OCORA

Open CCS On-board Reference Architecture

Economic Model

Model Description

Document ID: OCORA-BWS06-030

Version: 2.11

Date: 31.05.2023

Revision history

Version	Change Description	Initial	Date of change
1.01	First R1 version for WS review	CTR	16.11.2021
1.02	• R1 Version for Coreteam review	NPA	19.11.2021
2.00	Official version for OCORA Release R1	NPA	26.11.2021
2.01	Decoupled document from a specific OCORA release	NPA	10.06.2022
2.11	Checking typos and reformulating some sentences	VI	31.05.2023

Table of contents

Table of figures

Table of tables

References

Reader's note: please be aware that the numbers in square brackets, e.g. [1], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

[1] OCORA-BWS01-010 – Release Notes

[2] OCORA-BWS01-020 – Glossary

[3] OCORA-BWS01-030 – Question and Answers

[4] OCORA-BWS01-040 – Feedback Form

[5] OCORA-BWS03-010 – Introduction to OCORA

[6] OCORA-BWS04-010 – Problem Statements

[7] OCORA-BWS06-020 – Economic Model

1 Introduction

1.1 Purpose of the document

The purpose of this document is to complete the vision of the Economic Model [7] with a description of the functioning of its simulation process. It allows to understand the construction of the program and its operating conditions.

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the Feedback Form [4].

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, complete on-board CCS system, or on-board CCS replacements for functional upgrades or life-cycle reasons.

If you are an organisation interested in developing on-board CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

1.2 Applicability of the document

The document is currently considered informative but may become a standard at a later stage for OCORA compliant on-board CCS solutions. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration.

1.3 Context of the document

This document is published as part of the OCORA Release, together with the documents listed in the Release Notes [1]. Before reading this document, it is recommended to read them. If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [5], and the Problem Statements [6]. The reader should also be aware of the Glossary [2] and the Question and Answers [3].

2 Simulation Tool

The simulation of the deployment of fleets scenarios is made by a program in a VBA module of the Economic Model file [7]. This module is activated by a button “run simulation” in the scenarios sheet, which, once pressed, executes the function **SimulationTool**.

2.1 Definition and principles

The function **SimulationTool** makes sure the user wants to run the simulation before running the sub-function **Simu**. It does so by displaying a message box asking “Do you want to modify the cost assumption tables?”. If “yes” is clicked, the function exits the simulation, else it starts the simulation.

2.2 Sub-function

This function launches the sub-function **Simu**.

3 Read

As many abbreviations are used, it can be useful to have a function that can check if a word is contained in another location.

3.1 Definition and principles

Read (**R** in the script) is a function which takes two strings and check if the first is in the second.

Example: let T be any **String**,

R(“”, T) will return **True** as the empty string “” is present in every string.

3.2 Sub-function

The function uses the VBA function **Mid**.

Mid returns a **Variant** (**String**) containing a specified number of characters from a string.

3.3 Inputs

L and T two **Strings**.

3.4 Outputs

The function returns a **Boolean**, **True** if the first string is present in the second, else **False**.

4 Simulation

Simulation is the function which executes the simulation of the business model.

4.1 Definition and principles

Simulation (**Simu** in the script) is a procedure using the scenarios table to create a result table and filling it with the characteristics of each fleet, along with their acquisition costs, RTS costs and maintain costs.

4.2 Sub-function

The function uses the sub-functions:

- **CreaResultT:** Creation of Result Table;
- **AcqCostCal:** Acquisition Cost Calculation**;**
- **RTSCostCal:** RTS Cost Calculation**;**
- **MaintCostCal:** Maintenance Cost Calculation**;**
- **Sum**.

4.3 Assumptions and constraints

The function supposes that the first fleet in the scenarios table is in the third line of the table, and that the last line of the table is not a fleet.

4.4 Operation and implementation

The function will execute **CreaResultT** to create a Result Table.

Then, for each fleet of the scenarios table, it will execute **AcqCostCal**, **RTSCostCal** and **MaintCostCal** to calculate and fill the Result Table with respectively the acquisition, RTS and maintain costs.

To finish, the function will execute **Sum**.

5 Creation of result table

The function creates and fills the Result Table with the fleets information.

5.1 Definition and principles

CreaResultT fills the fourth columns of the result table with the type of cost, the fleet number, the class/name, and the product applied for each fleet.

				2025	2026	2027	2028	2029	2030	2031	2032	
Acquisition costs	Fleet 1	EMU/DMU C	EVC									
Acquisition costs	Fleet 2	EMU/DMU C	pre-O.									
Total Acquisition Costs												
other CCS Induced costs	Fleet 1	EMU/DMU C	EVC									
other CCS Induced costs	Fleet 2	EMU/DMU C	pre-O.									
Total other CCS induced Costs												
Maintenance costs	Fleet 1	EMU/DMU C	EVC									
Maintenance costs	Fleet 2	EMU/DMU C	pre-O.									
Total maintenance Costs												

Table 1 Result table of two fleets without results

5.2 Assumptions and constraints

The function supposes that the first fleet in the scenarios table is in the third line, also that the last line of the table is not a fleet.

Besides, the function does not modify the two first lines of the Result Table.

5.3 Inputs

None

5.4 Configuration data and parameters

The scenarios table

5.5 Operation and implementation

The function copies in a table the fleet number, the class/name, the product applied of each fleet from the scenarios table and paste it in the result table one time for each type of costs. The function adds a “Total” line at the end of each type of costs (“Total Acquisition Costs”, “Total other CCS induced Costs” and “Total maintenance Costs”).

6 Acquisitions costs calculation

The function **AcqCostCal** calculates for a fleet of train the acquisition costs for every year.

6.1 Definition and principles

For each year, the function will add up the acquisition cost of a fleet (if trains are added that year) and the update costs (if updates are foreseen that year).

6.2 Sub-function

The function uses **Read** to search “WS” or “HW” in the name of update to know if it applied.

6.3 Assumptions and constraints

The function supposes that:

- the product applied is in fourth column of scenarios table;
- the class/name of a fleet is in second column of scenarios table;
- the first fleet in the scenarios table is in the third line;
- the last line of the table is not a fleet;
- the columns of products are the columns 3 to 6 in costs assumption tables;
- if a product contains the sub-product (indicated in PBS) the case contains “x”;
- the parameter table is complete with all the products;
- the first column of the roadmaps in PBS table is 29;
- the last column of the roadmaps in PBS table is 58;
- the scenarios in scenarios table start after the third column and end before the thirty-fifth column.

6.4 Inputs

The function input is an **Integer** representing the line number of the fleet in scenarios table for which the acquisition costs will be calculated.

6.5 Configuration data and parameters

PBS roadmap, costs assumption tables, simulation table.

6.6 Outputs

A **Single** representing the values of acquisition costs on the line of the Result Table corresponding to the line of Scenarios Table.

6.7 Operation and implementation

To calculate the acquisition costs of a fleet the function will browse all the sub subsystems of the fleet product.

For each subsystem, **AcqCostCal** will calculate the costs between the last update (included) and the next update (excluded), while the dates of the updates are in the roadmap.

To calculate the acquisition costs of a subsystem, the function will browse the WBS activities of the subsystem.

For each cost of a WBS activity of a subfunction, the function will process:

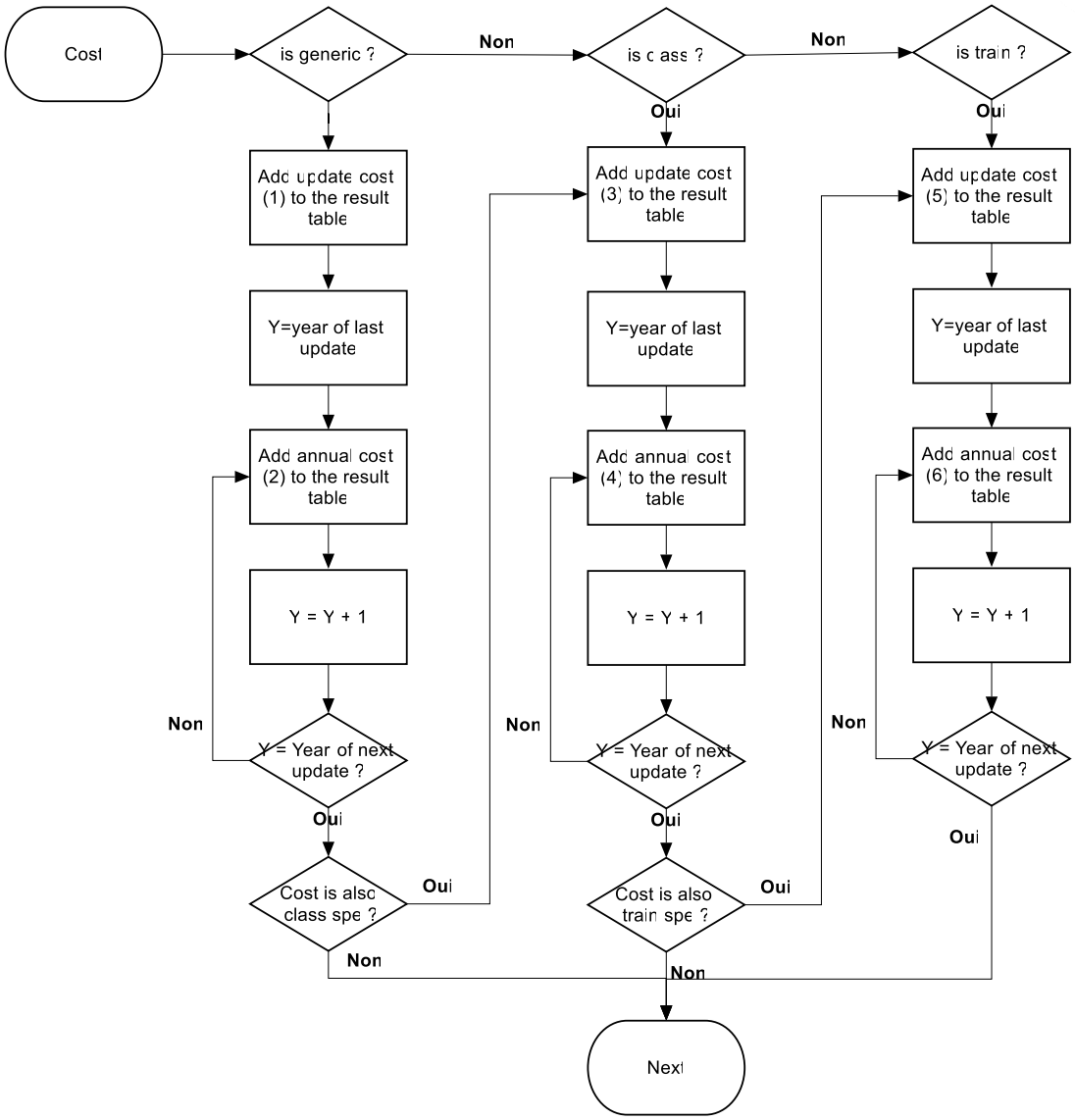

Figure 1 Programming flowchart of the cost calculation loop of AcqCostCal

Calculation of the costs shows configurable values defined in the Economic Model:

- *Pa* is the WBS parameter in the WBS table, it indicates the part of the cost applied generically, class specifically or train specifically;
- *PaU* is the value in parameter table, it is the coefficient to apply to the total cost for a type of update.

The calculations processed by the function are:

(1) Calculation of update cost for a generic WBS activity:

$$GeUpCost = \frac{Cost * Pa * PaU}{NbrG} * \sum_{first\ year}^{last\ update} NbrTrains$$

$NbrG$ is the total number of trains with the subsystem considered at the date of the year before the Next Update,

$NbrTrains$ is the number of trains of a year, the value in the scenario table.

(2) Calculation of annual cost for a generic WBS activity:

$$GeCost = \frac{Cost * Pa * NbrTrains}{NbrG}$$

(3) Calculation of update cost for a class specific WBS activity:

$$ClassUpCost = \frac{Cost * Pa * PaU}{NbrClass} * \sum_{first\ year}^{last\ update} NbrTrains$$

NbrClass is the total number of trains with the class considered at the date of the year before the next update.

(4) Calculation of annual cost for a class specific WBS activity:

$$ClassCost = \frac{Cost * Pa * NbrTrains}{NbrClass}$$

(5) Calculation of update cost for a train specific WBS activity:

$$TUpCost = Cost * Pa * PaU * \sum_{first\ year}^{last\ update} NbrTrains$$

(6) Calculation of annual cost for a train specific WBS activity:

$$TCost = Cost * Pa * NbrTrains$$

Except when $NbrTrains < 0$, if $NbrTrains<0$ and the WBS activity name contains “Removal”

$$TCost = Cost * Pa * - NbrTrains$$

Else (i.e. if $NbrTrains<0$ and the WBS activity name does not contain “Removal”), the calculation is skipped.

7 RTS Cost calculation

The function **RTSCostCal** calculates, for a fleet of train, the RTS costs for every year.

7.1 Definition and principles

For each year, the function will add up the RTS costs of a fleet (if trains are added that year) and the update costs of RTS (if updates are made that year).

7.2 Sub-function

The function uses **Read** to search WS or HW in the name of update to know if it applies.

7.3 Assumptions and constraints

The function supposes that:

- The product applied is in the fourth column of scenarios table

- The class/name of a fleet is in the second column of scenarios table
- The first fleet in the scenarios table is in the third line, also that the last line of the table is not a fleet
- The columns of products are the columns 3 to 6 in costs assumption tables
- If a product contains the sub-product (indicate in PBS) the case contains “x”
- The parameter table is complete with all the products
- The first column of the roadmaps in PBS table is 29
- The last column of the roadmaps in PBS table is 58
- The scenarios in scenarios table start after the third columns and end before the 35th column
- The background color of RTS WBS activities is different than the other activities
- The columns of RTS WBS activities are the lasts before “maintain CCS” activity

7.4 Inputs

The function input is the line number of the fleet in scenarios table as an **Integer**.

7.5 Configuration data and parameters

PBS roadmap, costs assumption table, simulation table.

7.6 Outputs

The values of RTS costs (as **Single**) on the line of result table corresponding to the line of scenarios table.

7.7 Operation and implementation

To calculate the RTS costs of a fleet, the function will browse all the sub subsystems of the fleet product.

For each subsystem, **RTSCostCal** will calculate the costs between the last update (included) and the next update (excluded), while the dates of the updates are in the roadmap.

To calculate the acquisition costs of a subsystem, the function will browse the RTS WBS activities of the subsystem.

For each cost of a WBS activity of a subfunction, the function will process:

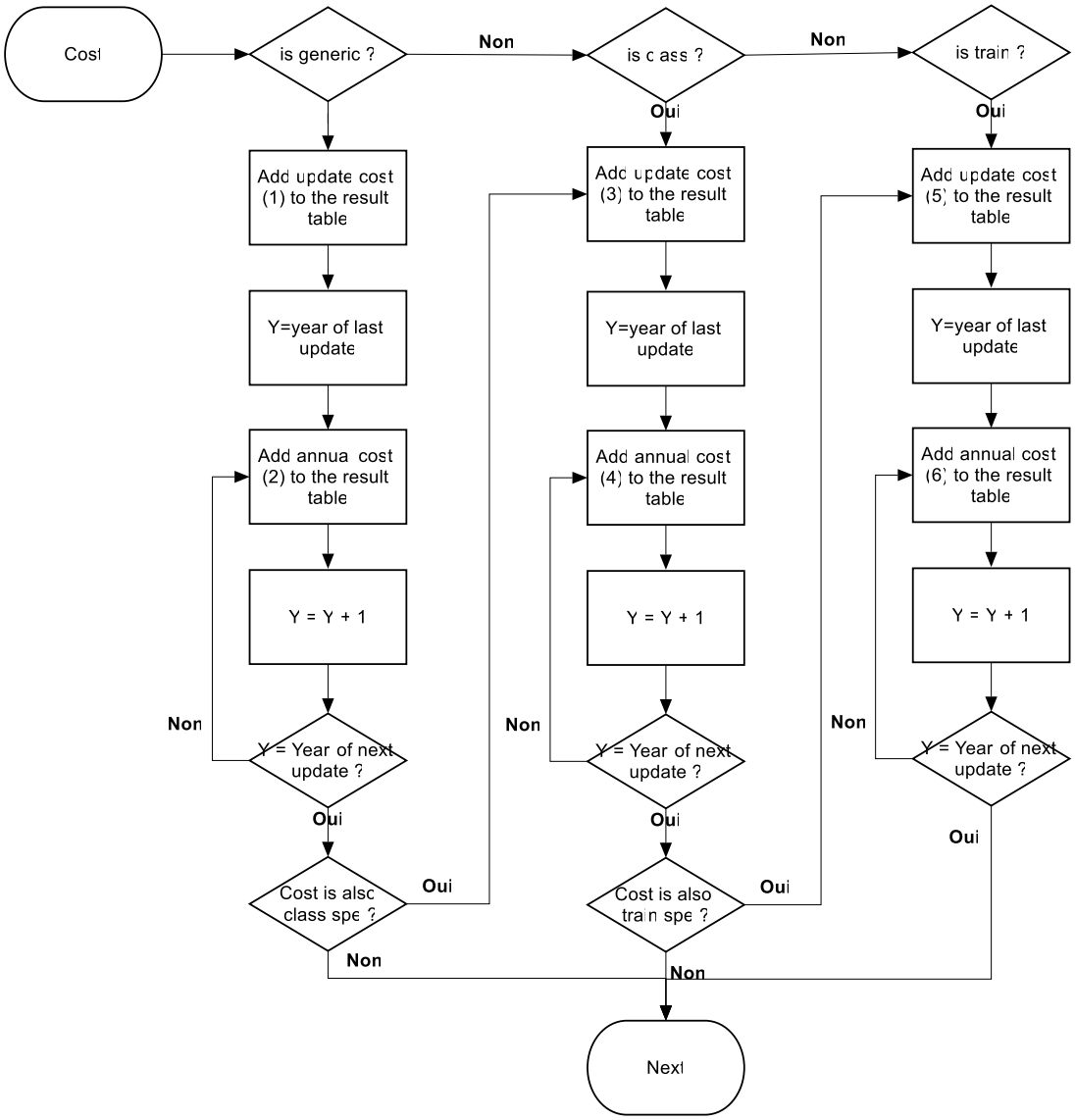

Figure 2 Programming flowchart of the cost calculation loop of RTSCostCal

Calculation of the costs show configurable values define in the Economic Model:

- *Pa* is the WBS parameter in the WBS table, it indicates the part of the cost applied generically, class specifically or train specifically;
- *PaU* is the value in parameter table, it is the coefficient to apply to the total cost for the type of update.

The calculations processing by the function are:

(1) Calculation of update cost for a generic WBS activity:

$$GeUpCost = \frac{Cost * Pa * PaU}{NbrG} * \sum_{first\ year}^{last\ update} NbrTrains$$

NbrG is the total number of trains with the subsystem considered at the date of the year before the Next Update,

NbrTrains is the number of trains of a year, the value in the scenario table.

(2) Calculation of annual cost for a generic WBS activity:

$$GeCost = \frac{Cost * Pa * NbrTrains}{NbrG}$$

(3) Calculation of update cost for a class specific WBS activity:

$$ClassUpCost = \frac{Cost * Pa * PaU}{NbrClass} * \sum_{first\ year}^{last\ update} NbrTrains$$

NbrClass is the total number of trains with the class considered at the date of the year before the next update.

(4) Calculation of annual cost for a class specific WBS activity:

$$ClassCost = \frac{Cost * Pa * NbrTrains}{NbrClass}$$

(5) Calculation of update cost for a train specific WBS activity:

$$TUpCost = Cost * Pa * PaU * \sum_{first\ year}^{last\ update} NbrTrains$$

(6) Calculation of annual cost for a train specific WBS activity:

$$TCost = Cost * Pa * NbrTrains$$

*if $NbrTrains < 0$ the calculation is skipped.

8 Maintenance costs calculation

The function **MaintCostCal** calculates for a fleet of train the maintenance costs for every year.

8.1 Definition and principles

For each year, the function will add up the maintenance costs of a fleet to the results.

8.2 Assumptions and constraints

The costs of maintain are in the last column of cost assumption table and in the fourth and the twenty-fourth line.

8.3 Inputs

The function input is the line number of the fleet in scenarios table as an **Integer.**

8.4 Configuration data and parameters

Costs assumption table, simulation table.

8.5 Outputs

The values of maintenance costs (as **Single**) on the line of result table corresponding to the line of scenarios table.

8.6 Operation and implementation

To calculate the maintain costs, the function will for each year calculate:

$$MaintCost = \sum Maintain\ CCS\ Costs * \sum_{first\ year}^{current\ year} NbrTrains$$

9 Sum

The function **Sum** calculates the sum of years costs specific.

9.1 Definition and principles

The function browses the result table years by years and sum the acquisition, RTS and maintain costs.

9.2 Assumptions and constraints

The calendar of results table begins in column 5.

9.3 Inputs

None.

9.4 Configuration data and parameters

Scenarios table.

9.5 Outputs

Totals of acquisition costs, RTS costs and maintenance costs for each year.