

# Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista



# Chapter 3

Eliciting requirements of software architectures

# Learning outcomes of this chapter

- You will learn:
  - the SysADL constructs for expressing requirements;
  - the underlying concepts needed for expressing dependencies between requirements; and
  - the SysADL constructs for documenting architectural decisions.

# The structure of this chapter

- Introduction
- The concept of requirement
- Requirement constructs
- Dependencies between requirements
- Requirements diagram
- Applying requirement constructs
- Summary

# Introduction

## Overview – 1/2

- In SysADL, a ***requirement specification*** identifies
  - a documented need that can be a **required capability** or a **required quality of a system-of-interest**.
  - according to the concerns, it designates a **requirement elicited** from a customer, a provider, or other stakeholder.
- SysADL provides:
  - **requirement constructs**
  - a **requirements diagram** for documenting the requirements of a system

# Introduction

## Overview – 2/2

- Regarding requirements, the questions are:
  - what are the **concepts needed for specifying the requirements** of a system and its related architectural decisions?
  - which are the **SysADL constructs** provided by the requirements diagram?
  - how to **apply these constructs for specifying requirements** and architectural decisions to satisfy the requirements?



# The Concept of Requirement

# Requirement

## Concept

- A **requirement** specifies a capability or a quality to be satisfied.
- In terms of architectural requirements, they may be related to
  - the **structure**,
  - the **behavior**, or
  - the **properties** a software architecture must (in the case of mandatory requirements) or should (in the case of desirable, but not mandatory requirements) satisfy
- A requirement may specify
  - a **service** that a system must perform (*functional requirement*) , or
  - a **quality of service** a system must achieve (an *extra-functional requirement* or also called *non-functional requirement*).



# Requirement

## Example

- As an example, you can elicit the following functional requirement for the *RTC* System from the concern of both the costumer and provider
  - **Functional Requirement:** the *RTC* system must be capable of maintaining the temperature in the room.
  - **Extra-functional requirement:** the *RTC* system must consume at least 20% of less energy than a manual system.



# Requirement Constructs

# Requirement

## Constructs

- In SysADL, we apply
  - the *requirement construct* to specify a requirement, and
  - the *rationale construct* to document a rationale for that requirement.
- A *requirement* is a **documented need related to a concern** of a stakeholder and the *rationale* is the reason for that need.



# Requirement

# Requirement

## SysADL notation

- We represent a *requirement* with a <<*requirement*>> stereotype with:
  - a name
  - two tags
    - **Id**: gives the unique identification of the requirement
    - **Text**: gives the expression of the requirement in a natural, semi-formal or formal language
- We will use **English as the language for specifying requirements** from stakeholders.
- As a convention, we use **the suffix FR** for functional requirements, and **NFR** for extra or non-functional requirements.

# Requirements

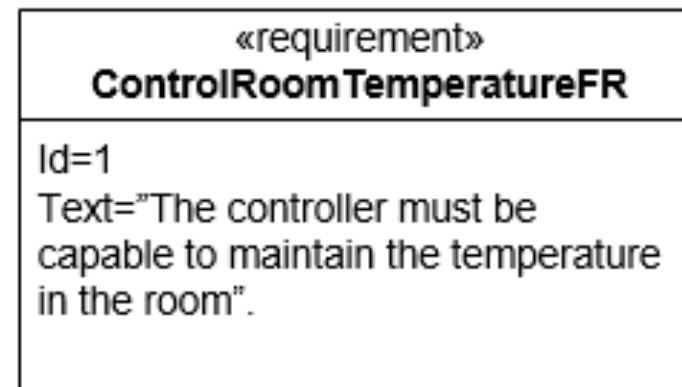
## Use in SysADL – 1/2

14

### Example Description

- In this example we have the following
  - **Functional Requirement:**  
the controller component must be capable to maintain the temperature set by the user in the room, defined by the *ControlRoomTemperatureFR* requirement

### SysADL notation



# Requirements

## Use in SysADL - Example

15

### Example Description

- In this example we have the following
  - **Non-Functional Requirement:** the system must satisfy some quality concerns such as modifiability, scalability, and accuracy, defined by the *QualityNFR* requirement

### SysADL notation

«requirement» QualityNFR
Id=2 Text="The system must satisfy the modifiability, scalability, availability, and accuracy concerns".



# Rationale



# Rationale

## SysADL notation

- We represent a *rationale* with a <<*rationale*>> stereotype attached to:
  - a **requirement**, to justify the decision for it; or
  - a **dependence** related to a requirement; or
  - an **architectural decision**
- We will use **English as the language** for specifying the rationale

# Rationale

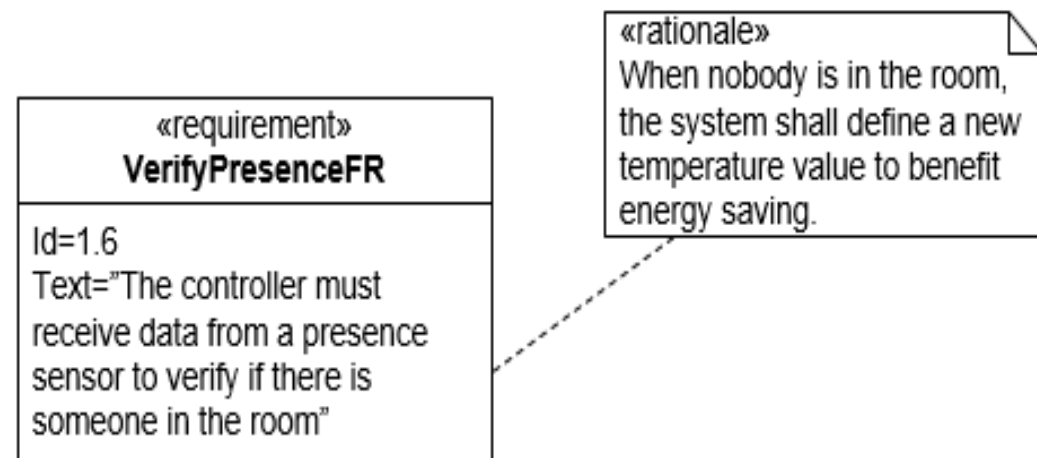
18

## Use in SysADL - Example

### Example Description

- In this example we have the specification of a rationale to the *VerifyPresenceFR* requirement

### SysADL notation





# Dependencies between requirements

# Dependencies

## Conceptual overview

- In SysADL, we can express the following dependencies between requirements
  - Containment
  - Derivation
  - Satisfaction
  - Verification

# Dependencies

## Containment

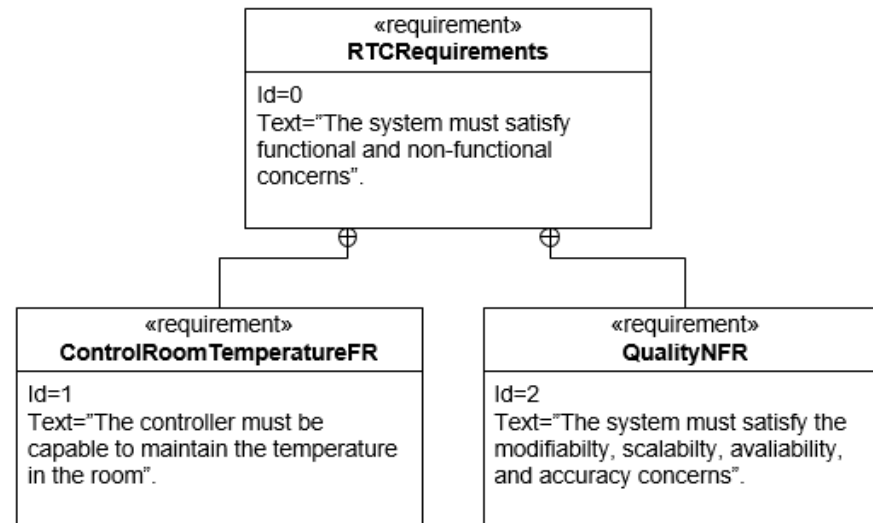
21

### Containment Definition

- A requirement can be **simple** or **composite**
  - **Composite requirements** contain sub-requirements
- The **containment** relationship allows decomposing a requirement into simpler requirements

### SysADL notation

- The ***RTCRequirements*** is decomposed into the
  - ***ControlRoomTemperatureF*** functional requirement
  - ***QualityNFR*** non-functional requirement

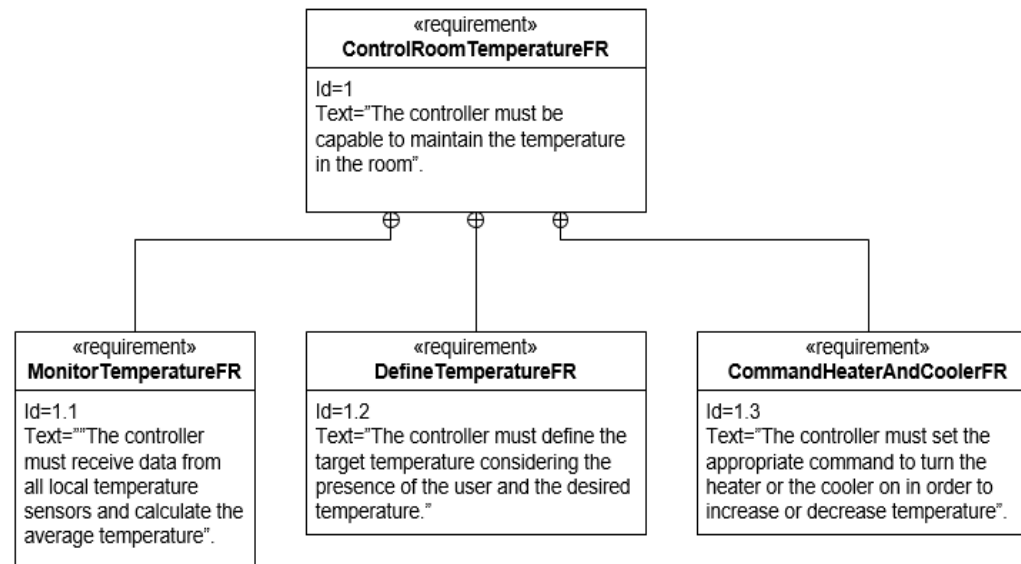


# Dependencies

## Containment among functional requirements

### Containment

- The **ControlRoomTemperatureFR** requirement is decomposed into three sub-requirements:
  - id 1.1 *MonitorTemperatureFR*,
  - id 1.2 *DefineTemperatureFR*, and
  - id 1.3 *CommandHeaterAndCooler*.



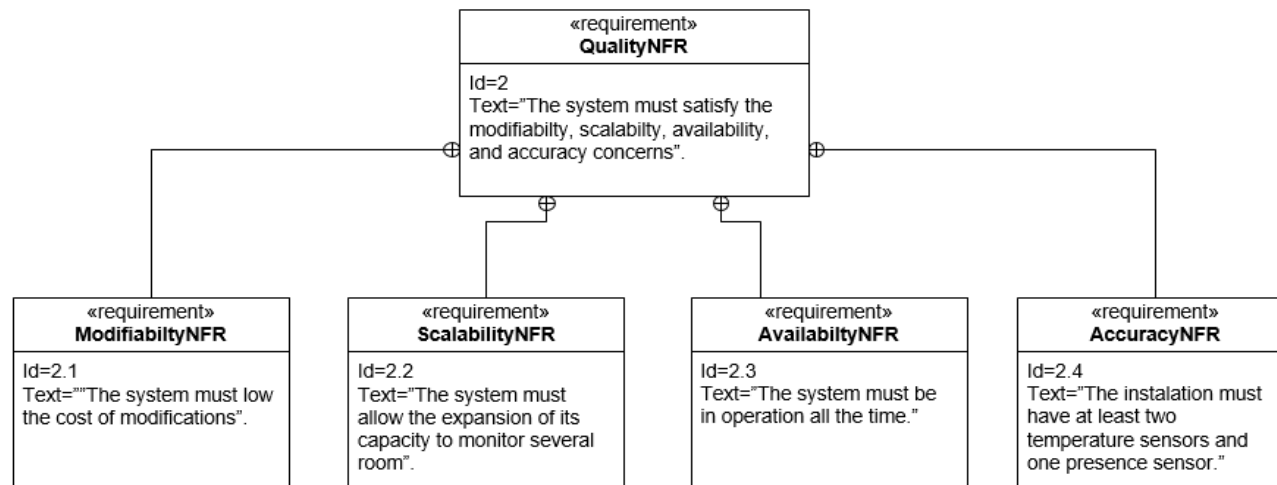
# Dependencies

## Containment among non-functional requirements

### Containment example

- The **QualityNFR** requirement is decomposed into three sub-requirements:
  - id 2.1 *ModifiabilityNFR*,
  - id 2.2 *ScalabilityNFR*,
  - id 2.3 *AvailabilityNFR*, and *AccuracyNFR*

### Containment Example



# Dependencies

## Derivation

24

### Derivation Definition

- A requirement may be **primitive** or **derived from another**
- The ***derivation*** relationship enables a requirement to be derived from another requirement

### SysADL notation

- The **VerifyPresenceFR** is derived from ***DefineRequirementFR***
  - It means that ***VerifyPresenceFR*** implies ***DefineRequirementFR***,
  - Therefore, if ***VerifyPresenceFR*** is satisfied, ***DefineRequirementFR*** is, by consequence, satisfied too





# Dependencies

## Satisfaction

25

### Satisfaction Definition

- A requirement is satisfied by an architectural decision.
- The *satisfy* dependence describes how an architectural element satisfies one or more requirements

### SysADL notation

- The **PresenceSensorCP** component satisfies the **VerifyPresenceFR** functional requirement
- This means that the architectural decision is to give to the identified architectural element – the presence sensor – the responsibility to satisfy that requirement – to verify the presence of somebody in the room



# Dependencies

## Verification

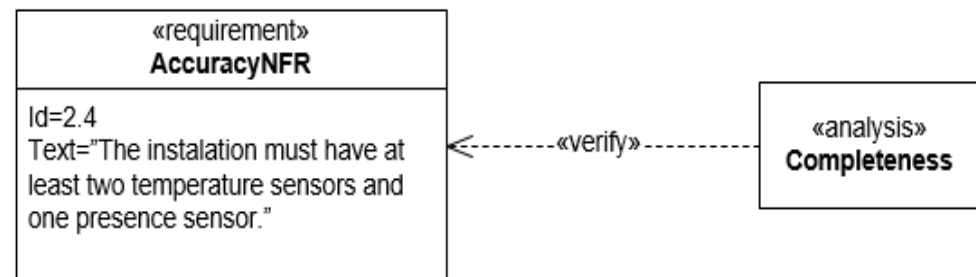
26

### Verification Definition

- A requirement needs to be verified, where this **verification may be carried out by different means**, e.g, it could be via a test case or by model checking
- An <<**analysis**>> stereotype or another stereotype representing a verification technique can be used to identify verification methods for e.g inspection, demonstration, or formal analysis

### SysADL notation

- **Completeness** is used to verify *AccuracyNFR*, a non-functional requirement





# Requirements Diagram

# Requirements Diagram

## Conceptual overview

- The requirements specification is described using one or several **requirements diagram**.
- It presents the set of **interrelated requirements**
  - using the requirement and rationale constructs and the different dependencies.
  - architectural decisions are represented in the requirements diagram by associating to the specified requirements, the architectural elements that satisfy these requirements.

# Requirements Diagram

## Diagram elements

- The header of a requirements diagram, **req**, is marked **[Requirements]** followed by the name of the diagram.
- The **suffix FRD** is a convention to state that this is a specification of functional requirement diagrams.

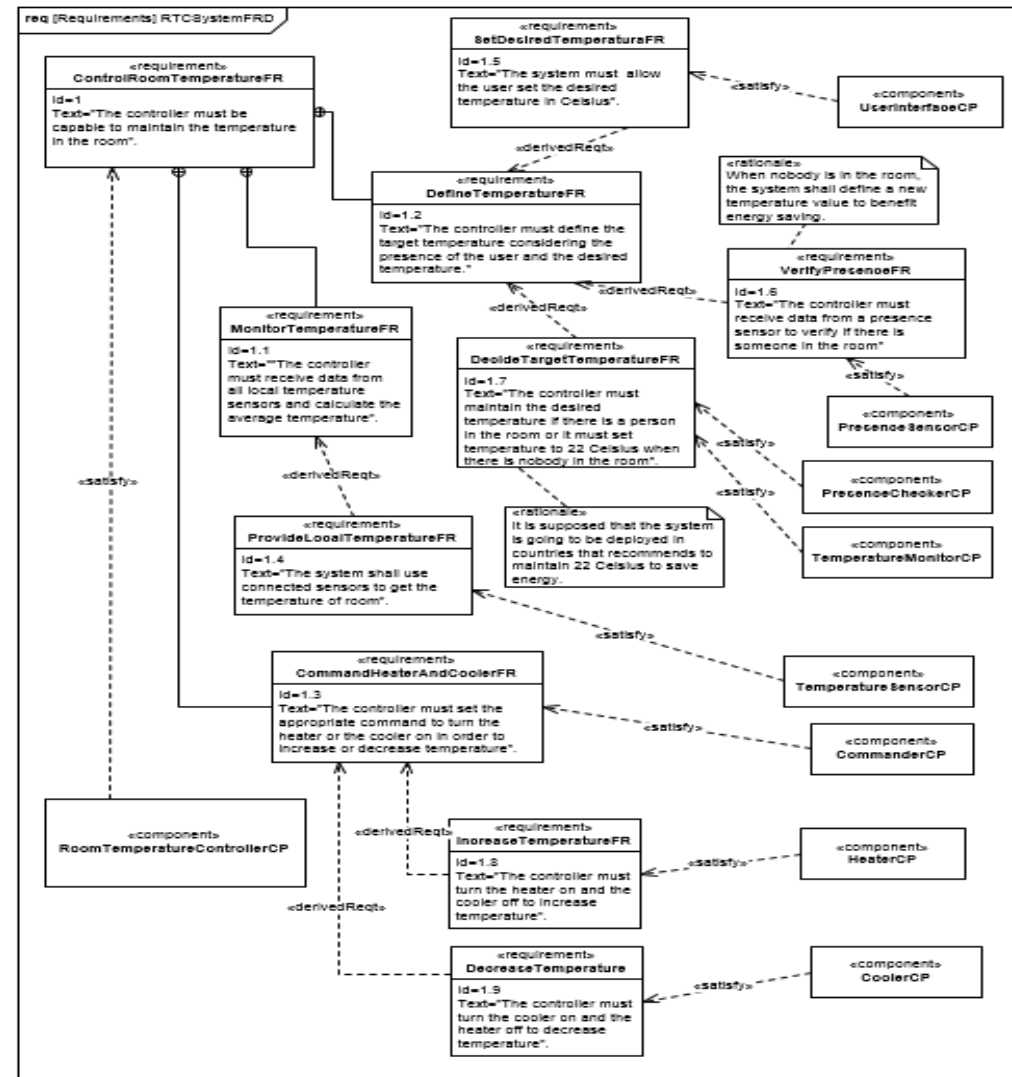
req [Requirements] RTCSystemFRD



# Applying Requirement Constructs

# Requirements in RTC

- The overall functional requirement of the RTC system is ***ControlRoomTemperature FR.***
- It is decomposed into three sub-requirements:
  - *id 1.1*  
*MonitorTemperatureFR,*
  - *id 1.2*  
*DefineTemperatureFR,* and
  - *id 1.3*  
*CommandHeaterAndCooler.*



# Summary

- In this chapter, you learned how to:
  - apply the SysADL constructs for expressing requirements;
  - express the dependencies between requirements;
  - document architectural decisions.



# For Further Reading

- Friedenthal, S., Moore, A.: A Practical Guide to SysML: The Systems Modeling Language, 3rd edn. The MK/OMG Press (2014)
- Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications (2009)
- Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer (2010)