UNDERSTAND THE PROBLEM BEFORE YOU BUILD THE SOLUTION

Creating Software that Matters
with Requirements Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Requirements Engineering

A Brief Introduction

Julian **Frattini**, Ph.D.
Chalmers University of Technology, Sweden
https://julianfrattini.github.io/

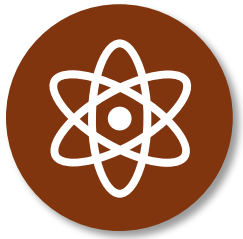
JulianFrattini/**seminar-refun** Public

Motivation

Starting a software development project

Minimizing **wasted effort** and costly rework

Goals



Understand the **fundamentals** and **definitions**

Understand the **impact** of requirements engineering

Apply **basic techniques** to specify requirements

Agenda

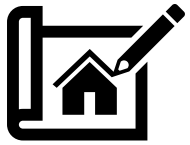
1. **Definitions** or: what exactly are requirements?
2. **Impact** or: why should I care about requirements?
3. **Application** or: how do I do requirements engineering?

Definitions

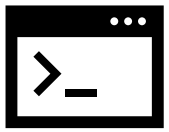
What exactly are Requirements?

Software Development Lifecycle

Requirements Specification



Architecture



Source Code

Test Cases

Product

Requirement & Requirements Artifact

Requirement:

1. A need or constraint imposed by a stakeholder.
2. A capability or property that a system shall have.

(Requirements) Artifact: A documented representation of a (1) need, constraint, (2) capability or property.

REQ1: When a user enters the webpage, the login option shall be highlighted.

REQ2: The system shall be secure and comply to data privacy guidelines

Adapted from [1] and [2]

Requirements Engineering

Requirements Engineering (RE) is the systematic, iterative, and disciplined approach to **develop an explicit requirements specification** that all stakeholders agree upon.

Based on [1]

Levels of Abstraction

What is the relationship between the following two statements?

REQ1: The system shall be secure.

is refined to

REQ2: Communication between users of the system shall not be accessible to external actors.

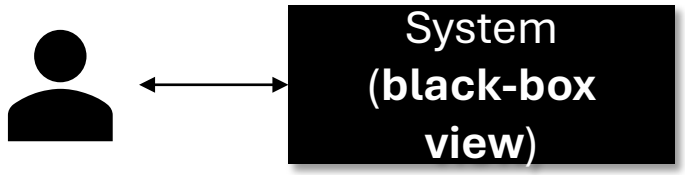
Statements exist on different **levels of abstraction**.

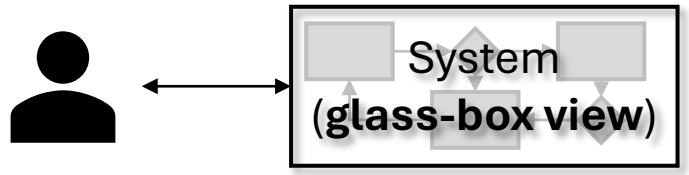
Levels of Abstraction

Context Layer (why?)	Requirements Layer (what?)	System Layer (how?)
Project scope, stakeholders, goals, ...	**REQ1**: The system shall b **REQ2**: Communication between users of the system shall not be accessible to external actors.	Data model, system architecture, ...

In scope of **requirements engineering**

In scope of **subsequent phases**, e.g., software architecture





Based on [3]

Impact

Why care about requirements?

Cost of Defect Removal

Underspecified Requirements Artifact

Suboptimal Architecture

Faulty Implementation

Misleading Test cases

Unsatisfying Product

The cost of removing a defect from an artifact **scales approximately by the factor 10** for each phase that it survives.

Based on [4] and [5]

Problem- versus Solution-Space

Problem-Space

Why should the system do something and *what* should it do?

Solution-Space

How should the system do it?

Problem- versus Solution-Space

Problem Space **Solution Space**

The system shall be secure.

The system shall perform well with a large number of concurrent users.

The system's architecture will contain a broker-pattern at the client-server interface with at least 5 subscribed servers

Large scale maintenance and/or an upgrade shall give the possibility to reach a lifetime of 50 years.

The primary data storage subsystem will adhere to active redundancy.

All subsystems shall not lose more than 4 hours of acquired or processed measurement data (not yet permanently stored) as a result of an outage in the external power supply.

All communication shall be encrypted with SHA-2.

Problem- versus Solution-Space

UNDERSTAND THE PROBLEM BEFORE YOU BUILD THE SOLUTION

REQ1: The purpose of the website is to make static information available.

SOL1: The website will be implemented using the *Hugo* framework.

Good choice for *static* websites

SOL2: The website will be implemented using the *Svelte* framework.

Good choice for *interactive* websites

For every solution-space statement you receive, **first determine the problem** you are trying to solve.

Insights

Every project has requirements …

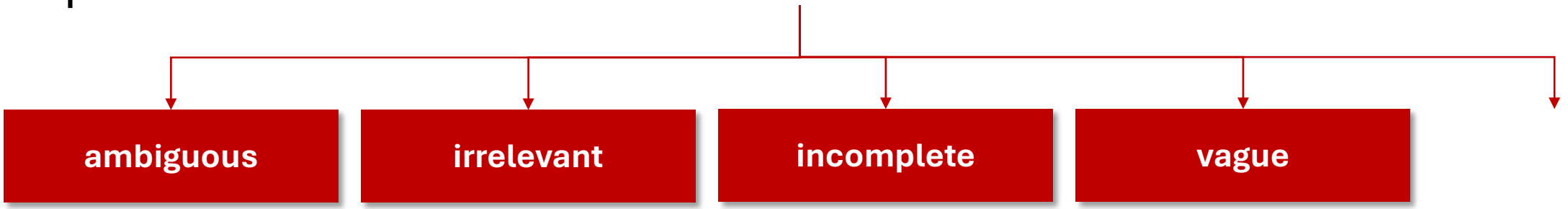
… but not every team decides **to write them down**.

Application

How to Requirements Engineering?

Motivation

It is desirable to specify requirements, but these requirements need to be **free of defects**.



Rather than eliciting requirements all-at-once, we can **incrementally** elicit and refine them.

Techniques

Stakeholder Elicitation

Goal Modeling

System Vision

Requirements Elicitation

Stakeholder Elicitation



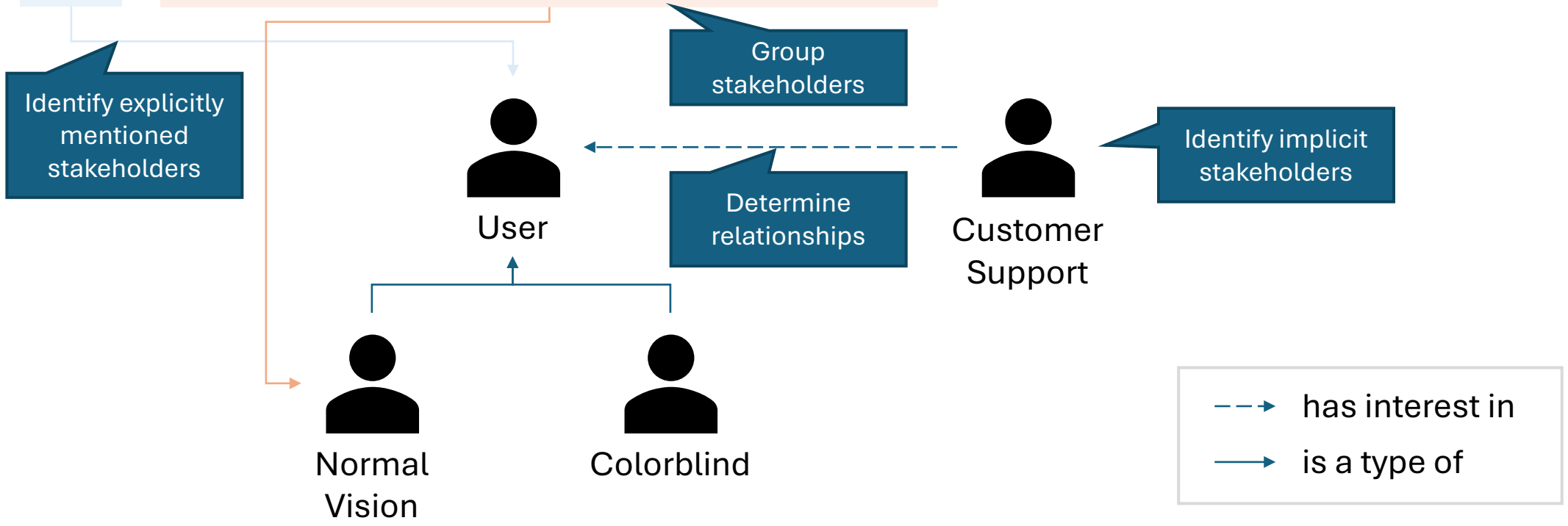

A **stakeholder** is a person or a group of persons, interest group, or organization that has to a certain extent **interest in the system** to be developed, or that takes/should take influence on the system's development.

Elementary Steps:

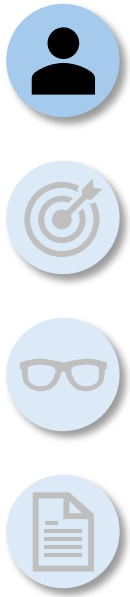
1. **Elicit stakeholders**: list all relevant stakeholders
2. **Elicit relationships**: make relationships between stakeholders explicit

Stakeholder Elicitation

”A user – regardless of their ability to see colors – can log into the system.”



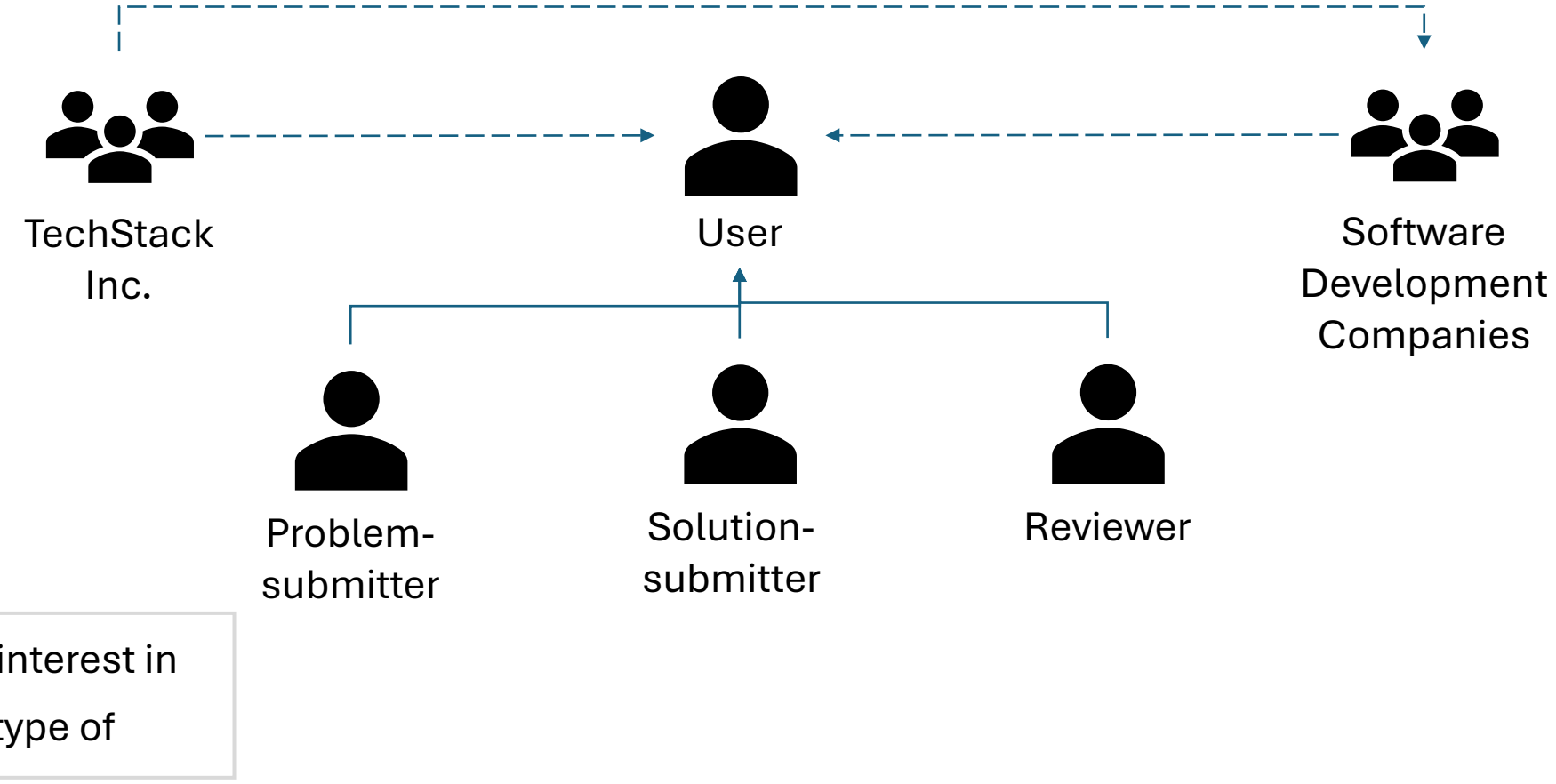
Stakeholder Elicitation



Identify all stakeholders in the following system description:

TechStack Inc. issued the development of a platform where users can upload coding problems. Other users can submit solutions to those problems and a third set of users reviews the solutions and ranks them by code quality. Problem-submitting users get to see the ranked solutions, solution-submitting users get credits depending on the quality of the solution, and reviewers get credit based on the overlap between their ranking and the overall ranking of solutions. Software companies can pay to get the contacts of well-performing users (for targeted hiring).

Stakeholders



Goal Modelling

A **goal** is a prescriptive **statement of intent**, i.e., it describes an abstract property that a system must fulfill.

Goal types:

1. **Usage Goals**: Goals with immediate relevance for end users which serve as basis for inference of user requirements

2. **System Goals**: Goals directed at system properties and capabilities (typically quality, i.e., non-functional
3. **Business Goals**: All organization-specific (strategic) goals with relevance to the project

Goal Modelling

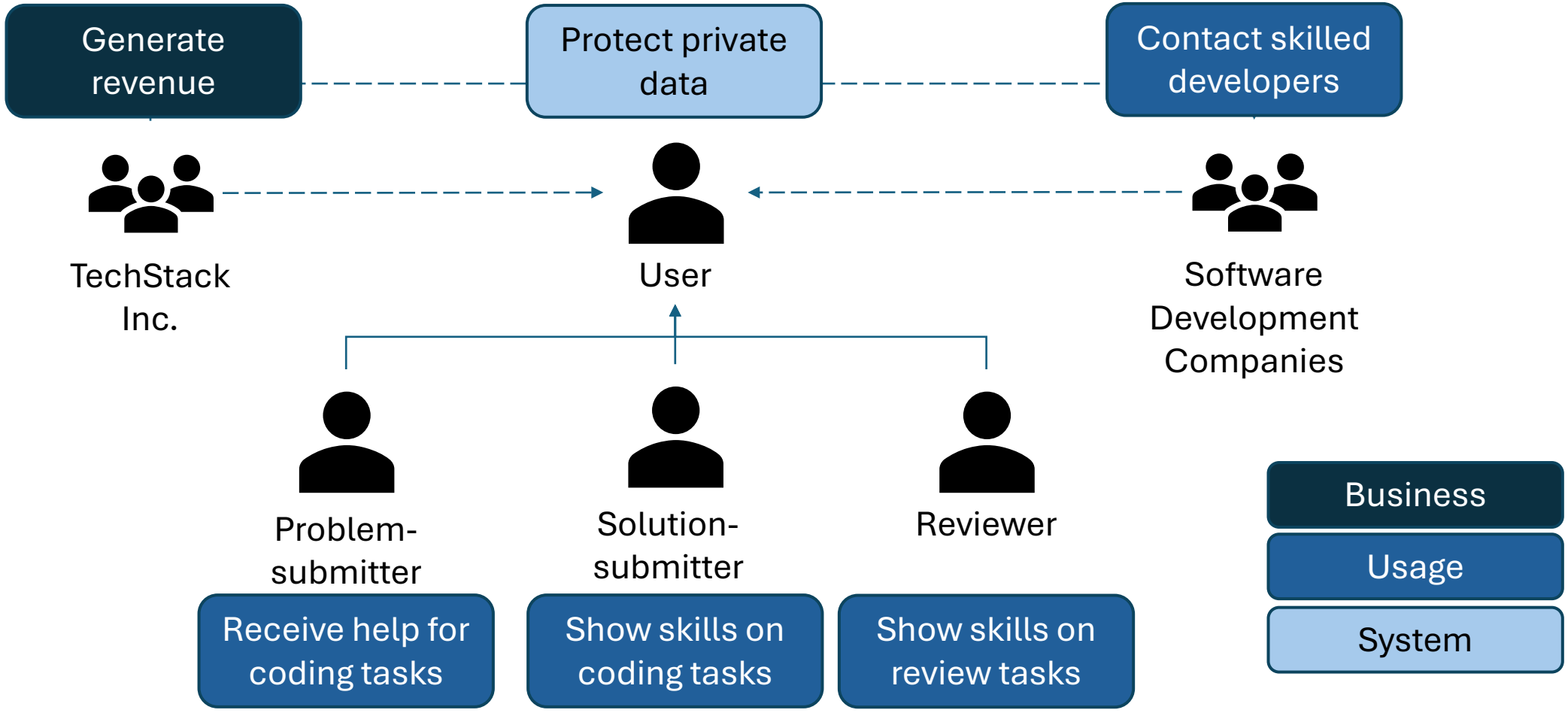
For each of the following stakeholders in the previously mentioned scenario, **identify one goal** and classify it as a usage, system, or business goal.

TechStack Inc.

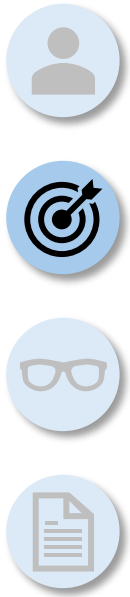
Solution-submitter

1. **Usage Goals**: Goals with immediate relevance for end users which serve as basis for inference of user requirements
2. **System Goals**: Goals directed at system properties and capabilities (typically quality, i.e., non-functional
3. **Business Goals**: All organization-specific (strategic) goals with relevance to the project

Goal Modelling



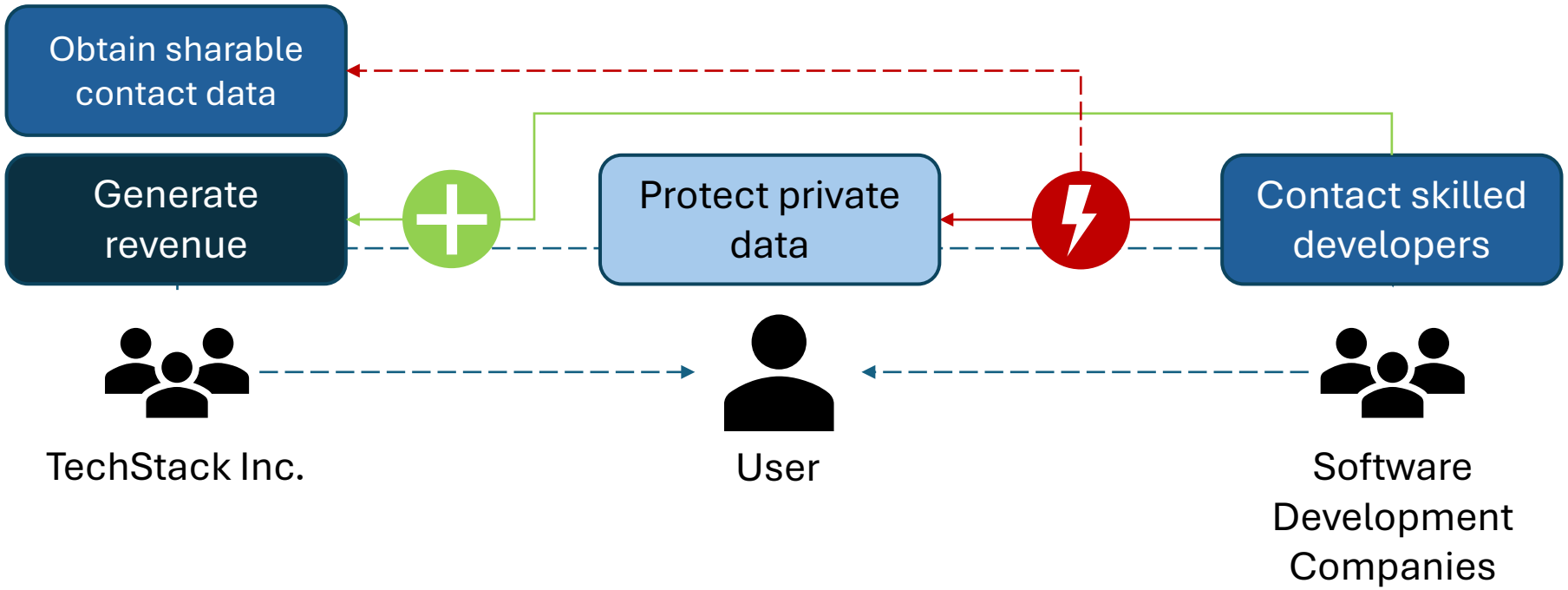
Goal Refinement



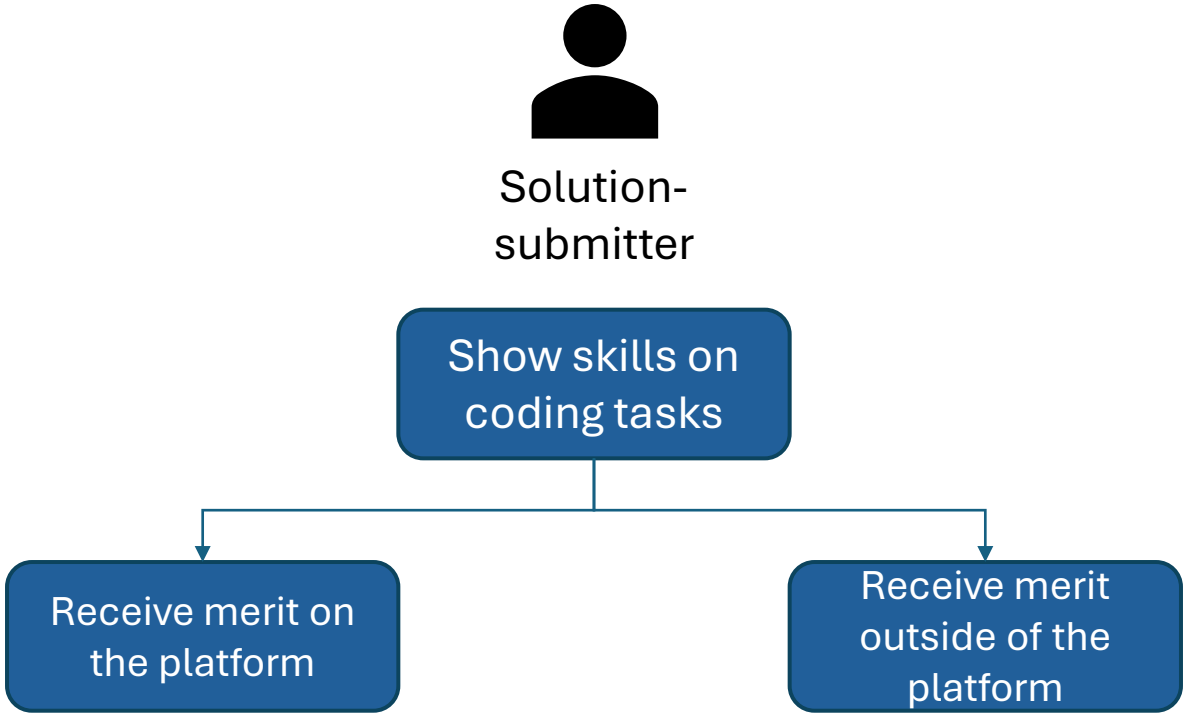
Goals can be further **refined** with the following relations:

- **Conflict**: one goal may conflict with another goal
- **Support**: one goal may support another goal
- **Refinement**: one goal can be decomposed into more specific sub-goals

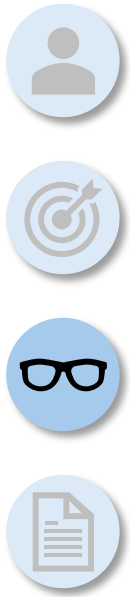
Goal Relationships



Goal Refinement



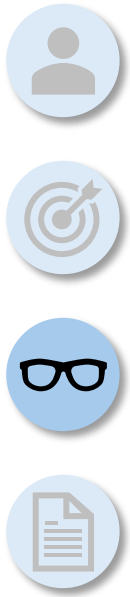
System Vision



A **system vision** is the transition point between the context specification and the requirements specification. Its main purpose is to

- give a **comprehensive overview** of the most important *use cases* and
- boundaries, thus it clearly defines the **scope of the system**. It clearly distinguishes which parts belong to the system and which parts are external.

System Vision

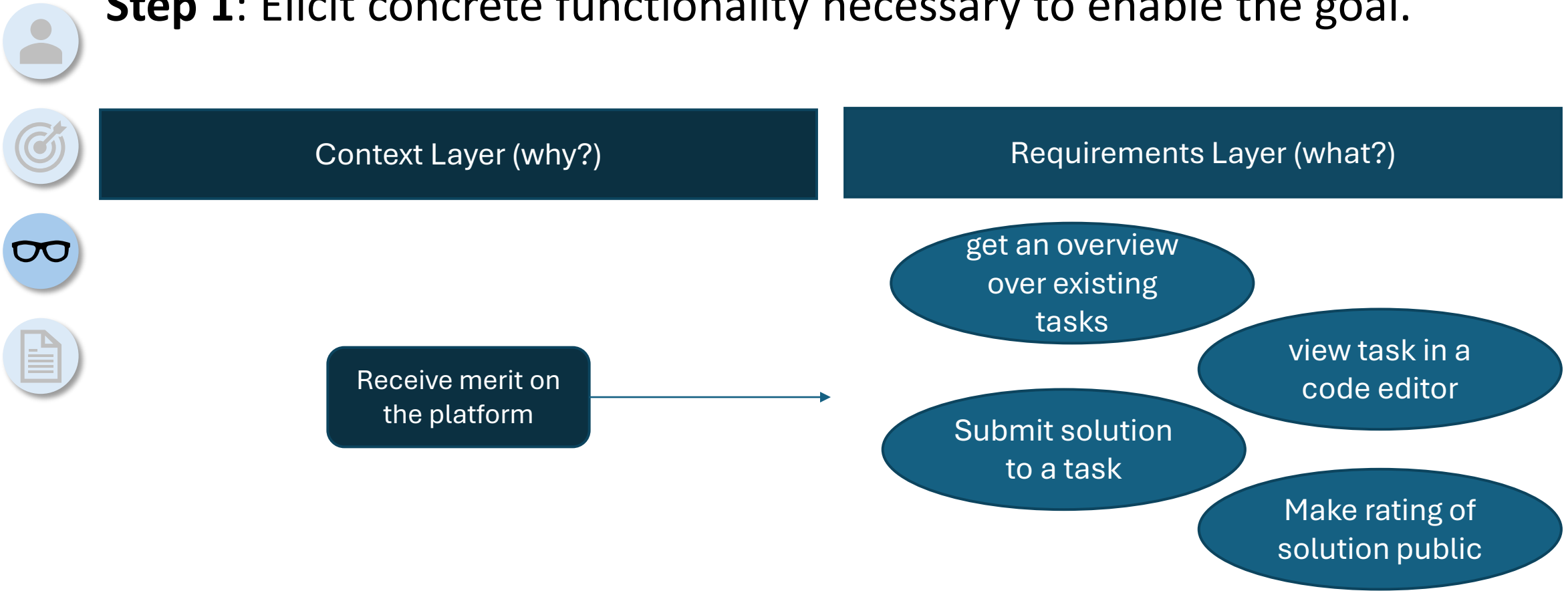


Use Case Diagram procedure:

1. Elicit **concrete functionality** necessary to enable the goal.
2. **Connect it to stakeholders** that are involved with that use case.
3. Determine, whether the use case is **part of the system or external.**

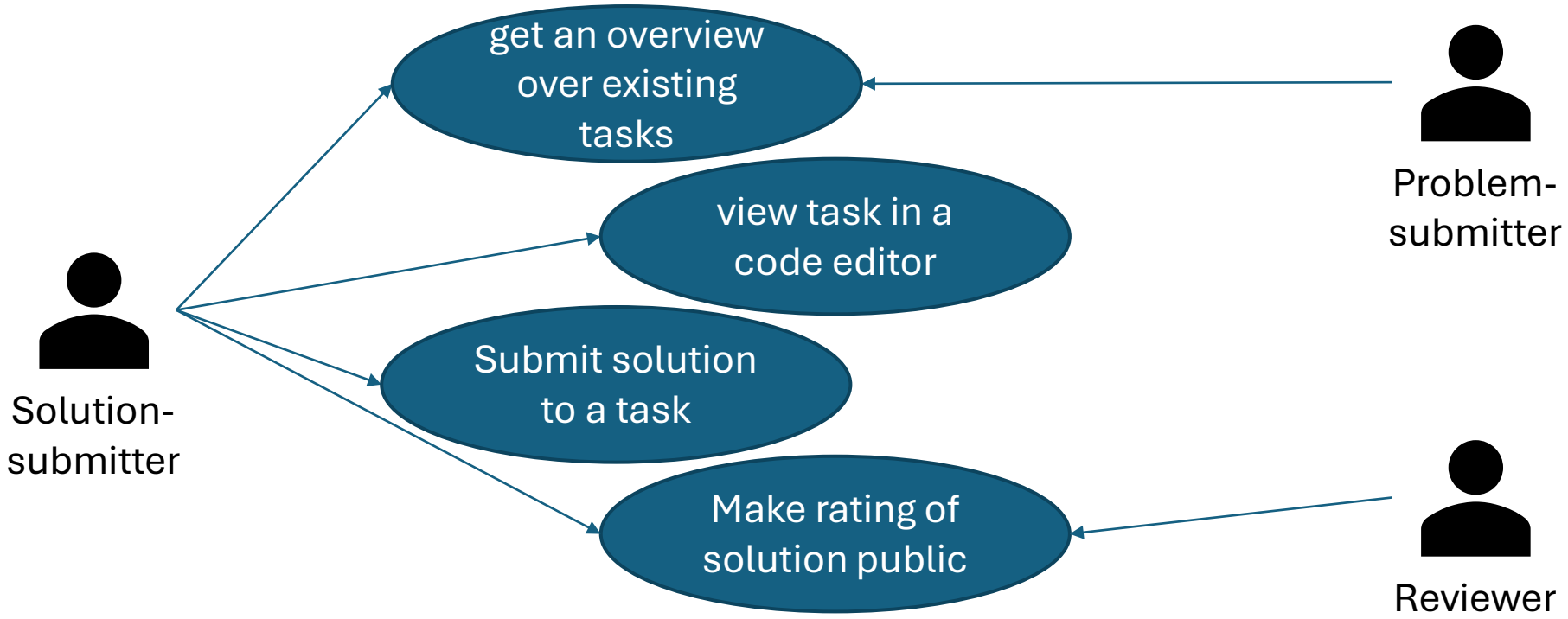
System Vision

Step 1: Elicit concrete functionality necessary to enable the goal.



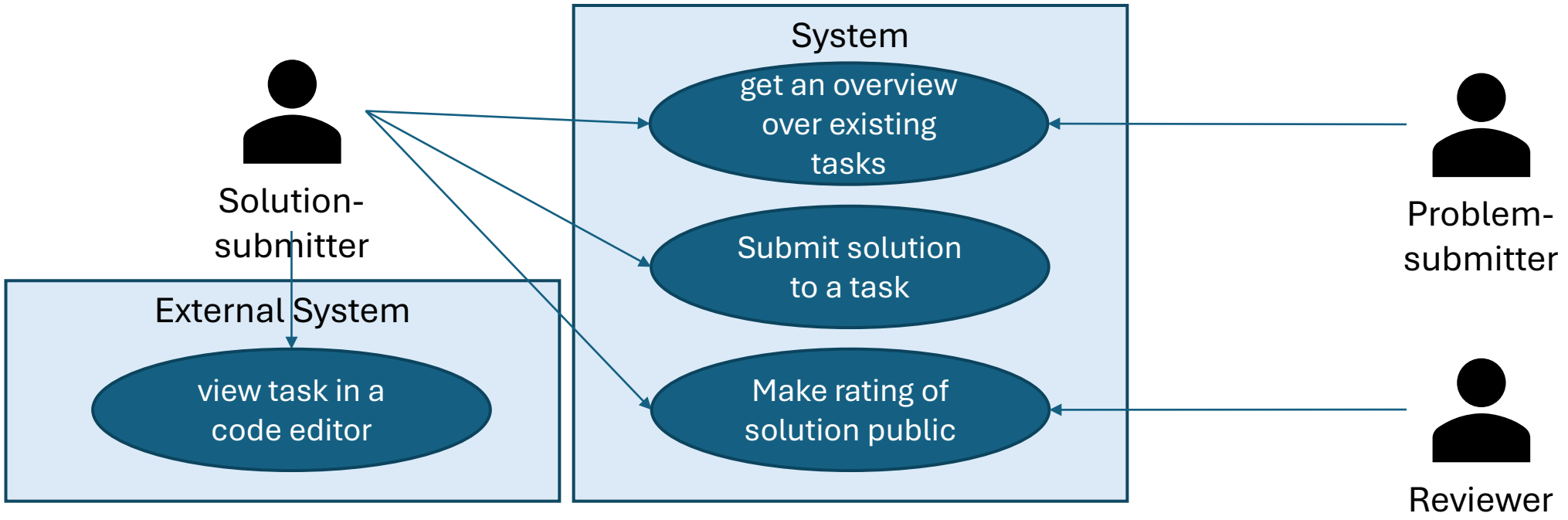
System Vision

Step 2: Connect it to stakeholders that are involved with that use case.

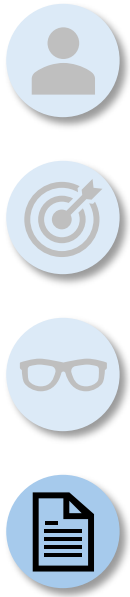


System Vision

Step 3: Determine, whether the use case is part of the system or external.



Requirements Elicitation



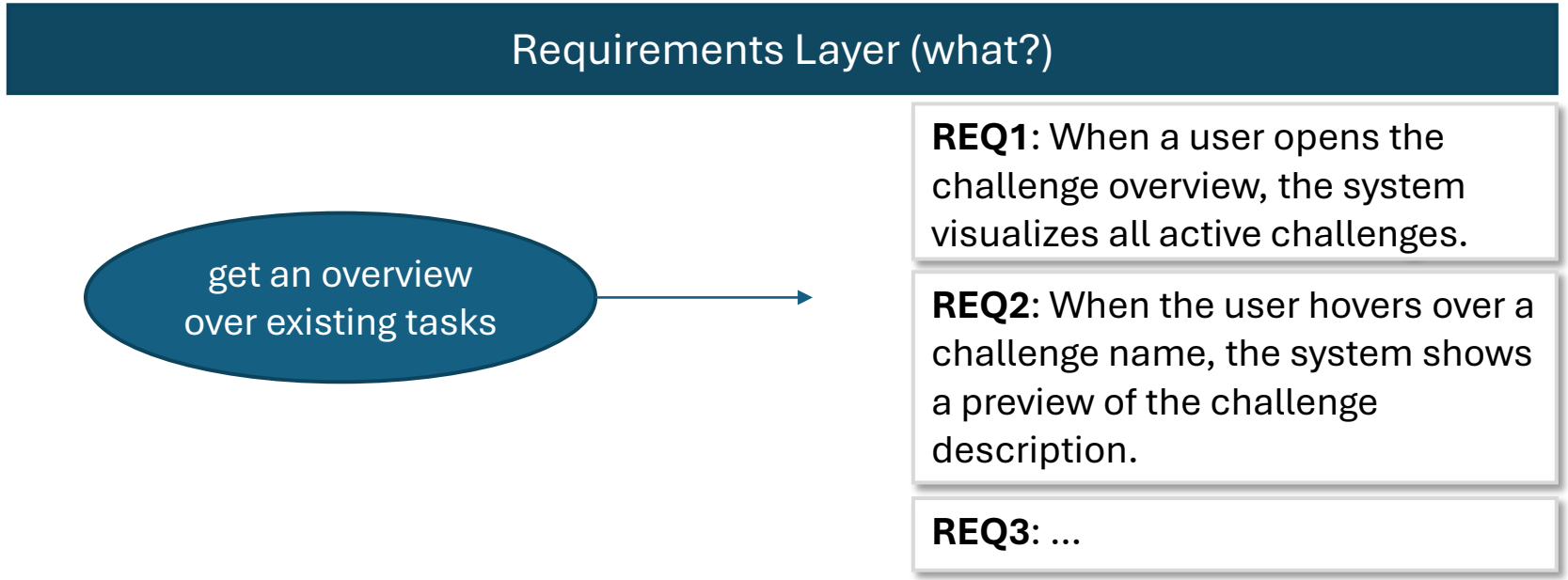
You can specify **functional requirements** using the following template:

> **REQ<id>**: When <stakeholder> <action>, then the system <reaction>.

Requirements Elicitation

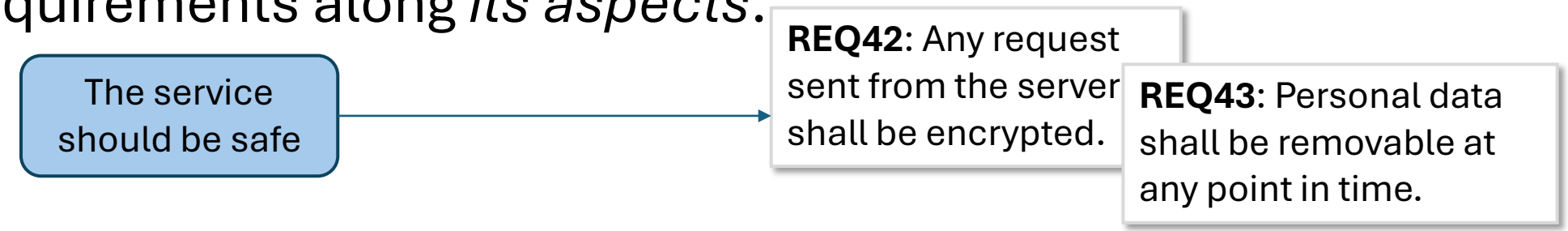
Refine features into **measurable, specific** requirements.

REQ<id>: When <stakeholder> <action>, then the system <reaction>.

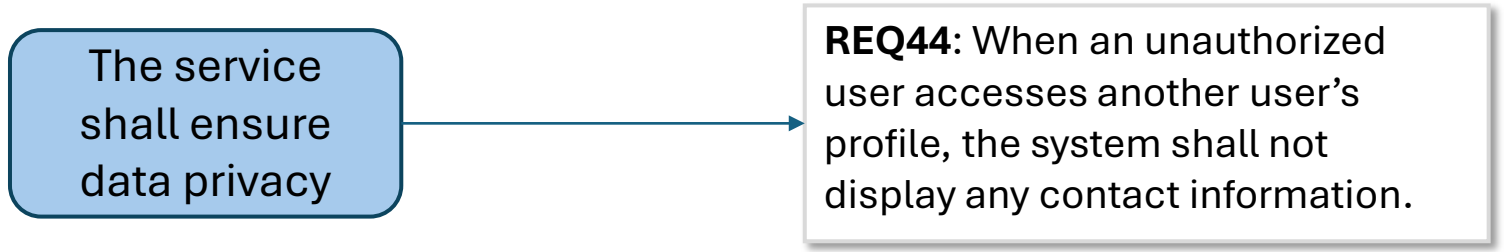


Requirements Elicitation for System Goals

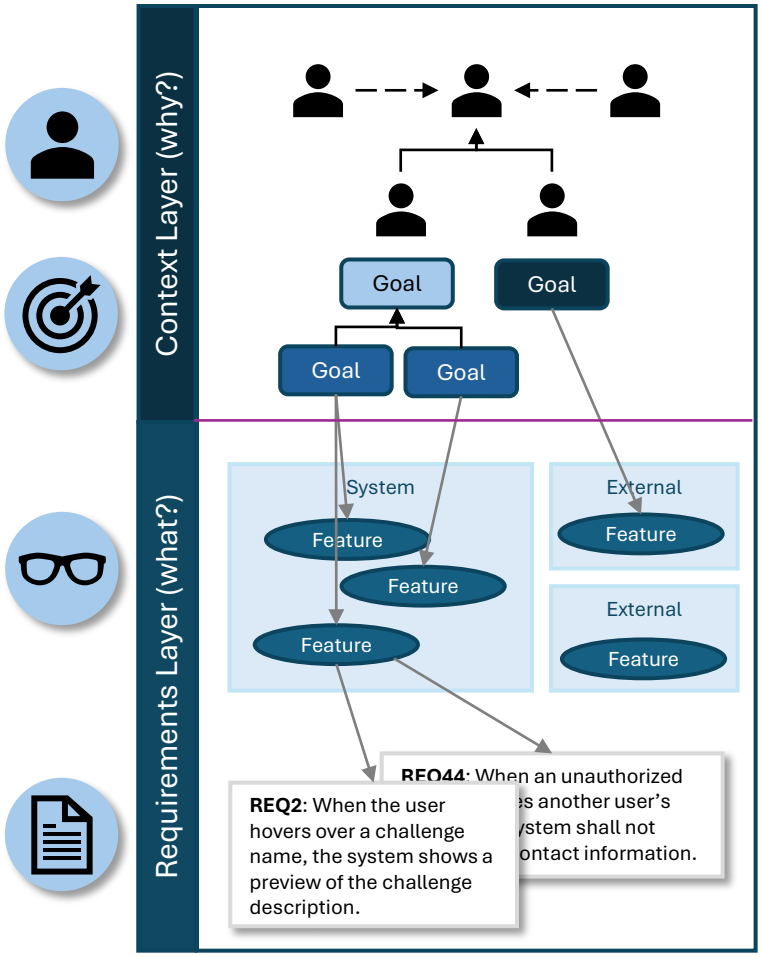
Approach 1: Decompose a system goal into measurable quality requirements along *its aspects*.



Approach 2: Specify a *misuse-case*, i.e., a functional requirement of what is not supposed to happen.



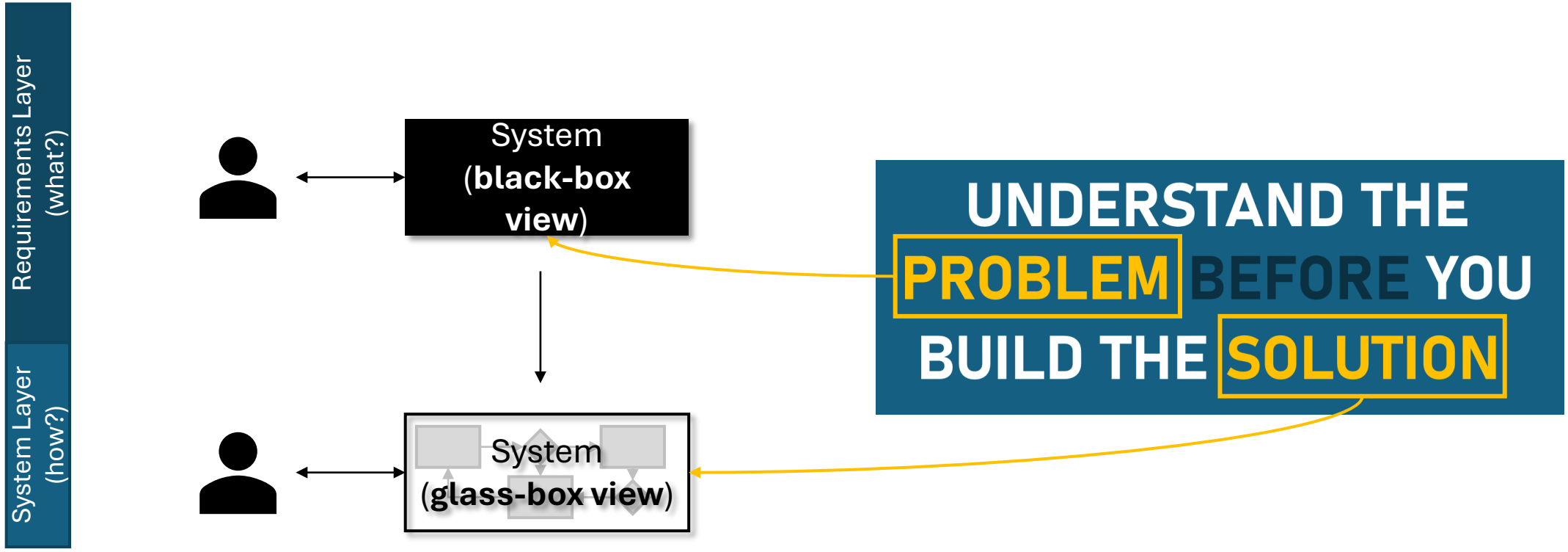
Requirements Engineering Template



Beyond Requirements Engineering

How to continue from Requirements

Using Requirements



Concluding Thoughts

Tailoring
there is no one-size-fits-all solution

Change
requirements are rarely static

Means-to-an-end
Requirements are no means-to-itself

Recommended Reading

Méndez Fernández, D., & Penzenstadler, B. (2015). Artefact-based requirements engineering: the AMDiRE approach. *Requirements Engineering*, *20*, 405-434. https://link.springer.com/article/10.1007/s00766-014-0206-y

Requirements Eng (2015) 20:405–434
DOI 10.1007/s00766-014-0206-y

CrossMark

ORIGINAL ARTICLE

Artefact-based requirements engineering: the AMDiRE approach

Daniel Méndez Fernández · Birgit Penzenstadler

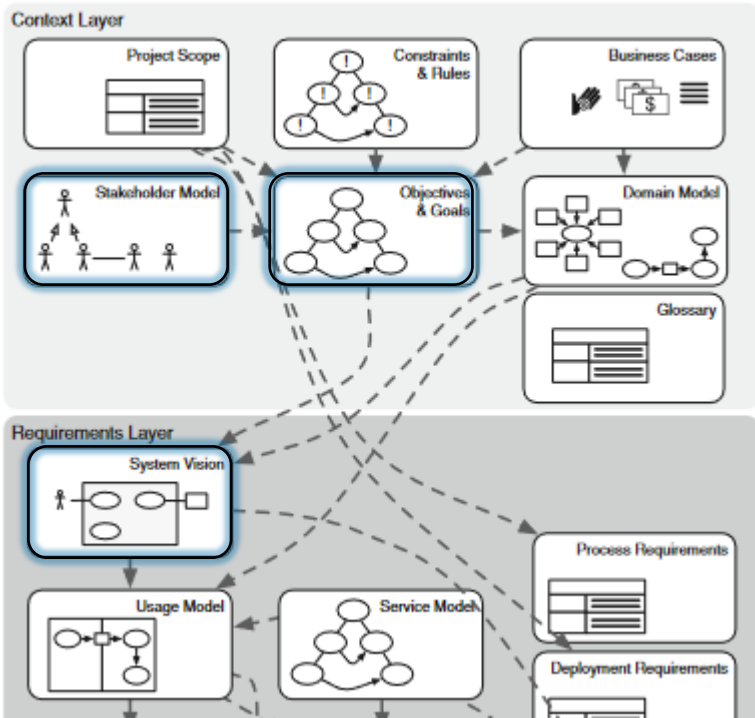
Received: 12 September 2013 / Accepted: 10 May 2014 / Published online: 28 May 2014

Abstract The various influences in the processes and application domains make requirements engineering (RE) inherently complex and difficult to implement. In general, we have two options for establishing an RE approach: We can either establish an activity-based RE approach, or we can establish an artefact-based one where project participants concentrate on the RE artefacts rather than on the way of creating them. While a number of activity-based RE

experiences we made during the development and different industrial evaluations, and lessons learnt.

Keywords Requirements engineering · Artefact orientation · Empirical evaluation

1 Introduction



References

[1] Glinz, M. (2011). A glossary of requirements engineering terminology. *Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version*, *1*, 56.

[2] Méndez Fernández, D., Böhm, W., Vogelsang, A., Mund, J., Broy, M., Kuhrmann, M., & Weyer, T. (2019). Artefacts in software engineering: a fundamental positioning. *Software & Systems Modeling*, *18*, 2777-2786.

[3] Méndez Fernández, D., & Penzenstadler, B. (2015). Artefact-based requirements engineering: the AMDiRE approach. Requirements Engineering, 20, 405-434.

[4] Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., ... & Wieringa, R. (2017). Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering*, *22*, 2298-2338.

[5] Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering*, (1), 4-21.