CMP 519 Software Engineering

UNIT 01 - INTRODUCTION

Instructor: Dr. Suresh Pokharel

DISCLAIMER

These slides are part of teaching materials for Software Engineering. These slides do not cover all aspect of learning Software Engineering, nor are these be taken as primary source of information. As the core textbooks and reference books for learning the subject has already been specified and provided to the students, students are encouraged to learn from the original sources.

Software Engineering Concepts

What is software engineering?

SOFTWARE ENGINEERING

- A multidisciplinary activity involving modeling, problem-solving, knowledge acquisition, and rationale capture.
- Key Features:
  - *Modeling*: Focuses on relevant details, ignoring others
  - *Problem-Solving*: Empirical methods for constrained resources
  - *Knowledge Acquisition*: Nonlinear, adaptive understanding
  - *Rationale*: Capturing decision contexts and alternatives
- Goal:
  - *Make quality software within a budget, within timeframe*

SOFTWARE ENGINEERING – MODELING

- Simplify complex systems for better understanding
- An abstract representation of a system
- Software modeling with *UML*

SOFTWARE ENGINEERING – PROBLEM SOLVING

- How you solve the any engineering problems?
- Engineering Process
  - Formulate the problem
  - Analyze the problem
  - Search for solutions
  - Decide on a solution
  - Specify the solution
- Development Activities:
  - Requirements, Analysis, Design, Implementation, Testing

SOFTWARE ENGINEERING – PROBLEM SOLVING

- Nonlinear Process
  - New information can invalidate prior knowledge
  - Sequential models (e.g., Waterfall) are limiting
- Alternative Approaches
  - Risk-Based Development: Focus on high-risk areas
  - Issue-Based Development: Parallel execution of activities
- Implication: Be prepared for iteration and adaptability

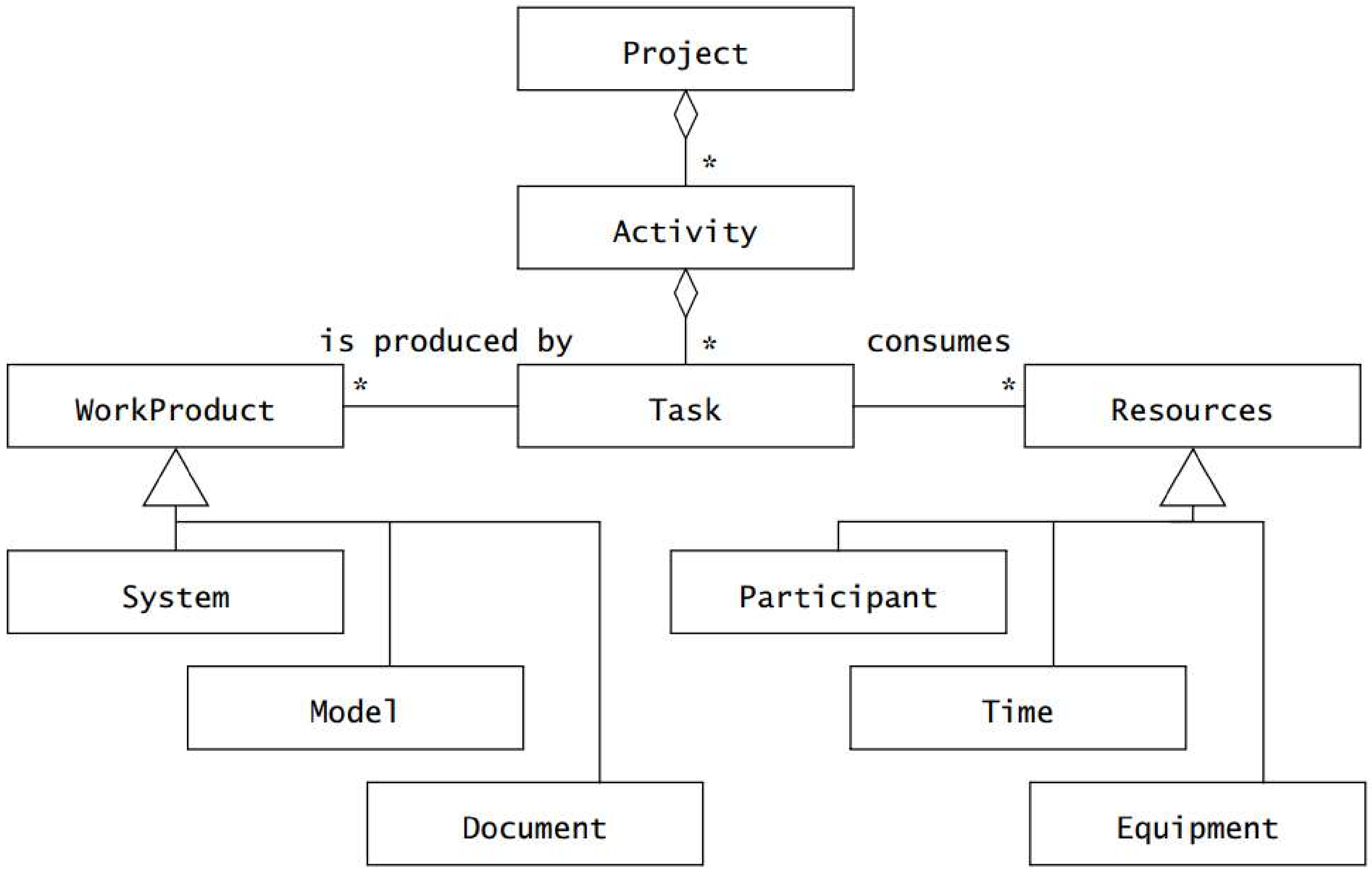
SOFTWARE ENGINEERING – RATIONALE

- Importance
  - Contextual understanding of decisions
  - Helps manage changes and updates effectively
- Challenges
  - Capturing tacit knowledge (intuition-based decisions)
  - Managing large volumes of alternatives and arguments
- Key Tasks
  - Document decision rationale
  - Adapt to technological and requirement changes

CURRENT TRENDS IN SOFTWARE ENGINEERING

- Docker and containerization
- Low-Code No-Code (LCNC)/Low-Code Development
- Continuous Delivery and Deployment
- Artificial Intelligence (AI)/Machine learning/Information Retrieval
- Data analytics
- Big data
- Internet of Things (IOT) and wearable computing
- Cloud Computing/ Serverless Computing (IAAS, PAAS, SAAS, DAAS)
- Blockchain

SOFTWARE ENGINEERING CONCEPTS

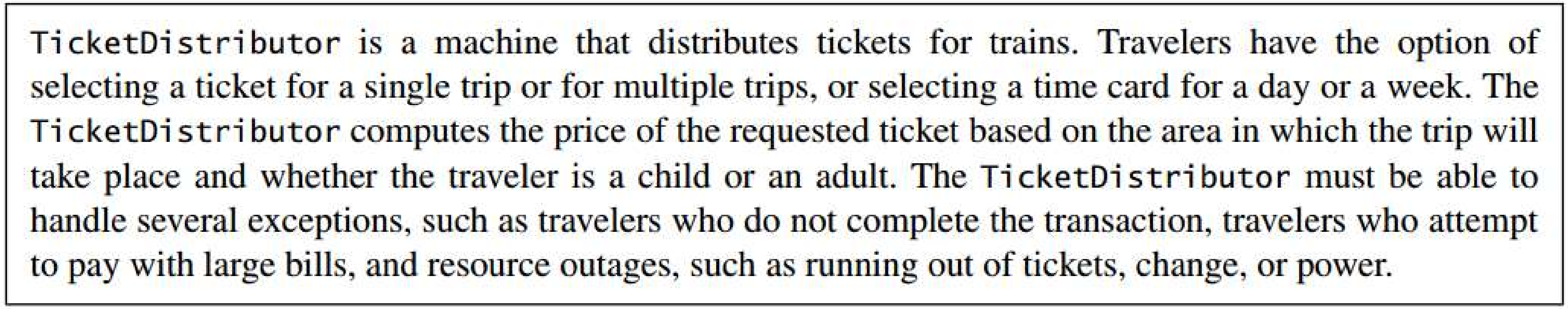


Software engineering concepts depicted as a UML class diagram [OMG, 2009].

PARTICIPANTS AND ROLES

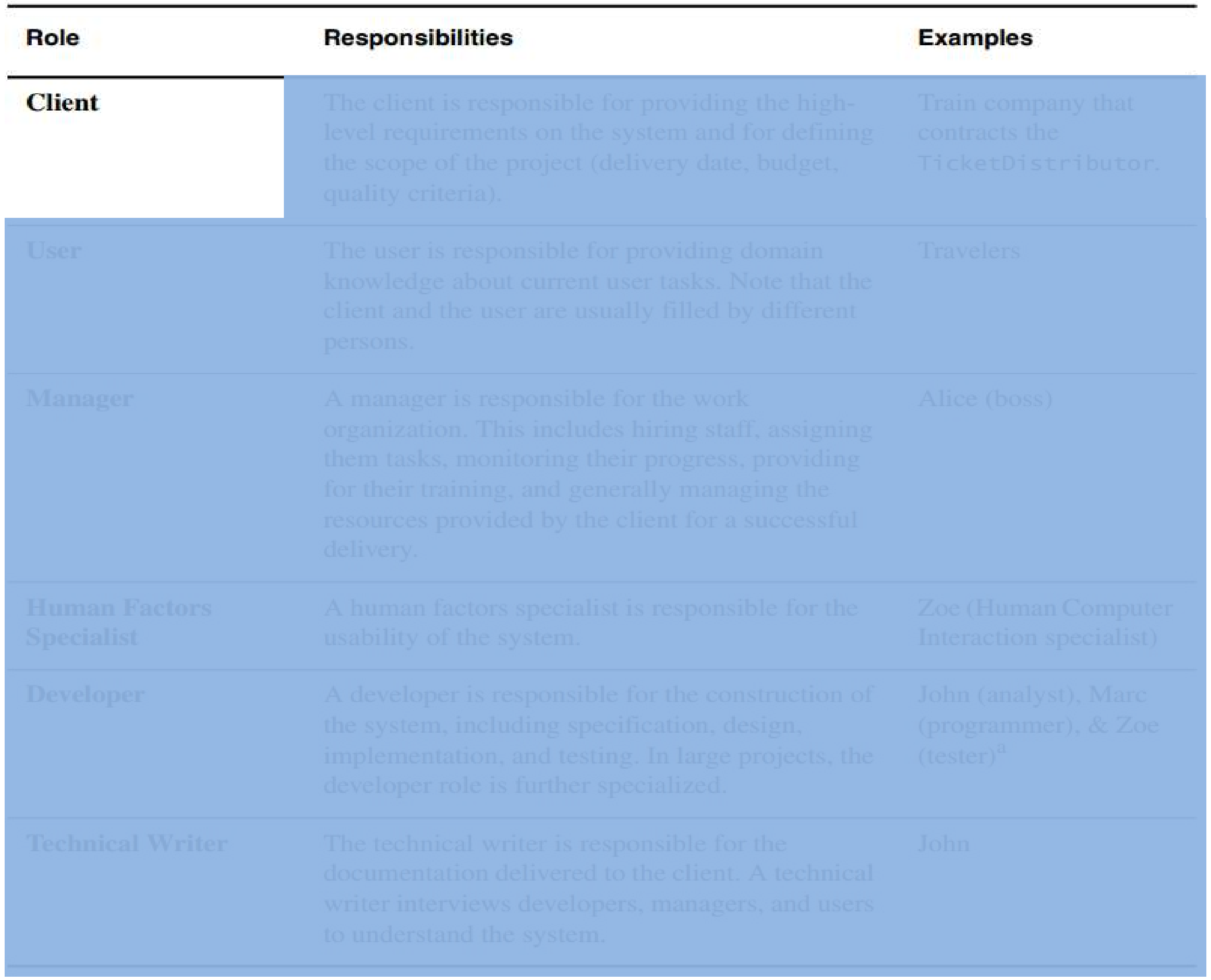
- **Participants**: Individuals involved in the project.
- **Roles**: Defined responsibilities assigned to participants, associated with specific tasks.
  - A participant can fulfill multiple **roles**.

Consider a `TicketDistributor` system:



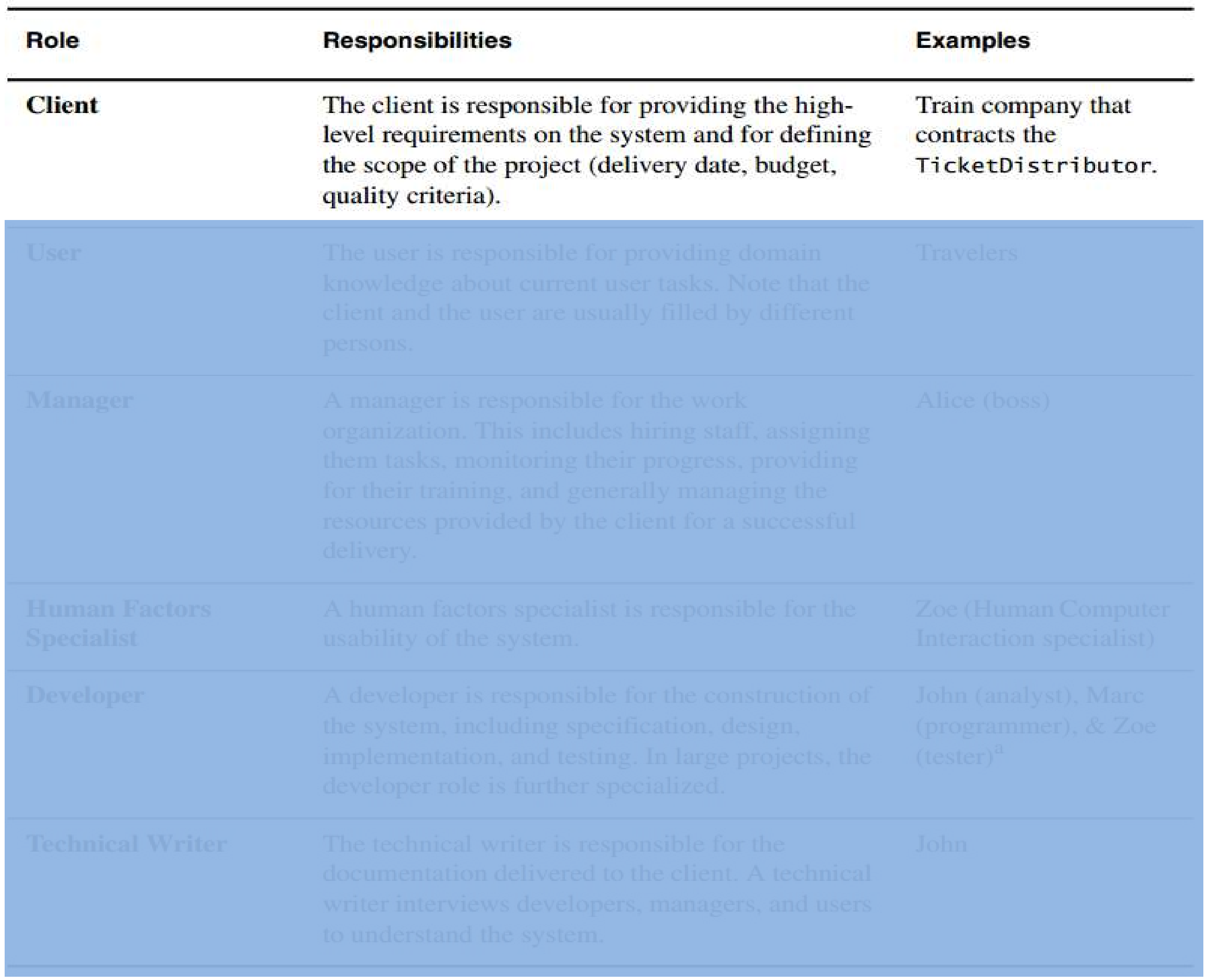
`TicketDistributor` is a machine that distributes tickets for trains. Travelers have the option of selecting a ticket for a single trip or for multiple trips, or selecting a time card for a day or a week. The `TicketDistributor` computes the price of the requested ticket based on the area in which the trip will take place and whether the traveler is a child or an adult. The `TicketDistributor` must be able to handle several exceptions, such as travelers who do not complete the transaction, travelers who attempt to pay with large bills, and resource outages, such as running out of tickets, change, or power.

SOFTWARE ENGINEERING – RATIONALE



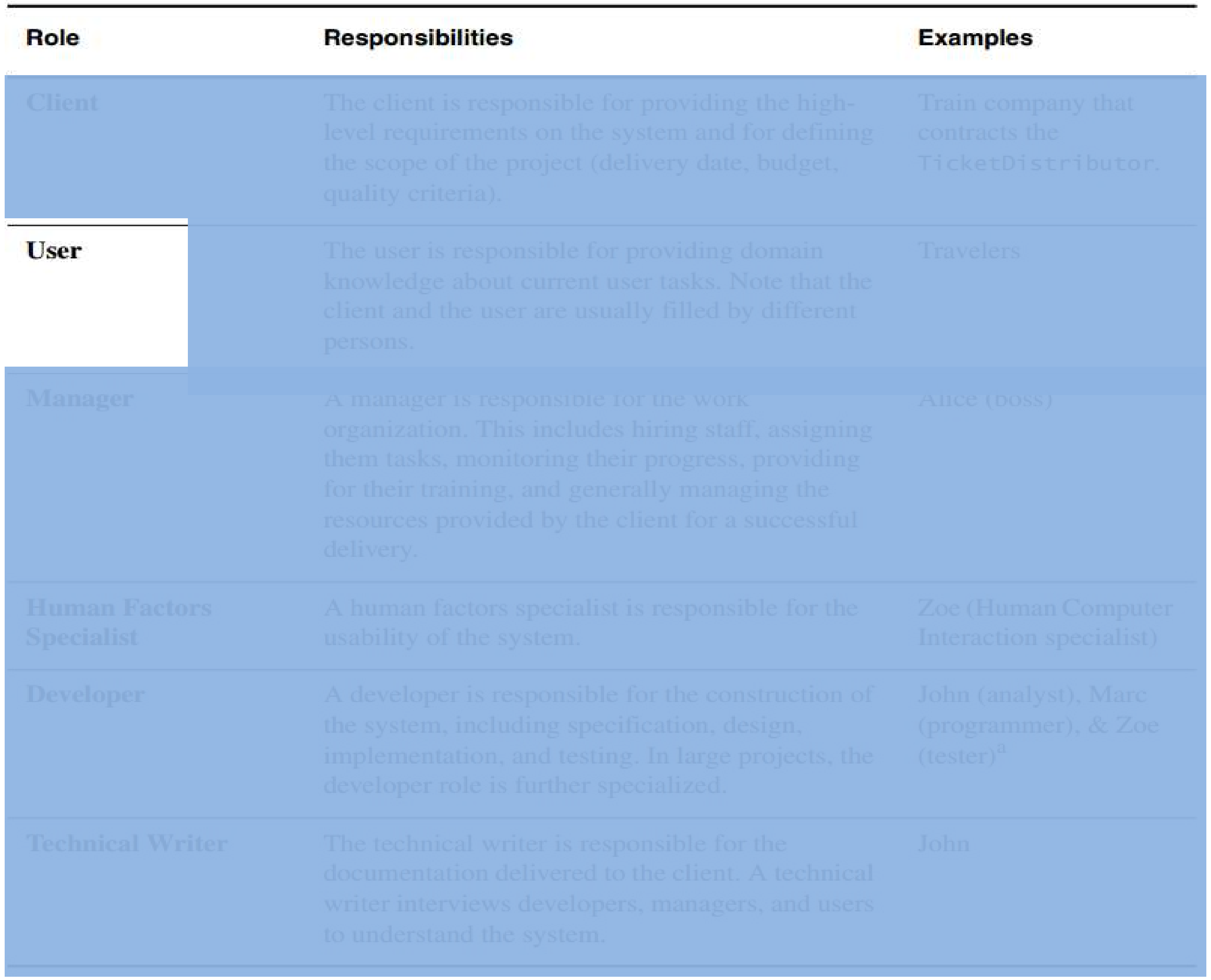
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



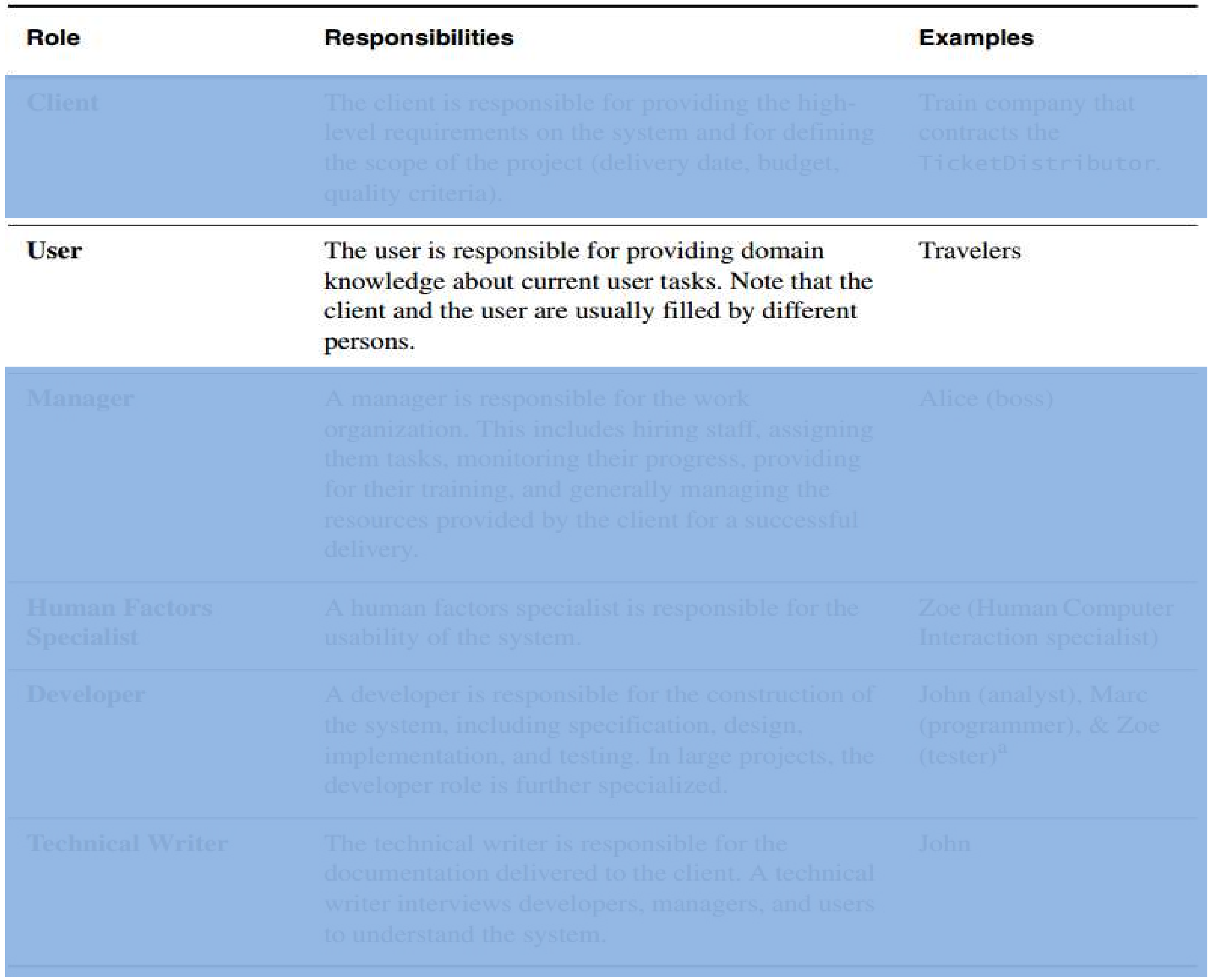
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



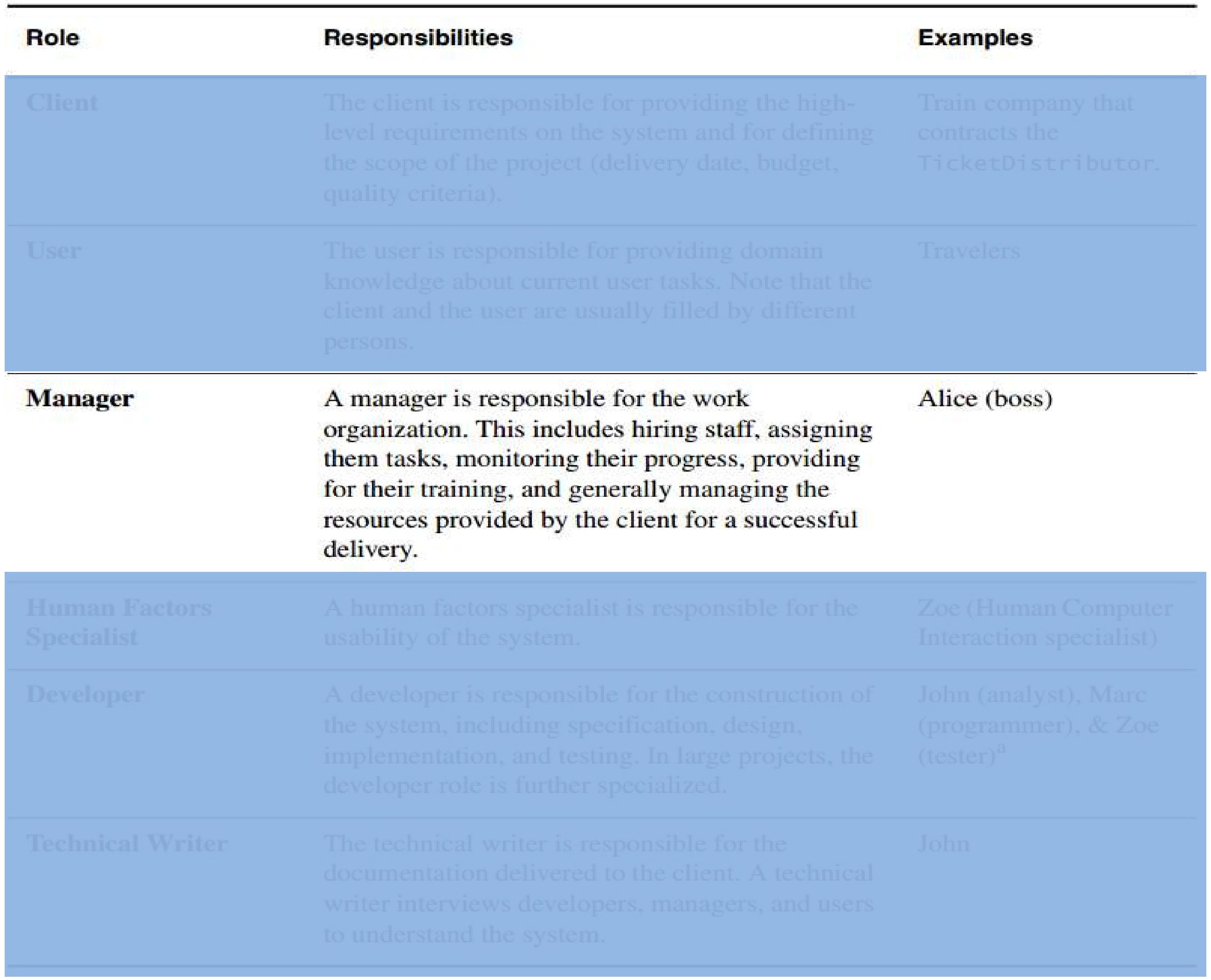
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



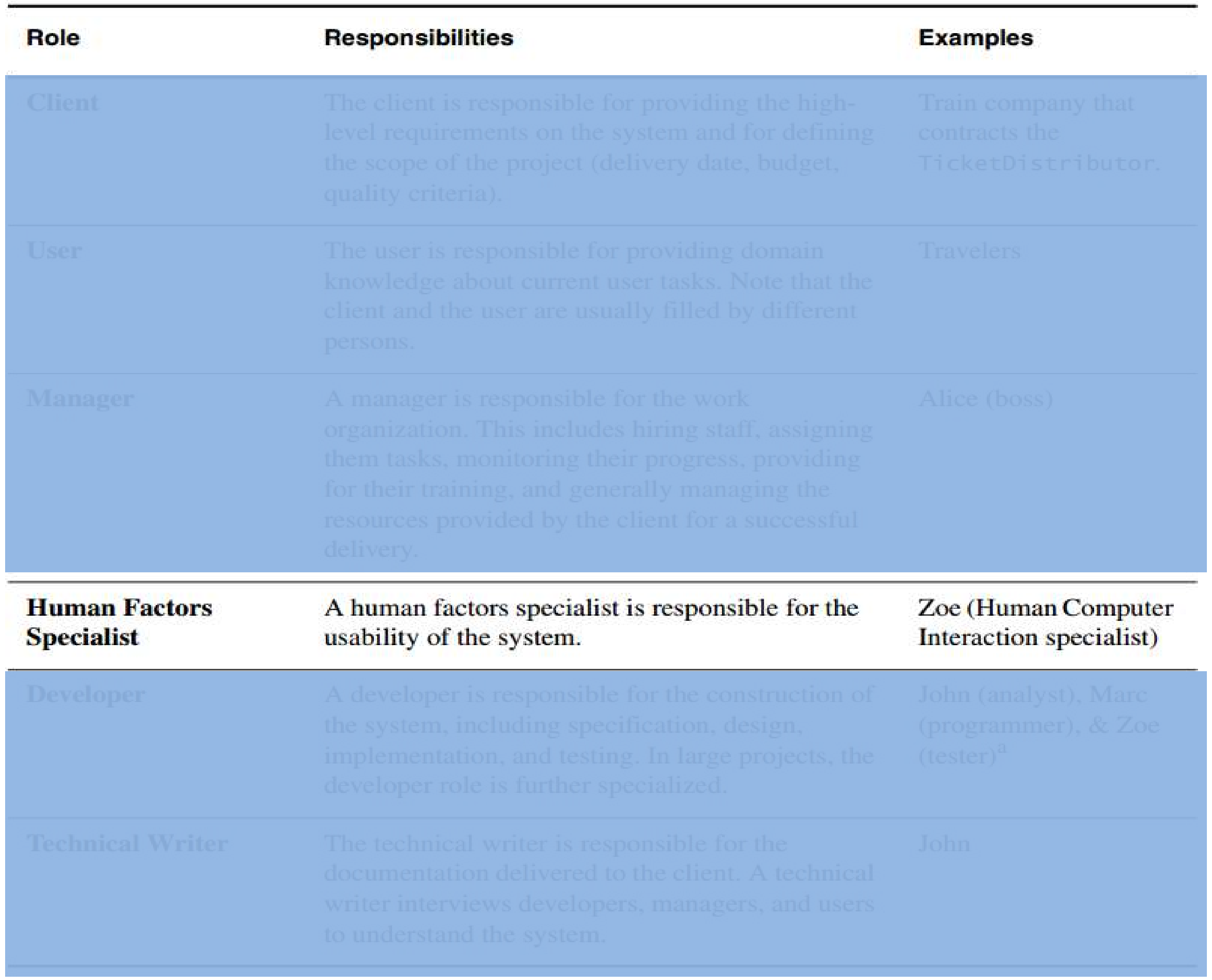
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



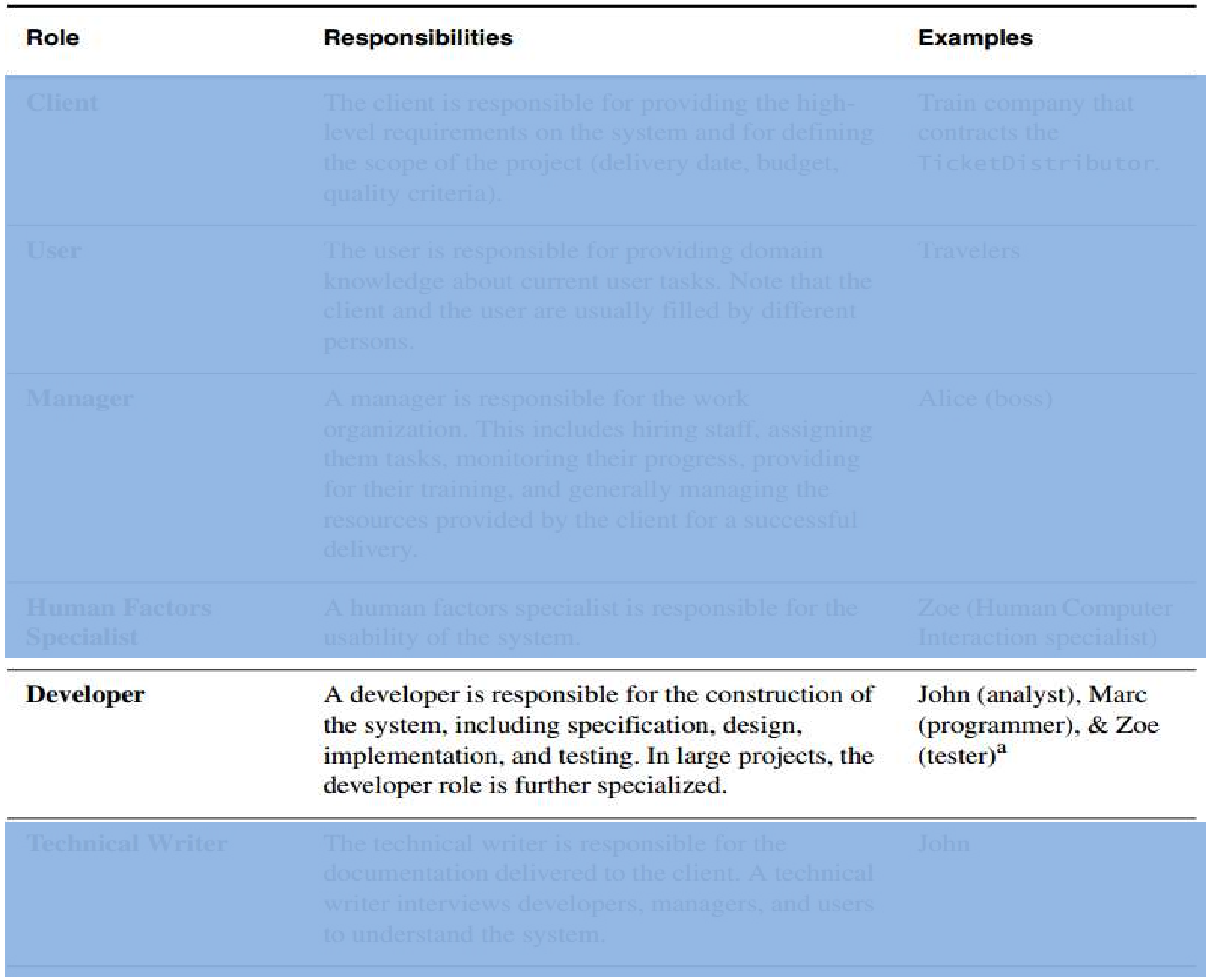
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



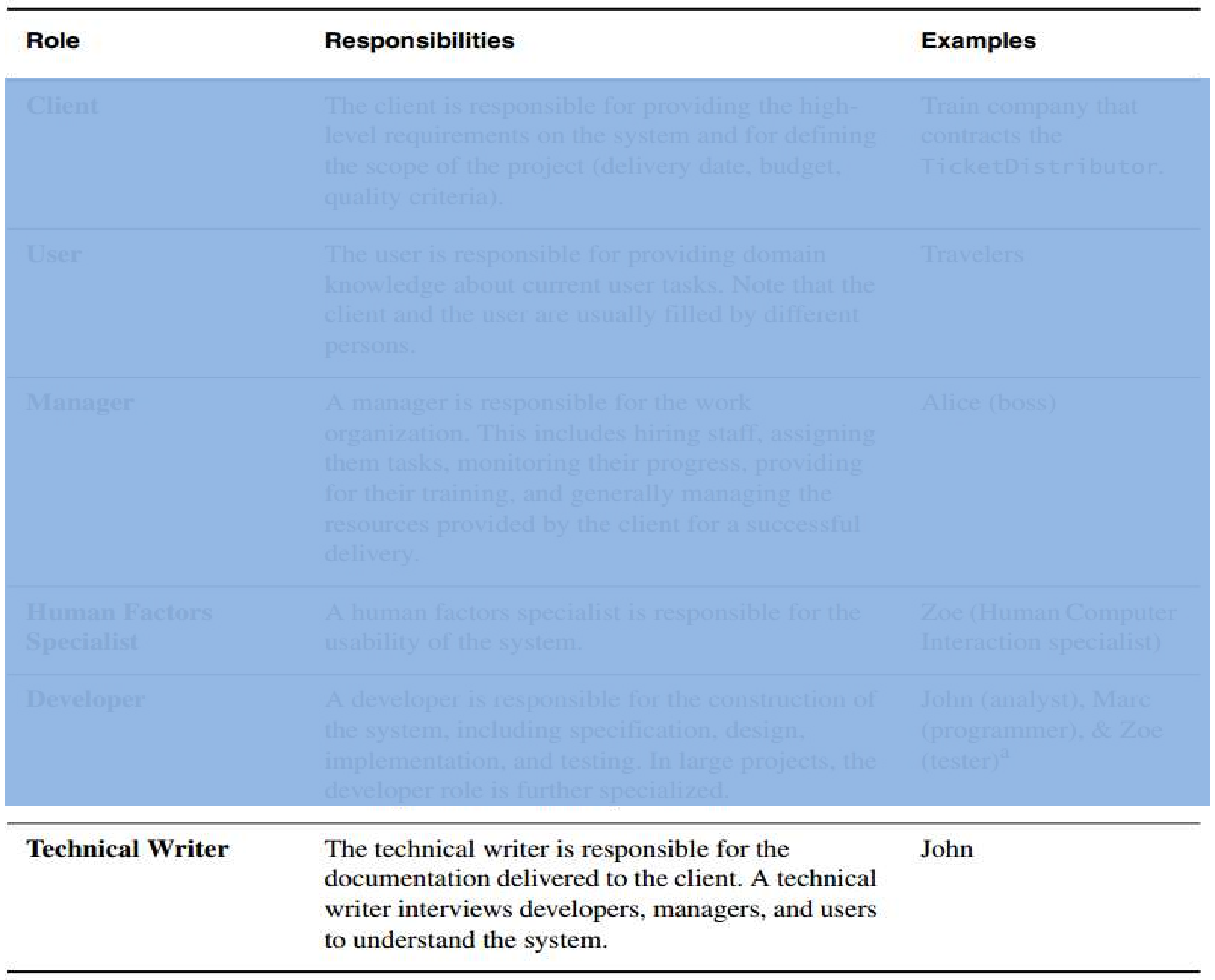
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



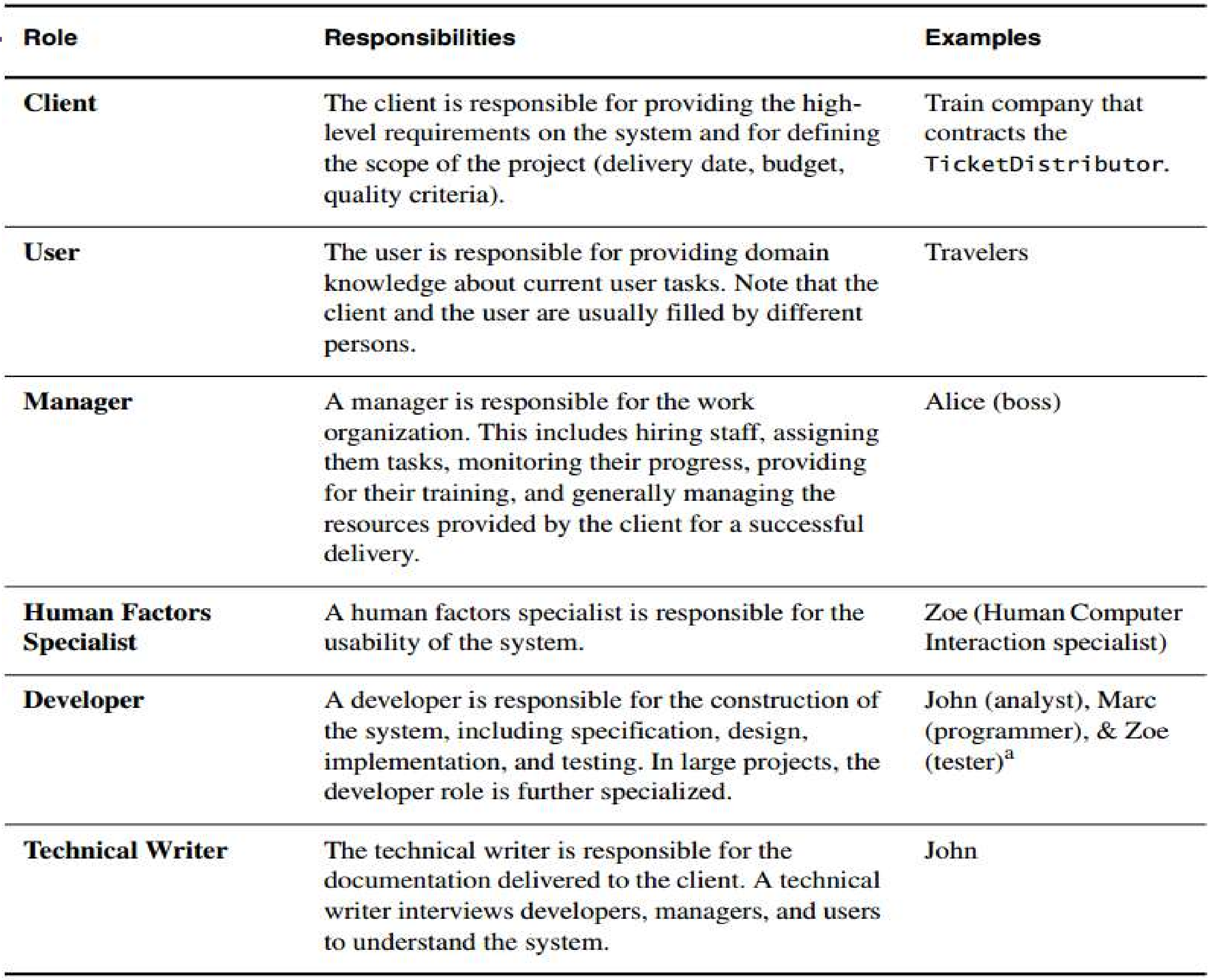
Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE



Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

SOFTWARE ENGINEERING – RATIONALE

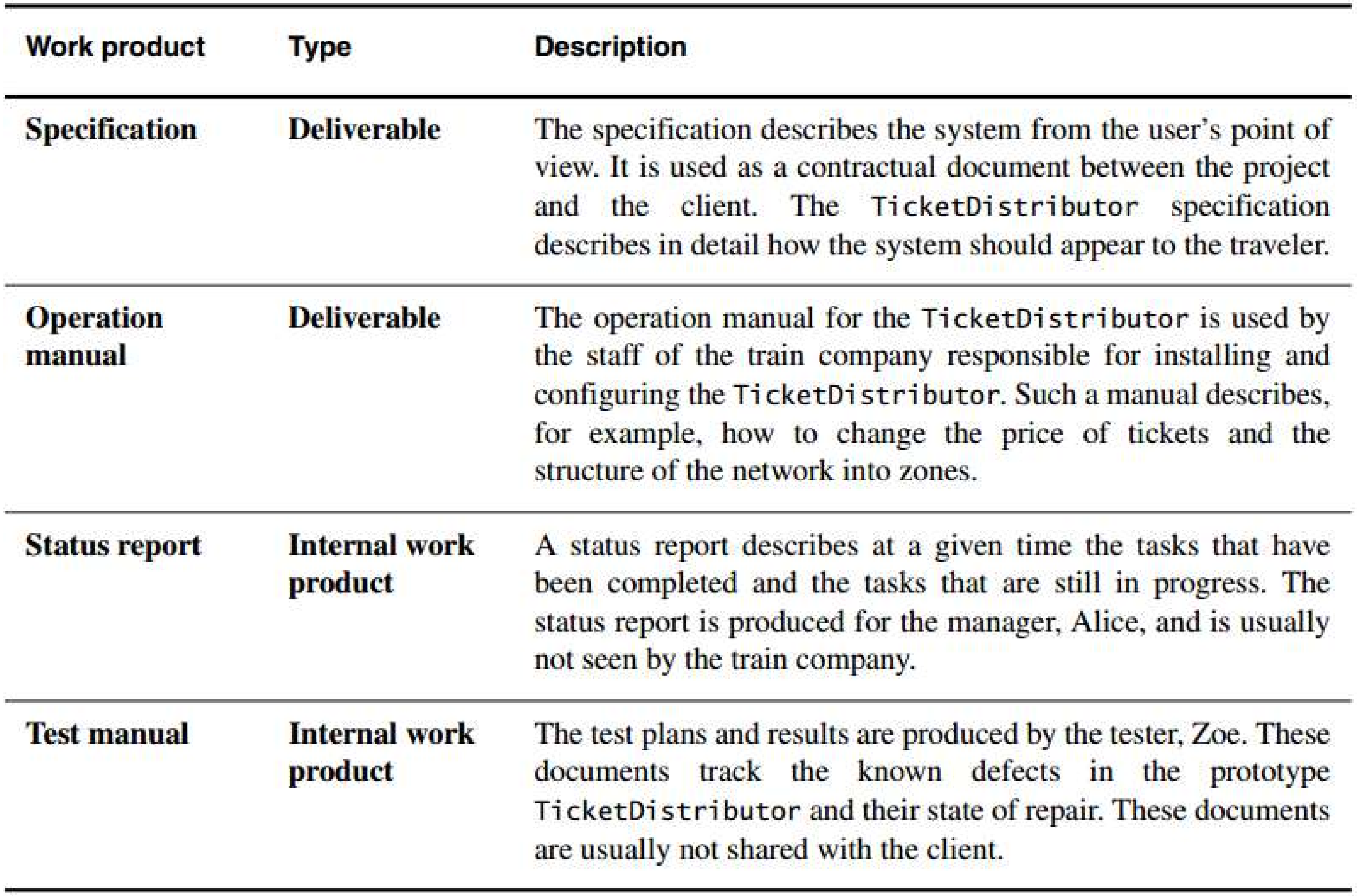


Role	Responsibilities	Examples
Client	The client is responsible for providing the high-level requirements on the system and for defining the scope of the project (delivery date, budget, quality criteria).	Train company that contracts the `TicketDistributor`.
User	The user is responsible for providing domain knowledge about current user tasks. Note that the client and the user are usually filled by different persons.	Travelers
Manager	A manager is responsible for the work organization. This includes hiring staff, assigning them tasks, monitoring their progress, providing for their training, and generally managing the resources provided by the client for a successful delivery.	Alice (boss)
Human Factors Specialist	A human factors specialist is responsible for the usability of the system.	Zoe (Human Computer Interaction specialist)
Developer	A developer is responsible for the construction of the system, including specification, design, implementation, and testing. In large projects, the developer role is further specialized.	John (analyst), Marc (programmer), & Zoe (tester)[a]
Technical Writer	The technical writer is responsible for the documentation delivered to the client. A technical writer interviews developers, managers, and users to understand the system.	John

WORK PRODUCTS

- A ***work product*** is any artifact created during development, such as documents or software, intended for developers or clients.
- Types of Work Products
  - ***Internal Work Product***: Created for internal use within the project.
  - ***Deliverable***: A work product delivered to the client, defined before the project begins.
- Deliverables and Contracts
  - Deliverables are specified by a contract, ensuring developers meet client requirements.

WORK PRODUCTS

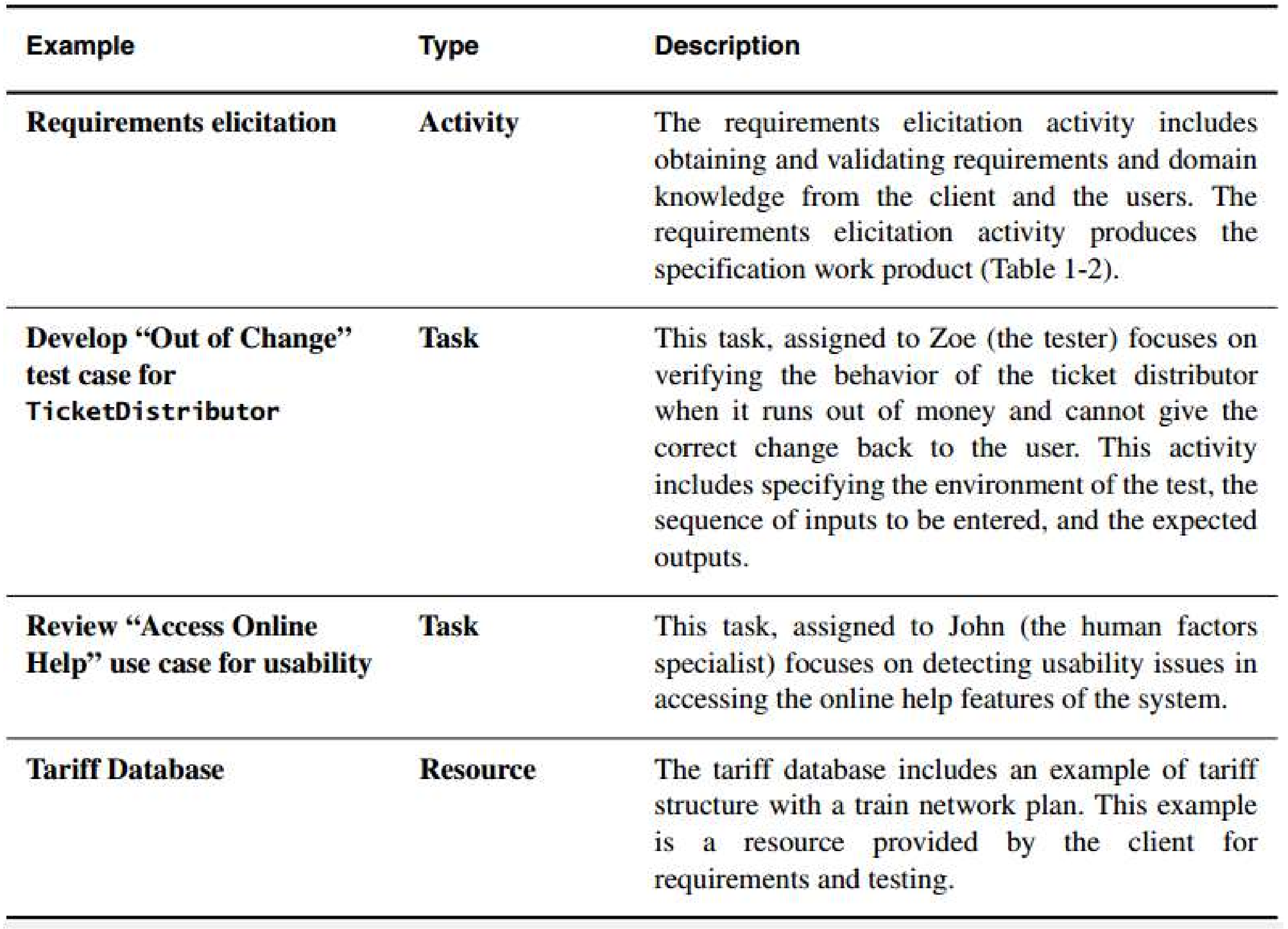


Work product	Type	Description
Specification	**Deliverable**	The specification describes the system from the user's point of view. It is used as a contractual document between the project and the client. The `TicketDistributor` specification describes in detail how the system should appear to the traveler.
Operation manual	**Deliverable**	The operation manual for the `TicketDistributor` is used by the staff of the train company responsible for installing and configuring the `TicketDistributor`. Such a manual describes, for example, how to change the price of tickets and the structure of the network into zones.
Status report	**Internal work product**	A status report describes at a given time the tasks that have been completed and the tasks that are still in progress. The status report is produced for the manager, Alice, and is usually not seen by the train company.
Test manual	**Internal work product**	The test plans and results are produced by the tester, Zoe. These documents track the known defects in the prototype `TicketDistributor` and their state of repair. These documents are usually not shared with the client.

Activities, Tasks, and Resources

- ***Activity***: An activity is a set of tasks performed toward a specific purpose, such as requirements elicitation, delivery, or project management.
- ***Resources*** are assets that are used to accomplish work. Resources include time, equipment, and labor.
- ***Tasks***: Tasks are atomic units of work within activities, consuming resources like time, equipment, and labor, and producing or depending on work products.

Activities, Tasks, and Resources



Example	Type	Description
Requirements elicitation	**Activity**	The requirements elicitation activity includes obtaining and validating requirements and domain knowledge from the client and the users. The requirements elicitation activity produces the specification work product (Table 1-2).
Develop "Out of Change" test case for `TicketDistributor`	**Task**	This task, assigned to Zoe (the tester) focuses on verifying the behavior of the ticket distributor when it runs out of money and cannot give the correct change back to the user. This activity includes specifying the environment of the test, the sequence of inputs to be entered, and the expected outputs.
Review "Access Online Help" use case for usability	**Task**	This task, assigned to John (the human factors specialist) focuses on detecting usability issues in accessing the online help features of the system.
Tariff Database	**Resource**	The tariff database includes an example of tariff structure with a train network plan. This example is a resource provided by the client for requirements and testing.

Notations, Methods, and Methodologies

- A ***notation*** is a graphical or textual set of rules for representing a model. The Roman alphabet is a notation for representing words.
- A ***method*** is a repeatable technique that specifies the steps involved in solving a specific problem. A recipe is a method for cooking a specific dish. A sorting algorithm is a method for ordering elements of a list. Rationale management is a method for justifying change. Configuration management is a method for tracking change.
- A ***methodology*** is a collection of methods for solving a class of problems and specifies how and when each method should be used. A seafood cookbook with a collection of recipes is a methodology for preparing seafood if it also contains advice on how ingredients should be used and what to do if not all ingredients are available.

Traditional Models

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

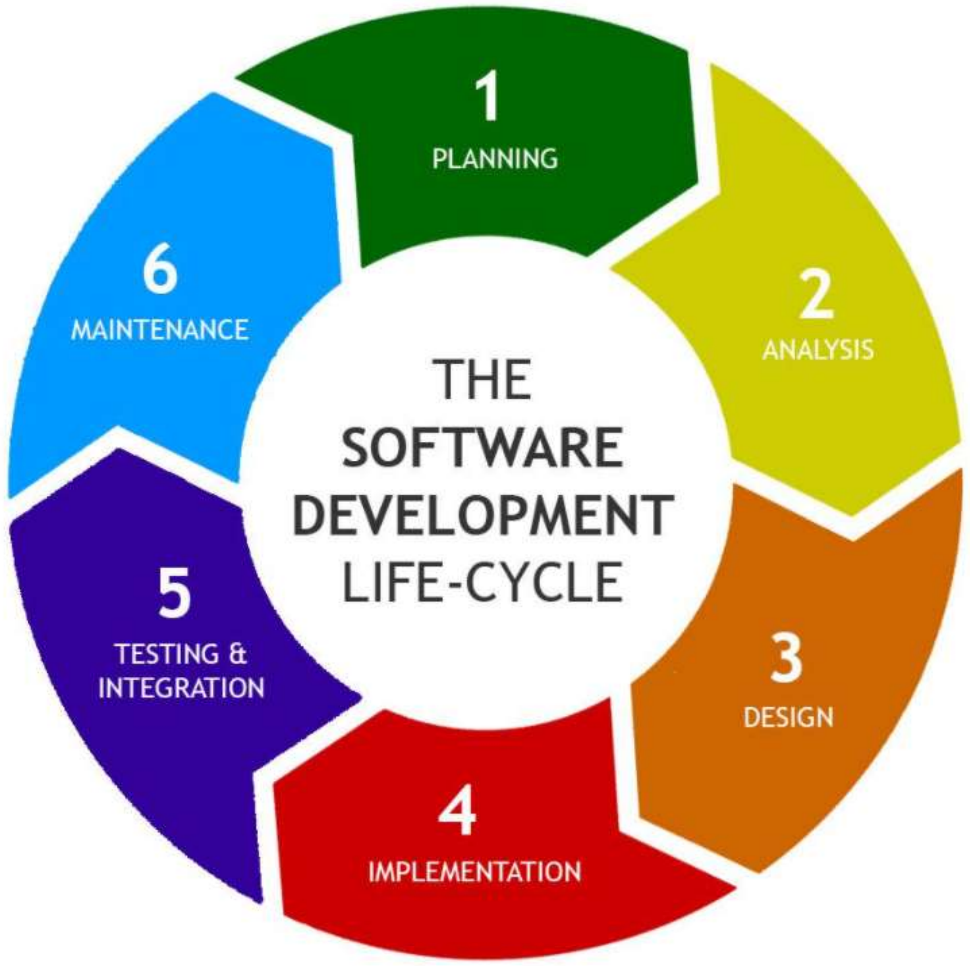
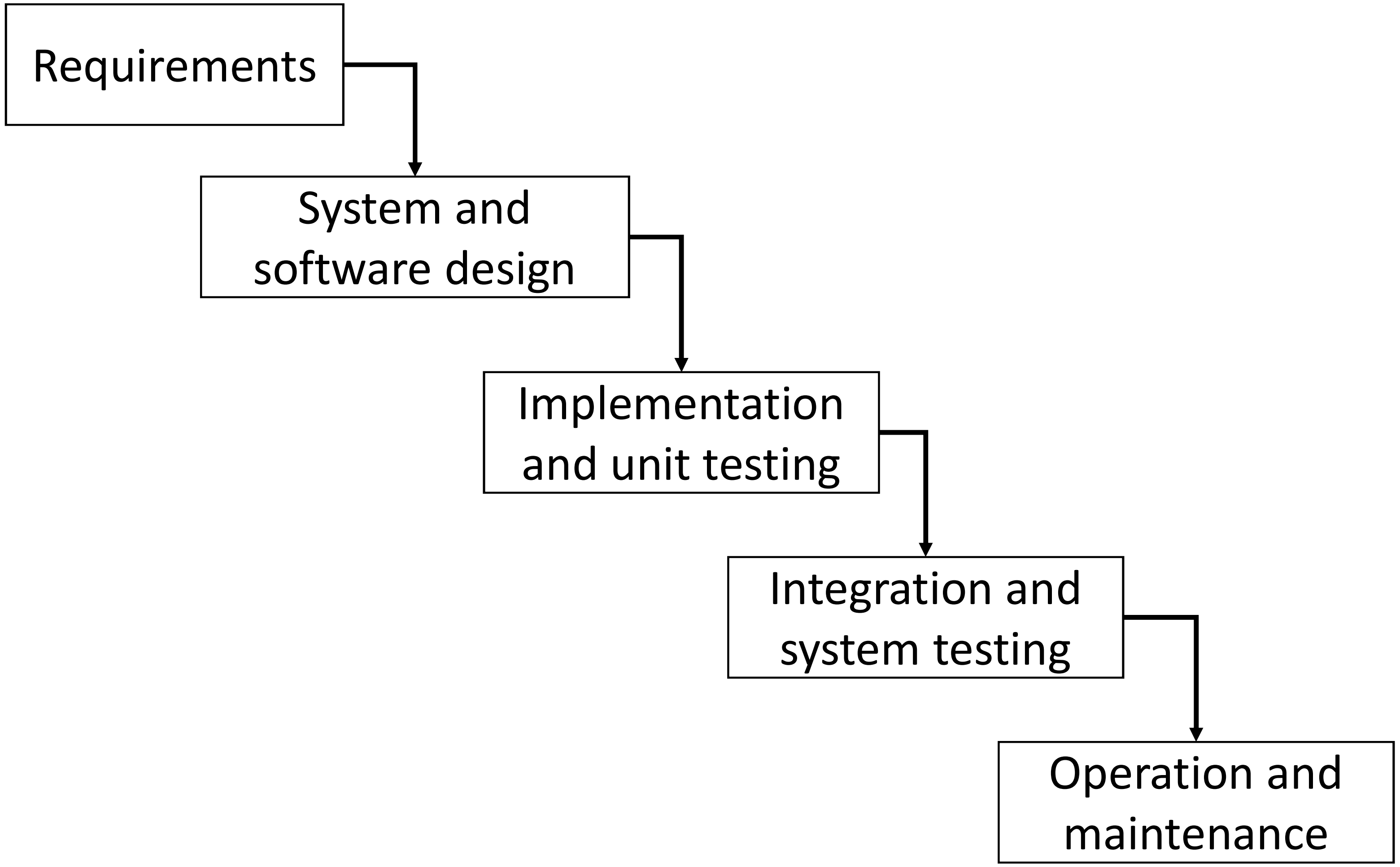


Image: https://arkbauer.com/blog/software-development-life-cycle-sdlc/

SOME SDLC MODELS

- Waterfall Model
- Iterative Model
- Spiral Method

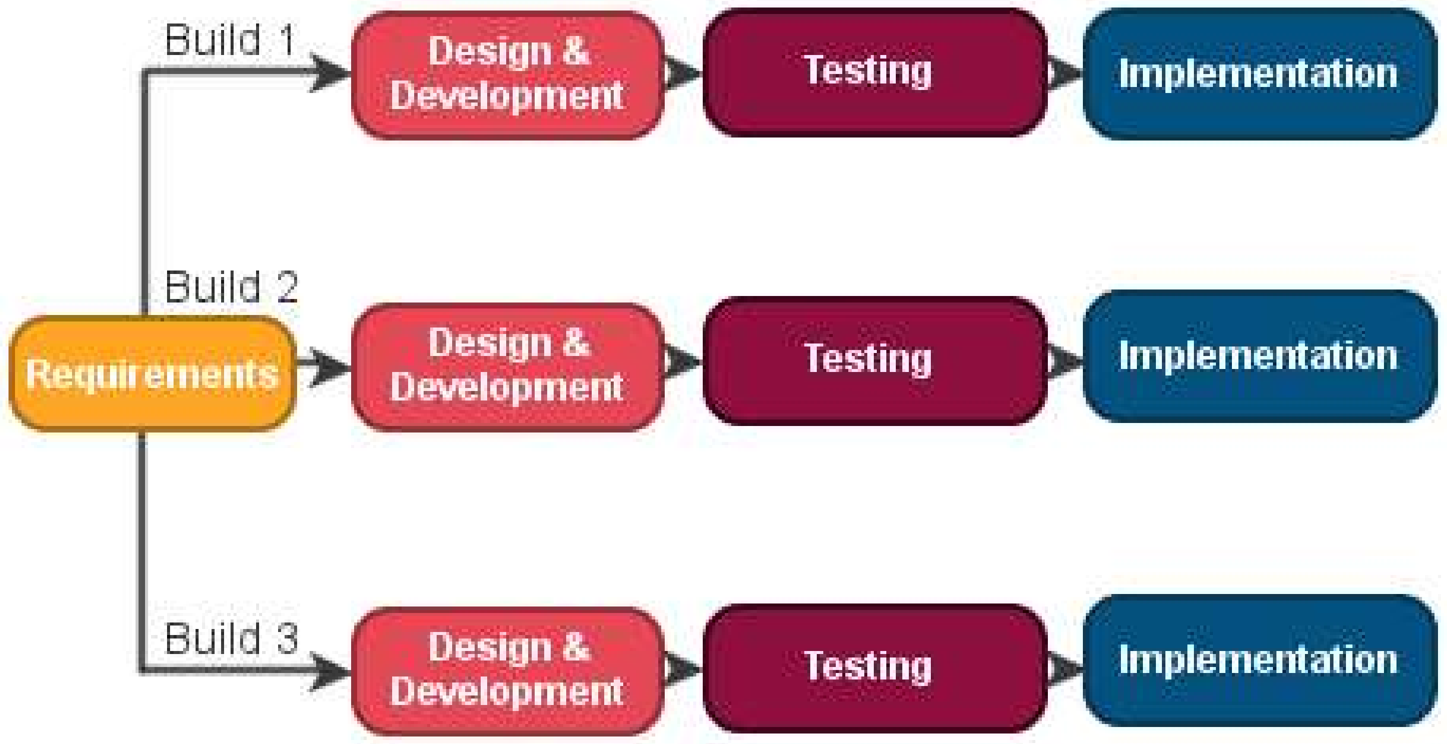
WATERFALL MODEL


Requirements
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

WATERFALL MODEL

Advantages and Disadvantages of waterfall model?

ITERATIVE MODEL

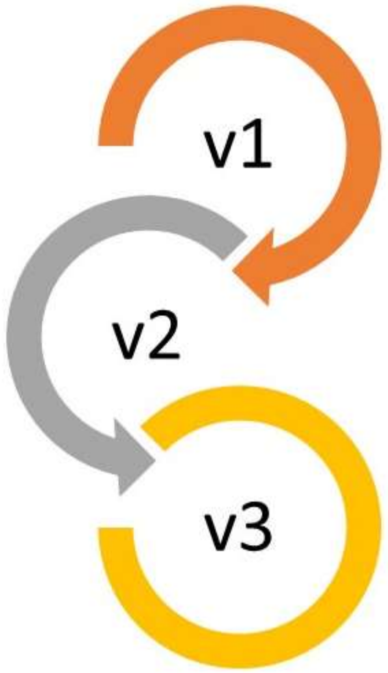


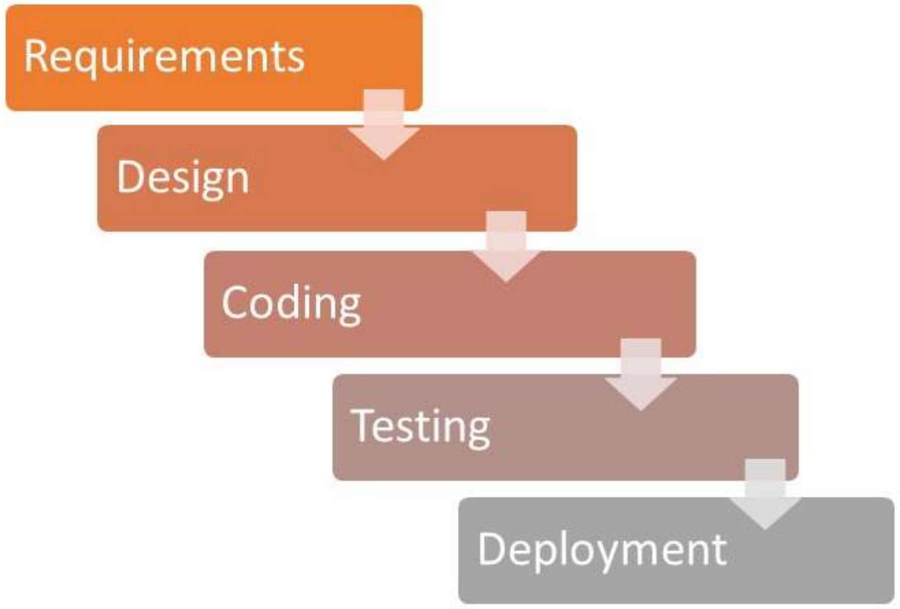
https://www.professionalqa.com/iterative-model

ITERATIVE MODEL

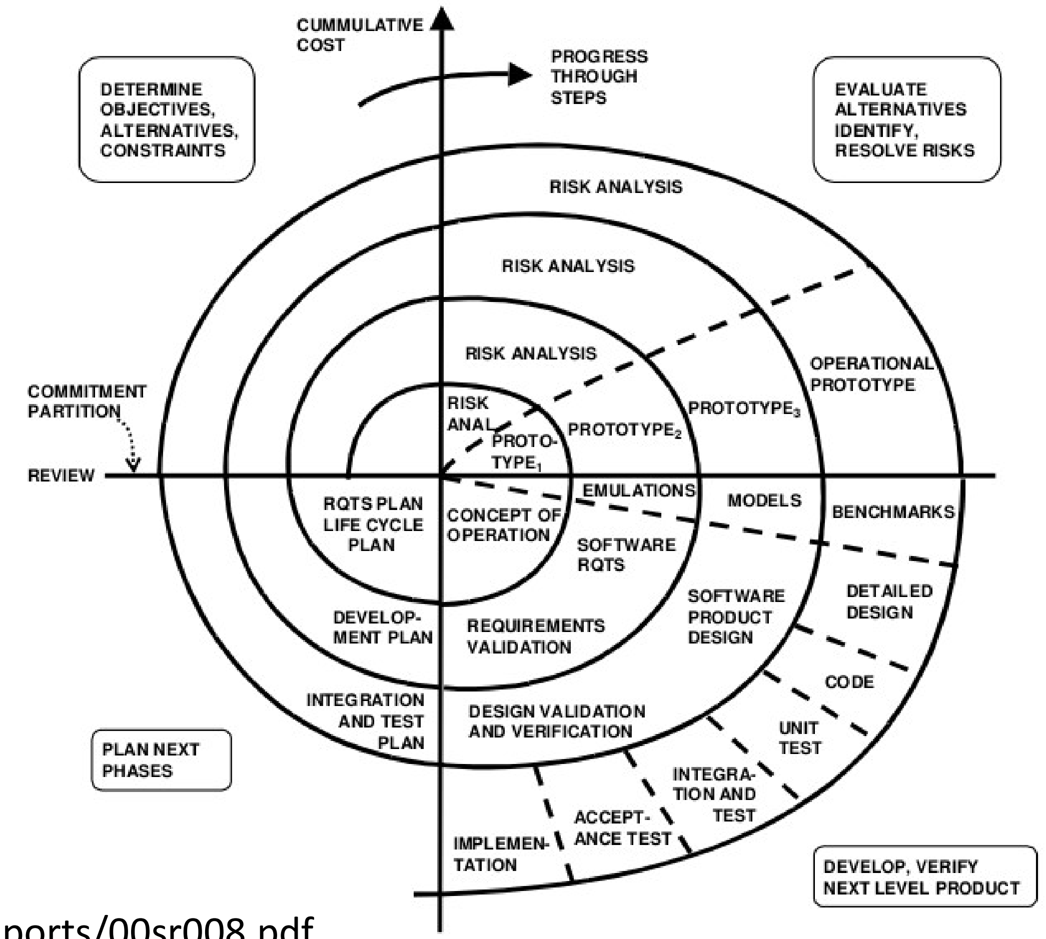
Advantages and Disadvantages of iterative model?

Waterfall vs Iterative





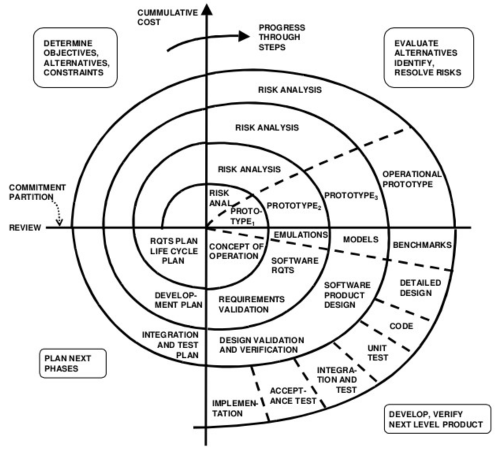
SPIRAL MODEL

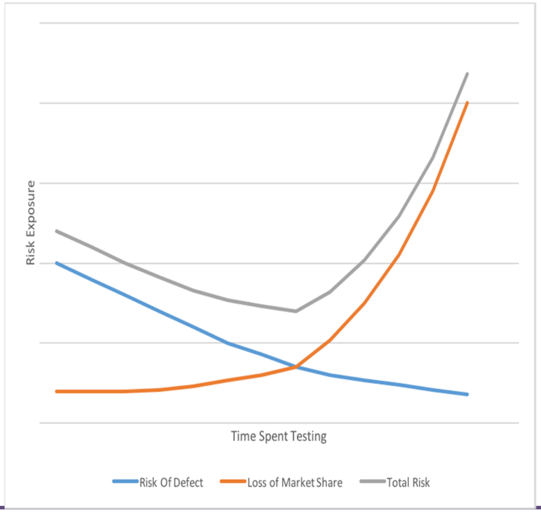


http://www.sei.cmu.edu/reports/00sr008.pdf
Image- https://airbrake.io/blog/sdlc/spiral-model

SPIRAL MODEL: Six invariant characteristics

1. Define Artifacts Concurrently (AKA “Plan everything, then re-plan those plans, then plan some more.”)
2. Four Essential Spiral Tasks
    - Consider critical-stakeholder objectives and constraints.
    - Elaborate and evaluate alternatives for achieving your objectives.
    - Identify and resolve risks attendant on choices of alternative solutions.
    - Stakeholders’ review and agree to proceed based on satisfaction of their critical objectives and constraints.
3. Risk determines level of effort
4. Risk determines degree of details- the potential for risk should determine how much attention you pay to the details of the project you’re working on.
5. Use the anchor point milestones
    - Life Cycle Objectives (LCO)
    - Life Cycle Architecture (LCA)
    - Initial Operational Capability (IOC)
6. Focus on the system and its life cycle (Or “Software Isn’t Everything”)





SPIRAL MODEL

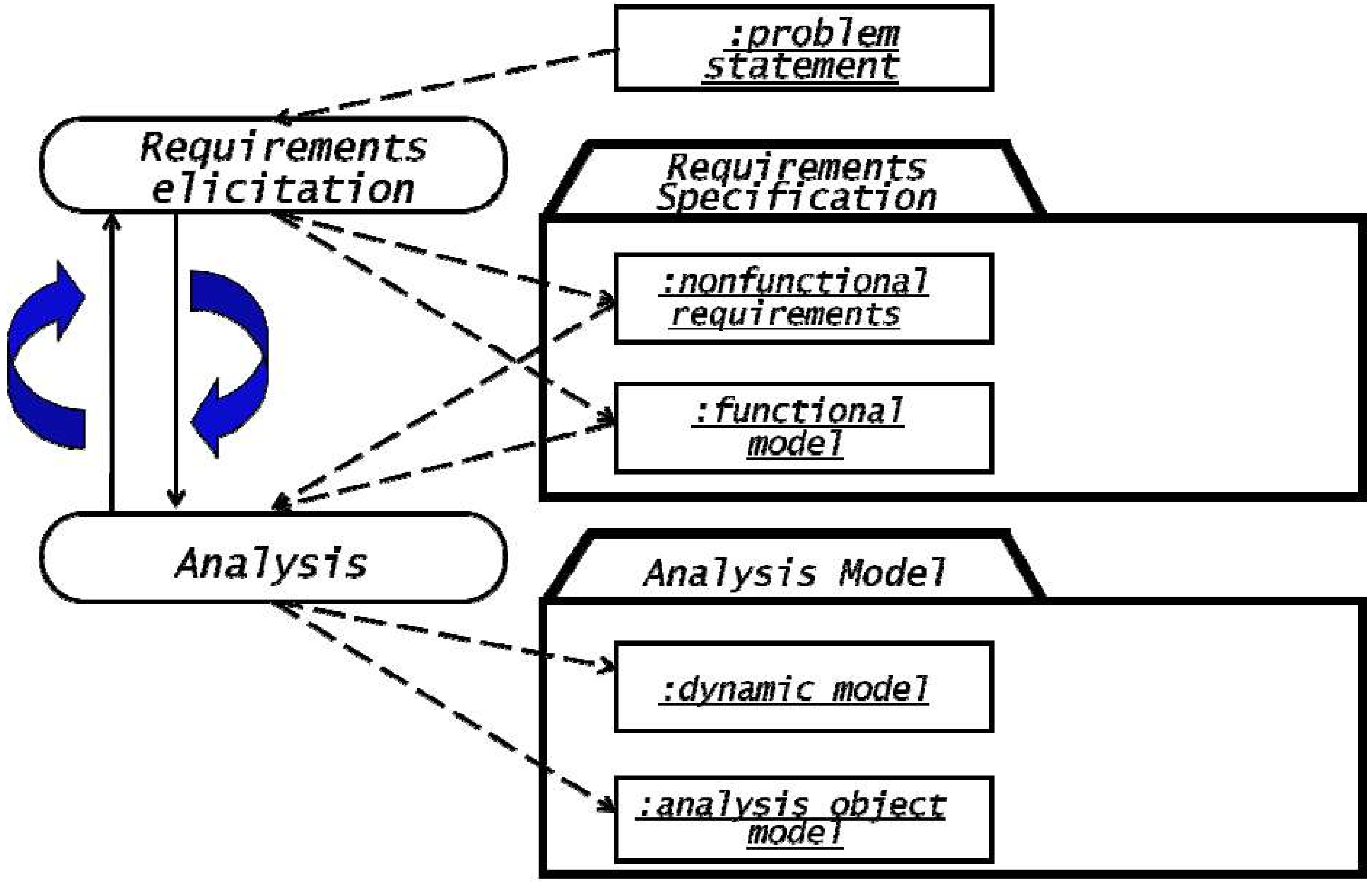
Advantages and Disadvantages of spiral model?

SPIRAL MODEL

Discussions: Spiral Model Vs Iterative Model

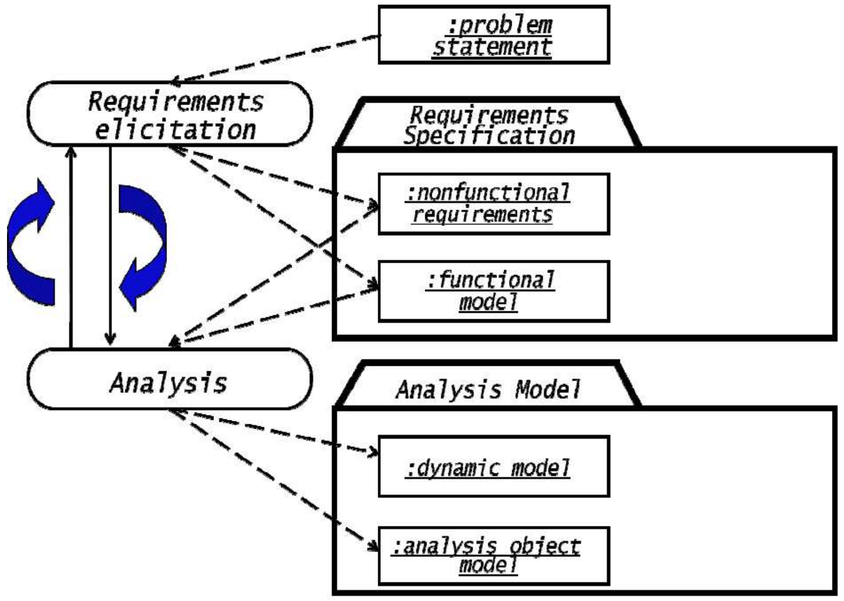
Introduction to Requirements Engineering

REQUIREMENTS ELICITATION AND ANALYSIS



REQUIREMENTS ELICITATION AND ANALYSIS

- A systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system
- In short, doing it iteratively and skillfully, and not being sloppy
- Prime challenges
  - Find
  - Communicate
  - Remember



REQUIREMENTS GATHERING METHODS/TECHNIQUES

- **One-on-One Interviews**
  - Identify stakeholders to be interviewed
  - Sit down with the clients and ask them what they need
  - Ask open-ended questions, as well as closed-ended questions

- **Group Interviews**
  - Similar to one-on-one interviews except more than one person being interviewed
  - Need more preparation and more formality
  - Generally agreed issues and differ issues
  - Advantages: fast- more requirements generated within short time, overlooked issues discovered
  - Disadvantages: difficult to find common time of many people

REQUIREMENTS GATHERING METHODS/TECHNIQUES

- **Questionnaires/Surveys**
  - To collect information from many people in relatively short amount of time
  - Automated tools can be used
  - Useful when stakeholders in remote locations or those that will have only minor input into the overall requirements
  - When using questionnaires, the questions should be focused and organized by a feature or project objective
  - Questionnaires should be not be too long, to ensure that users will complete them
  - How, where, when, who, what, and why
  - For how: *"How will you use this feature?" "How might we meet this business need?" "How will we know this is complete?"*
  - For where: *"Where does the process start?" "Where would the user access this feature?" "Where would the results be visible?"*
  - For when: *"When will this feature be used?" "When will the feature fail?" "When will we be ready to start?"*
  - For who: *"Who will use this feature?" "Who will deliver the inputs for the feature?" "Who will deliver the outputs of the feature?"*
  - For what: *"What do I know about this feature?" "What does this feature need to do?" "What is the end result?" "What must happen next?"*

REQUIREMENTS GATHERING METHODS/TECHNIQUES

- **User Observation**
  - The analyst may observe the user themselves (working culture and practices)
  - Very useful when client unable to explain what they want
  - Useful for understanding environment

- **Analyzing Existing Documents**
  - Reviewing the current process and documentation can help the analyst understand the business, or system, and its current situation
  - Information about the titles and names of stakeholders who are involved with the system
  - Help to formulate questions for interviews or questionnaires
  - Useful for supplement to information obtained from interviews, questionnaires, and observations

REQUIREMENTS GATHERING METHODS/TECHNIQUES

- **Joint Application Design (JAD)**
  - Application Development (JAD) was introduced in the late 1970s
  - JAD method reported up to 40% saving time for design
  - User and other key members are heavily involved in the process
  - Participants: JAD session leader (also known as the facilitator), users, managers, sponsors, systems analysts, scribe, and other IS staff members.
  - Facilitator manages the entire process, manage agenda and resolving conflicts
  - Facilitators do not contribute ideas; they solicit the ideas of the group.
  - Benefits:
    - Consolidation of months of work into a structured workshop
    - Clarifies specification requirements in an environment of consensus
    - Identifies open issues and parties responsible for their resolution
    - Ties together the steps of the design process in a concise document
    - Increases user satisfaction by directly involving users in the design process
    - Builds commitment through the use of the executive sponsor
    - Builds a sense of belonging and helps create a cohesive team

REQUIREMENTS GATHERING METHODS/TECHNIQUES

- **Although JAD (AJAD)**
  - A traditional JAD setting will be in a conference room of some type, with a “U” shaped table. There will be documents on the table, white boards used to write ideas, and agenda posted on the wall and a screen for projecting.
  - An in AJAD session, users be still be seated in a “U” shaped table, but will have computers in front of them, instead of documents.
  - Purpose - promote team interaction
  - Uses groupware packages during their sessions
  - Groupware capabilities: information generation, information analysis/decision making, and information documentation
- **Prototyping**
  - Iterative process that heavily involves the users
  - User will be sitting side by side
  - Prototype is build based on user’s requirements
  - The last prototype will be used as a model to build the actual system

Final Note: The method in gathering the requirements may vary depending on the situation and constraints but using the various methods to supplement each other will help in achieving complete requirements.

REQUIREMENTS GATHERING: NON-FUNCTIONAL

FURPS

F = Functional

U = Usability

R = Reliability

P = Performance

S = Supportability

FURPS+

Functional
Usability
Reliability
Performance
Supportability
Implementation
Interfaces
Operation
Packaging
Legal

- The FURPS+ model adds a few more categories.

FURPS+

Functional

- Features
- Capabilities
- Security

Usability

- Level of user expertise
- User interface standards
- Documentation

FURPS+

Reliability

- Safety and security requirements
- Availability, robustness, and reliability of the system
- Exception handling
- Mean time between failures
- Error tolerance
- Data loss tolerance

$$\text{MTBF} = \frac{\text{Total uptime}}{\text{Number of failures}}$$

FURPS+

Performance

- Number of concurrent users supported
- Response time
- Number of transactions per second

Supportability

- How will system be extended?
- Who maintains the system?

FURPS+

Implementation

- Which platform?

Interfaces

- Interfaces to existing systems.
- Protocols used.

Operation

- Who manages the running system?

FURPS+

Packaging

- Who installs the system?
- How many installations are there?

Legal

- Licensing?
- Liability issues?
- Licensing fees or liabilities incurred from using third-party components or algorithms?

REQUIREMENTS ELICITATION – SOURCE OF INFO

Source of information

- Client-supplied documents
- Manuals
- Technical documentation of legacy systems
- End-user discussions

IMPORTANCE OF REQUIREMENT ENGINEERING

- Prevents Miscommunication
  - Bridges the gap between stakeholders and developers.
  - Avoids misunderstandings about what the system should do.
- Saves Time and Costs
  - Early detection of issues reduces rework and delays.
  - Minimizes risks associated with unclear or incomplete requirements.
- Enhances Software Quality
  - Clear requirements lead to better design and testing.
  - Aligns system functionality with user expectations.
- Facilitates Project Success
  - Helps manage scope and expectations.
  - Improves stakeholder satisfaction and trust.

END OF UNIT 01