

# Requirement Engineering

## Lecture 11: Traceability

Prof. Dr. Benjamin Leiding  
M.Sc. Anant Sujatanagarjuna

# General Requirements Engineering Process Overview

Requirements Engineering					
Requirements Analysis				Requirements Management	
Elicitation	Negotiation	Documentation	Validation	Change Management	Tracing



# **Lecture 11: Traceability**

## **Content**

1. Introduction
2. Classification
3. Documentation



# INTRODUCTION

## Introduction

### Traceability in a Nutshell

**What** happened **when** to a/the requirement(s)?

## Introduction

### Definition – Requirements Traceability

“Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases).”

## Introduction

### Advantages of Traceable Requirements

- Change management → Which other artefacts are affected by a change?
- Process improvements → Trace problems in the development process back to their cause
- Reuse
  - Identify development artefacts associated with a requirement → If requirement is reused, the development artefact might also be reused
- Accountability
  - Calculate/estimate the development effort to implement a requirement
- Maintenance
  - Simplified cause-effect analysis, impact analysis, etc.

## Introduction

### Advantages of Traceable Requirements

- Verifiability
  - Easy to verify whether a requirement has been implemented or not
- Identification of gold-plated solutions in the system
  - Gold-plated = unnecessary attention to details
  - Reverse function to “verifiability” → Checks for each function whether it implements a requirement
- Identification of gold-plated solutions in the requirements
  - Tracing requirements to their origin
  - Analysis whether a requirement contributes to a goal



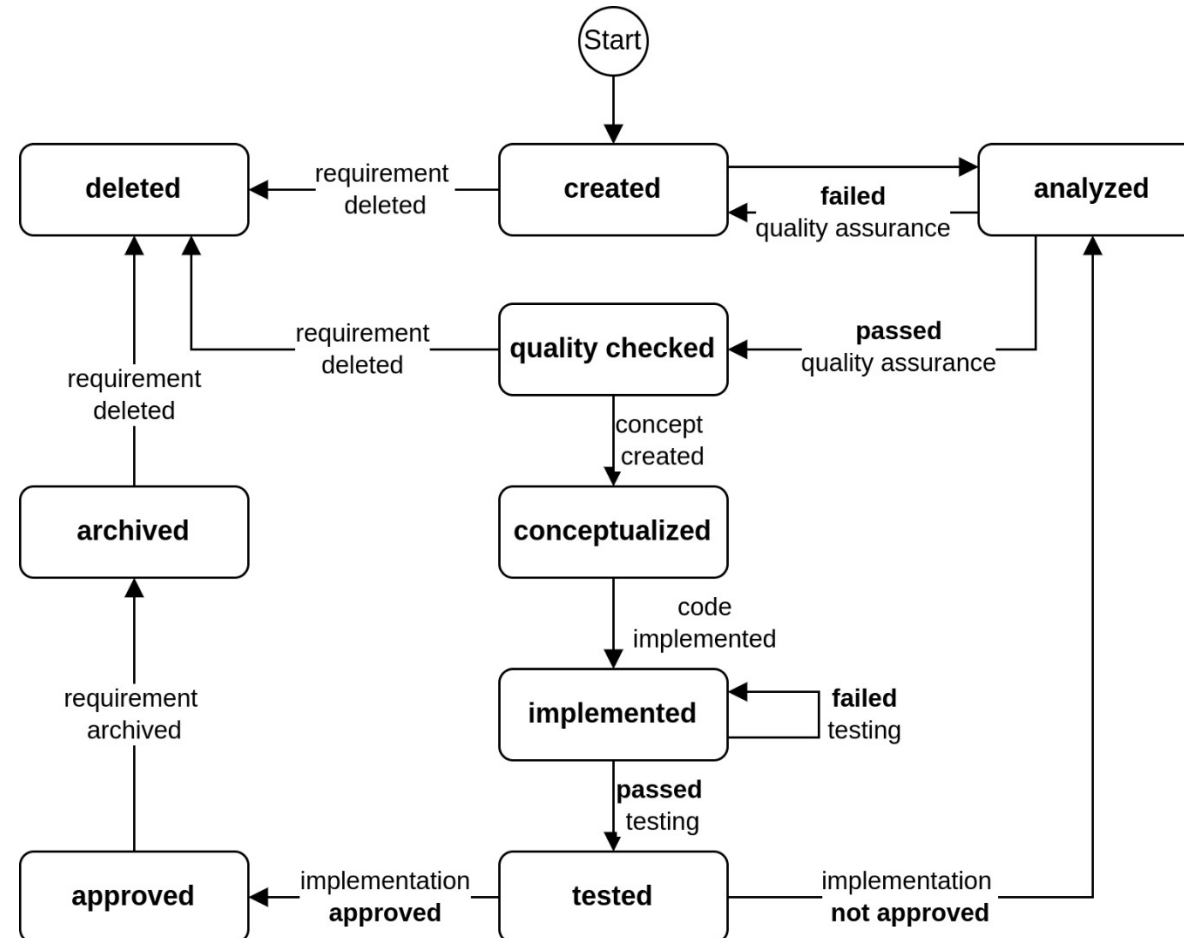
# Introduction

## Purpose-driven Tracing

- Extensive tracing is expensive
- Purpose-driven!
- Do not trace everything
- Trace according to needs → Too much/little information (sufficient level of detail)

# Introduction

## State Changes of a Requirement

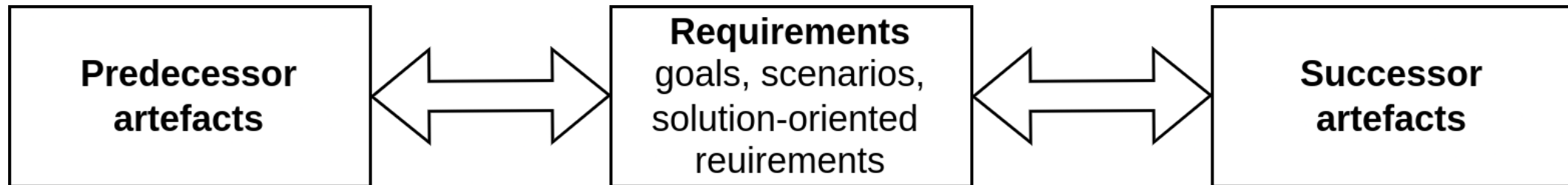




# CLASSIFICATION

## Classification

### Pre- and Post-Traceability



# Classification

## Overview

- Pre-requirements-specification (pre-RS) traceability
- Post-requirements-specification (post-RS) traceability
- Traceability among requirements
  - E.g., requirement **A** refines/generalized/replaces requirement **B**

# Classification

## Classes of Traceability Relationships

1 Condition

2 Content

3 Abstraction

4 Evolution

5 Miscellaneous

## Classification

### Traceability Relationships – Condition

- constraint:
  - E.g., artefact **A** defines a constraint on artefact **B**
- precondition:
  - E.g., artefact **A** defines a condition that must be fulfilled before artefact **B** can be realized

## Classification

### Traceability Relationships – Content

- similar:
  - Two associated artefacts are similar in content
- compares:
  - Artefact **A**<sub>1</sub> represents the result of a comparison of the artefacts **A**<sub>2</sub> ... **A**<sub>n</sub>
- contradicts:
  - Two artefacts cannot be realized together
- conflicts:
  - Artefact **A** may hinder (but not necessarily exclude) the realization of artefact **B**



## Classification

### Traceability Relationships - Abstraction

- classifies:
  - Artefact **A** classifies a set of artefacts **B<sub>1</sub> ... B<sub>n</sub>** → e.g., a goal classifies a set of solution-oriented requirements
- aggregates:
  - Artefact **A** is an aggregation of a set of other artefacts **B<sub>1</sub> ... B<sub>n</sub>**
- generalizes:
  - Artefact **A** is a generalization of (one or) several other artefacts → e.g., an abstract scenario (e.g., a type scenario) is a generalization of a set of more concrete scenarios (e.g., instance scenarios)

## Classification

### Traceability Relationships - Evolution

- replaces:
  - Artefact **B** replaces artefact **A**
- based\_on:
  - Artefact **A** has influenced the definition of artefact **B**
- formalizes:
  - Artefact **A** is a formal documentation of artefact **B** → e.g., relate a solution-oriented requirements model to a set of textual requirements
- refines:
  - Artefact **A** refines artefact **B**
- derived:
  - Artefact **A** was derived based on (a set of) other artefact(s)

## Classification

### Traceability Relationships - Miscellaneous

- example\_of:
  - Artefact **A** contains exemplary aspects of a set of artefacts → e.g., relates an interaction scenario to a set of solution-oriented requirements to document an exemplary sequence of interactions that a system implementing the solution-oriented requirements will support
- verifies:
  - Test artefact **A** verifies requirement artefact **B**
- rationale:
  - Artefact **A** justifies artefact **B** → e.g., text fragment contains justification for the existence of a scenario

## Classification

### Traceability Relationships - Miscellaneous

- responsible\_for:
  - Stakeholder (or role) **A** is responsible for the associated artefact **B**
- background:
  - Assign background information to a requirement artefact → e.g., standardization document relating to a solution-oriented requirement
- comment:
  - Relates any kind of information to a requirements artefact – use sparingly!

# DOCUMENTATION

# Documentation Overview

- 1 Textual references
- 2 Hyperlinks
- 3 Traceability models
- 4 Matrix
- 5 Graph

## Documentation

### Textual References

R2-17: For selecting the trip destination, the navigation system shall display the last ten trip destinations. [based\_on→R1-17] [...]

## Documentation

### Hyperlinks

R2-17: For selecting the trip destination, the navigation system shall display the last ten trip destinations.

hyperlink (type: conflicts)

R3-11: The system shall not store any information about the destinations of previous trips



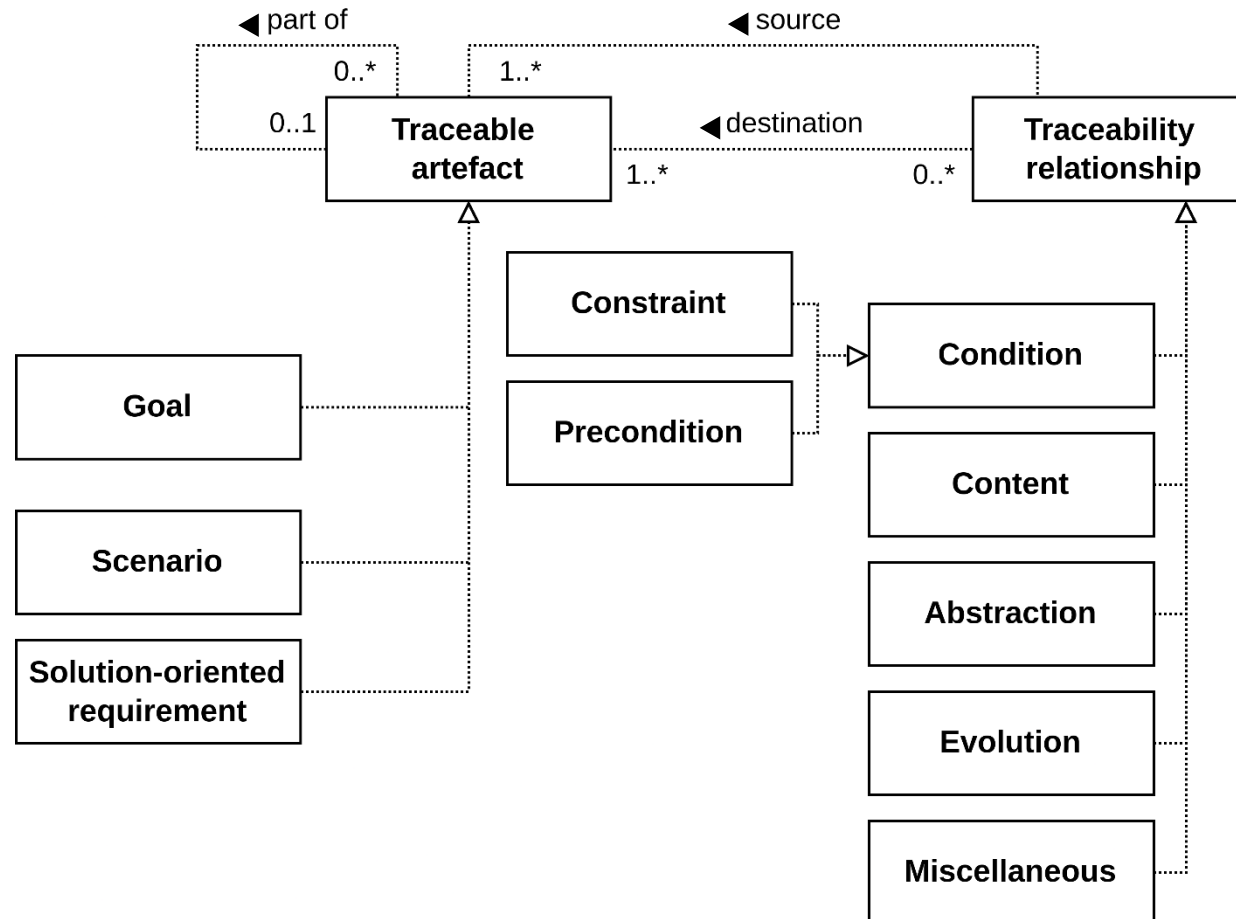
## Documentation

### Textual References & Hyperlinks

- Simple and easy
- Links are textually part of the requirements themselves
- Disadvantages:
  - Maintenance is time-consuming and tedious
  - Bidirectionality is difficult to achieve/maintain

# Documentation

## Traceability Models



## Documentation

### Traceability Matrix

		Target artefacts				
Source artefacts	satisfies	Goal 1	Goal 2	Goal 3	Goal 4	Goal 5
	Scenario 1					
	Scenario 2					
	Scenario 3			Traceability		
	Scenario 4			relationships		
	Scenario 5					

## Documentation

### Traceability Matrix

		Target artefacts				
Source artefacts		Goal 1	Goal 2	Goal 3	Goal 4	Goal 5
	Scenario 1	satisfies				
	Scenario 2	based_on	conflicts		satisfies	
	Scenario 3		satisfies			
	Scenario 4	conflicts		satisfies		satisfies
	Scenario 5		satisfies		based_on	

## Documentation

### Traceability Matrix

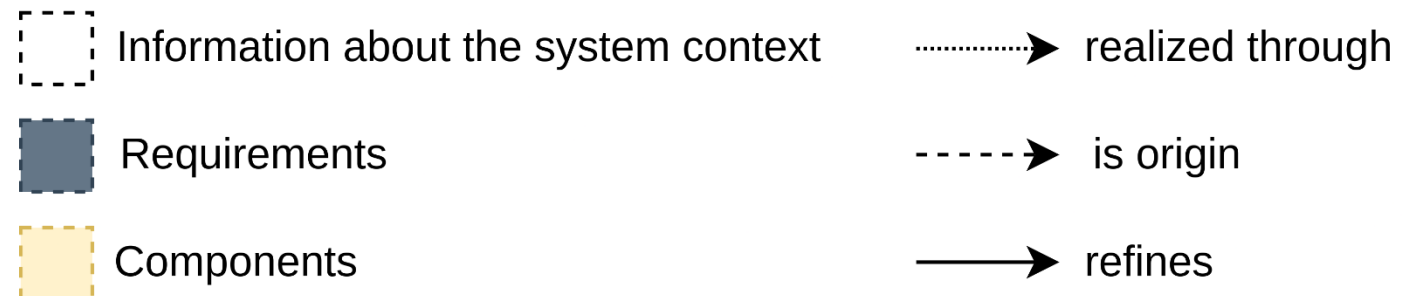
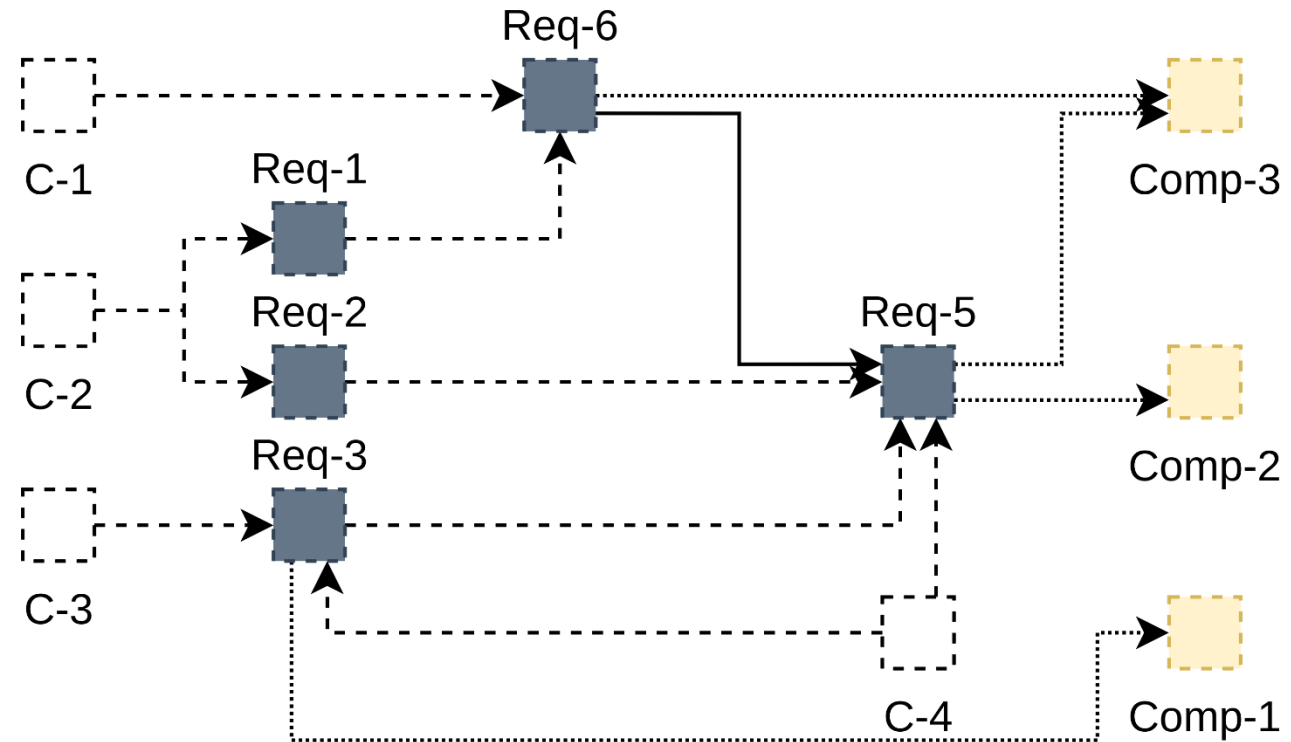
- Documents traceability in a matrix
- Rows represent the initial artefact
- Columns represent the target artefact
  - Sources of requirements
  - Development artefacts
  - Requirements

## Documentation

### Traceability Matrix

- Advantages:
  - Good overview
  - Separation → One matrix per traceability aspect
- Disadvantages:
  - Difficult to maintain (might be very large)
  - Multiple matrices required

## Documentation Traceability Graph



## Documentation

### Traceability Graph

- Graphical notation for traceability
- Nodes represent development artefacts
- Edges represent traceability relations
- Infeasible to create and maintain manually → Requires tool support



# SUMMARY

## Summary

- Analysis and understanding of the relations among
  - Requirements
  - Requirements sources
  - Development artefacts
- Supports other activities
  - Especially useful for maintenance
  - E.g., analyze impact of (requirement) changes
- Good traceability is difficult to maintain
  - Tool support might help



# Questions?