



Software Architecture in Action

Flavio Oquendo, Jair C Leite, Thais Batista



Chapter 6

Specifying Executable Software Architectures

Learning outcomes of this chapter

■ You will learn:

- the architectural constructs to express the *executable body of the actions*
- the *data and control flow concepts* used in the executable body of actions
- the SysADL notation to represent these constructs

The structure of this chapter

- Introduction
- Executable viewpoint and views
- Action language
- Summary
- For further reading



Introduction

Introduction (1/3)

Conceptual Overview

■ Viewpoints:

- the *structural viewpoint* declaratively describes the structure of the architecture
- the *behavioral viewpoint* declaratively describes the behavior of the architecture
- **the *executable viewpoint*** adds the executable elements of the architecture description enabling to execute architecture descriptions

Introduction (2/3)

Conceptual Overview

- The *executable viewpoint* is expressed by describing the body of the actions expressing the computation
 - The SysADL notation to represent the body of the actions is based on ALF, part of UML and SysML

Introduction (3/3)

Conceptual Overview

- From the executable viewpoint, the questions we will address are:
 - which are the SysADL constructs provided by the executable viewpoint?
 - how to apply these constructs for describing the executable view of the architecture?



Executable viewpoint and views

Executable viewpoint and views

Executable viewpoint

- The **executable viewpoint** enables the description of the executable elements of a software architecture to achieve the required system functionality:
 - To be able to execute a software architecture description, an action semantics is needed
- Main executable concepts are:
 - **Activity**: Activities define the behavior of a component and connectors, and activities are defined in terms of actions that are specified in terms of pre- and post-conditions
 - **Action**: An action is expressed by a sequential control flow of statements

Executable viewpoint and views

Executable views

- An **executable view** shows a subset of the executable elements of a software architecture description
- Executable views are defined using constructs of the executable viewpoint

Executable viewpoint and views

Executable constructs

Executable construct

- We apply the *executable* construct to specify the action body
- An **executable** depicts the action body by expressing:
 - its *parameters*: the pins that consume and produce data;
 - its *body*: the statements that execute how the output pin is computed from the input pins

SysADL notation

«executable» CalculateAverageTemperatureEX
<i>parameters</i> in t1:CelsiusTemperature in t2:CelsiusTemperature out :Temperature
<i>body</i> return ((t1 + t2)/2);

Executable viewpoint and views

Executable constructs

Executable construct

- **Body:** the body is specified by an action language, part of SysADL, based on ALF

SysADL notation

«executable» CompareTemperatureEX
<i>parameters</i> in averageTemp: CelsiusTemperature in targetTemp: CelsiusTemperature out result: Commands
<i>body</i> <pre> let heater:Command = Command::off; let cooler:Command = Command::off; if (averageTemp > targetTemp) { heater = Command::off; cooler = Command::on; } else if averageTemp < targetTemp { heater = Command::on; cooler = Command::off; } return new Commands(heater=>heater, cooler=>cooler); </pre>



Action language

Action language

Statements

Statements

- There are several kind of statements:
 - declaration
 - assignment
 - if
 - for
 - while
 - switch
 - expression
 - return

SysADL notation

```
Statement ::=  
( LocalNameDeclaration |  
    Assignment |  
    If |  
    For |  
    While |  
    Switch |  
    Expression |  
    Return |  
) ';'
```

Action language

Variable

Declaration & assignement

- **Variable** declaration requires the specification of a *TypeUse* and an optional *Expression*
- *Assignment* are supported by expressions
- *Examples:*

```
defaultTemperature:CelsiusTemperature=
  22;
```

```
temperatures:CelsiusTemperature[] =
  new CelsiusTemperature[] {20,22};
```

```
targetTemperature =
  defaultTemperature - 2;
```

SysADL notation

LocalNameDeclaration ::=
TypeUse ('=' Expression)?

TypeUse ::=
ID ':' ID

Assignment ::=
ID '=' Expression

Action language

While

While statement

- while is used to specify iterative execution based on evaluation of a Boolean expression
- *Example :*

```
found = false;
i = 1;
while(not found) {
    if (searchedTemp == temps[i]) ;
        found = true;
    else
        i++;
}
```

SysADL notation

```
While ::=

'while' '(' Expression ')'

'{'

Statement*

'}
```

Action language

For

For statement

- For is used to specify iterative execution based on evaluation of a Boolean expression
- *Example :*

```
sum = 0;
for (t in temps) {
    sum = sum + t;
}
```

SysADL notation

```
For ::=
    'for' ForControl '{'
    Statement*
    '}'

ForControl ::=
    LoopVariableDefinition*

LoopVariableDefinition ::=
    LoopTypeUse 'in' Expression ('..'
    Expression)?
    | LoopTypeUse ':' Expression

LoopTypeUse ::= ID ID
/* type use and variable name */
```

Action language

If

If statement

- **If** is a conditional construct that defines an expression to be evaluated during the execution

- *Example :*

```
if(counter <> 0) {
    average = sum/counter;
}
```

SysADL notation

ConditionalBlock ::=

'if' (Expression)

('{' (Statement) '}'*

| Statement)

(ElseBlock)?

ElseBlock ::=

'else' '{' Statement '}'*

Action language

Switch

Switch statement

- **Switch** is a control construct to express the choice among different guarded branches

- *Example:*

```
switch(fanLevel) {
    case Level::low :
        fancommand(1);
    case Level::medium :
        fancommand(2);
    case Level::high :
        fancommand(3);
}
```

SysADL notation

SwitchStatement ::=

'switch' '(' Expression ')'

*'{' SwitchClause**

SwitchDefaultClause? '}'

SwitchClause ::=

*SwitchCase+ Statement**

SwitchCase ::=

'case' Expression ':'

SwitchDefaultClause ::=

*'default' ':' Statement**

Action language

Return

Return statement

- **Return** returns the results of the action given by an expression

- *Examples:*

```
return (sum/counter);
```

```
return average;
```

SysADL notation

Return ::=

'return' Expression

Summary

- In this chapter, you learned how to:
 - define the executable body of actions
 - use the action language for expressing the data and control flows of actions
 - apply the SysADL notation to express the executable body of the actions

For further reading

- Concrete Syntax For A UML Action Language: Action Language For Foundational UML™ (ALF™).
<http://www.omg.org/spec/ALF/>
- Semantics Of A Foundational Subset For Executable UML Models (FUML™).
<http://www.omg.org/spec/FUML/>
- Precise Semantics Of UML Composite Structures™ (PSCS™). <http://www.omg.org/spec/PSCS/1.0/>