



Katholieke  
Universiteit  
Leuven

Department of  
Computer Science

# Shared Internet Of Things Infrastructure Platform: ADD Application Software Architecture (H09B5a and H07Z9a) – Part 2a

FILIPCIKOVA-HALILOVIC

Monika Filipcikova (r0683254)  
Armin Halilovic(r0679689)

Academic year 2016–2017

# Contents

<b>1</b>	<b>Attribute-driven design documentation</b>	<b>2</b>
1.1	Decomposition 1: SIO TIP System (Av3, P2, UC11, UC14, UC15, UC18) . . . . .	2
1.1.1	Module to decompose . . . . .	2
1.1.2	Selected architectural drivers . . . . .	2
1.1.3	Architectural design . . . . .	2
1.1.4	Instantiation and allocation of functionality . . . . .	3
1.1.5	Interfaces for child modules . . . . .	4
1.1.6	Data type definitions . . . . .	4
1.1.7	Verify and refine . . . . .	4
1.2	Decomposition 2: Module (drivers) . . . . .	6
1.2.1	Module to decompose . . . . .	6
1.2.2	Selected architectural drivers . . . . .	6
1.2.3	Architectural design . . . . .	6
1.2.4	Instantiation and allocation of functionality . . . . .	6
1.2.5	Interfaces for child modules . . . . .	6
1.2.6	Data type definitions . . . . .	7
1.2.7	Verify and refine . . . . .	7
<b>2</b>	<b>Resulting partial architecture</b>	<b>8</b>

# 1. Attribute-driven design documentation

## 1.1 Decomposition 1: SIoTIP System (Av3, P2, UC11, UC14, UC15, UC18)

### 1.1.1 Module to decompose

In this run we decompose the SIoTIP System.

### 1.1.2 Selected architectural drivers

The non-functional drivers for this decomposition are:

- *Av3*: Pluggable device or mote failure
- *P2*: Requests to the pluggable data database

The related functional drivers are:

- *UC11*: Send pluggable device data (P2)  
This use case stores pluggable device data in the pluggable device data storage. This could be a sensor reading, or an actuator status.
- *UC14*: Send heartbeat (Av3)  
This use case checks whether or not motes and pluggable devices are still operational.
- *UC15*: Send notification (Av3)  
This use case sends a notification to a registered user.
- *UC18*: Check and deactivate applications (Av3)  
This use case deactivates any application that requires deactivation, because of unavailability of essential pluggable devices or unassigned mandatory roles.

**Rationale** We chose Av3 first since it had high priority and it was more relevant to the core of the system (pluggable device data) than attributes M1 and U2. We chose P2 along with Av3 as it would force us to think about the way sensor data is handled. We believe this combination of pluggable device connectivity and storage of sensor data is the most defining feature of the system, and that handling this combination would give a better starting point than M1+U2 for later ADD iterations.

### 1.1.3 Architectural design

**Application redundancy settings for Av3** Discussion of the solution selected for (a part of) one of the architectural drivers.

**Failure detection for Av3** timers?  
heartbeat/timestamp tactic

**Application deactivation for Av3** same as Application redundancy settings for Av3?  
degradation/removal from service tactic

**Notifications for Av3** from application manager to cust orgs and apps  
from gateway to infr owners  
notifications tactic

**Scheduling for P2** dynamic priority scheduling

tactics: schedule resource, prioritize events, also limit event response?

**Pluggable data separation for P2** "pluggable data has no impact on other data"

**Alternatives considered**

**Alternatives for X** A discussion of the alternative solutions and why that were not selected.

#### 1.1.4 Instantiation and allocation of functionality

**Decomposition** Main aspects of the resulting decomposition.

**PluggableDeviceFacade** send heartbeats

**GatewayFacade** receive heartbeats, send heartbeats/device lists, send application shutdown, send notification trigger (Av3)  
forward data to applications

**HeartbeatProcessor** process heartbeats: check timers (Av3)

**PluggableDeviceManager** check list of devices and see if there are pluggable devices for applications check redundancy in the available pluggable devices contains application preferences (e.g. amount of sensors required) can send command to deactivate application If failure detected, notify inf owner (Av3). reactivate application if new/needed hardware detected

**RequestHandler** handles requests from gateways and sends them to other components handles requests from other components and sends them to gateways

**ApplicationManager** deactivate apps ???? check mandatory user roles set redundancy in the available pluggable devices If application suspended or re-activated, notify cust. org. If application uses failed plugg device, notify application (Av3)

**NotificationHandler** Send notifications.  
stored by system -¿ contact DB?  
lookup communication channel  
users choose delivery method?

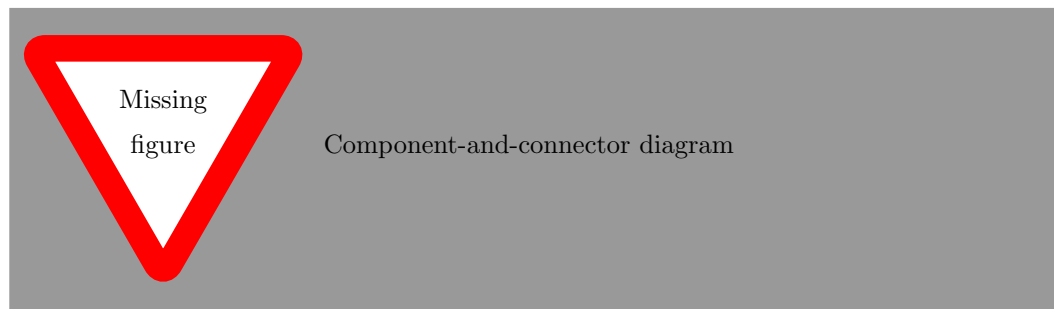


Figure 1.1: Component-and-connector diagram of this decomposition.

**Behaviour** If needed and explanation of the behaviour of certain aspects of the design so far.

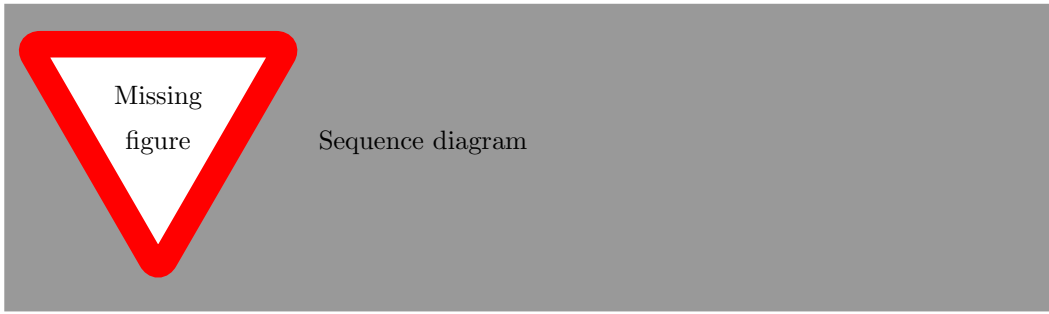


Figure 1.2: Sequence diagram illustrating a key behavioural aspect.

**Deployment** Rationale of the allocation of components to physical nodes.

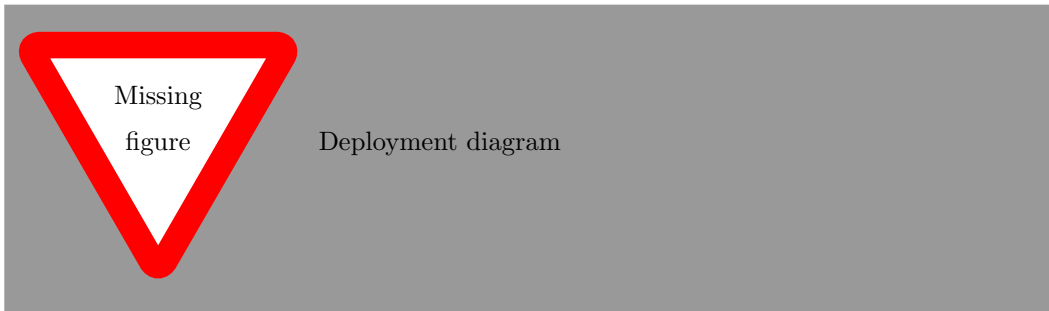


Figure 1.3: Deployment diagram of this decomposition.

### 1.1.5 Interfaces for child modules

#### ModuleB

- InterfaceA
  - returnType operation1(ParamType param1) throws TypeOfException
    - \* Effect: Describe the effect of calling this operation.
    - \* Exceptions:
      - TypeOfException: Describe when this exception is thrown.
  - returnType operation2()
    - \* Effect: Describe the effect of calling this operation.
    - \* Exceptions: None

### 1.1.6 Data type definitions

Describe per complex data type used in the interfaces what it represents.

**returnType** This data element represents X.

**ParamType** This data element represents Y.

### 1.1.7 Verify and refine

This section describes per component which (parts of) the remaining requirements it is responsible for.

## ModuleB

- *Z1*: name
- *UCd*: name

## ModuleC

- *UCba*: name  
Description which part of the original use case is the responsibility of this component.

## 1.2 Decomposition 2: Module (drivers)

### 1.2.1 Module to decompose

In this run we decompose .

### 1.2.2 Selected architectural drivers

The non-functional drivers for this decomposition are:

- $x: \mathbf{x}$

The related functional drivers are:

- $x: \mathbf{x}$

#### Rationale

### 1.2.3 Architectural design

**Topic** Discussion of the solution selected for (a part of) one of the architectural drivers.

#### Alternatives considered

**Alternatives for solution** A discussion of the alternative solutions and why that were not selected.

### 1.2.4 Instantiation and allocation of functionality

**Decomposition** Main aspects of the resulting decomposition.

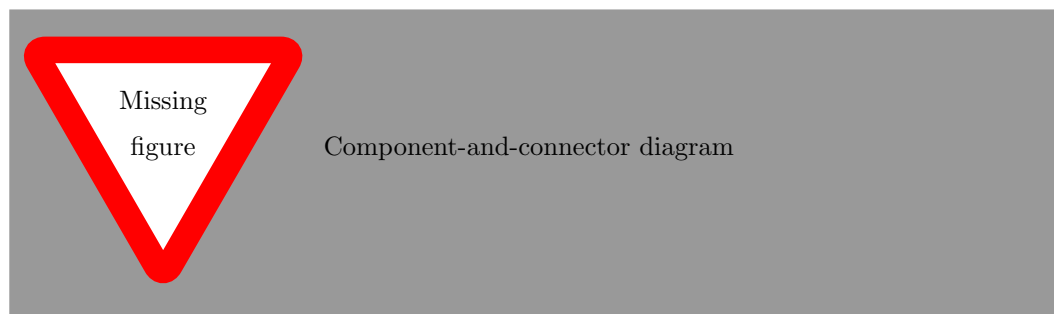


Figure 1.4: Component-and-connector diagram of this decomposition.

**Behaviour** If needed and explanation of the behaviour of certain aspects of the design so far.

**Deployment** Rationale of the allocation of components to physical nodes.

### 1.2.5 Interfaces for child modules

#### ModuleB

- InterfaceA
  - `returnType operation1(ParamType param1)` throws `TypeOfException`
    - \* Effect: Describe the effect of calling this operation.

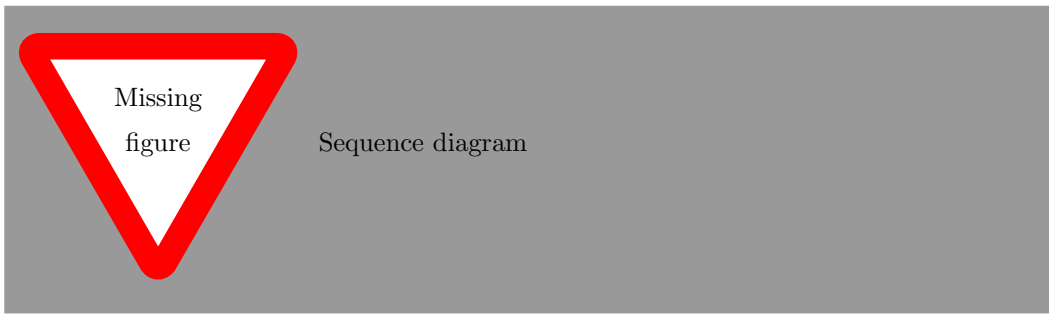


Figure 1.5: Sequence diagram illustrating a key behavioural aspect.

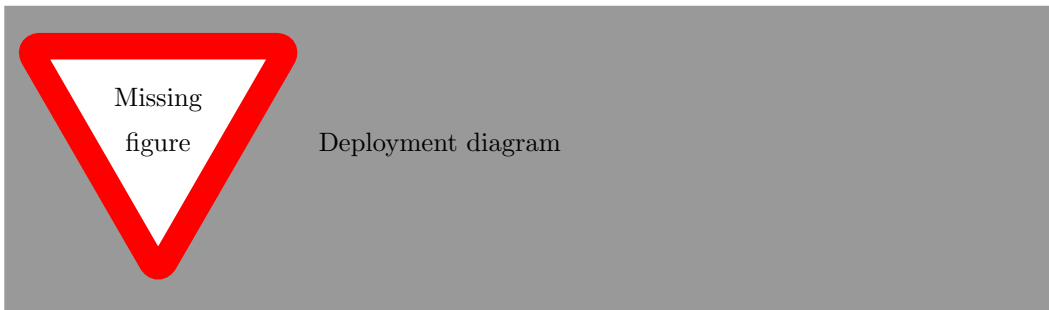


Figure 1.6: Deployment diagram of this decomposition.

- \* Exceptions:
  - TypeOfException: Describe when this exception is thrown.
- returnType operation2()
  - \* Effect: Describe the effect of calling this operation.
  - \* Exceptions: None

### 1.2.6 Data type definitions

Describe per complex data type used in the interfaces what it represents.

**returnType** This data element represents X.

**ParamType** This data element represents Y.

### 1.2.7 Verify and refine

This section describes per component which (parts of) the remaining requirements it is responsible for.

#### ModuleB

- *Z1*: name
- *UCd*: name

#### ModuleC

- *UCba*: name  
Description which part of the original use case is the responsibility of this component.



## 2. Resulting partial architecture

~~This section provides an overview of the architecture constructed through ADD.~~

*"Since you are a two-student team, you can skip the final step of the assignment/report ("2. Resulting architecture  
This section should present the component diagram of the overall system (after two decompositions). At this point,  
you are not required to provide the deployment diagram of the overall system.")"*