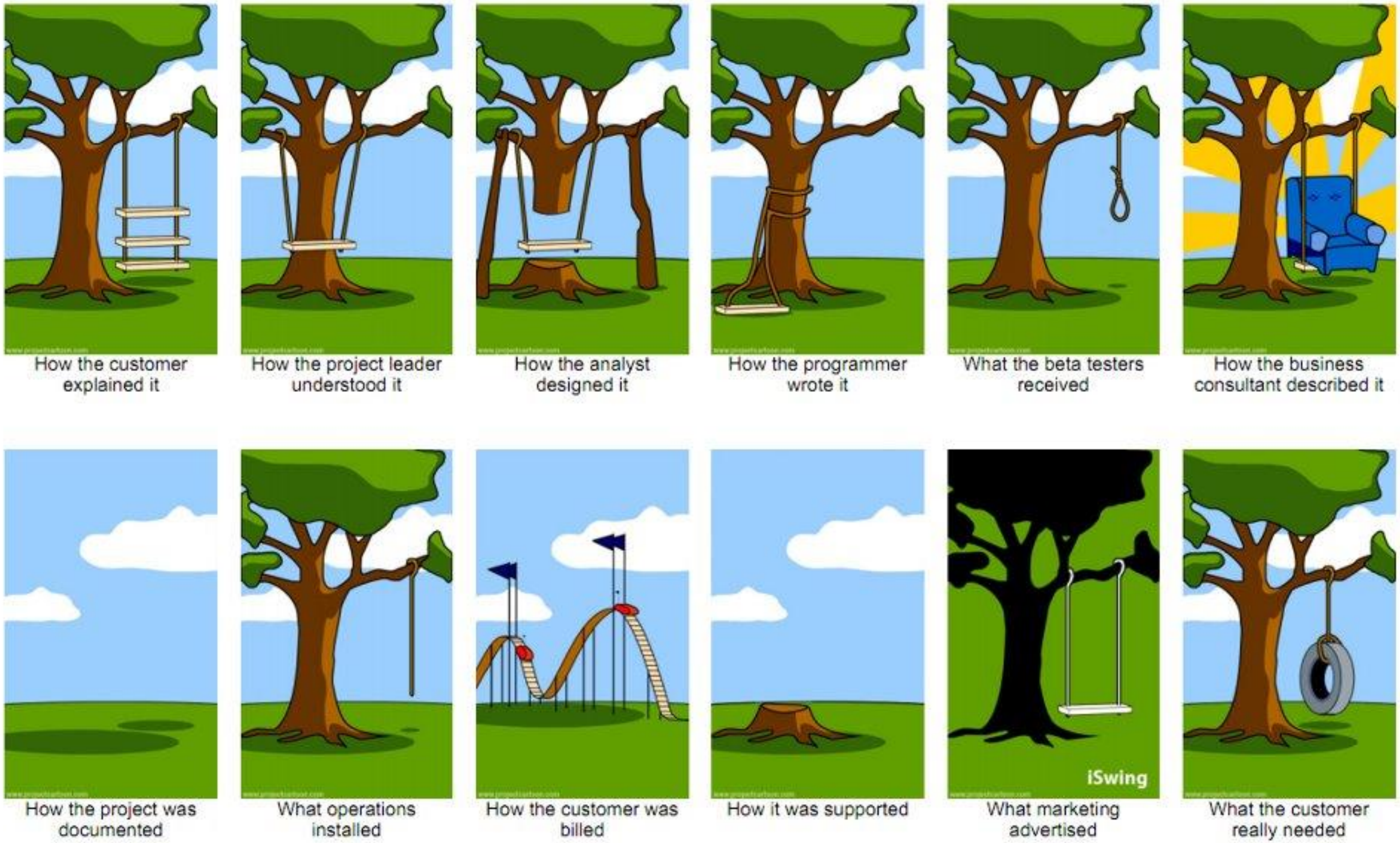
Need for SRS (3/3)

- Good SRS reduces the development cost
  - SRS errors are expensive to fix later
  - Req. changes can cost a lot (up to 40%)
  - Good SRS can minimize changes and errors
  - Substantial savings; extra effort spent during req. saves multiple times that effort

- An Example
  - Cost of fixing errors in req. , design, coding, acceptance testing and operation are 2, 5, 15, 50, 150 person-months

Requirement Engineering

The Problems with our Requirements Practices (1/3)


How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

The Problems with our Requirements Practices (2/3)

- 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

The Problems with our Requirements Practices (3/3)

- 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

A Solution: Requirements Engineering

- **The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed**
- 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 he 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

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

Inception Task

First set of questions	Next set of questions	Final set of questions
These questions focus on the customer, other stakeholders, the overall goals, and the benefits	**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**	**These questions focus on the effectiveness of the communication activity itself**
• 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?	• 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?	• 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?

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

Elaboration Task

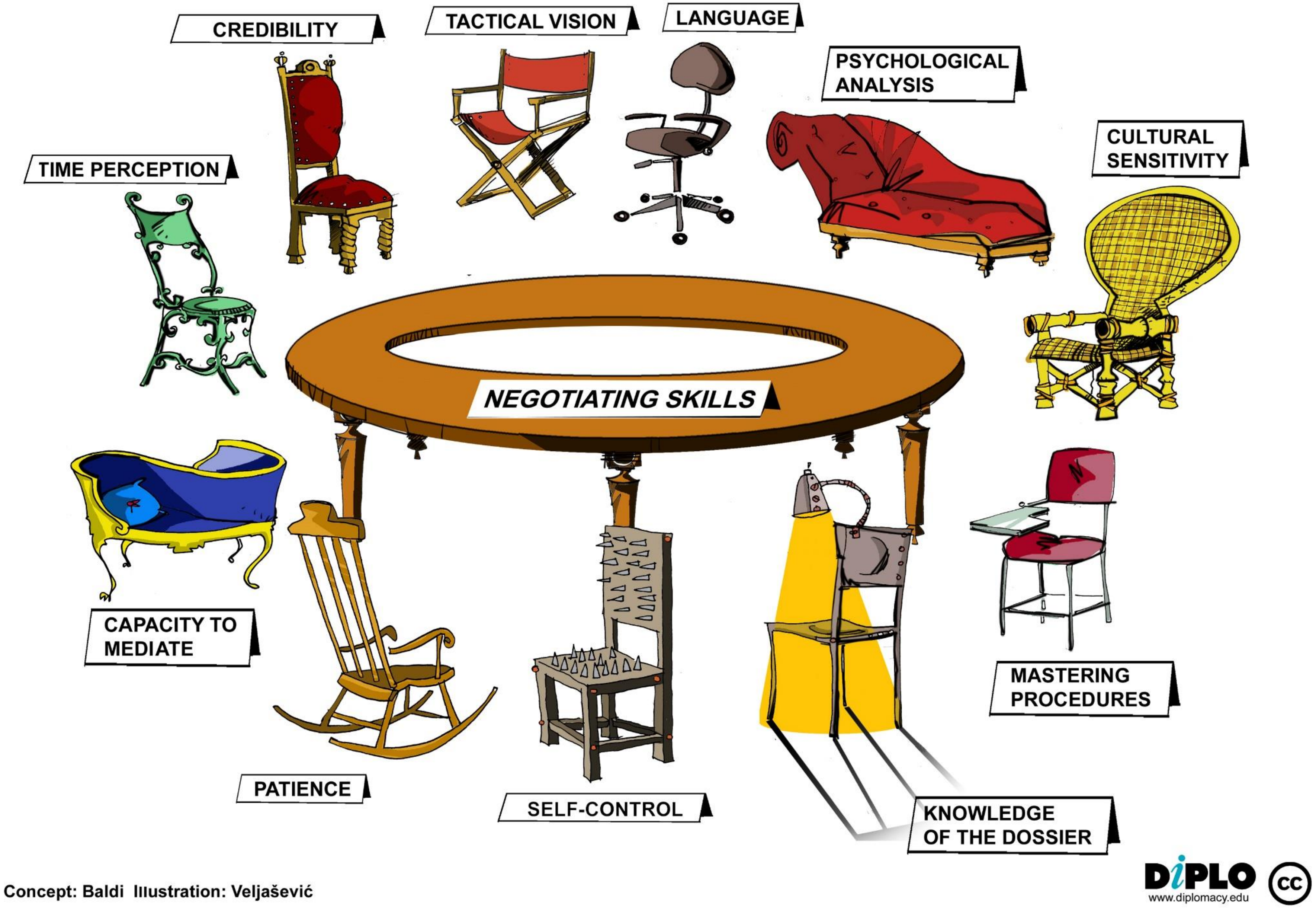
- During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- It is an analysis modeling task
  - Use cases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

Analysis Model
(Will be covered from slide- 55)

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

Negotiation skills



Concept: Baldi Illustration: Veljašević

5. Requirement specification

- Final output of requirements task is the SRS
- Why are DFDs, OO models, etc not SRS ?
  - SRS focuses on external behavior, while modeling focuses on problem structure
  - UI etc. not modeled, but have to be in SRS
  - Error handling, constraints etc. also needed in SRS
- Transition from analysis to specification is not straight forward
- Knowledge about the system acquired in analysis used in specification

5. Characteristics of an SRS

- Correct
- Complete
- Unambiguous
- Consistent
- Verifiable
- Traceable
- Modifiable
- Ranked for importance and/or stability

5. Characteristics

- Correctness
  - Each requirement accurately represents some desired feature in the final system
- Completeness
  - All desired features/characteristics specified
  - Hardest to satisfy
  - Completeness and correctness strongly related
- Unambiguous
  - Each req has exactly one meaning
  - Without this errors will creep in
  - Important as natural languages often used

5. Characteristics

- Verifiability
  - There must exist a cost effective way of checking if SW satisfies requirements
- Consistent
  - Two requirements don’t contradict each other
- Traceable
  - The origin of the req, and how the req relates to software elements can be determined
- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements

5. Components of an SRS

- What should an SRS contain ?
  - Clarifying this will help ensure completeness

- An SRS must specify requirements on
  - External interfaces
  - Functionality
  - Performance
  - Design constraints
  - System attributes

5. External Interface

- All interactions of the software with people, hardware, and SW
- User interface most important
- General requirements of “friendliness” should be avoided
- These should also be verifiable

5. Functional requirements

- Heart of the SRS document; this forms the bulk of the specs
- Specifies all the functionality that the system should support
- Outputs for the given inputs and the relationship between them
- All operations the system is to do
- Must specify behavior for invalid inputs too

5. Performance Requirements

- All the performance constraints on the software system
- Generally on response time , throughput etc => dynamic
- Capacity requirements => static
- Must be in measurable terms (verifiability)
  - E.g. response time should be <1sec in 99% of the time

5. Design Constraints

- Factors in the client environment that restrict the choices
- Some such restrictions
  - Standard compliance and compatibility with other systems
  - Hardware Limitations

5. System attributes

- The requirements in this section specify the
  - Required reliability
  - Availability
  - Security and maintainability of the software system.

5. Specification Language

- Language should support desired characteristics of the SRS
- Formal languages are precise and unambiguous but hard
- Natural languages mostly used, with some structure for the document
- Formal languages used for special features or in highly critical systems

5. Structure of an SRS

- 1. Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Definitions
  - 1.4 References
  - 1.5 Overview
- 2. General Description
  - 2.1 Product Perspective
  - 2.2 Product Functions
  - 2.3 User Characteristics
  - 2.4 General Constraints
  - 2.5 Assumptions and Dependencies

5. Structure of an SRS

- 3. Specific Requirements
  - 3.1 Functional Requirements
    - 3.1.1 Func Req 1
    - 3.1.1.1 Introduction
    - 3.1.1.2 Inputs
    - 3.1.1.3 Processing
    - 3.1.1.4 Outputs
    - 3.1.2 Func Req 2
    - …
  - 3.2 External Interface Requirements
    - 3.2.1 User Interface
    - 3.2.2 Hardware Interfaces
    - 3.2.3 Software Interfaces
    - 3.2.4 Communication Interfaces
  - 3.3 Performance Requirements
  - 3.4 Design Constraints
    - 3.4.1 Standards Compliance
    - 3.4.2 Hardware Limitations
  - 3.5 Software System Attributes
    - 3.5.1 Security
    - 3.5.2 Maintainability
  - 3.6 Other Requirements
    - 3.6.1 Database

Requirement Specification (Template and Case Study)

[BJIT] SRS Template

Acrobat Document

SRS Case Study2

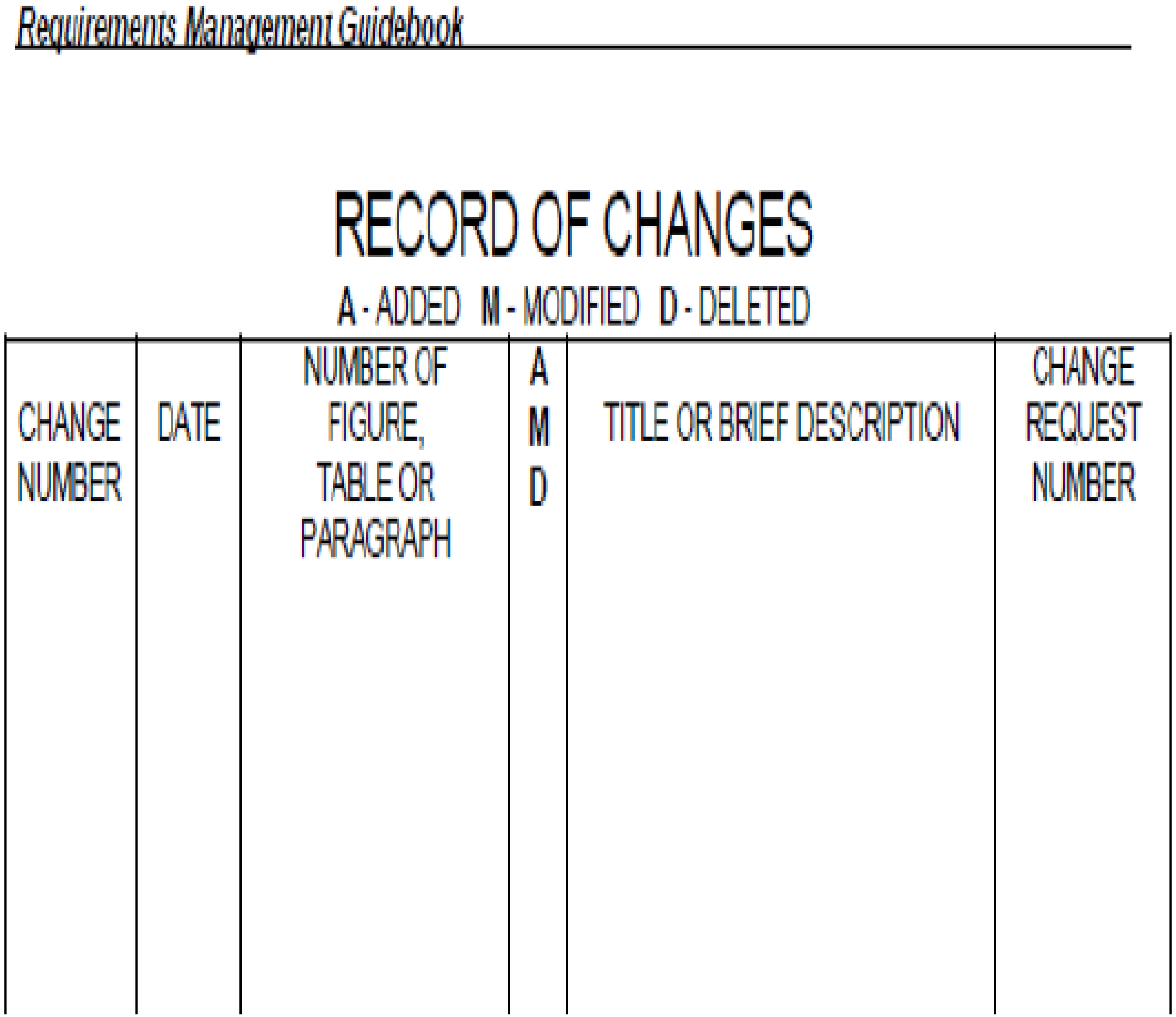
6. Validation Task

- 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
- 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?

7. 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

Sample requirement management table



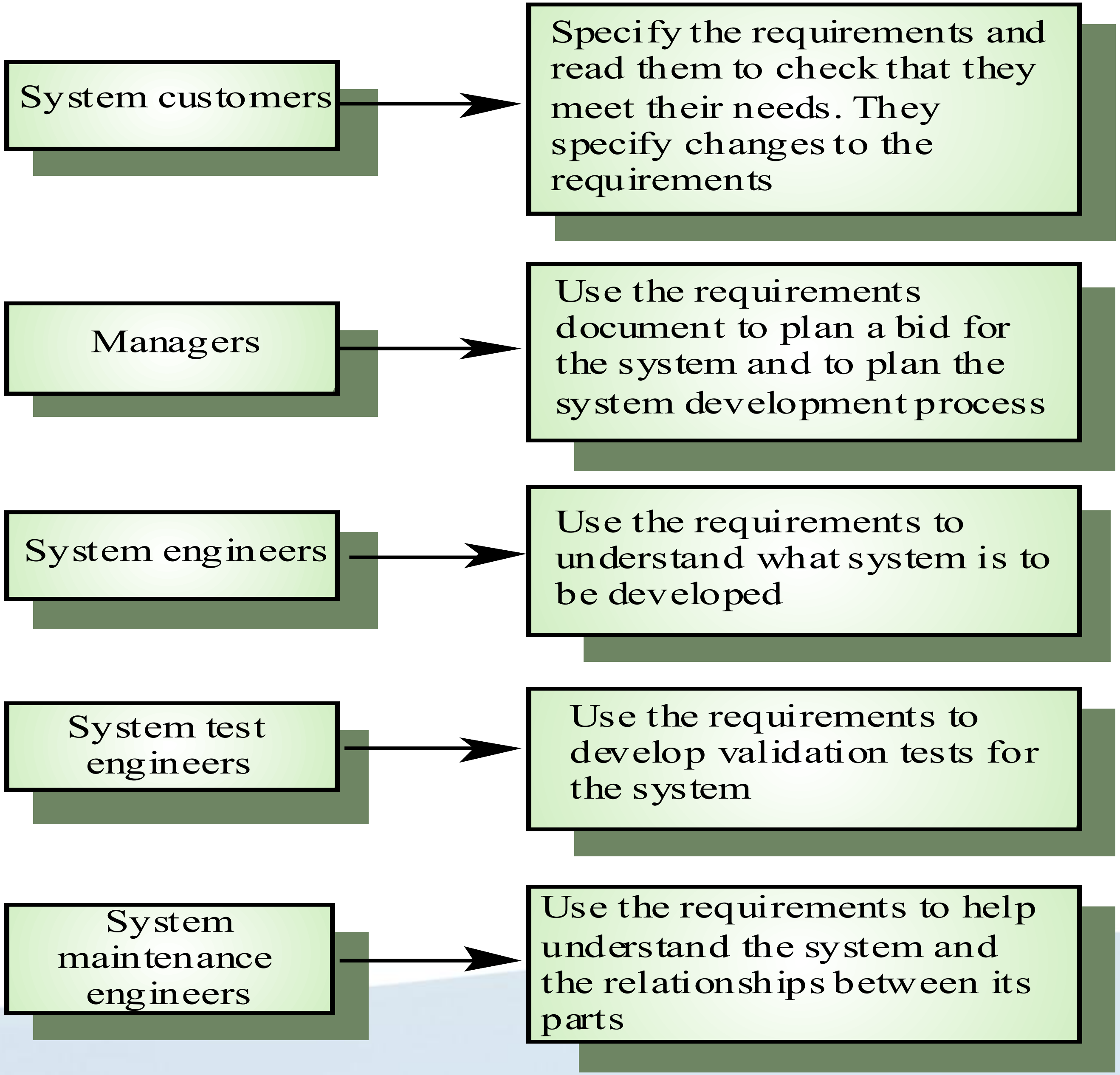
Requirements Management Guidebook

RECORD OF CHANGES

A - ADDED M - MODIFIED D - DELETED

CHANGE NUMBER	DATE	NUMBER OF FIGURE, TABLE OR PARAGRAPH	A M D	TITLE OR BRIEF DESCRIPTION	CHANGE REQUEST NUMBER

SRS users



System customers

Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements

Managers

Use the requirements document to plan a bid for the system and to plan the system development process

System engineers

Use the requirements to understand what system is to be developed

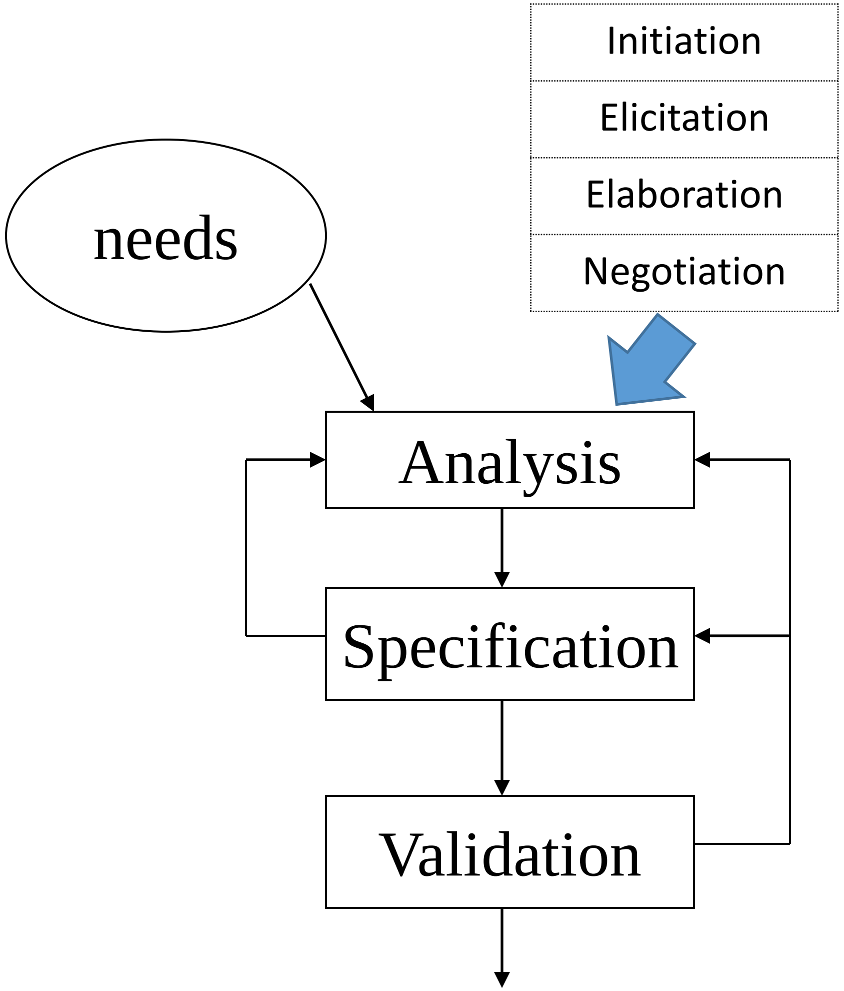
System test engineers

Use the requirements to develop validation tests for the system

System maintenance engineers

Use the requirements to help understand the system and the relationships between its parts

Requirement engineering practices



- Process is not linear, it is iterative and parallel
- Overlap between phases - some parts may be analyzed and specified
- Specification itself may help analysis
- Validation can show gaps that can lead to further analysis and spec

Informal Approach to Analysis

- No defined methodology; info obtained through analysis, observation, interaction, discussions,…
- No formal model of the system built
- Obtained info organized in the SRS; SRS reviewed with clients
- Relies on analyst experience and feedback from clients in reviews
- Useful in many contexts

Analysis Model

Modeling

- A model is an abstract view of a system
- We create a model to gain better understanding of an entity, for example a model of a plane is a small plane.
- When the entity is software, the model takes a different form.

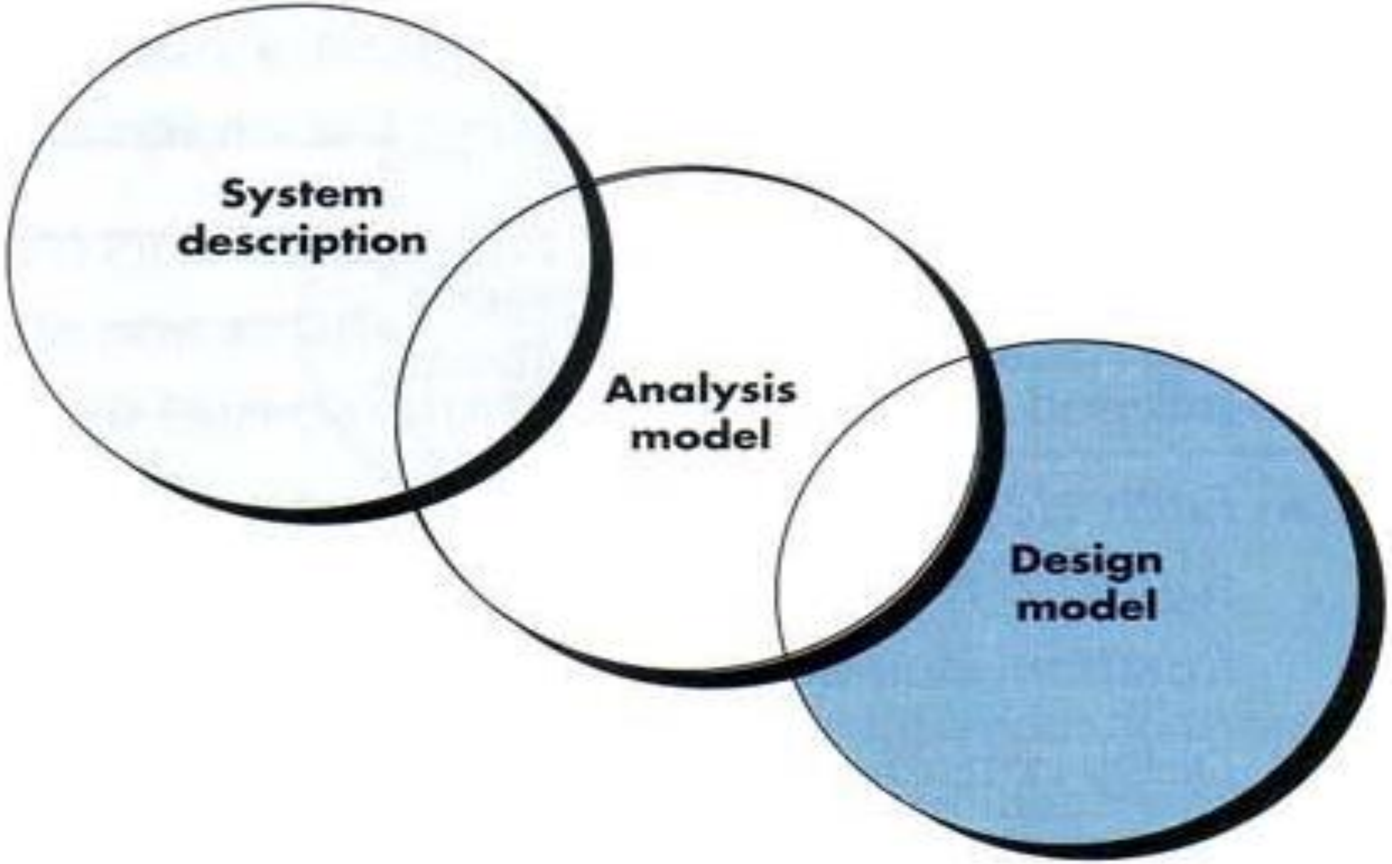
Analysis Modeling Objectives

- Three Primary Objectives:
  - Describe what the customer requires.
  - Establish a basis for the creation of a software design.
  - Devise a set of requirements that can be validated once the software is built.
- Its bridges the gap between a system-level description that describes overall system functionality and a software design.

Guidelines :

- Graphics should be used whenever possible.
- Differentiate between the logical (essential) and physical (implementation) considerations.

Analysis Model - A Bridge



Requirement Modeling Rules of Thumb

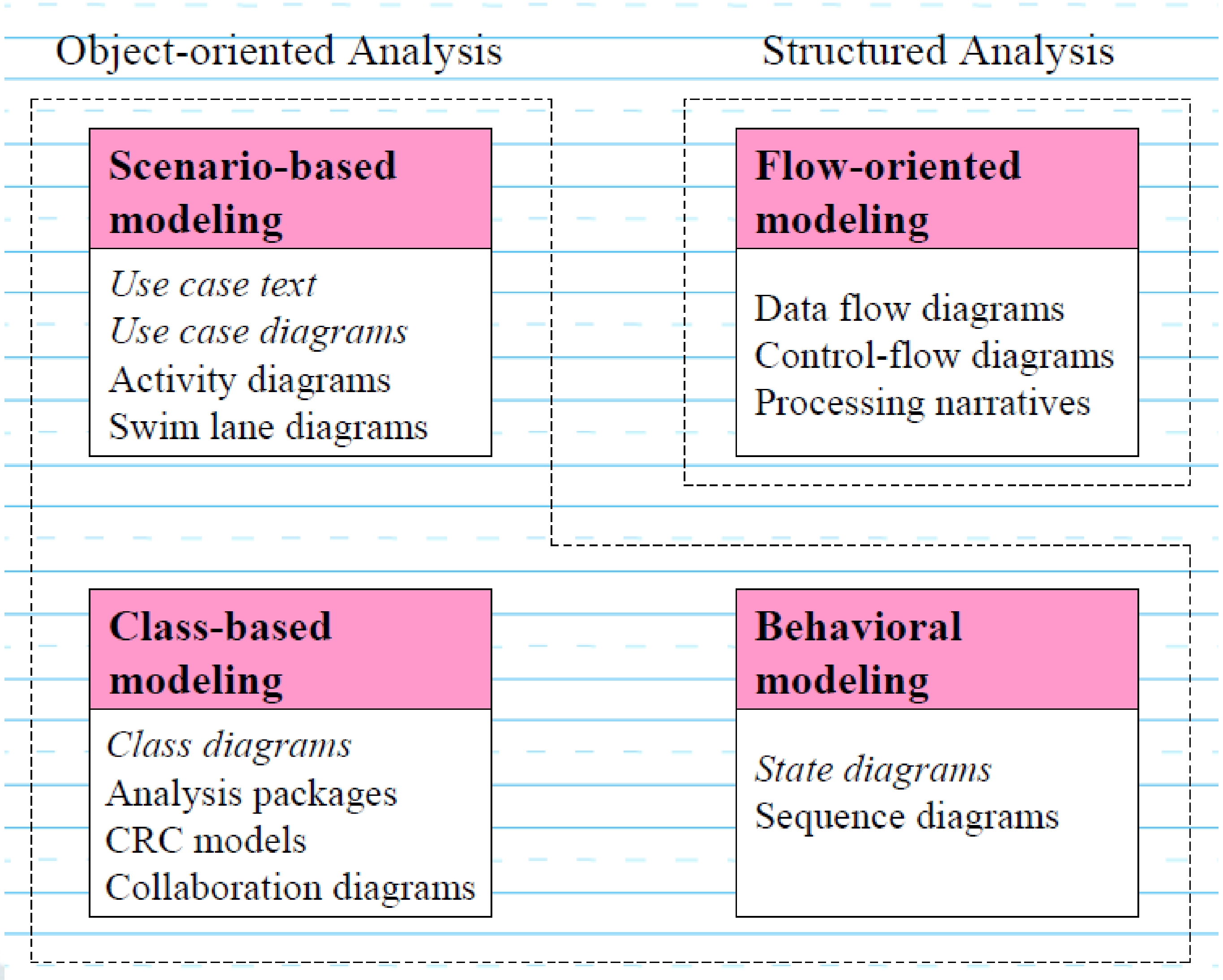
- The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high. – Don’t get bogged into the details.
- Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system.
- Delay consideration of infrastructure and other non-functional models until design.
- Minimize coupling throughout the system. - If level of interconnectedness is high, efforts should be made to reduce it.
- Assured that the analysis model provides value to all stakeholders. – business stakeholder should validate requirement, Designers should use the model as a basis for design.
- Keep the model as simple as it can be. - No need to use additional diagram and use notations.

Elements of Analysis model

- There are two approaches

1. Structured Analysis:-
   - Data objects are modeled in a way that defines their attributes and relationships.
   - Processes that manipulate data objects in a manner that shows how they transform data as data objects flow through the systems.
2. Object Oriented Analysis :-
   - Focuses on the definition of classes and the manner in which they collaborate with one another.
   - UML is predominantly object oriented.

Elements of Analysis model



Object-oriented Analysis

Structured Analysis

Scenario-based modeling

Use case text
Use case diagrams
Activity diagrams
Swim lane diagrams

Flow-oriented modeling

Data flow diagrams
Control-flow diagrams
Processing narratives

Class-based modeling

Class diagrams
Analysis packages
CRC models
Collaboration diagrams

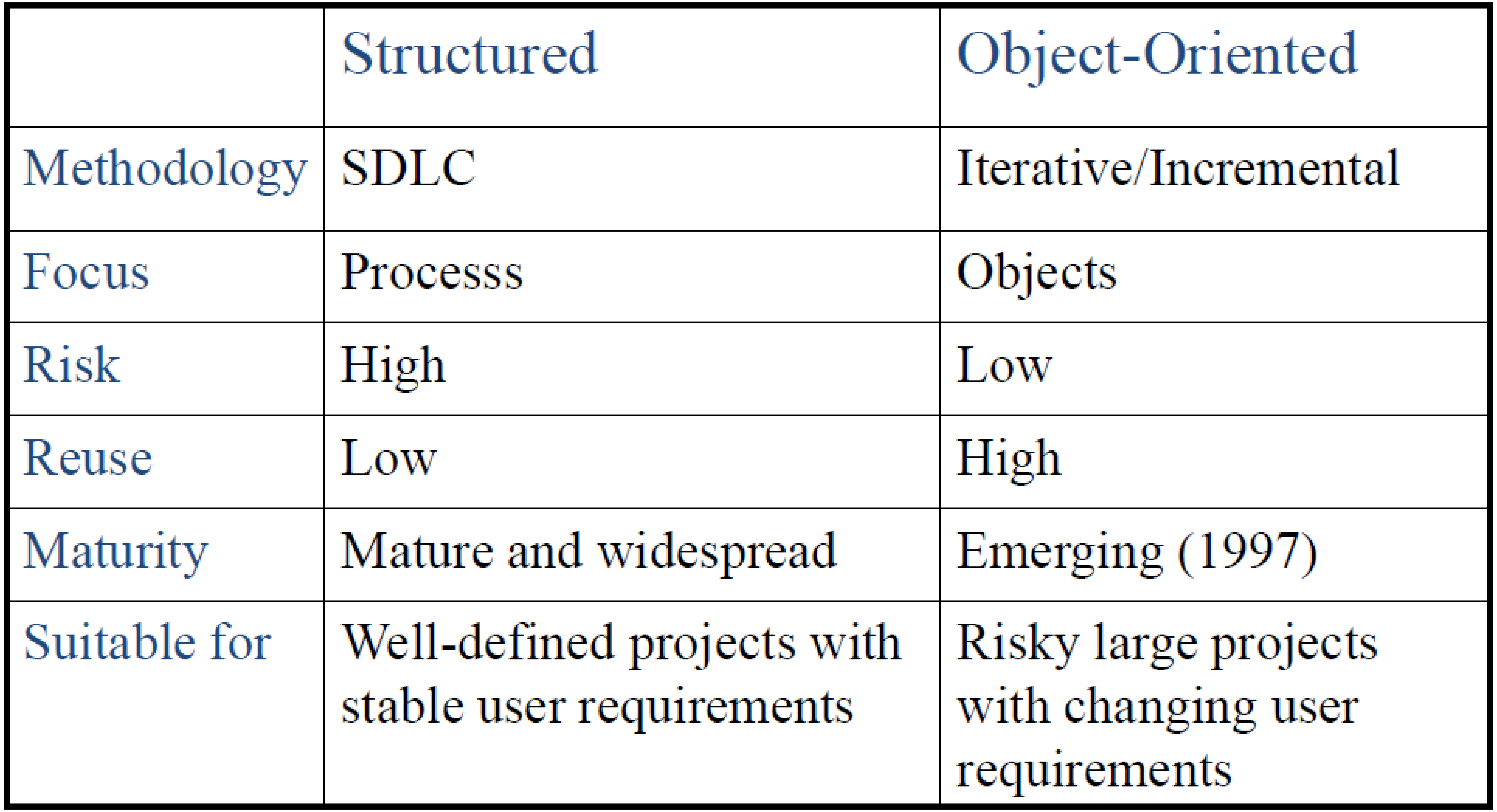
Behavioral modeling

State diagrams
Sequence diagrams

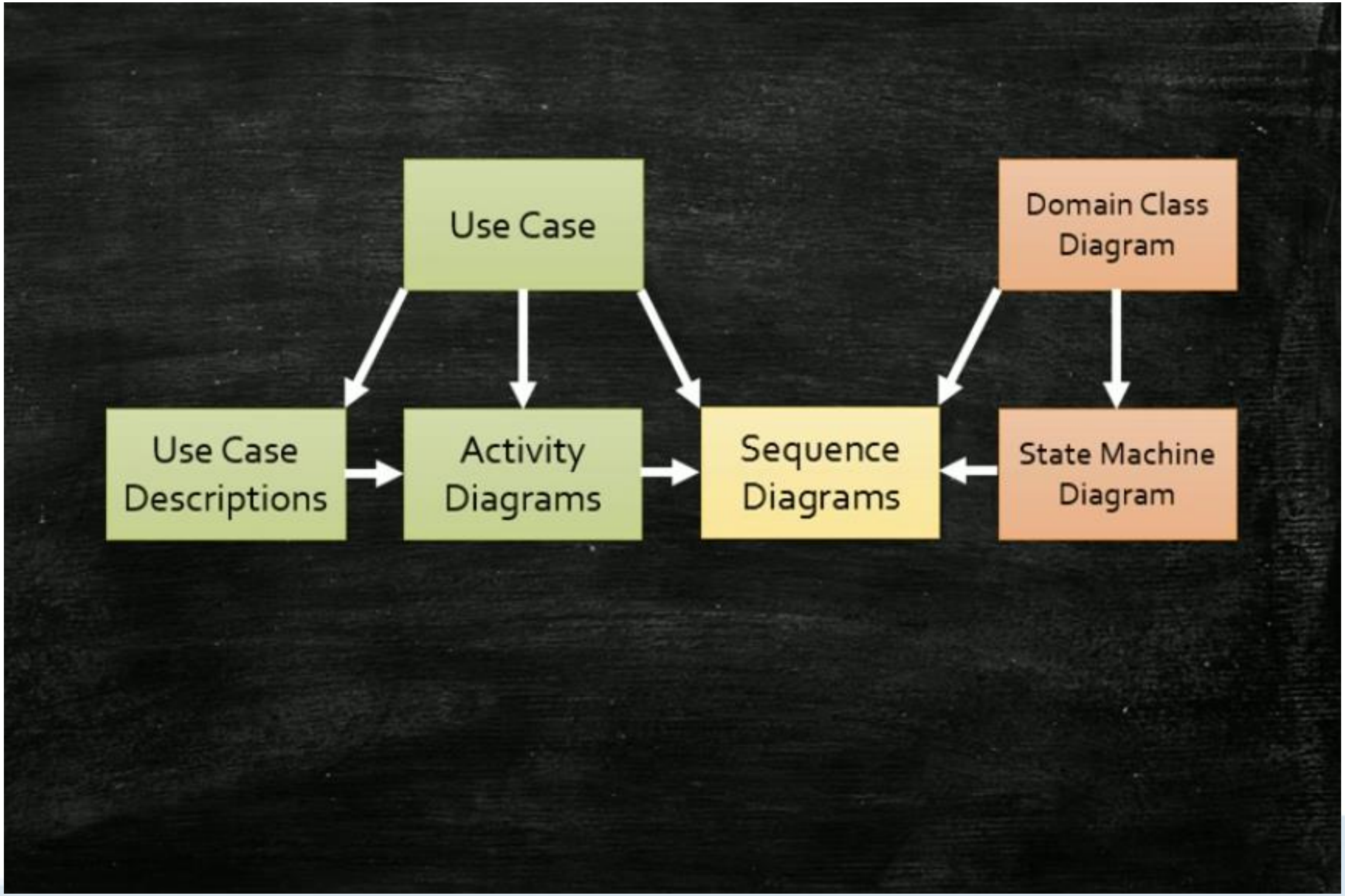
CRC-Class Responsibility Collaborator

- **Flow-oriented modeling** – provides an indication of how data objects are transformed by a set of processing functions
- **Scenario-based modeling** – represents the system from the user's point of view
- **Class-based modeling** – defines objects, attributes, and relationships
- **Behavioral modeling** – depicts the states of the classes and the impact of events on these states

Structured vs OO analysis



	Structured	Object-Oriented
Methodology	SDLC	Iterative/Incremental
Focus	Processs	Objects
Risk	High	Low
Reuse	Low	High
Maturity	Mature and widespread	Emerging (1997)
Suitable for	Well-defined projects with stable user requirements	Risky large projects with changing user requirements


Use Case
Domain Class Diagram
Use Case Descriptions
Activity Diagrams
Sequence Diagrams
State Machine Diagram