Katholieke
Universiteit
Leuven

**Department of
Computer Science**

DOCUMENT PROCESSING

The complete architecture

Software Architecture (H09B5a and H07Z9a) – Part 2b

Jeroen Reinenbergh (r0460600)
Jonas Schouterden (r0260385)

Academic year 2014–2015

Contents

1 Introduction

The goal of this project was to develop an architecture for a system for document processing. This part of the project consisted of

2 Overview

2.1 Architectural decisions

Briefly discuss your architectural decisions for each non-functional requirement. Pay attention to the solutions that you employed (in your own terms or using tactics and/or patterns).

ReqX: requirement name Provide a brief discussion of the decisions related to *ReqX*.
Employed tactics and patterns: List all patterns and tactics used to achieve ReqX, if any.

2.2 Discussion

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.

3 Attribute-driven design documentation

3.1 Decomposition 1: eDocs (X1, Y3, UCa, UCb, UCc)

3.1.1 Module to decompose

In the first run, the eDocs System is decomposed as a whole

3.1.2 Selected architectural drivers

The non-functional drivers for this decomposition are:

- *X1*: name
- *Y3*: name

The related functional drivers are:

- *UCa*: name
- *UCb*: name
- *UCc*: name

Rationale A short discussion of why these drivers were selected for this decomposition.

3.1.3 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.

3.1.4 Instantiation and allocation of functionality

Decomposition Main aspects of the resulting decomposition.

ModuleB Per introduced component a paragraph describing its responsibilities.

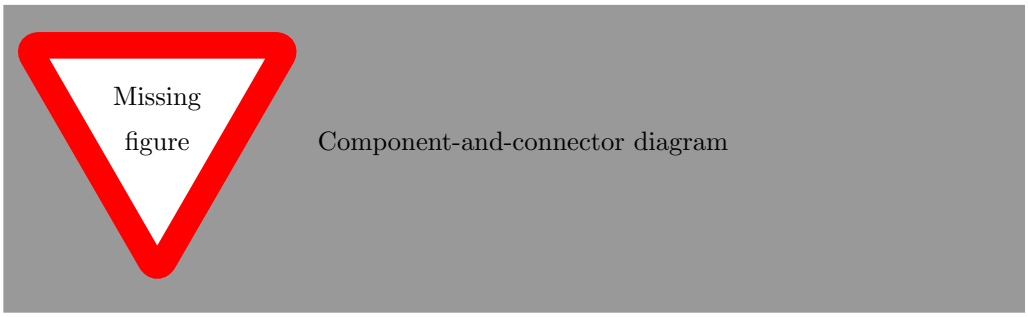


Figure 1: Component-and-connector diagram of this decomposition.

ModuleC Per introduced component a paragraph describing its responsibilities.

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

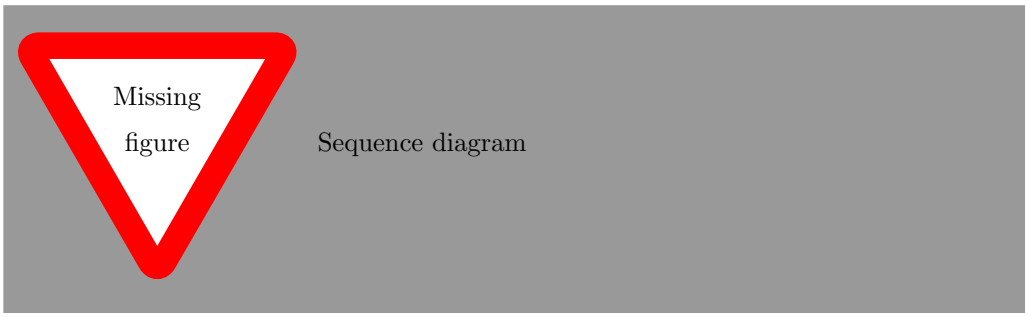


Figure 2: Sequence diagram illustrating a key behavioural aspect.

Deployment Rationale of the allocation of components to physical nodes.

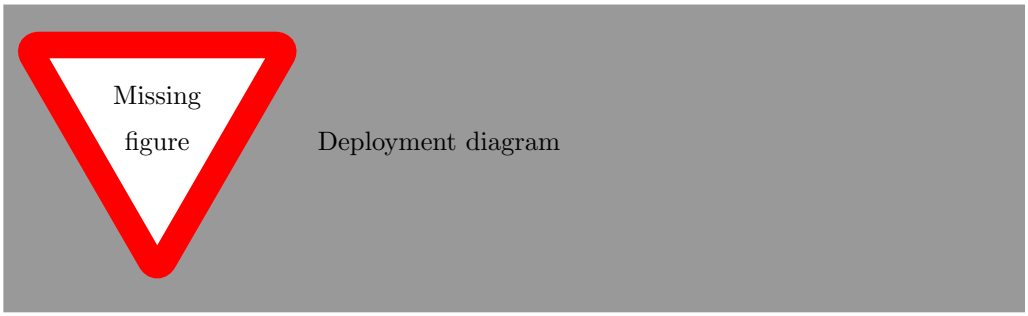


Figure 3: Deployment diagram of this decomposition.

3.1.5 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

3.1.6 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.

3.1.7 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.

3.2 Decomposition 2: OtherFunctionality(P2, UC12, UC13, UC14, UC15)

3.2.1 Module to decompose

In this run we decompose `Otherfunctionality`.

3.2.2 Selected architectural drivers

The non-functional drivers for this decomposition are:

- *X1*: name
- *Y3*: name

The related functional drivers are:

- *UCa*: name
- *UCb*: name
- *UCc*: name

Rationale A short discussion of why these drivers were selected for this decomposition.

3.2.3 Architectural design

Extended functionality of PDSLongTermDocumentManager for P2 Discussion of the solution selected for (a part of) one of the architectural drivers.

Separate document and link mapping database for P2

Sharding for document database for P2 Note that there must be a (sub)component monitoring the requests to the different shards, to cap the number of requests. We want to have the response time of active replication. But actively replicating the whole database might have to high a cost, so we choose sharding. This keeps the fast response time, but has less hardware required.

The shards themselves are

Pingen is nodig voor write bij sharding. DocumentStorageManager nodig want twee opslagkanalen, de PDSDB en de DocumentDB

extra methode in PDSDBMngmt

Alternatives considered

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

3.2.4 Instantiation and allocation of functionality

Decomposition Main aspects of the resulting decomposition.

ModuleB Per introduced component a paragraph describing its responsibilities.

ModuleC Per introduced component a paragraph describing its responsibilities.

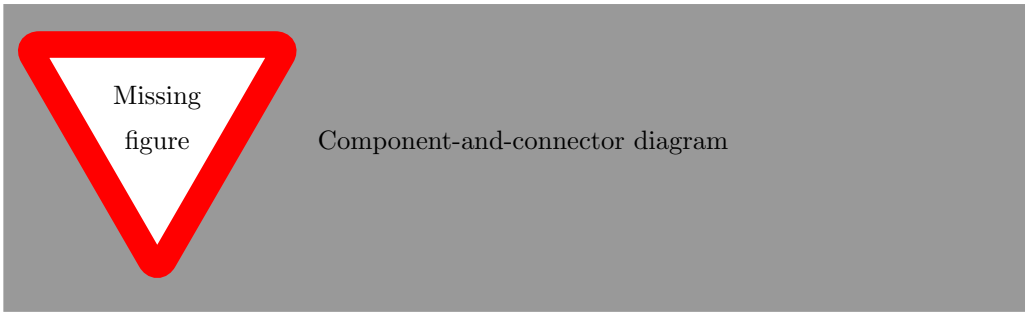


Figure 4: Component-and-connector diagram of this decomposition.

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

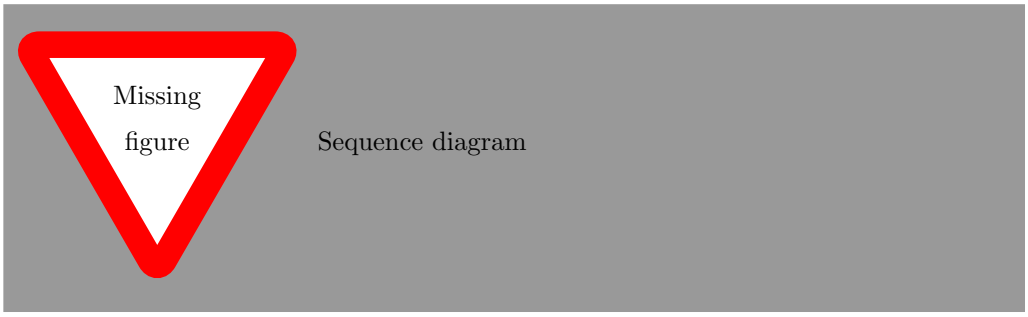


Figure 5: Sequence diagram illustrating a key behavioural aspect.

Deployment Rationale of the allocation of components to physical nodes.

3.2.5 Interfaces for child modules

ModuleB

- InterfaceA
  - `returnType operation1(ParamType param1)` throws TypeOfException

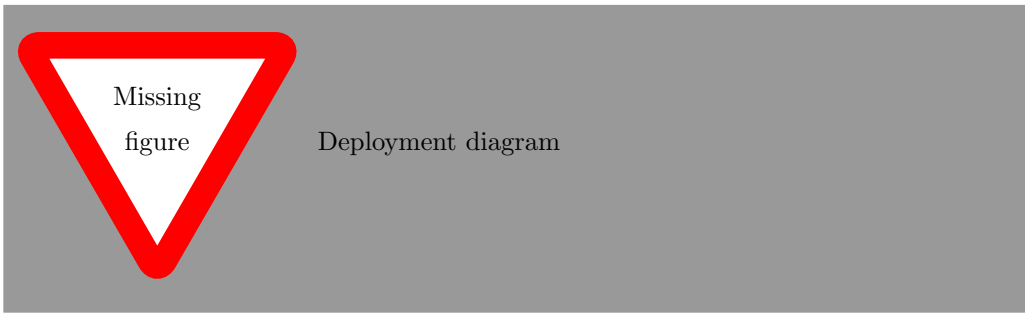


Figure 6: Deployment diagram of this decomposition.

    - 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

3.2.6 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.

3.2.7 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.

3.3 Decomposition 3: ModuleA (X1, Y3, UCa, UCb, UCc)

3.3.1 Module to decompose

In this run we decompose `ModuleA`.

3.3.2 Selected architectural drivers

The non-functional drivers for this decomposition are:

- *X1*: name
- *Y3*: name

The related functional drivers are:

- *UCa*: name

- *UCb*: name
- *UCc*: name

Rationale A short discussion of why these drivers were selected for this decomposition.

3.3.3 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.

3.3.4 Instantiation and allocation of functionality

Decomposition Main aspects of the resulting decomposition.

ModuleB Per introduced component a paragraph describing its responsibilities.

ModuleC Per introduced component a paragraph describing its responsibilities.

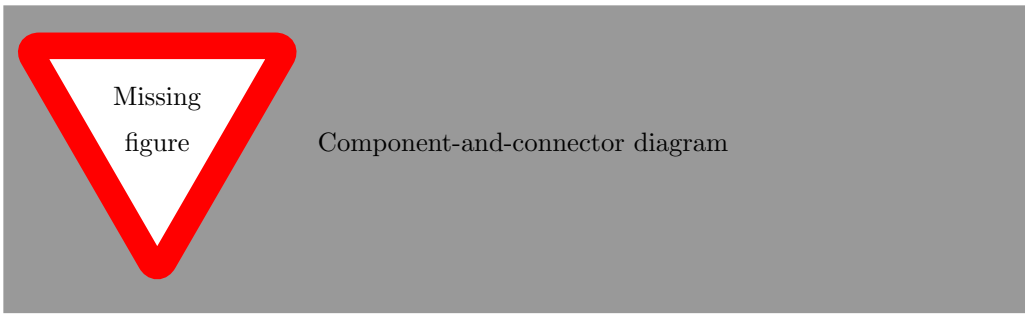


Figure 7: Component-and-connector diagram of this decomposition.

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

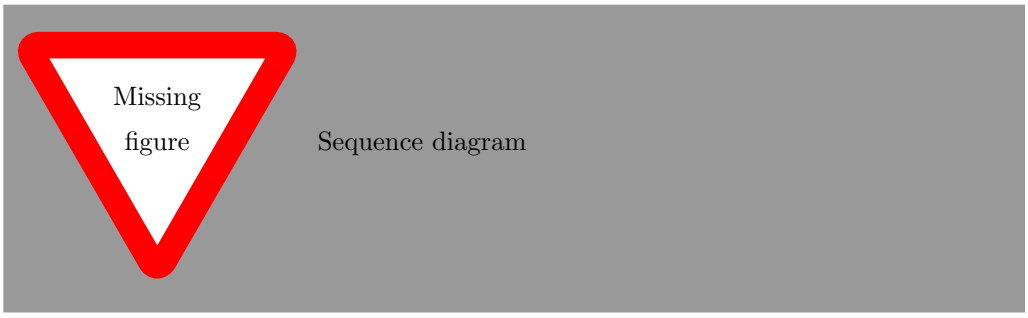


Figure 8: Sequence diagram illustrating a key behavioural aspect.

Deployment Rationale of the allocation of components to physical nodes.

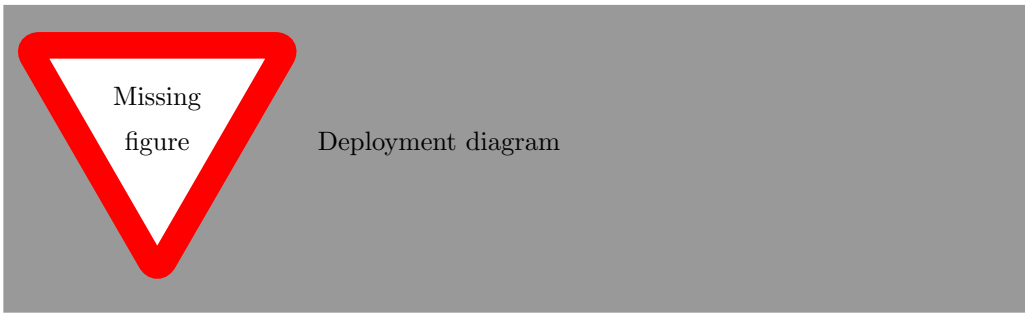


Figure 9: Deployment diagram of this decomposition.

3.3.5 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

3.3.6 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.

3.3.7 Verify and refine

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

ModuleB

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

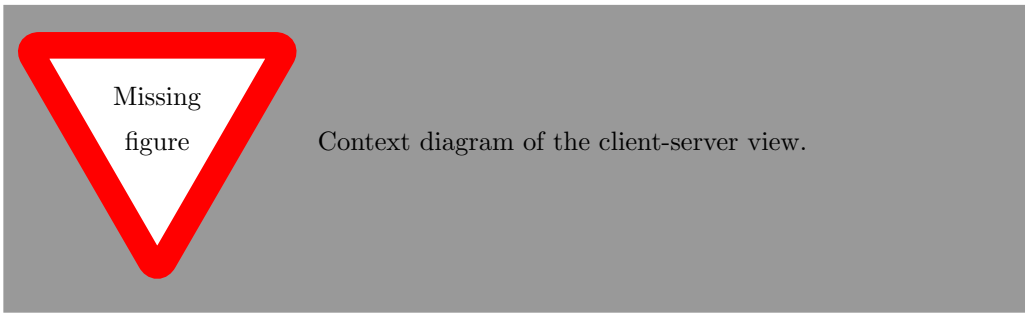
ModuleC

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

4 Client-server view (UML Component diagram)

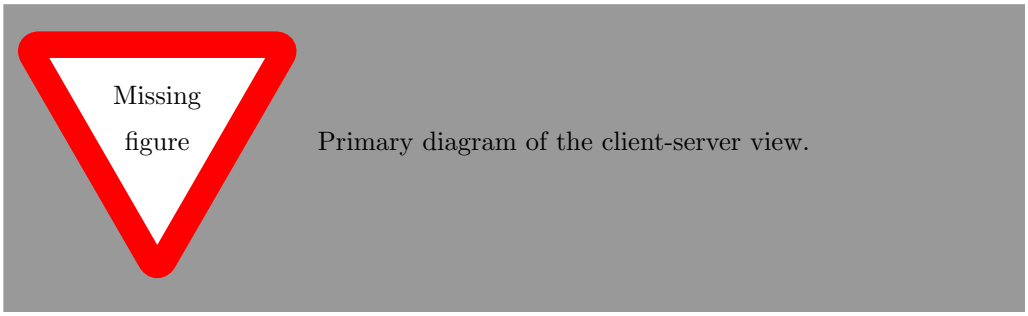
The context diagram of the client-server view. Discuss which components communicate with external components and what these external components represent.

The primary diagram and accompanying explanation.



Context diagram of the client-server view.

Figure 10: Context diagram for the client-server view.



Primary diagram of the client-server view.

Figure 11: Primary diagram of the client-server view.

4.1 Main architectural decisions

Discuss your architectural decisions for the most important requirements in more detail using the components of the client-server view. Pay attention to the solutions that you employed and the alternatives that you considered. The explanation here must be self-contained and complete. Imagine you had to describe how the architecture supports the core functionality to someone that is looking at the client-server view only. Hide unnecessary details (these should be shown in the decomposition view).

4.1.1 ReqX: requirement name

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

Alternatives considered

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

5 Decomposition view (UML Component diagram)

Discuss the decompositions of the components of the client-server view which you have further decomposed.

5.1 ComponentX

Describe the decomposition of `ComponentX` and how this relates to the requirements.

6 Deployment view (UML Deployment diagram)

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

The primary deployment diagram itself and accompanying explanation. Pay attention to the parts of the deployment diagram which are crucial for achieving certain non-functional requirements. Also discuss any alternative deployments that you considered.

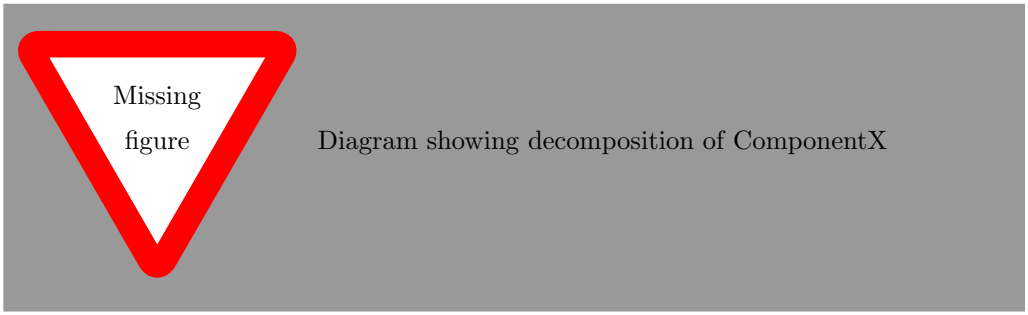
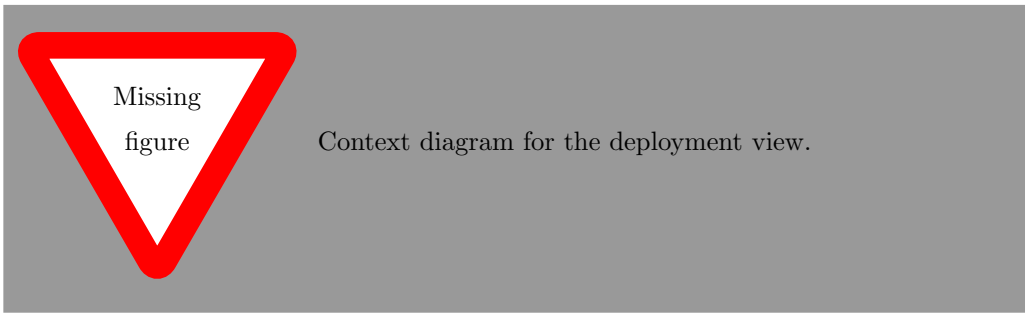


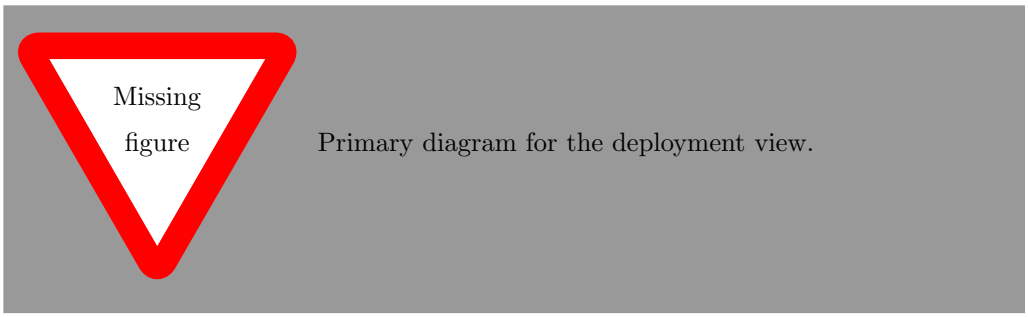
Diagram showing decomposition of ComponentX

Figure 12: Decomposition of `ComponentX`



Context diagram for the deployment view.

Figure 13: Context diagram for the deployment view.



Primary diagram for the deployment view.

Figure 14: Primary diagram for the deployment view.

7 Scenarios

Illustrate how your architecture fulfills the most important data flows. As a rule of thumb, focus on the scenario of the domain description. 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 you architecture at this point.

7.1 Scenario 1

Shortly describe the scenario shown in this subsection. Show the complete scenario using one or more sequence diagrams.

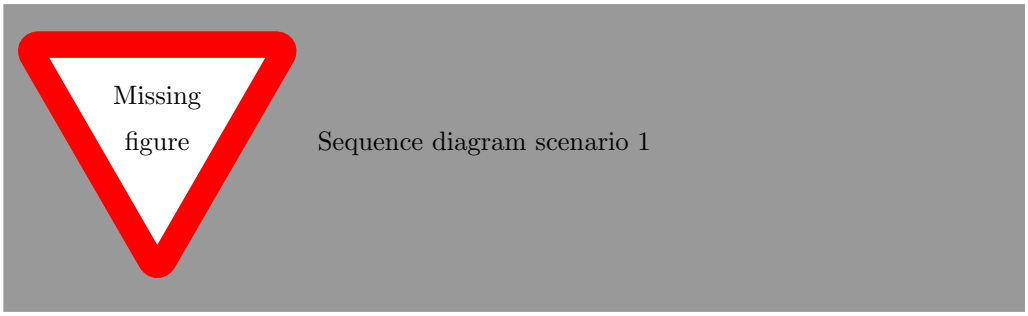


Figure 15: The system behavior for the first scenario.

A Element catalog

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.

A.1 Component 1

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

Provided interfaces

- InterfaceA
  - `returntType1 operation1(ParamType param) throws SomeException`
    - Effect: Describe the effect of the operation
    - Exceptions:
      - SomeException: Describe when the exception is thrown.
    - `void operation2(ParamType2 param)`
      - Effect: Describe the effect of the operation
      - Exceptions: None
- InterfaceB
  - `returntType2 operation3()`
    - Effect: Describe the effect of the operation
    - Exceptions: None

B Defined data types

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.