

Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista

Motivation

- In this book you can learn the main software architecture concepts and practices.
- We use an architecture description language, SysADL, to illustrate the concepts and techniques using practical examples.

The structure of the book

- Part I: Fundamentals
- Part II: Quality-based Architectures
- Part III: Style-based Architectures
- Part IV: Textual Description of Architectures



Part I

Fundamentals

The structure of Part I

- Chapter 1: Introduction to Software Architecture
- Chapter 2: Viewpoints for Describing Software Architectures
- Chapter 3: Eliciting Requirements of Software Architectures
- Chapter 4: Specifying the Structure of Software Architectures
- Chapter 5: Specifying Behavior of Software Architectures
- Chapter 6: Specifying Executable Software Architectures
- Chapter 7: Executing Software Architectures



Chapter 1

Introduction to Software Architecture

Learning outcomes of this chapter

- You will learn:
 - the concept of software architecture;
 - the rationale for defining SysADL;
 - the approach provided by SysADL to address the description, analysis, and execution of software architectures.

The structure of the chapter

- The concept of Software Architecture
- Language for modeling Software Architecture
 - Why conceiving SysADL
 - Introducing SysML for SysADL
 - SysADL as a specialization of SysML for architecture modeling
- Designing software architecture with SysADL
 - Describing software architectures
 - Designing quality-based software architectures
 - Designing style-based software architectures
 - Textually representing software architectures
- Running case study to illustrate software architecture
- Summary



The Concept of Software Architecture

Software Architecture

Concept – 1/2

- The concept of architecture
 - has been used originally for buildings as “**the art or practice of designing and constructing buildings**” (Oxford English Dictionary), and
 - evolved to mean nowadays “**the complex structure of something**”, where this “something” could be a for instance a building, a ship, or a software.
- Since 2011 we have an official definition of software architecture [ISO/IEC Standard 42010]
 - *“The fundamental properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution”.*

ISO/IEC Standard 42010 - Systems and Software Engineering – Architecture description. www.iso-architecture.org/42010/index.html

Software Architecture

Concept – 2/2

- To understand the concept and definition of software architecture, you should think about different interrelated notions
 - **structure:** what is the form of the interrelated elements
 - **behavior:** what is the functionality that the interrelated elements provide
 - **execution:** how the functionality provided by the interrelated elements is carried out
 - **analysis:** what are the non-functional properties that the interrelated elements satisfy

ISO/IEC Standard 42010 - Systems and Software Engineering – Architecture description. www.iso-architecture.org/42010/index.html

Software Architecture

Concepts

- **Structure** is the organization of elements and their relationships for executing the functionalities while satisfying the properties of the system.
- **Behavior** is the detailed specification of activities allocated to the structural elements, which provides the functionality of the architected system
- **Execution** is the detailed specification of atomic actions that supports the actual execution of the described behavior

Software Architecture

Properties

- **Properties** complete the description of structure, behavior, and execution with the quality of service.
- **The non-functional properties** need to be analyzed against the architecture description to ensure their satisfaction

Software Architecture

ISO/IEC Standard 42010 definition (1/3)

- Three matters in an architecture
 - Elements
 - Relationships between elements
 - Principles in the design and evolution of these interrelated elements

Software Architecture

ISO/IEC Standard 42010 definition (2/3)

- The architecture comprises the implied properties of these constituents, basically
 - The **structure of each element**, its behavior, its execution semantics, and its non-functional properties
 - The **structure of each relationship between elements**, its behavior, its execution semantics, and its non-functional properties
 - The **structure of the interrelated elements**, its behavior, its execution semantics and its non-functional properties.

Software Architecture

ISO/IEC Standard 42010 definition (3/3)

- These concerns are embodied in:
 - ***component*** as the software architectural concept of element, and
 - ***connector*** as the software architectural concept of relationship between elements.
- The principles in the design and evolution of the architecture will be described through ***non-functional properties*** and encoded in **architectural styles**.

Software Architecture

Concept

- Software architecture is an essential activity for the development of software-intensive systems enabling to reason about system properties very early in the development lifecycle.
- The issue is on how to organize a system to, simultaneously:
 - provide the required functional services,
 - guarantee the required quality of service.



Language for Modeling Software Architecture

Software Architecture

Description

- Two lines of work emerged for software architecture description
 - Based on the definition of new languages for describing the architecture of software-intensive systems – the so-called "***Architecture Description Languages (ADLs)***"
 - Based on the use of general-purpose modeling languages, in particular UML (*Unified Modeling Language*) and, recently, **SysML** (*Systems Modeling Language*)

ISO/IEC Standard 42010 - Systems and Software Engineering – Architecture description. www.iso-architecture.org/42010/index.html

Software Architecture

Problems of existing proposals (1/5)

- Although many ADLs have been proposed since the 1990s
 - none of them **have a broad adoption in the industry** (Malavolta et al., 2013)
 - even if a few has been adopted in specific domains, as for instance AADL, developed for the field of avionics and automotive applications.

Software Architecture

Problems of existing proposals (2/5)

- In 1997, the Unified Modeling Language (UML) emerged as an OMG Standard (UML 1.0) and gained popularity with a high acceptance by the software development community and industry. However,
 - the first versions of UML **did not include support for describing software architectures.**
 - UML 2.0 (2005) has improved the language with some architectural modeling features (the notion of architectural component and composite structures, but however not the one of architectural connector), it is **still limited for describing software architectures.**

Software Architecture

Problems of existing proposals (3/5)

- In 2007 (ten years after UML), SysML was published as an evolution of UML for systems engineering, and it **has been increasingly used** by software-intensive systems engineers, inheriting the popularity of UML.
- SysML enriches UML with new concepts, diagrams, and it has been widely adopted to describe software-intensive systems. However:
 - in terms of architectural description, **SysML inherits the limitations of UML**: architectural constructs are basically the same as UML, limited to the notion of component and composite structures.

Software Architecture

Problems of existing proposals (4/5)

- UML was borne in the mid-1990s:
 - In the 1980s and beginning of 1990s **too many modeling languages** were proposed by the community, each industry adopting one or several of them, and even some been adopted only internally.
 - the **need was to unify all modeling languages** around a unique, common and shared language: the Unified Modeling Language. Designed for software modeling, UML has been extended for systems modeling (including software-intensive systems modeling) and issued as SysML.

Software Architecture

Problems of existing proposals (5/5)

- For ADLs, we are in a similar and even worst situation
 - along the years more than 120 languages have been proposed by the research community
 - only very few of them being adopted in the industry for particular application domains, e.g. AADL in the avionics and automotive industry.
- Current existing architectural languages,
<http://www.di.univaq.it/malavolta/al/> (accessed 04/2015)

Software Architecture

Motivation for SysADL – 1/2

- As UML, SysML is also a semi-formal modeling language, with semantic gaps, the so-called “**semantic variation points**”, for supporting the specialization of SysML (as well as of UML) for different purposes.
- The **semantic variation points** together with the **profile mechanism** enable to specialize the syntax and semantics of SysML for architecture description.

Software Architecture

Motivation for SysADL – 1/2

- We have designed SysADL
 - as a specialization of SysML (in the sense that all architecture descriptions expressed using SysADL are also valid SysML models),
 - adding new architectural concepts (completing SysADL in terms of ADL expressiveness) by filling architecture-related semantic gaps of SysML.

Software Architecture

SysADL – 1/2

- SysADL reconciles
 - the expressive power of ADLs, with
 - the use of a common syntax in line with the SysML standard.
- SysADL copes with the architectural concepts defined in the ISO/IEC/IEEE 42010 Standard in terms of multiple viewpoints.
 - according to the Standard, ***an architectural viewpoint establishes conventions for the construction, interpretation, and use of architecture views to frame specific systems concerns.***

Software Architecture

SysADL – 2/2

- SysADL has a rigorous operational semantics, which allows
 - the analysis (in terms of verification of both structure and behavior)
 - the execution (in terms of executable specifications for validation) of the architecture.



Why conceiving SysADL?

SysADL

Why conceiving SysADL

- SysML is more expressive than UML, however
 - it is **not an ADL** in the sense that it does not provide the minimal set of architectural concepts of an ADL.
 - it is a **general-purpose modeling language**
 - encompassing some of the architectural concepts (but not all, and not consistently assorted)
 - mixed with design concepts and implementation ones.

ISO/IEC Standard 42010 - Systems and Software Engineering – Architecture description. www.iso-architecture.org/42010/index.html

SysADL

What would be the suitable ADL to specify the architecture of systems?

- The answer is that the suitable ADL is the one that solves the following tradeoff:
 - the ADL **must satisfy the architecture** needs and concepts expressed in the ISO/IEC/IEEE 42010 standard;
 - the ADL **must comprise the core architectural concepts** according to different viewpoints, in particular the structural and behavioral;
 - the ADL **should be based on a standard notation** whenever possible.

This ADL does not exist yet!

SysADL

Characteristics

- We defined SysADL as a specialization of SysML to cope with these three key requirements:
 - SysADL is **based on the standard framework** for architecture description, as defined by the ISO/IEC/IEEE 42010 standard;
 - SysADL is **based on well-established concepts drawn from the R&D on software architecture**.
 - SysADL is **based on the SysML standard**, by being a compliant specialization of SysML.



Introducing SysML for SysADL

SysML

Introduction

- SysML was **jointly defined by** the Object Management Group (**OMG**) and the International Council on Systems Engineering (**INCOSE**).
- It is a **general-purpose modeling language** for systems engineering, including in particular software-intensive systems engineering.
- It supports the **specification, analysis, design, verification and validation** with a diagrammatic notation that is more expressive and flexible than UML while being smaller.

SysML

Diagrams (1/2)

- It provides nine kinds of diagram, of which seven are borrowed from UML. Of these seven
 - **four are used as they are**
 - package diagram,
 - use case diagram,
 - sequence diagram, and
 - state machine diagram
 - **three were extended**
 - block definition diagram generalizes class diagram;
 - internal block diagram generalizes composite structure diagram. and
 - activity diagram was extended with more general features.

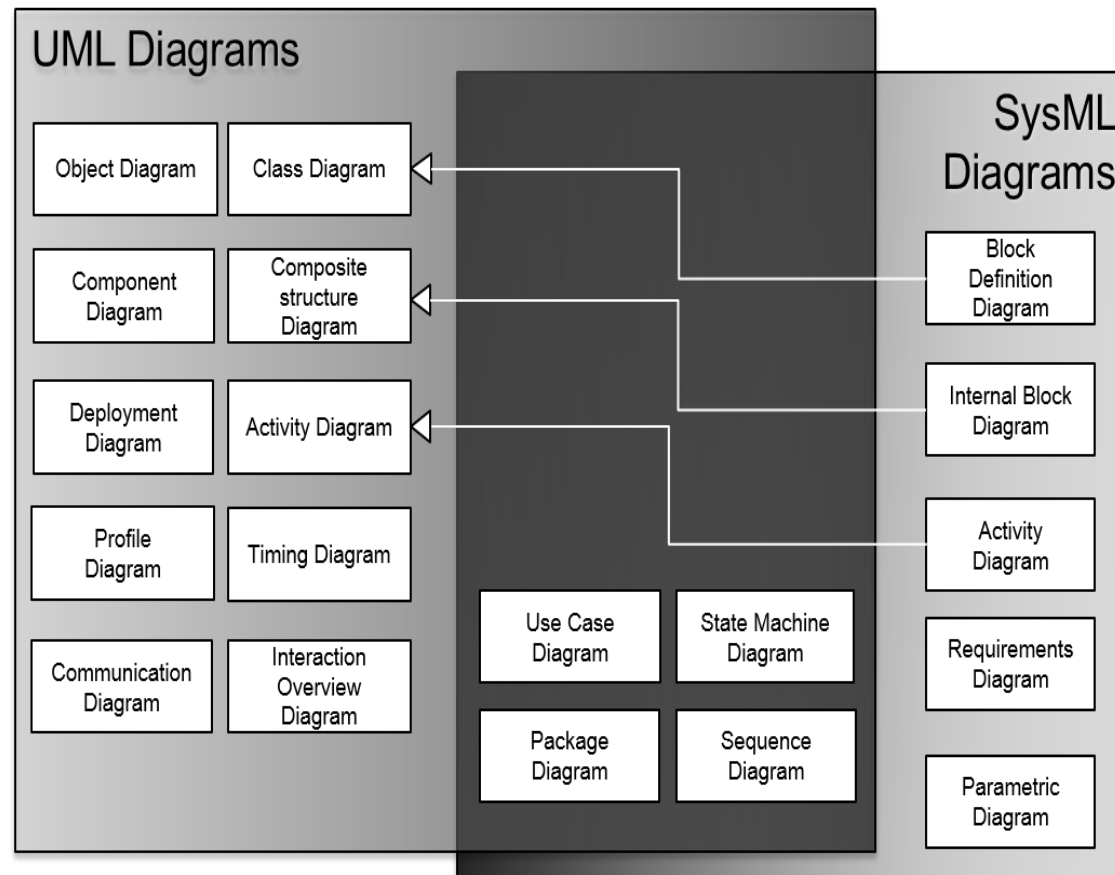
SysML

Diagrams (2/2)

- The two new diagrams provided by SysML are
 - the **requirement diagram** for expressing the definitions and dependencies of requirements, and
 - the **parametric diagram** mostly for defining quantitative constraints.

SysML x UML Diagrams

37





SysADL as a
specialization
of SysML for
architecture
modeling

SysADL

Specialization of SysML (1/3)

- SysML provides a suitable foundation for the definition of an ADL.
- SysADL is thereby defined as a **subset of SysML** specialized for architecture description and analysis.

SysADL

40

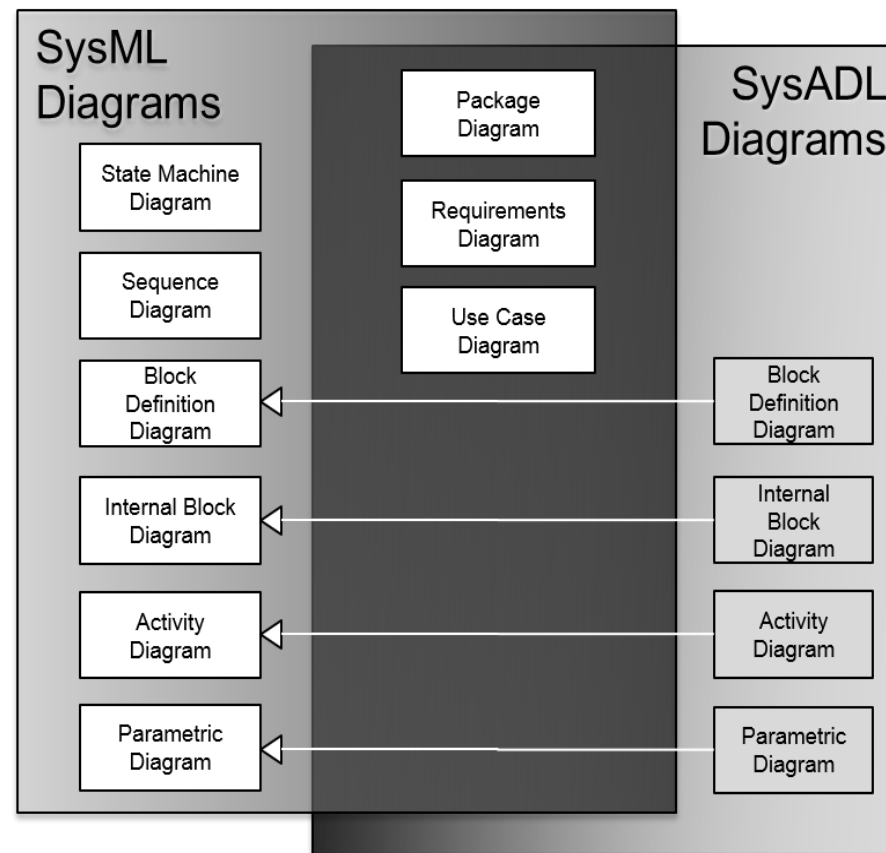
Specialization of SysML (2/3)

- SysADL reuses the requirement diagram and specializes four diagrams of SysML:
 - the **block definition diagram (bdd)** for modelling the structure of architectural components and connectors;
 - the **internal block diagram (ibd)** for modeling the structure of architectural configurations;
 - the **activity diagram** for modelling the behavior of architectural components and connectors;
 - the **parametric diagram** for expressing qualitative as well as quantitative architectural properties.

SysADL x SysML

41

Diagrams



Specialization of SysML (3/3)

- SysADL also extends SysML with the **OMG Action Language for Foundational UML (ALF)** focusing on architectural relevant constructs.
 - It **provides a specialization of ALF** adapted from UML to SysML for specifying executable models of software architecture.
 - An executable architecture description modelled with SysADL provides **an executable specification** detailed enough (but independent of any implementation platform or programming language) that can effectively be run as it was a “program”.
- In summary, regarding architectural modeling, SysADL **provides an extremely more powerful language** than full SysML.



Designing Software Architecture with SysADL

Software Architecture with SysADL

how to use SysADL in practice for designing software architecture?

- This question is answered along the different parts of this book.
 - Describing software architectures
 - Designing quality-based software architectures
 - Designing style-based software architectures
 - Textually representing software architectures

Describing Software Architectures

- In Part I of the book, we present how to describe a software architecture.
 - In Chapter 2, we introduce the concepts of viewpoint and view in the description of software architecture.
 - In Chapter 3 we present how to represent software architecture requirements in SysADL.
 - In Chapter 4, we present how to describe the software architectures from the structural viewpoint
 - In Chapter 5, we present how to describe the software architectures from the behavioral viewpoint
 - In Chapter 6, we present how to describe the software architectures from the executable viewpoint.
 - In Chapter 7, we present the execution semantics of architecture descriptions expressed with SysADL.

Designing quality-based software architectures

- In Part II of the book, we present how to design a software architecture with SysADL for achieving quality attributes
 - In Chapter 8, we introduce the concept of quality attribute and after we present three cases of design
 - In Chapter 9 we present how to design a software architecture that is easily modifiable (*modifiability* quality attribute)
 - In Chapter 10, we present how to design a software architecture that is easily scalable (*scalability* quality attribute)
 - In Chapter 11, we present how to design a software architecture that is fault tolerant (*fault tolerance* quality attribute)

Designing style-based software architectures

- In Part III of the book, we present how to define software architecture styles in SysADL, that will afterwards be used to describe a software architecture.
 - In Chapter 12, we introduce the concept of *architectural style*. Next, we present, four largely used architectural styles.
 - In Chapter 13, we describe how to define and use the *Pipe-Filter Architectural Style*
 - In Chapter 14, we present the *Client-Server Architectural Style*
 - In Chapter 15, we describe the *Feedback Control Loop Architectural Style*; and
 - In Chapter 16, we present the *Blackboard Architectural Style*

Textually representing software architectures

- In the last part of the book (Part IV), we present the textual notation to the SysADL constructs represented in the diagrams.
- In Chapter 17, we present how to textually represent software architecture models in complement to the visual notation.



Running Case Study

Case Study

Overview – 1/2

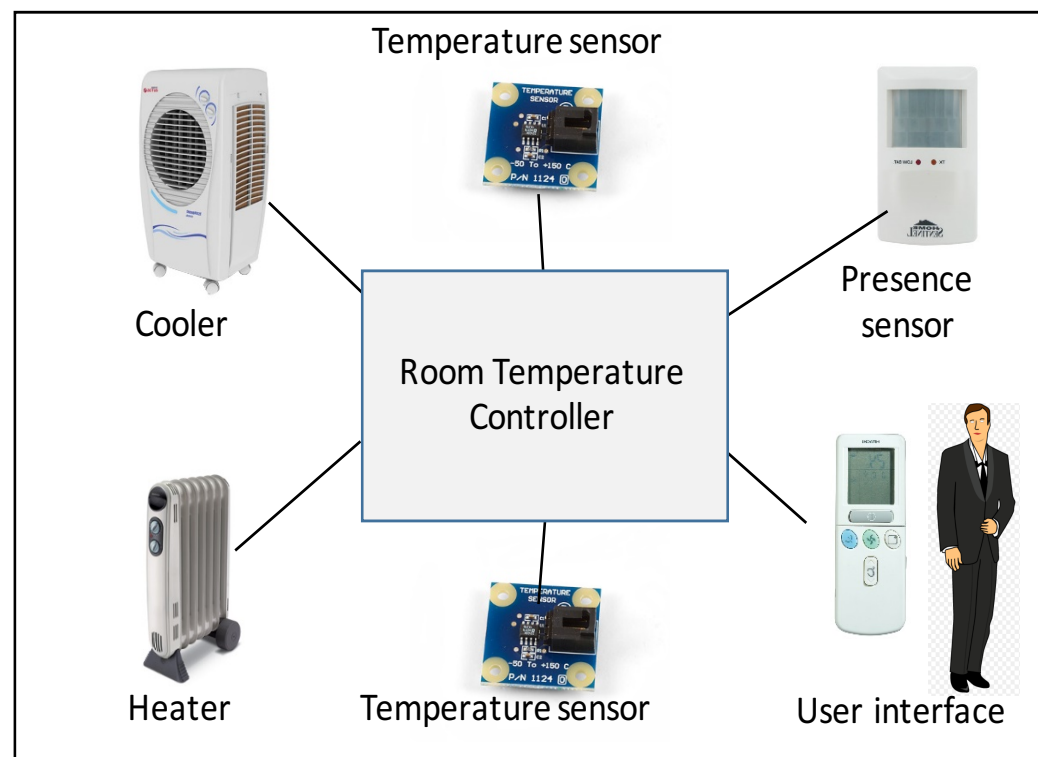
- To illustrate the different concepts and constructs of software architecture, including architecture description and analysis, we present hereafter a case study for architecting a ***Room Temperature Controller (RTC)*** system.

Case Study

Overview – 2/2

- The system has
 - two *temperature sensors* to capture the *current temperature* in different areas.
 - an **user** can set the desired temperature.
 - a *central controller* receives the values from the *temperature sensors*, compares them with the desired temperature and turn the *cooler* or the *heater* on or off.
 - a *presence sensor* to detect if there is someone in the room.
 - In case of presence, the system operates to provide the desired temperature.
 - Otherwise, the system operates to maintain 22 Celsius.

RTC System



Summary

- In this chapter you learnt
 - the principles underlying the concept of software architecture
 - the principles underlying the definition of SysADL as an ADL derived from SysML
 - how SysADL will be presented in this book, along the chapters, for designing a software architecture

For Further Reading

- Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., Tang, A.: What industry needs from architectural languages: a survey. *IEEE Trans. Softw. Eng.* 39(6) (2013)
- Medvidovic, N., et al.: A classification and comparison framework for software architecture description languages. *IEEE Trans. Softw. Eng.* 26(1), 70–93 (2000)
- Medvidovic, N. et al.: Modeling Software Architecture in the Unified Modeling Language. *ACM Trans. Softw. Eng. Methodol. (TOSEM)*. 11(1), 2–57 (2002)