Project Software Engineering: CO2-friendly printing system

1Ba INF 2023-2024

1 Introduction

The University of Antwerp currently owns and manages an extensive systems of printers and scanners. These devices can be found in the libraries, study areas, and hallways of the various campuses of the University, and are accessible both to students and employees through the myPrint system.

Figure 1: Printer devices in the library.

With an eye on a climate-friendly and sustainable future, the University of Antwerp has developed an ambitious plan to become climate-neutral by 2030. As part of this initiative, the University wishes to get a clear view on the CO2-emissions of the printing devices across the different campuses.

In order realise this, the lab of Serge Demeyer, part of the AnSyMo research group, has been tasked with developing new printing software that is user-friendly to both students and employees while at the same time providing a good overview of all CO2-emissions of the system.

Most importantly, this system needs to be **state-of-the-art**. This means that all best practices of Software Engineering and C++ programming have to be applied. This includes: good planning, excellent communication between the customer, developers, and management, extensive testing, object-oriented programming, as well as good documentation and clearly defined contracts.

2 Use case format

Important: It is possible that some things in this specification are unclear. In the field of software development, this is the norm. If you are not sure about the meaning of something, or if you suspect it is an error, you are expected to do the following:

1. **Reflect**: What is missing? Why do you think it is an error? Why is it not clear?
2. Try to come up with some **alternatives**: what are the different ways to interpret the specification? In what different ways can it be implemented?
3. Present your question, together with the different interpretations and alternatives you propose, to one of the assistants.

The requirement specification consists of so-called use cases. Each use case describes a specific functionality of the system. Throughout the project, you will have to implement multiple use cases. It might also be possible that some use cases are modified or revised in later versions of the specification. A use case may contain the following components:

- **Use case number and title:**
  Every use cases has a title and a number. This is used to refer to that use case.
- **Priority:**
  This specification will contain too many use cases to implement during the semester. This is why the use cases come with priorities. Possible priorities are: REQUIRED (implementing these is obligatory), IMPORTANT (not obligatory, but highly preferred), USEFUL (interesting, but may be left out).
- **Goal:**
  A small description of the use case: what it is supposed to do, why is it there.
- **Precondition:**
  What needs to be present in the system before the use case functionality can be used. This will tell you exactly under what conditions the use case should work.
- **Postcondition:**
  Description of what the use case will have accomplished in case of the "happy day scenario". That is, everything went well and no errors occurred.
- **Steps:**
  A sequential description of the steps that the use case will take as part of the "happy day scenario". The steps are numbered, and may contain control flow (e.g., WHILE, IF, ELSE, ...).
- **Exceptions:**
  A list of possible problems that might occur, and how they should be handled. Each exception will specify the following:
  (a) the number of the step where the problem can occur,
  (b) a condition that specifies when the problem has occurred,
  (c) a short description of how the exception should be handled.
- **Example:**
  An example of the input or output of the use case.

Some use cases will be extensions of other use cases. In order to implement them, you will have to adjust functionalities that you have implemented earlier. In that case, the following things will be specified:

- **Extension:**
  A reference to the original use case.
- **Steps:**
  A description of how the steps of the original use case should be modified.

3 Overview

Use-Case	**Priority**
1: Input	
Use Case 1.1: Reading printers and jobs	REQUIRED
Use Case 1.2: Reading printers and jobs with different types	REQUIRED
Use Case 1.3: Reading scanners and scanning jobs	REQUIRED
Use Case 1.4: Reading printing costs	IMPORTANT
2: Output	
~~Use Case 2.1: Simple output (old)~~	~~REQUIRED~~
Use Case 2.2: Simple output (new)	REQUIRED
Use Case 2.3: Advanced textual output	IMPORTANT
Use Case 2.4: 3D graphical output	IMPORTANT
3: Simulation	
~~Use Case 3.1: Manual processing (old)~~	~~REQUIRED~~
~~Use Case 3.2: Automated processing (old)~~	~~IMPORTANT~~
Use Case 3.3: Manual processing met type (new)	REQUIRED
Use Case 3.4: Automated processing with type (new)	REQUIRED
Use Case 3.5: Multiple devices with the same type	REQUIRED
Use Case 3.6: CO2-friendly job scheduling	IMPORTANT
Use Case 3.7: Tracking CO2 emissions	REQUIRED
Use Case 3.8: CO2 compensation	IMPORTANT
Use Case 3.9: Statistical calculations	USEFUL
4: User interface	
Use Case 4.1: Admin GUI	USEFUL
Use Case 4.2: User GUI	IMPORTANT
Use Case 4.3: GUI with CO2 emissions overview	USEFUL

Use Case 1.1: Reading printers and jobs

Priority: REQUIRED

Goal: The printing system should be read from an XML file. The file reader should be able to handle all the information in the file, read it correctly, check for errors, and inform the user of any errors in the input.

Precondition: An XML file in ASCII format needs to be presented to the system. The XML file needs to contain information about the devices and jobs. See appendix A for a detailed description of the input format.

Postcondition: The system contains a printing system with printer devices and printing jobs. All devices and jobs in the system need to conform to the information specified in the XML file.

Steps:

1. Open input file
2. WHILE the file is not yet fully read:
   2.1. Recognize element (DEVICE, JOB)
   2.2. Read information about the element (see appendix A)
   2.3. IF the information is correct
      2.3.1. THEN add element to the printing system
      2.3.2. ELSE print error message; go to next element in the input file.
3. Verify the consistency of the printing system (see appendix A).
4. Close input file

Exceptions:

Step 2.1. [Unrecognizable element] Print error message. Go to step 2 and begin reading the next element.
Step 2.2. [Invalid information] Print error message. Go to step 2 and begin reading the next element.
Step 3 [Inconsistent printing system] Print error message. Go to step 4.

Example: Example printing system with one printer, and two printing jobs.

```
<SYSTEM>
    <DEVICE>
        <name>Office_Printer5</name>
        <emissions>3</emission>
        <speed>40</speed>
    </DEVICE>
    <JOB>
        <jobNumber>89751</jobNumber>
        <pageCount>2</pageCount>
        <userName>SergeDemeyer</userName>
    </JOB>
    <JOB>
        <jobNumber>2189</jobNumber>
        <pageCount>3</pageCount>
        <userName>anonymous_user</userName>
    </JOB>
</SYSTEM>
```

Use Case 1.2: Reading printers and jobs with different types

Priority: REQUIRED

Goal: The system should be able to read two different types of printer devices and jobs: black-and-white printing, and color printing.

Precondition: Same as `Use Case 1.1: Reading printers and jobs`.

Postcondition: Same as `Use Case 1.1: Reading printers and jobs`.

Extensions: This use case extends `Use Case 1.1: Reading printers and jobs`.

Steps: You will have to modify step 2.2. During this step you will, additionally, have to take into account the `type` attribute to support both black-and-white printing (`bw`), and color printing (`color`). See appendix A for detailed information.

Example: Example printing system with one printer for each type, and two printing jobs of different types.

```
<SYSTEM>
    <DEVICE>
        <name>Office_Printer5</name>
        <emissions>3</emission>
        <type>bw</type>
        <speed>40</speed>
    </DEVICE>
    <DEVICE>
        <name>ColorPrinter</name>
        <emissions>6</emission>
        <type>color</type>
        <speed>32</speed>
    </DEVICE>
    <JOB>
        <jobNumber>89751</jobNumber>
        <pageCount>2</pageCount>
        <type>color</type>
        <userName>SergeDemeyer</userName>
    </JOB>
    <JOB>
        <jobNumber>2189</jobNumber>
        <pageCount>3</pageCount>
        <type>bw</type>
        <userName>anonymous_user</userName>
    </JOB>
</SYSTEM>
```

Use Case 1.3: Reading scanners and scanning jobs

Priority: REQUIRED

Goal: The system should also be able to handle scanner devices and jobs.

Precondition: Same as `Use Case 1.1: Reading printers and jobs`.

Postcondition: Same as `Use Case 1.1: Reading printers and jobs`.

Extensions: This use case extends `Use Case 1.1: Reading printers and jobs`.

Steps: You will have to modify step 2.2. During this step you will, additionally, have to take into account the `type` attribute to support scanners. See appendix A for detailed information.

Example: Example system with a scanner and a scanning job.

```
<SYSTEM>
    <DEVICE>
        <name>Default Scanner</name>
        <emissions>8</emission>
        <type>scan</type>
        <speed>11</speed>
    </DEVICE>
    <JOB>
        <jobNumber>98632</jobNumber>
        <pageCount>10</pageCount>
        <type>scan</type>
        <userName>Peter Selie</userName>
    </JOB>
</SYSTEM>
```

Use Case 1.4: Reading printing costs

Priority: IMPORTANT

Goal: In order to properly calculate how much it costs to run the printing system, the system should keep track of the costs of each job.

Precondition: Same as `Use Case 1.1: Reading printers and jobs`.

Postcondition: Same as `Use Case 1.1: Reading printers and jobs`.

Extensions: This use case extends `Use Case 1.1: Reading printers and jobs`.

Steps: You will have to modify step 2.2. During this step you will, additionally, have to take into account the `cost` attribute of a `DEVICE` element. See appendix A for detailed information.

Example: Example printing system with one printer, and two printing jobs.

```
<SYSTEM>
    <DEVICE>
        <name>Office_Printer5</name>
        <emissions>3</emission>
        <speed>40</speed>
        <cost>20</cost>
    </DEVICE>
    <JOB>
        <jobNumber>89751</jobNumber>
        <pageCount>2</pageCount>
        <userName>SergeDemeyer</userName>
    </JOB>
    <JOB>
        <jobNumber>2189</jobNumber>
        <pageCount>3</pageCount>
        <userName>anonymous_user</userName>
    </JOB>
    <DEVICE>
        <name>Default Scanner</name>
        <emissions>8</emission>
        <type>scan</type>
        <speed>11</speed>
        <cost>110</cost>
    </DEVICE>
    <JOB>
        <jobNumber>98632</jobNumber>
        <pageCount>10</pageCount>
        <type>scan</type>
        <userName>Peter Selie</userName>
    </JOB>
</SYSTEM>
```

Use Case 2.1: Simple output (old)

Priority: REQUIRED

Goal: The system should print an easy-to-read status report.

Precondition: A valid printing system is loaded.

Postcondition: The system has created a text file (ASCII) that contains all the necessary information about the system.

Steps:

1. Create output file.
2. WHILE printers left
   2.1. Print information about the printer to the output file.
3. WHILE jobs left
   3.1. Print information about the job to the output file.
4. Close output file.

Example:

```
NEW-Printer (CO2: 5g/page):
    * Current:
        [#789|KasperEngelen]
    * Queue:
        [#423|Peter Selie]
        [#98712|New-user]
```

Use Case 2.2: Simple output (new)

Priority: REQUIRED

Goal: To gain better insight into the system, the University wants to have a more extensive output. For example, the jobs should be printed in more detail, and there needs to be more detailed info about the amount of pages.

Note: Depending on which other use cases have been implemented, this use case might display different information!

Precondition: Same as `Use Case 2.1: Simple output (old)`.

Postcondition: The system has created a text file (ASCII) that contains all the necessary information about the system.

Extension: This use case extends `Use Case 2.1: Simple output (old)`

Steps:

1. Create output file.
2. WHILE devices left
   2.1. Print information about the device to the output file.
3. WHILE jobs left
   3.1. Print information about the job to the output file.
4. WHILE climate compensation initiatives left
   4.1. Print information about the initiative to the output file.
5. Close output file.

Example:

```
# === [System Status] === #

-=== Devices ==-=

High-resolution:
* CO2: 2g/page
* 12 pages / minute
* Scanner
* 60 cents / page

Office_Groundfloor:
* CO2: 6g/page
* 60 pages / minute
* Black-and-white printer
* 10 cents / page

-=== Jobs ==-=

[Job #5487]
* Owner: SergeDemeyer
* Device: Office_Groundfloor
* Status: 12 pages done
* Total pages: 65 pages
* Total CO2: 390g CO2
* Total cost: 650 cents

[Job #345]
* Owner: User_65643147
* Device: Office_Groundfloor
* Status: WAITING #1
* Total pages: 3 pages
* Total CO2: 18g CO2
* Total cost: 30 cents
* Compensation: The CO2 Reduction Company

-=== Co2 Compensation initiatives ==-=

The CO2 Reduction Company [#78465]
UN Climate Comittee [#3659]

# ======================= #
```

Use Case 2.3: Advanced textual output

Priority: IMPORTANT

Goal: The current state of the system is graphically rendered in an ASCII format.

Precondition: A valid printing system has been loaded.

Postcondition: The system has saved a text-file that contains a graphical ASCII-representation of the system.

Steps:

1. Create output file.
2. Write the graphical representation to the file.
3. Close the output file.

Example: You will have to come up with a format yourself. Below you find an example. Every other line contains the name of a device. Below the device, you can see the current status. On the left of the `|` you see the current job, and how many pages are still left to process. On the right of the `|`, you see the jobs in the queue. Each job displays the total mount of pages.

```
Office printer
    [21/23] | [40][3][44]
Scanner_In_Hallway7
    |
Device_192.168.32.102
    [1/3] | [1][1545][19]
```

Use Case 2.4: 3D graphical output

Priority: IMPORTANT

Goal: In order to properly present the project, there should be a 3D visualisation of the system. You should make use of the graphics engine you made for the Computer Graphics course.

Precondition: A valid printing system is loaded.

Postcondition: The processing of jobs is visualised in a 3D environment.

Use Case 3.1: Manual processing (old)

Priority: REQUIRED

Goal: The printing system should be able to take a job and have to processed by a printer.

Precondition: A valid printing system is loaded.

Postcondition: All the pages of the job have been printed.

Steps:

1. WHILE pages left:
   1.1. Print page.
2. Print a message to the screen.

Example: If a job with 3 pages and job number 13989 was submitted by "John Doe" to the printer "Library Printer 5", the following message would be printed after the job was finished:

```
Printer "Library Printer 5" finished job:
    Number: 13989
    Submitted by "John Doe"
    3 pages
```

Use Case 3.2: Automated processing (old)

Priority: IMPORTANT

Goal: The system should be able to automatically run multiple jobs one after the other.

Precondition: A valid printing system is loaded.

Postcondition: All jobs are finished.

Steps:

1. WHILE unfinished jobs left
   1.1. Pick job
   1.2. Perform `Use Case 3.1: Manual processing (old)`

Use Case 3.3: Manual processing met type (new)

Priority: REQUIRED

Goal: The system should be able to handle multiple types of jobs and have them processes by the appropriate device.

Precondition: Same as `Use Case 3.1: Manual processing (old)`.

Postcondition: Same as `Use Case 3.1: Manual processing (old)`.

Extension: Extends `Use Case 3.1: Manual processing (old)`.

Steps:

1. WHILE pages left:
   1.1. IF job type is `bw_print`:
      1.1.1. Print black and white page.
   1.2. ELSE IF job type is `color_print`:
      1.2.1. Print colored page.
   1.3. ELSE IF job type is `scan`:
      1.3.1. Scan page.
2. Print a message to the screen that depends on the type of the job.

Example: If a color-printing job with 3 pages and job number 13989 was submitted by "John Doe" to the printer "Library Printer 5", the following message would be printed after the job was finished:

```
Printer "Library Printer 5" finished color-printing job:
    Number: 13989
    Submitted by "John Doe"
    3 pages
```

Use Case 3.4: Automated processing with type (new)

Priority: REQUIRED

Goal: When automatically processing all jobs one by one, the type of the job needs to be taken into account.

Precondition: Same as `Use Case 3.2: Automated processing (old)`.

Postcondition: Same as `Use Case 3.2: Automated processing (old)`.

Steps:

1. WHILE unfinished jobs left
   1.1. Pick job
   1.2. Choose device according to job type
   1.3. Perform `Use Case 3.3: Manual processing met type (new)`

Extension: Extends `Use Case 3.2: Automated processing (old)`

Example: an example with three jobs:

```
Printer "Library Printer 5" finished color-printing job:
    Number: 13989
    Submitted by "John Doe"
    3 pages

Printer "Library Printer 5" finished black-and-white job:
    Number: 132
    Submitted by "New User"
    60 pages

Printer "Brand new scanner" finished scanning job:
    Number: 368
    Submitted by "Serge Demeyer"
    5 pages
```

Exceptions: The following exception can occur:

Step 1.2 [No device exists for the specified job type] Print an error message that the job could not be printed.

Use Case 3.5: Multiple devices with the same type

Priority: REQUIRED

Goal: The system should support multiple devices of the same type. This means that there are, for example, two scanners, five color printers, etc.

Precondition: A valid printing job has been loaded by the system.

Postcondition: Given a new job, the system will choose a device that is capable of printing the job.

Extension: Extends `Use Case 3.4: Automated processing with type (new)`.

Steps: Step 1.2 will have to be modified:

[1.2.] IF there exists more than one device of the required type:
- [1.2.a.] THEN:
  - [1.2.a.1.] Choose the device with the **least pages** left to print (= pages of the active job and the jobs in the queue).
  - [1.2.a.2.] Perform `Use Case 3.3: Manual processing met type (new)`
- [1.2.b.] ELSE:
  - [1.2.b.1.] Choose the unique device of the required type.
  - [1.2.b.2.] Perform `Use Case 3.3: Manual processing met type (new)`

Exceptions: The following exception can occur:

Step 1.2.a.1 [No device exists for the specified job type] Print an error message that the job could not be printed.

Step 1.2.b.1 [No device exists for the specified job type] Print an error message that the job could not be printed.

Use Case 3.6: CO2-friendly job scheduling

Priority: IMPORTANT

Goal: When the scheduler assigns jobs to devices, it should take into account the CO2-emissions of the devices.

Precondition: Same as `Use Case 3.5: Multiple devices with the same type`.

Postcondition: The jobs are scheduled, taking into account the CO2-emissions of the devices.

Extension: Extends `Use Case 3.5: Multiple devices with the same type`.

Steps: Step 1.2 will now have to be modified in a different way:

[1.2.] IF more than one device of the required type:
- [1.2.a.] THEN:
  - [1.2.a.1.] FOR each device `dev` that does not exceed the CO2-limit:
    - [1.2.a.1.a.] `value(dev) += pages left in active job + pages in queue`
    - [1.2.a.1.b.] `value(dev) = value(dev) * CO2 per page`
  - [1.2.a.2.] Choose the device with minimal `value(dev)` and that does not exceed the CO2-limit.
  - [1.2.a.3.] Perform `Use Case 3.3: Manual processing met type (new)`
- [1.2.b.] ELSE:
  - [1.2.b.1.] Choose the unique device of the required type that does not exceed the CO2-limit.
  - [1.2.b.2.] Perform `Use Case 3.3: Manual processing met type (new)`

Exceptions: The following exception can occur:

Step 1.2.a.1 [No device exists for the specified job type] Print an error message that the job could not be printed.

Step 1.2.a.1 [All devices of the required type exceed the CO2 limit] Print error message and remove job from the system.

Step 1.2.b.1 [No device exists for the specified job type] Print an error message that the job could not be printed.

Step 1.2.b.1 [The unique devices of the required type exceed the CO2 limit] Print error message and remove job from the system.

Use Case 3.7: Tracking CO2 emissions

Priority: REQUIRED

Goal: The system should be able to keep track of the CO2 missions of all executed jobs.

Precondition: Same as `Use Case 3.4: Automated processing with type (new)`.

Postcondition: The system has stored the total CO2 emissions of all the jobs that were executed.

Extension: This use-case extends `Use Case 3.4: Automated processing with type (new)`.

Steps: The following steps have to be added to the use-case:
[After step 1.3] Increment the CO2-counter with the value of the `emissions` attribute of the device.

Use Case 3.8: CO2 compensation

Priority: IMPORTANT

Goal: In order for the system to be CO2-neutral, the system needs to offer users a way to compensate the CO2 emissions of their jobs.

Precondition: Same as `Use Case 1.1: Reading printers and jobs` and `Use Case 3.3: Manual processing met type (new)`.

Postcondition: The system allows the user to specify a way to compensate the CO2 emissions of a job. If the user has specified this, then the compensation is carried out.

Extension: This use case extends `Use Case 1.1: Reading printers and jobs`.

Steps: The following steps have to be modified:

Step 2.1. Take into account the `COMPENSATION` elements.

Step 2.2. Take into account the `compNumber` attribute of the `JOB` elements.

Example:

```
<SYSTEM>
...
    <JOB>
        <jobNumber>13989</jobNumber>
        <pageCount>3</pageCount>
        <type>color</type>
        <userName>John Doe</userName>
        <compNumber>545684</compNumber>
    </JOB>
    <COMPENSATION>
        <compNumber>545684</compNumber>
        <name>Students 4 Climate</name>
    </COMPENSATION>
...
</SYSTEM>
```

Extension: This use case extends `Use Case 3.3: Manual processing met type (new)`.

Steps: The following steps have to be added to the use-case:

3. IF compensation was specified
   3.1. Print a message with the name of the compensation.

Example:

```
Printer "Library Printer 5" finished color-printing job:
    Number: 13989
    Submitted by "John Doe"
    3 pages
Job 13989 was made CO2 neutral with the support of "Students 4 Climate".
```

Use Case 3.9: Statistical calculations

Priority: USEFUL

Goal: While the system runs and jobs are completed by the devices, the system collects interesting statistics. Examples are

- the total operating costs of the system,
- which devices are used the most,
- the total CO2-emissions of the system,
- the average CO2-emissions per page,
- Which CO2-compensation initiatives are selected the most often,
- …

Which statistics are collected also depends on which use-cases are implemented or not.

Precondition: A valid printing system is loaded.

Postcondition: A report with the statistics has been saved to a file, as well as displayed to the screen.

Steps:

1. WHILE the system is running
   1.1. Collect information
2. Calculate statistics
3. Save the report.
4. Print the report.

Extension: This use case extends the following use case: `Use Case 3.4: Automated processing with type (new)`

Use Case 4.1: Admin GUI

Priority: USEFUL

Goal: A user-friendly interface for managing the system. A number of functionalities need to be present;

- Buttons to go forward or backwards.
- A button to start or pause the automatic simulation of the system.
- Overview of all the devices.
- Overview of all jobs.
- For each device it is possible to see information about the CO2 emissions about that device.
- Devices with a CO2 emission greater than their prescribed limit need to be marked in red (see appendix A).

Precondition: A valid printing system is loaded.

Postcondition: The administrator can manage the printing system via the GUI.

Use Case 4.2: User GUI

Priority: IMPORTANT

Goal: A user-friendly GUI for a user of the system. The user should be able to add extra jobs.

Precondition: A valid printing system is loaded.

Postcondition: The user sees which jobs have been added, how far they are finished, etc.

Steps:

1. The user specifies a username.
2. The user specifies the job: type, number of pages.
3. The user can, optionally, select a way to compensate the CO2 emissions.
4. The user has to select a device that is compatible with the job type (printer, scanner).

Exceptions:

Step 3. [The specified CO2-compensation does not exist] Display an error message that says that the specified CO2-compensation does not exist.

Step 4. [The selected device does not exist] Display an error message that says that the selected device does not exist.

Step 4. [The selected device exceeds the allowed CO2 limit (see appendix A)] Display an error message that says that the selected device is not CO2-friendly.

Use Case 4.3: GUI with CO2 emissions overview

Priority: USEFUL

Goal: The goal is to provide a user-friendly GUI that displays information about the CO2-emissions of the system.

Precondition: A valid printing system is loaded.

Postcondition: The user can see statistics about the CO2-emissions of the system.

Extension: This use-case extends `Use Case 3.7: Tracking CO2 emissions` and `Use Case 3.9: Statistical calculations`.

A Input format

The input format for the printing system is made in such a way that new elements and attributes easily be added. All input files will be specified as XML files.

```
File = "<SYSTEM>" ElementList "</SYSTEM>"
ElementList = Element { Element ... }
Element = "<" ElementType ">" AttributeList "</" ElementType ">"
ElementType = (see table)
AttributeList = Attribute { Attribute ... }
Attribute = "<" AttributeName ">" AttributeValue "</" AttributeName ">"
AttributeName = (see table)
AttributeValue = Integer | String | Float
Integer = Digit { Digit ... }
Float = Digit { Digit ... } "." Digit { Digit ... }
Digit = "0" | ... | "9"
String = Character { Character ... }
Character = "a" | ... | "z" | "A" | ... | "Z"
```

There are different types of elements that can be added to the input file. Each type of element has some required attributes, which have to be specified, as well as optional attributes, that the user may or may not specify.

Element type	Attributes (required)	Attributes (optional)
`DEVICE`	`name`, `type*`, `emissions`, `speed`, `cost**`	N/A
`JOB`	`jobNumber`, `type*`, `pageCount`, `userName`	`compNumber***`
`COMPENSATION***`	`compNumber***`, `name***`	N/A

Which elements and attributes have to be supported will depend on which use cases have been implemented:

- `*` `Use Case 1.2: Reading printers and jobs with different types` and/or `Use Case 1.3: Reading scanners and scanning jobs`
- `**` `Use Case 1.4: Reading printing costs`
- `***` `Use Case 3.8: CO2 compensation`.

Each attribute has a specific type of value that can be assigned to it:

Attribute	Type of value
`name`, `type`, `userName`	String
`jobNumber`, `emissions`, `speed`, `pageCount`, `compNumber`	Integer
`cost`	Float

The `emissions`, `speed`, and `cost` are specified in the following units:

Attribute	Unit
`emissions`	gram CO2 per page
`speed`	pages per minute
`cost`	euro cents (= 1/100 euros)

Printers and scanners have different amount of CO2 caps:

Device type	**CO2 limit**
Black-and-white printers	8g CO2 per page
Color printers	23g CO2 per page
Scanner	12g CO2 per page

Note that the opening tag has to correspond to the closing tag. For more information, see for example the Wikipedia page of the XML format.

There is the additional requirement that every input file, has to be **consistent**:

- The attributes `emissions`, `pageCount`, `cost`, `speed`, `compNumber`, and `jobNumber` must not be negative.
- If a job specifies a `compNumber`, then there must exist a `COMPENSATION` element with that `compNumber`.
- Every `COMPENSATION` element needs to have a unique `compNumber`.
- Every `JOB` element needs to have a unique `jobNumber`.

Any and all input files that do not conform to the above requirements are not valid.