



Katholieke
Universiteit
Leuven

Department of
Computer Science

Shared Internet Of Things Infrastructure Platform:

The Complete Architecture

Software Architecture (H09B5a and H07Z9a) – Part 2b

FILIPCIKOVA-HALILOVIC

Monika Filipcikova (r0683254)
Armin Halilovic(r0679689)

Academic year 2016–2017

Contents

1	Architectural Decisions	2
1.1	ReqX: Requirement Name	2
1.2	Other decisions	2
1.2.1	Decision 1	3
1.3	Discussion	3
2	Client-server view (UML Component diagram)	4
3	Decomposition view (UML Component diagram)	6
4	Deployment view (UML Deployment diagram)	8
5	Scenarios	10
6	Element Catalog and Datatypes	11
6.1	Element catalog	11
6.1.1	ComponentA	11
6.1.2	ComponentZ	11
6.2	Common interfaces	12
6.3	Defined Exceptions	12
6.4	Defined data types	12

1. Architectural Decisions

Note: This section discusses *all* your architectural decisions *in-depth*. First, *all* decisions related to the non-functionals are discussed in detail. Next, *all* other decisions are listed and discussed.

Hint: Don't just say *what* you have done. Explain *why* you have done it.

1.1 ReqX: Requirement Name

TODO: Use this section structure for each requirement

Key Decisions

TODO: Briefly list your key architectural decisions. Pay attention to the solutions that you employed (in your own terms or using tactics and/or patterns).

- decision 1
- ...

Employed tactics and patterns: ...

Rationale

TODO: Describe the design choices related to *ReqX* together with the rationale of why these choices were made.

Considered Alternatives

Alternative(s) for choice 1 Explain what alternative(s) you considered for this design choice and why they were not selected.

Deployment Decisions

...

Considered Deployment Alternatives

...

1.2 Other decisions

TODO: *Optional* If you have made any other important architectural decisions that do not directly fit in the sections of the other qualities you can mention them here. Follow the same structure as above.

1.2.1 Decision 1

KeyDecisions

...

Rationale

...

Considered Alternatives

...


Deployment Decisions

...

Considered Deployment Alternatives

...

1.3 Discussion

 **TODO:** Use this section to discuss your architecture in retrospect. For example, what are the strong points of your architecture? What are the weak points? Is there anything you would have done otherwise with your current experience? Are there any remarks about the architecture that you would give to your customers? Etc.

2. Client-server view (UML Component diagram)

Figures

2.1	Context diagram for the client-server view.	4
2.2	Primary diagram of the client-server view.	5

✓ **Hint:** No need to just repeat what we can see on the diagram.

Don't do this: *As you can see on fig. x: comp A consists of B and C, and C connects to D.*

But, please do explain if there is anything non-trivial (e.g., a custom mapping from actors to external components on the context diagram).

✓ **Hint:** Add any essential information, necessary for interpreting the figure, in the caption. Be sure to add a separate short title for inclusion in the list of figures: `\caption[shorttitle]{longtitle}`.

If your explanation becomes too long for the caption, you can create a separate subsection. Don't forget to refer to the figure and vice versa.

✓ **Hint:** If you have any doubts about the size of your figures, it is better to make your figure too large than too small. Alternatively, you can test the readability by printing it.

⚠ **Attention:** With regard to the context diagram, recall the lectures on what it means and should contain. Be sure not to miss any elements here. This is a frequent source of errors.

⚠ **Attention:** Make sure your main component-and-connector and context diagrams are consistent.

📄 **TODO:** The context diagram of the client-server view: Discuss which components communicate with external components and what these external components represent.

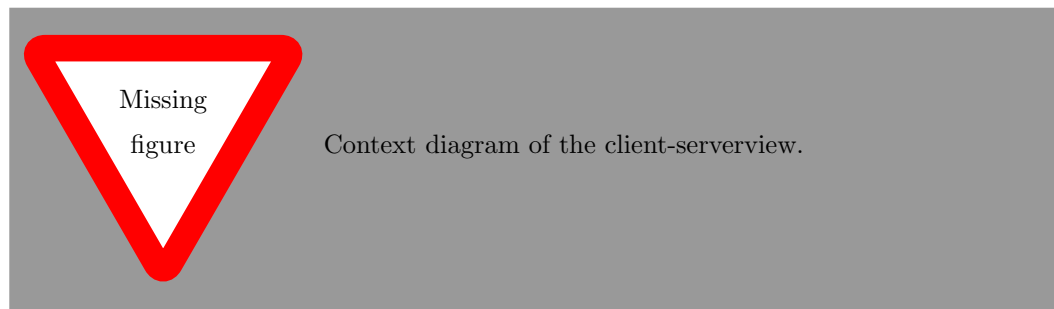


Figure 2.1: Context diagram for the client-server view.

📄 **TODO:** The primary diagram and accompanying explanation.

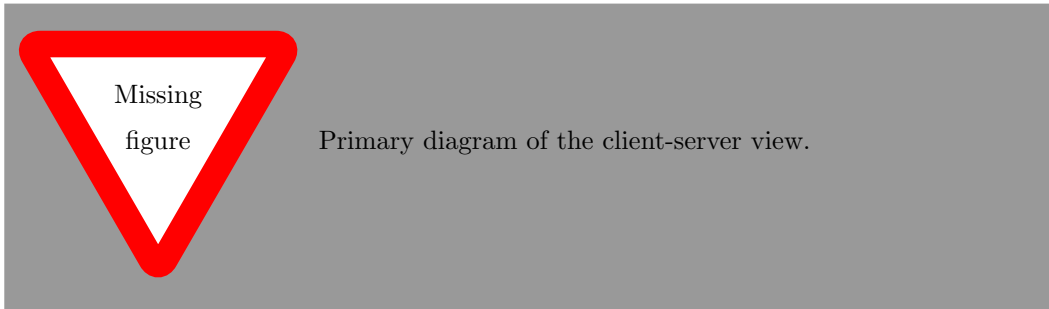


Figure 2.2: Primary diagram of the client-server view.

3. Decomposition view (UML Component diagram)

Figures

3.1	Decomposition of <code>ComponentX</code>	6
3.2	Decomposition of <code>ComponentY</code>	7

✔ **Hint:** No need to just repeat what we can see on the diagram.
Don't do this: *As you can see on fig. x: comp A consists of B and C, and C connects to D.*
But, please do explain if there is anything non-trivial (e.g., a custom mapping from actors to external components on the context diagram).

✔ **Hint:** Add any essential information, necessary for interpreting the figure, in the caption. Be sure to add a separate short title for inclusion in the list of figures: `\caption[shorttitle]{longtitle}`.
If your explanation becomes too long for the caption, you can create a separate subsection. Don't forget to refer to the figure and vice versa.

⚠ **Attention:** *Consistency between views!* Be sure to check for consistency between the client-server view and your decompositions.

⚠ **Attention:** *Consistency of a single decomposition!* Make sure that every interface provided or required by the decomposed component, is provided or required by a subcomponent in the decomposition.

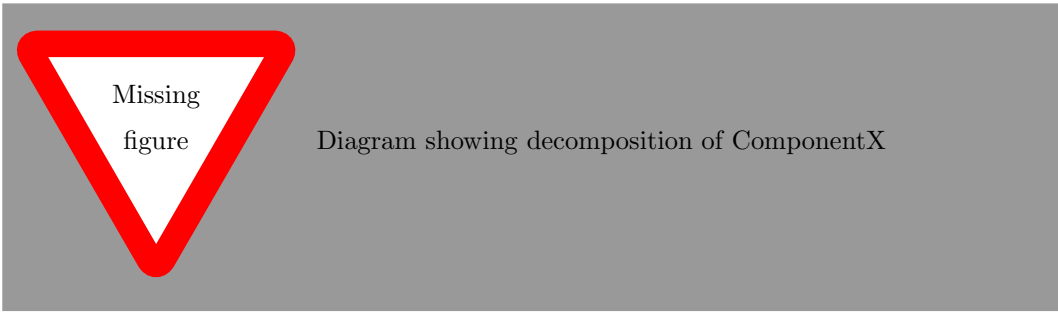


Figure 3.1: Decomposition of `ComponentX`

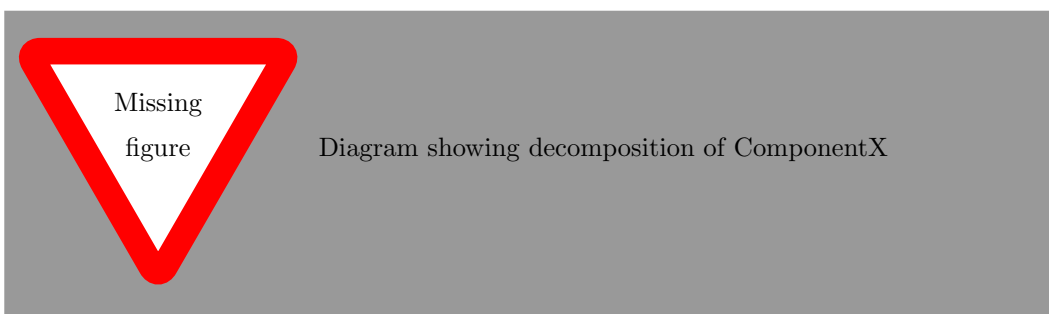


Figure 3.2: Decomposition of **ComponentY**.

This caption contains a longer explanation over multiple lines. This additional explanation is not shown in the list of figures.

4. Deployment view (UML Deployment diagram)

Figures

4.1	Context diagram for the deployment view.	8
4.2	Primary diagram for the deployment view.	9

✓ **Hint:** No need to just repeat what we can see on the diagram.

Don't do this: *As you can see on fig. x: components A and B are deployed on node C.*

But, please do explain if there is anything non-trivial (e.g., a custom mapping from actors to external components on the context diagram).

✓ **Hint:** Add any essential information, necessary for interpreting the figure, in the caption. Be sure to add a separate short title for inclusion in the list of figures: `\caption[shorttitle]{longtitle}`.

If your explanation becomes too long for the caption, you can create a separate subsection. Don't forget to refer to the figure and vice versa.

⚠ **Attention:** Connect nodes on the deployment diagram, *not* components.

⚠ **Attention:** *Consistency between views!* Be sure to check for consistency between the client-server/decomposition view and your deployment view.

📖 **TODO:** Describe the context diagram for the deployment view. For example, which protocols are used for communication with external systems and why?

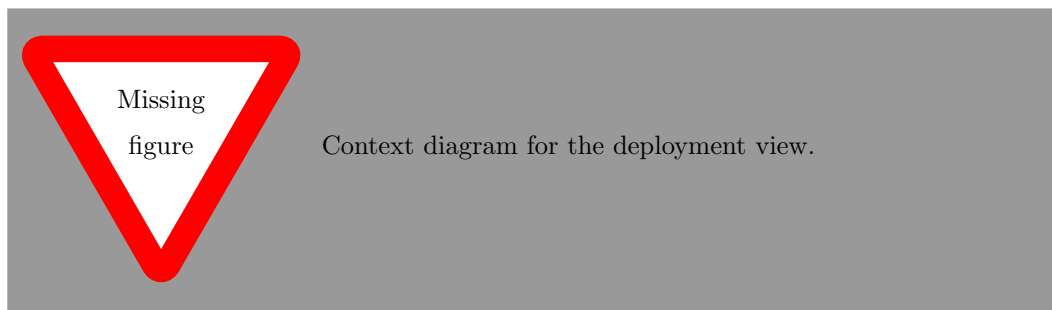


Figure 4.1: Context diagram for the deployment view.

📖 **TODO:** The primary deployment diagram itself. This discussion on the parts of the deployment diagram which are crucial for achieving certain non-functional requirements, and any alternative deployments that you considered, should be in the architectural decisions chapter.

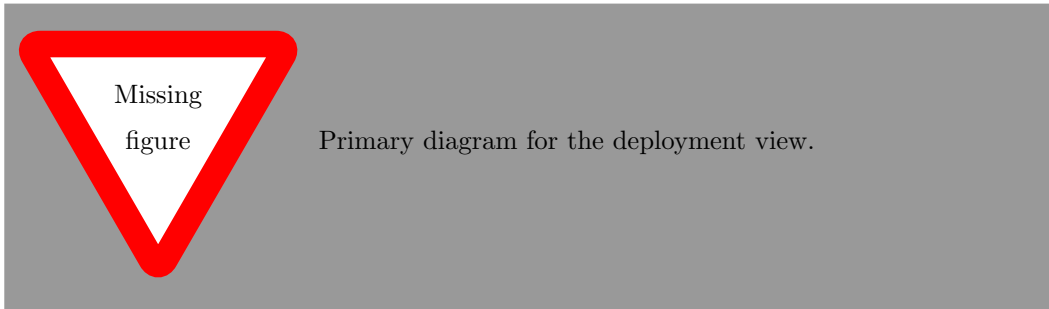


Figure 4.2: Primary diagram for the deployment view.

5. Scenarios

Figures

5.1 Scenario 1	10
--------------------------	----

✓ **Hint:** No need to just repeat what we can see on the diagram.

Don't do this: *As you can see on fig. x: component A calls operation b, next component C calls operation d.* But, please do explain if there is anything non-trivial (e.g., a custom mapping from actors to external components on the context diagram).

✓ **Hint:** Add any essential information, necessary for interpreting the figure, in the caption. Be sure to add a separate short title for inclusion in the list of figures: `\caption[shorttitle]{longtitle}`. If your explanation becomes too long for the caption, you can create a separate subsection. Don't forget to refer to the figure and vice versa.

⚠ **Attention:** Do include a list of which sequence diagrams together illustrate a which scenario from the assignment.

✓ **Hint:** Don't only model the 'happy path' in your sequence diagrams. Take into account the quality attributes. For example, what happens when a certain component fails (Av) or overloads (P)? Use the sequence diagrams to illustrate how you have achieved the qualities in your architecture.

📌 **TODO:** Illustrate how your architecture fulfills the most important data flows. As a rule of thumb, focus on the scenario of the assignment. Describe the scenario in terms of architectural components using UML Sequence diagrams and further explain the most important interactions in text. Illustrating the scenarios serves as a quick validation of the completeness of your architecture. If you notice at this point that for some reason, certain functionality or qualities are not addressed sufficiently in your architecture, it suffices to document this, together with a rationale of why this is the case according to you. You do not have to further refine your architecture at this point.

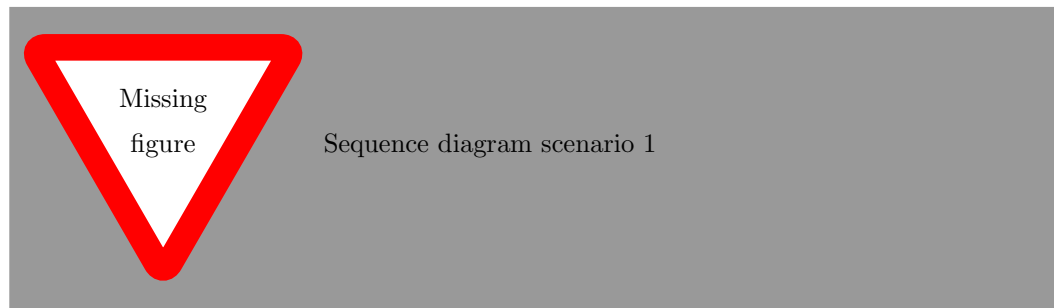


Figure 5.1: The system behavior for the first scenario.

6. Element Catalog and Datatypes

✓ **Hint:** Make sure the elements are sorted alphabetically. You can use the `\componentItem{name}{contents}` command for this in your report. Note that you cannot use newlines in the `componentItem` content, but you can use `\\`.

✓ **Hint:** Common interfaces such as, for example, `ping` can be described separately so you don't have to repeat them for each component that provides it.

✓ **Hint:** Similarly, you can describe the exceptions separately as well, so you don't have to repeat what they mean for each operation that can throw them.

⚠ **Attention:** Don't forget to include the exceptions in the method signature in the element catalog!

⚠ **Attention:** Interfaces are uniquely identified by their name, regardless of the context (e.g., the component that provides it). In other words, two interfaces with the same name are considered identical.

⚠ **Attention:** Don't forget to document the external interfaces!

6.1 Element catalog

📌 **TODO:** List all components and describe their responsibilities and provided interfaces. Per interface, list all methods using a Java-like syntax and describe their effect and exceptions if any. List all elements and interfaces alphabetically for ease of navigation.

6.1.1 ComponentA

- **Responsibility:** Responsibilities of the component.
- **Super-component:** The direct super-component, if any.
- **Sub-components:** the direct sub-components, if any.

Provided interfaces

- InterfaceC
 - `returnType1 operation1(ParamType param) throws SomeException`
 - * Effect: Describe the effect of the operation
 - `void operation2(ParamType2 param)`
 - * Effect: Describe the effect of the operation
- InterfaceD
 - `returnType2 operation3()`
 - * Effect: Describe the effect of the operation


6.1.2 ComponentZ

- **Responsibility:** Responsibilities of the component.
- **Super-component:** The direct super-component, if any.
- **Sub-components:** the direct sub-components, if any.


Provided interfaces

- InterfaceA
 - `returnType1 operation1(ParamType param) throws SomeException`
 - * Effect: Describe the effect of the operation
 - `void operation2(ParamType2 param)`
 - * Effect: Describe the effect of the operation
 - * Exceptions: None
- InterfaceB
 - `returnType2 operation3()`
 - * Effect: Describe the effect of the operation


6.2 Common interfaces

 **TODO:** If you have any common interfaces used by multiple components you may define them here and refer to them.

6.3 Defined Exceptions

 **TODO:** Instead of describing the exceptions with each operation, you may define common exceptions here and refer to them from the operation definition.

6.4 Defined data types

 **TODO:** List and describe all data types defined in your interface specifications. List them alphabetically for ease of navigation.

- Paramtype1: Description of data type.
- Paramtype2: Description of data type.
- returnType1: Description of data type.