EUCALC

Explore sustainable European futures

28+1 – Country version of the EU Calculator model

D8.2

April/2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730459.

Project Acronym and Name	EU Calculator: trade-offs and pathways towards sustainable and low-carbon European Societies - EUCalc
Grant Agreement Number	730459
Document Type	Other
Work Package	WP8
Document Title	28+1 – Country version of the EU Calculator model
Main authors	V. Matton, M. Cornet, J. Pestiaux
Partner in charge	CLIMACT
Contributing partners	
Release date	
Distribution	*All involved authors and co-authors agreed on the publication.*

Short Description
This deliverable is a technical appendix detailing the key choices in terms of general architecture, modelling choices and programming development.

Quality check	
Name of reviewer	**Date**
Luis Costa	**27/04/2018**
Ivana Rogulj	**30/04/2018**

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Contents

List of Tables

List of Figures

List of abbreviations

EU – European Union

MS – Member States

Glossary

Ambition	The degree of low carbon mitigation effort.
Levers	Sliders which move from a minimal abatement position (level 1) to an extremely ambitious position (level 4).
Model	The model performing the calculation to provide results for the specified lever positions.
Module	Unit of the Model. The modules are connected together to form the model. The modules are developed by different partners of the consortium and are most of the time called sectors.
Working groups	A series of working groups is coordinating the assessment of key questions: • "Lever list and incompatibilities", • "lever Ambition definition", • "Sector Feedbacks loops", • "Evaluation of economic/policy pathways", "Societal Impacts", • "Country/World Coupling", • "Spatial and Time resolution", • "User Interface", • "Modules interface", • "Uncertainty and complexity for model", and • "Alignment with potential user group needs"
CORE Model	The CORE model is the first version of the model working with the CORE modules. Those modules form the central part of the model and are the following: - 1.1 Lifestyle - 1.2 Technology - 2.1 Buildings - 2.2 Transport - 3.1 Industry - 4.3 Agriculture - 5.0 Power - 5.3 Biomaterials
Non-CORE additions to the model	The future versions of the model will include some additional features in the CORE modules.

	In addition, the future versions of the model will include the following modules: - 1.2 Climate - 3.2 CCUS - 4.2 Minerals - 4.4 Water - 4.5 Biodiversity - 5.4 Water-energy nexus - 6.1 Energy security - 6.2 Education - 6.3 Human health and safety - 6.4 Employment - 6.5 Working conditions - 7.1 baseline projection - 7.2 GTAP-EUCalc interface - Policy - Economy - EU & Rest of World
Pathway Explorer	The web interface to the model.
KNIME	An open source visual programming tool. It views the model as a calculation flow.
Compilation	The process of translating from a programme description (in KNIME) to an executable programme (in Python).
Python	The programming language in which the compiled model runs.
Node	A KNIME calculation element or group of calculation elements.
Metanode	Metanodes are nodes that contain sub-workflows, i.e. in the workflow they look like a single node, although they can contain many nodes and even more meta nodes.
CalcNode	A KNIME node which the project has written in Python.

1 Executive Summary

This deliverable 8.2 is a technical description detailing the key choices in terms of general architecture, modelling choices and programming development. By contrast, Deliverable 8.1 contains practical content with examples, prototype description, images and figures without going through thought process.

Through the general architecture section, we describe the characteristics and structure of this simulation model at EU level and MS level which enables instant pathway simulations on energy, emissions, materials, resources and socio-economic indicators. The content in this section is wider than the WP8 scope. We place it here to support the overall consistency of the project.

In the modelling choices section, we describe the key technical choices; we upgrade from an Excel structure[1] to visual programming through KNIME, which provides both a collaborative way of working in teams and a visual overview of the calculation flows. This enables the model to be much more transparent to non-modelling energy stakeholders. We keep a separation between the modelling language and the program executable. To ensure a high calculation speed, we make the program executable in Python. To improve the Python code quality, both for calculation performance and to align the code across work packages, the most used KNIME building blocks (which we call CalcNodes) will be directly programmed in Python. To make the KNIME modelling even more accessible, we are evaluating the possibility to develop a tool to view the KNIME flows through a simple web browser without installing any software. The KNIME-to-Python converter is currently being developed. Its overall architecture has been reviewed and we are now in the process of validating it for the KNIME nodes in use.

In the programming development section, we describe the various processes undertaken to make the model development progress. Progress dashboards are extensively being used to monitor progress and address bottlenecks.

A combination of one-on-one sessions with partners responsible for individual modules and best practice documents have been used to ensure a proficient mastery of KNIME by the teams. A first quality review of the KNIME flows calculation and of its documentation has been performed. Further iterations are still required and will be conducted throughout the project.

Robustness processes, such as quality checks between modules, are being incorporated to ensure mutual understanding of data flows between modules and towards the Pathways Explorer (both in terms of which variable types are transferred, and their expected value range).

A development environment has been installed with determined processes to perform module specific models, to store files, to specify layers, to consolidate the general model and compile it to Python.

[1] A technology commonly used in previous national and global calculators.

2 Introduction

This deliverable serves as a supporting backup to the deliverable D8.1 which illustrates the initial model prototype, its progress and its content. The deliverable is structured as technical appendix detailing the key choices in terms of general architecture, modelling choices and programming development.

In the general architecture section, it describes the model philosophy (its characteristics, the classification to which model family it belongs, and the main building blocks of its structure). The content of this "general architecture" section goes further than the WP8 scope. Preliminary discussion relatives to these architecture choices are compiled here to support the overall consistency of the project.

In the modelling choices section, we describe the technical choices undertaken in terms of programming and of visual interaction with stakeholders. Complexity management is also assessed here. In the programming development section, we discern on the various processes undertaken to make the model development progress.

An overview of the model development progress is illustrated in Figure 1.

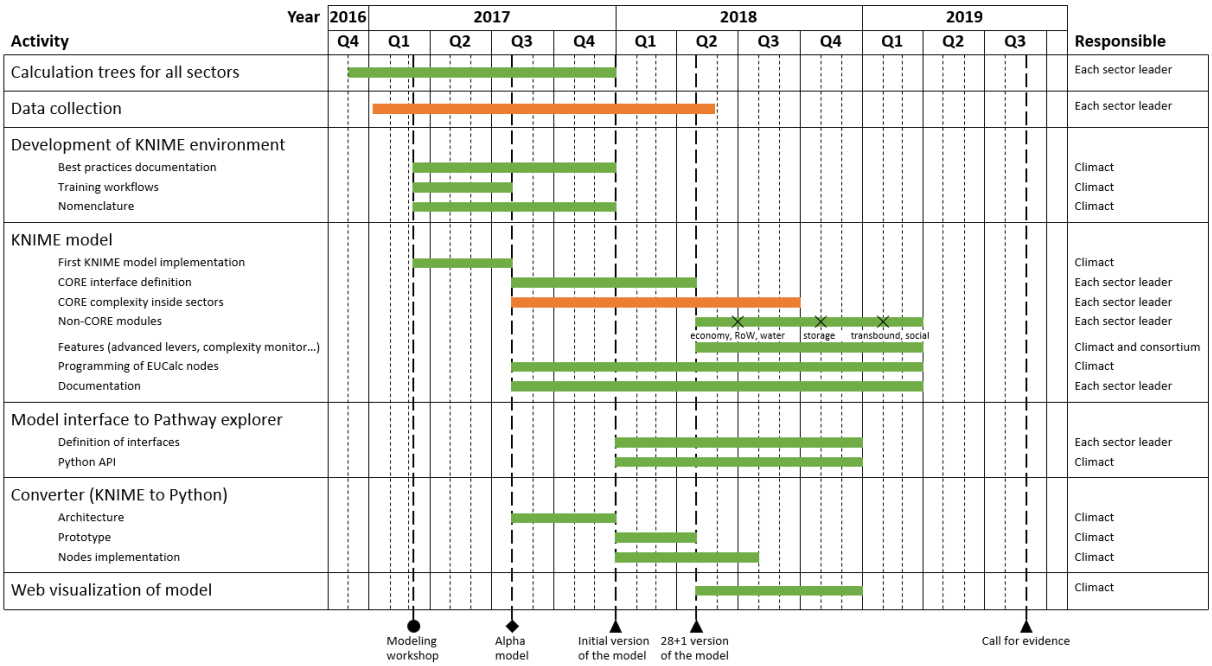


Figure 1. Agenda of the Model development

3 General architecture

We compile here the preliminary discussion relatives to the model characteristics, classification and internal structure. We place it here to support the overall consistency of the project.

3.1 Model characteristics

The proposal defines 6 model characteristics summarized in Table 1. Our goal is to create a model which:

Answers questions in real time	The 'time' is the delay experienced by the user between user input and the moment the results from the model calculations are visible. The Pathway Explorer has to propose a real-time experience to the user while the model may run in hours/days to pre-compile data.
Covers a wide array of topic	See Figure 2 with WP architecture.
Is granular enough to provide specific answers	The model enables to set the sliders for Europe as a block and for each of the 28 countries separately. The model also assesses interactions between MS countries and with the rest of the world.
Goes deep enough to provide added value	The model enables to answer key questions regarding the energy transition. The model provides enough analysis depth to be considered credible by stakeholders. In addition, the model aims to provide as much analysis depth as possible within the given constrains.
Is collaboratively build	The model is built in parallel by the 10 organisations of the consortium. These 10 organisations should not be required to have high programming skills.
Is transparent enough to enhance stakeholder buy in	The model is developed through a participative process involving stakeholders in each of the domains addressed. Stakeholders consulted have a good technical understanding of the main modelling choices performed in their area of expertise. The model is as transparent as possible to enable stakeholder buy in. Transparency can be observed in: • The input data used in the model; • The rationale of the calculation applied on the input data; • The ease of use by the end user of the tools used; • (The open-source access of the tools)

Table 1. Summary of model characteristics

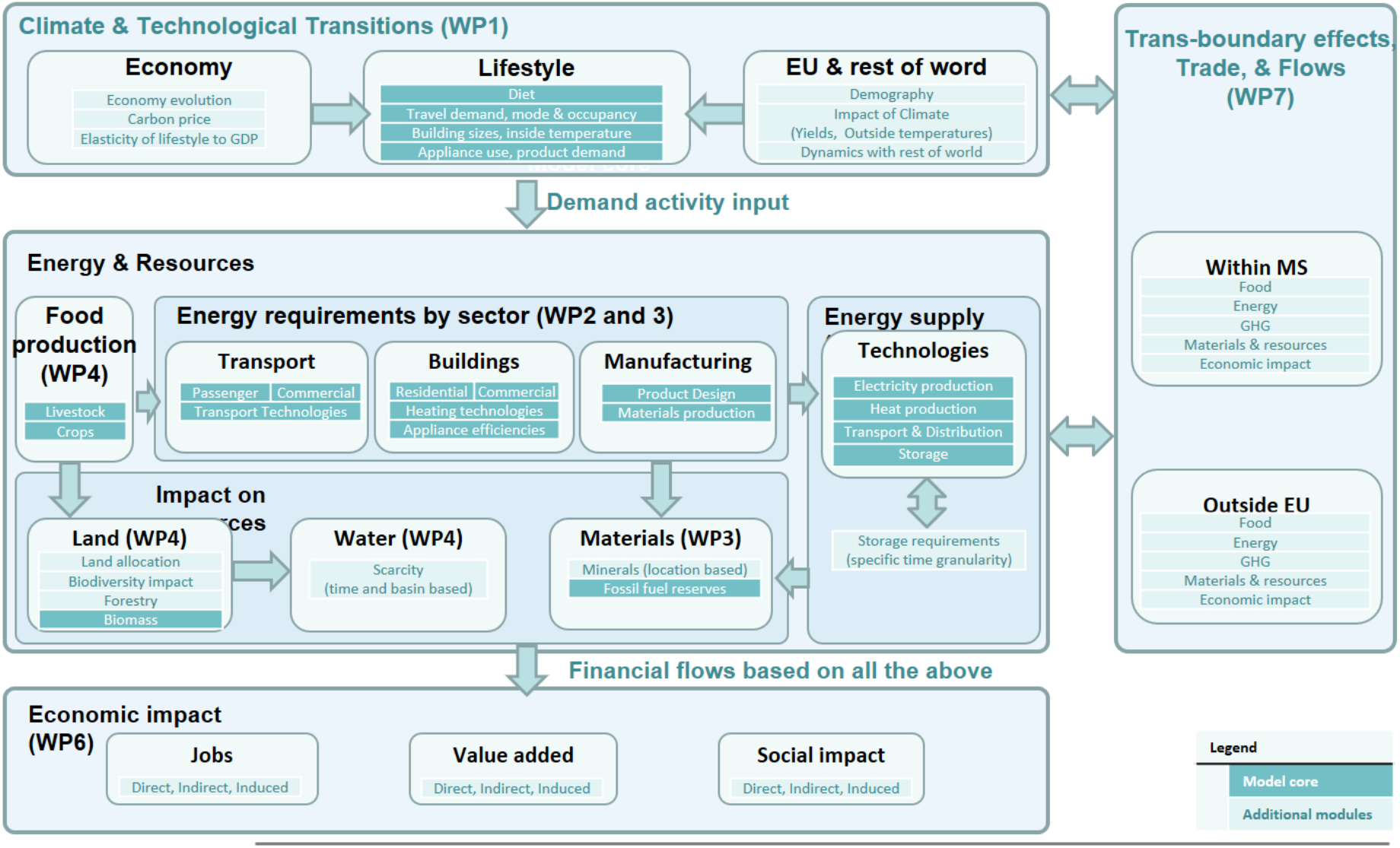


Figure 2. Work Package architecture

3.2Model classification

3.2.1 A simulation model driven by lever positions

The model has been described as a simulation model calibrated with effort ambitions. The EUCalc is a simulation model, it is driven by 'lifestyle', 'technology', and 'physical' changes and it allows a high flexibility in the simulation of pathways.

To initiate a simulation, users change the inputs to the calculator by making choices using a number of "levers" (usually between 10 and 40). These levers typically make a change in either the supply or demand of energy in a particular sector, for example building nuclear power stations, or reducing the distance people travel by car. Let's take as example in the table below the proportion of people travelling by car. The levers are transparently described, are exclusive, and the levers can be freely moved by the user within the defined range.

Ambition 1	Minimal abatement Car are the dominant mode of transport in urban areas (65% in 2050)
Ambition 2	Ambitious *Cars account for 53% of km travelled in urban areas*
Ambition 3	Very ambitious *Cars account for 43% of km travelled in urban areas*
Ambition 4	Extremely ambitious *Cars account for 29% of km travelled in urban areas*

Table 2. Ambition definition example

The model is 'activity demand' driven in the sense that it starts by assessing energy demand based on the lifestyle and activities performed in the different sectors, for example the diet, the passenger distance, or the room heating temperature. An economic module specifies the link between GDP and the activity demand drivers (the demand elasticity to price). Then, from the demand drivers, the model defines how the food, materials and energy are produced, which technologies are used and what is their efficiencies. Finally, an economic model assesses the economic impacts of these changes in terms of jobs and value added.

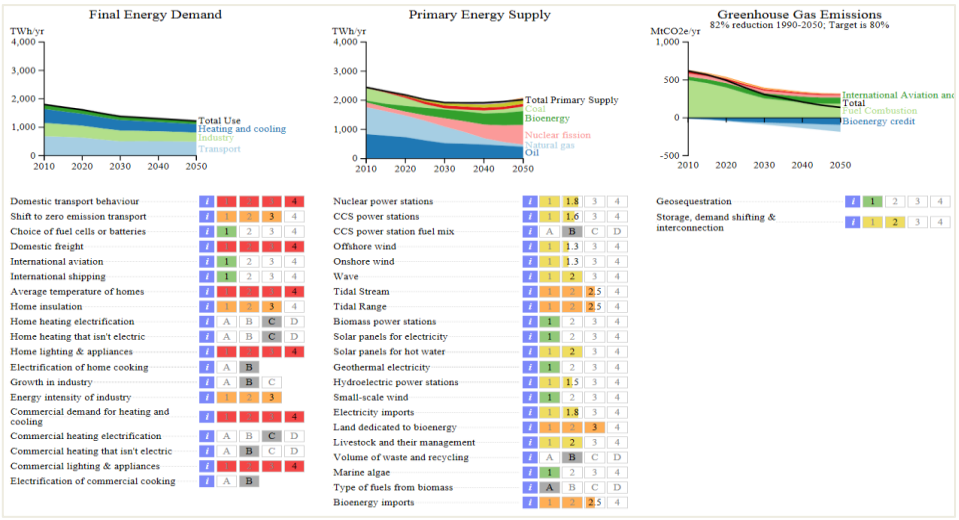


Figure 3. Pathway explorer illustration- Link of Energy demand and supply to the levers

A combination of all the lever choices creates a scenario. The model outputs for a given input scenario are named "pathways" because the focus is on the final impact and overall evolution trend[2]. For each pathway, the calculator displays the implications over time (for example in terms of energy, emissions, resource use, job creation and land-use). Figure 3 shows an example of 'Pathway explorer'.

3.2.2 Lever choices

Refining how the levers are used in the model is an ongoing work. There are 3 sub-tasks: a) defining the levers (under the coordination of working group B), b) associating an ambition to each lever position (under the coordination of working group C), c) refining along time how the ambition is reached (under the coordination of working group C).

3.2.2.1 Defining the list of levers

The initial list of levers is listed in in the appendix 7.1 in Table 9. This task is under the coordination of working group B.

The levers should:

- Not be too numerous to enable the Pathway Explorer to be easy to use and accessible
- Be adapted to the various user audiences of the EUCalc
- Be granular enough to enable to simulate scenarios corresponding to key questions for the stakeholders

3.2.2.2 Associating an ambition to each lever position

This task is under the coordination of working group C.

Many energy models are based on economics, looking at what effect changing prices, demand or supply could have on the market, and in turn what the optimal energy system would be under these circumstances. These models use complex equations and assumptions about the behaviour of individuals and firms. They are sometimes called "black box" models because

[2] We would name the scenarios "trajectories" if the milestones were detailed and the trajectories were operationalized in more depth.

inputs go in and outputs come out, but it is not clear what is happening in the middle because they are so complicated. This means that typically only a few experts can properly use them, and consequently trust in them can be low.

The EUCalc Model adds an engineering feasibility dimension to the economic forces. Users choose from a range of options for the future without really needing to consider the set of circumstances that would cause them to come about. It avoids any assumptions about what motivates behaviour and instead allows the user to see what the impact of changing behaviour directly would be. **It is about what is possible, not what is probable**. It can also be described as enabling the user to model "desirable futures".

Because we provide a simulation model, we outsource a major part of the model complexity to the user. The users take into account a high number of variables when selecting a position for each lever. Each choice the user performs on a lever position can in other models be the results of an optimisation. It is therefore important to ensure model users are provided with guidance on the meaning of each lever position and on potentially incompatible lever positions. In our experience, non-experts (e.g. ministers, students, CEOs…) can start using the tool efficiently after 30 minutes of guidance.

For this reason, the levers are calibrated along ambition levels which reflect either the ambition of the technology or the behavioural change. They range from a minimum to a maximum ambition. The fact that common definitions are applied to all simulation levers constitutes both a strength (since it helps the user define consistent positions for levers) and a weakness (since it limits the lower and upper bounds of each lever) of the tool.

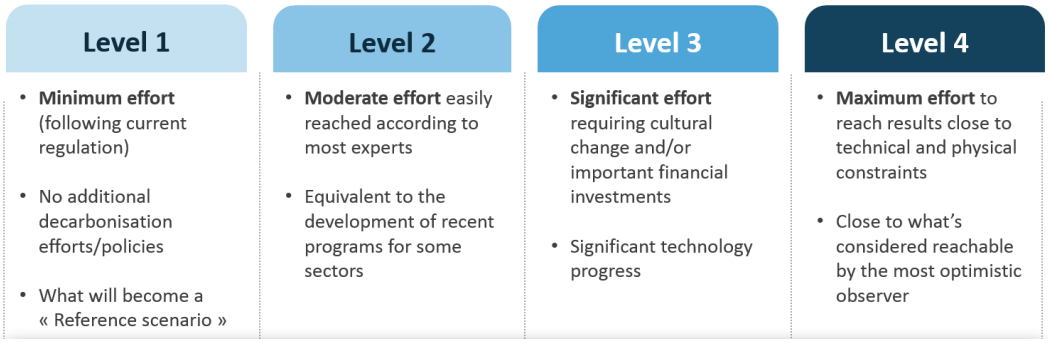


Figure 4. Illustration of ambition levels

In each sector modelled, these levers enable the end users to decompose in real time the specific impact of various parameters, greatly reducing the "black box" perception.

We apply a common definition of the ambition levels across all different sectors and countries. This will depend on the questions to be answered by the model. Here are several examples of definitions used in the past:

	Ambition levels				
Source	**1**	**2**	**3**	**4**	**5**
Global calculator	*Minimal abatement*	*Ambitious*	*Very ambitious*	*Extremely ambitious*	-
Industry federations calculators	*No effort*	*Important effort considered reachable by most stakeholders*	*Maximal effort with existing technologies, and without changing the paradigm*	*Disruptive change (or transformational) (technology, behaviour, or economic model)* *Significant breakthroughs can be assumed*	-
Japan calculator	*No effort (existing capacity, same technology, no change in consumption)*	*intermediary*	*intermediary*	*Great efforts* *(increased renewable energy, advanced technology, reduced unit energy service demand)*	*Physical limit/ Technical potential (renewables)*
Economists	*Carbon price of w*	*Carbon price of x*	*Carbon price of y*	*Carbon price of z*	-
EU Calculator	*This level is considered as a BAU scenario. The projections are aligned and coherent with either the historical trends or with the EU Reference scenario 2016 (when the results are available).*	*This level is an intermediate scenario, more ambitious than BAU but not reaching the full potential of available solutions.*	*This level is considered as very ambitious but realistic scenario, given the current technology evolutions and the best practices observed in some geographical areas.*	*This level is considered as transformational and requires some additional breakthrough or efforts such as important costs reduction for some technologies, very fast and extended deployment of infrastructures, major technological advances, strong societal changes, etc.*	-
European Climate Change Climate Transparency initiative	*Aligned to EU Calc*	*Aligned to EU Calc*	*Aligned to EU Calc*	*Aligned to EU Calc*	-

Table 3. Levers definition across various projects

3.2.2.3 Incorporating time in the ambition definition

This task is under the coordination of working group C. During the first phase of the model build up and consultations, lever ambitions will illustrate trajectories. By trajectories, we mean general evolutions, with less attention to the value at each point in time. Trajectories tend to look like Figure 5.

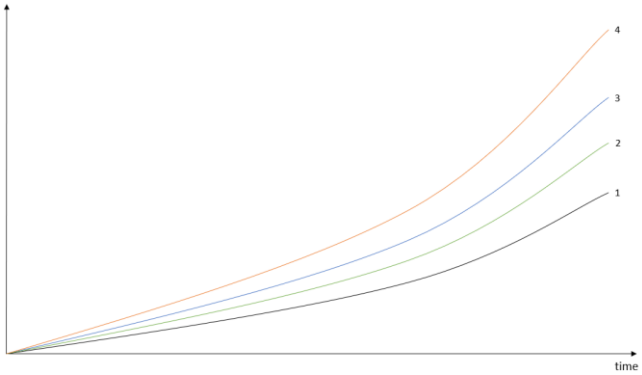


Figure 5. Original trajectories

One first improvement to these trajectories is to better reflect the notion of time in the ambition levels. We could move towards trajectories which look then more like Figure 6.

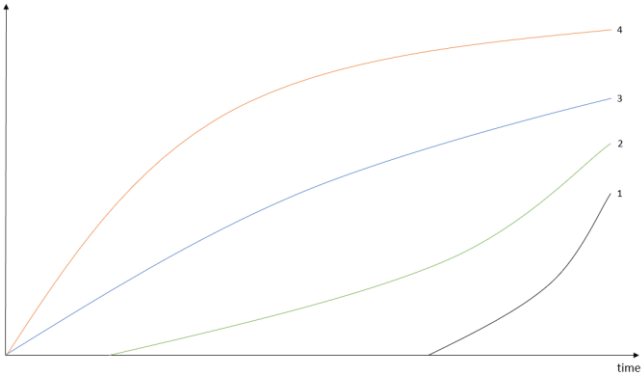


Figure 6. Incorporation of more "time notions" in the ambition level trajectories

Common stakeholder feedback is to assess "the impact of postponing the implementation" or "the impact of ramping in the lever effect over a longer period". For this reason, we are considering incorporating these assessments in the EU Calc by defining each lever through 4 parameters:

1. Start date	• When the level ambition begins to be applied
2. Implementation duration	• The time period between the start date and reaching the end goal
3. End goal	• The ambition after the implementation duration
4. Curve type	• The shape of the evolution during the implementation duration. It can for example be 1) Linear, 2) An S-curve or, 3) a hockey stick curve

Table 4. Lever parameters

These parameters are illustrated in Figure 7

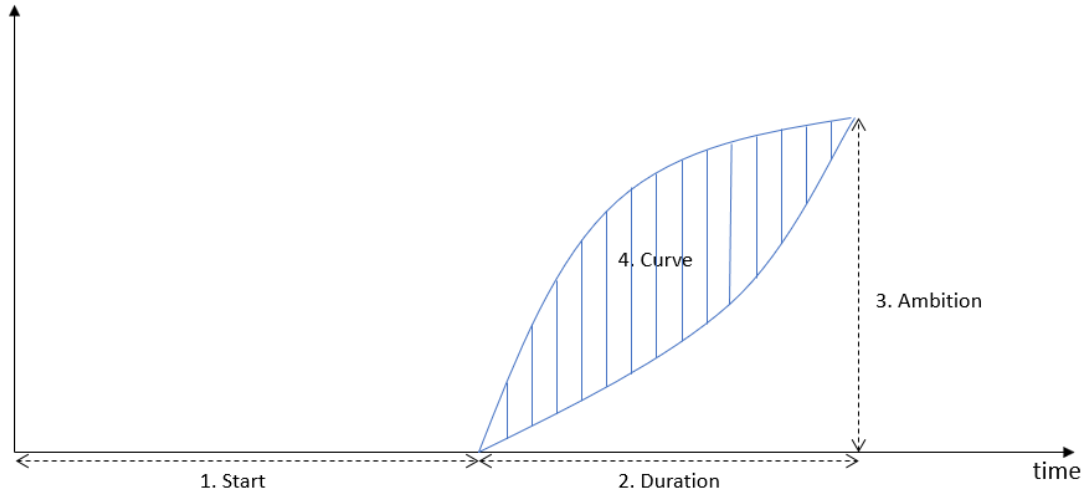


Figure 7. Refined ambition levels

These smarter ambition levels have already been implemented by Climact in the context of the ECF Climate Transparency initiative. The EUCalc model will switch to them in case we have enough insight on the level ambitions through the consultations.

3.3Internal structure

3.3.1 A Model based on module-level perspective

The unit of the EU Calculator (conceptually) is the 'module'. Each module has its own mini-model and modifies variables. Variables are associated to ambition levels which determine the 'scale' of each modification through time. The calculation flow at the module-level is illustrated in Figure 8 through a calculation tree.

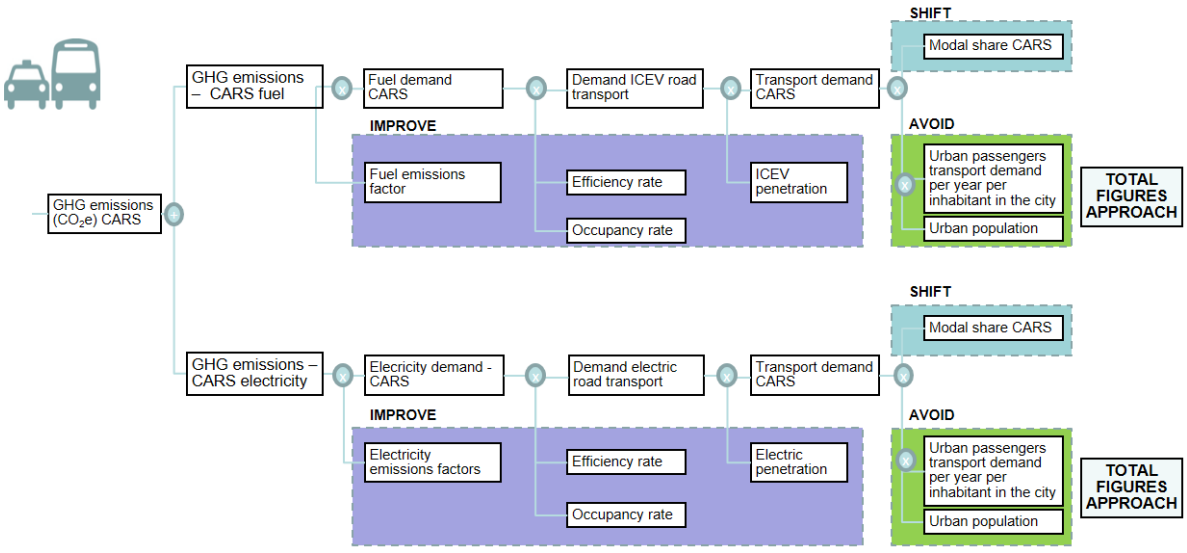


Figure 8. Example of calculation tree

All the modules are then linked together to form the EUCalc in a similar model as shown in the Figure 2. The modules are the building blocks of the model. Here is the current list:

#	Module name	Lead partner
1.1	Lifestyle	PIK
1.2	Climate	UEA
1.3	Technology	OGUT
2.1	Building	BPIE
2.2	Transport	Climact
3.1	Industry	OGUT
3.2	CCUS	EPFL
4.1	Land use	Imperial
4.2	Minerals	Imperial
4.3	Agriculture	Imperial
4.4	Water	UEA
4.5	Biodiversity	UEA
5.1	Electricity	PANNON
5.2	Fossil Fuel	PANNON
5.3	Biomass	Imperial
5.4	Water-energy nexus	EPFL
6.1	Energy security	TU Delft
6.2	Education	TU Delft
6.3	Human health and safety	TU Delft
6.4	Employment	EPFL
6.5	Working conditions	TU Delft

Table 5. List of modules

It is important to notice that the development of each module is led by one of the partners. This partner has final responsibility to validate the model (assumptions, hypothesis, formulas, calculation trees, robustness…). Each module was firstly defined using the following form. Figure 9 illustrates the layout of the form.

WP	#	Module	#	Module name	xx
Leader	partner name	contact	contact name		
Description	Description of the module				
Levers	list of levers for the module				
Inputs	list of inputs from other modules				
Outputs	list of outputs to other modules				
Status	status of the module (ongoing – not started yet – collecting datas…)				
KNIME model	status of the knime model				

Figure 9. Modules definition layout

This first approach was then converted into several working documents that are described in the next sections (e.g. interfaces definitions, KNIME workflows)

3.3.2 Interfaces between modules

The interaction between the sectors is crucial to build a coherent global model. We have therefore built the interfaces iteratively and work with 3 depths of interfaces with different purposes.

3.3.2.1 Depth 1 interfaces: Tree visualization

The general purpose of this first depth is to create a general model overview, to align the discussion topics and to give the opportunity for everyone to understand the general calculation flows of the model. By looking at the links between each module we created a matrix of interaction, the adjacency matrix, as illustrated on Figure 10.

WP# / Module / Node	unit	validation check	Lifestyle	Climate	Technology	Building	Transport	Industry	CCUS	Land use	Minerals	Agriculture	Water	Biodiversity	Electricity	Oil Refining	Biomass	Water-energy nexus	Energy security	Education	Human health and safety	Employment	Working conditions	baseline projection	GTAP-EUCalc interface	OUTPUT
			1.1	1.2	1.3	2.1	2.2	3.1	3.2	4.1	4.2	4.3	4.4	4.5	5.1	5.2	5.3	5.4	6.1	6.2	6.3	6.4	6.5	7.1	7.2	
WP1 1.1 Lifestyle						x	x	x		x		x	x								x	x				
Demography						x	x	x		x		x	x									x				
Population aging							x																			
Urban population						x				x																
GDP						x	x	x				x										x				
Food demand												x														
Meat demand												x														
Calories consumed																					x					
(Transport demand)							x																			
(Travel time budget)							x																			
(Household size)						x																				
1.2 Climate						x																				
Wind speed						x																				
Air temp						x																				
Land temp						x																				
1.3 Technology						x	x	x							x	x	x	x								

Figure 10. Adjacency matrix

This supports the discussion between the modules, ensures added coherence between modules and given the first indication of strong dependencies between modules that each module leader has to account for. From this information, we can build a graphical representation of the matrix, where each module becomes a box (node) in a graph and where flows become represented by arrows. This brings us to the following graphical representation of the model, from a general view on the positioning of each module in the overall architecture (Figure 11) to a detailed representation of the interactions of one specific module with the modules it connects to (Figure 13).

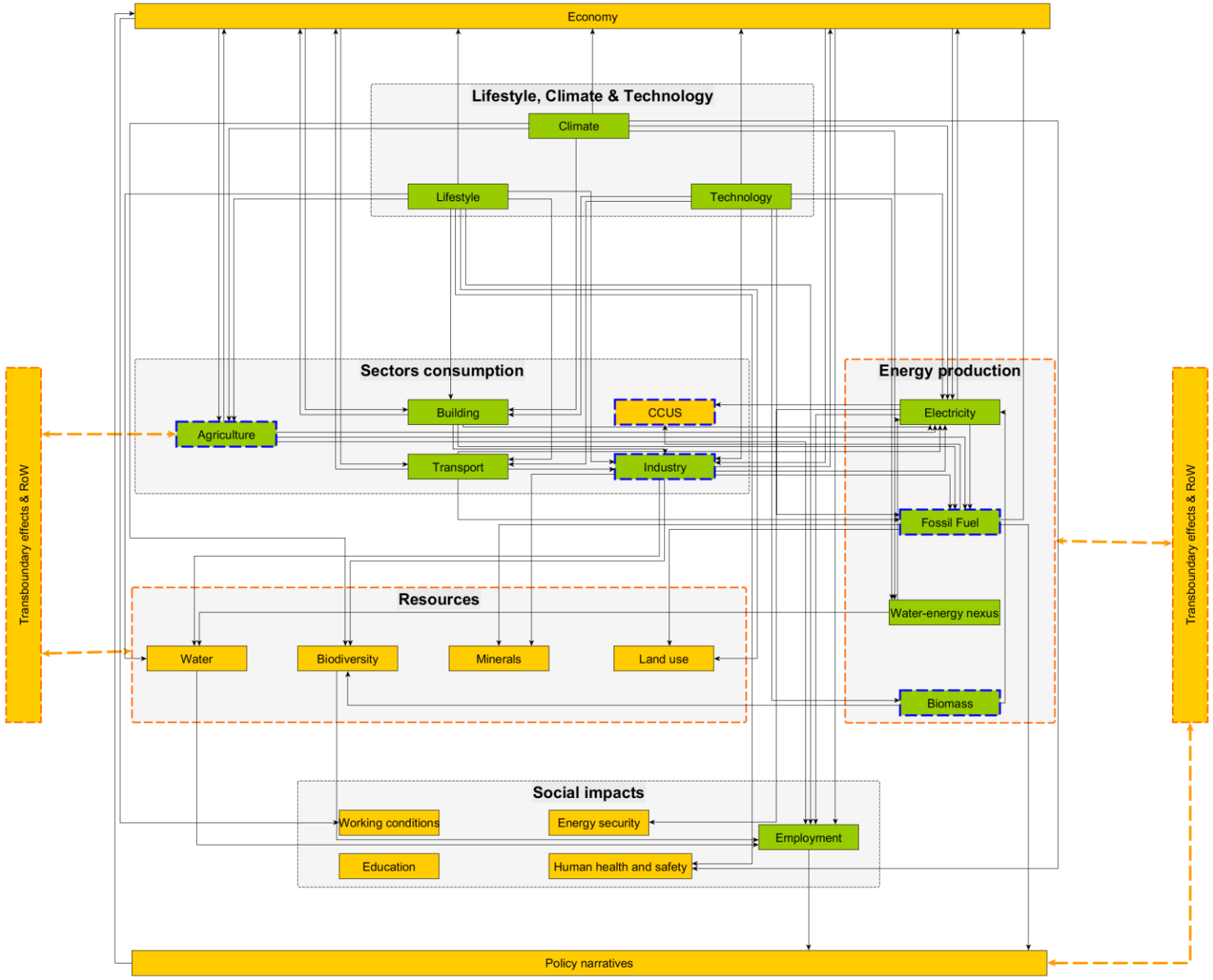


Figure 11. Overall model with module granularity. The green nodes are the CORE modules.

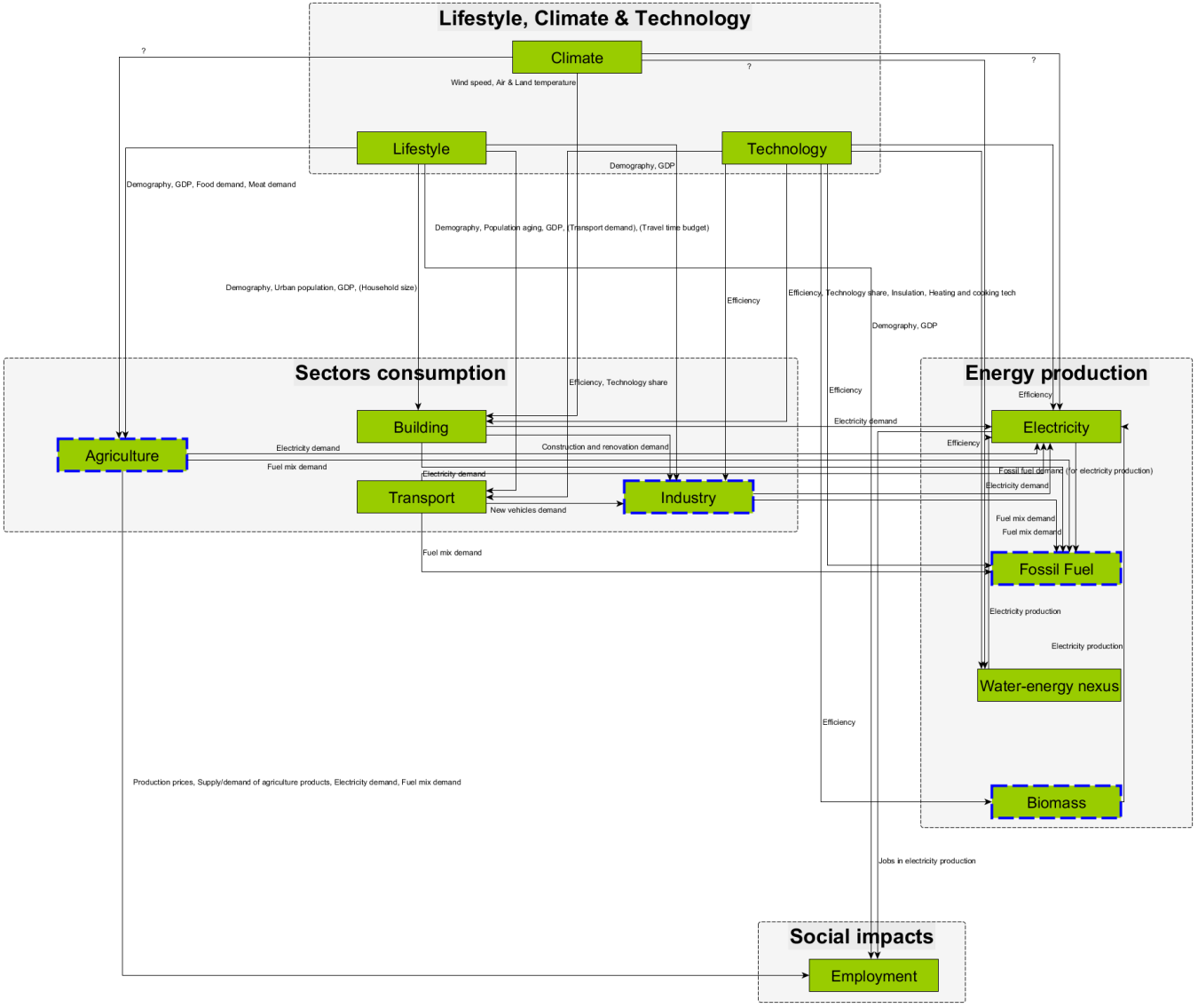


Figure 12. CORE model

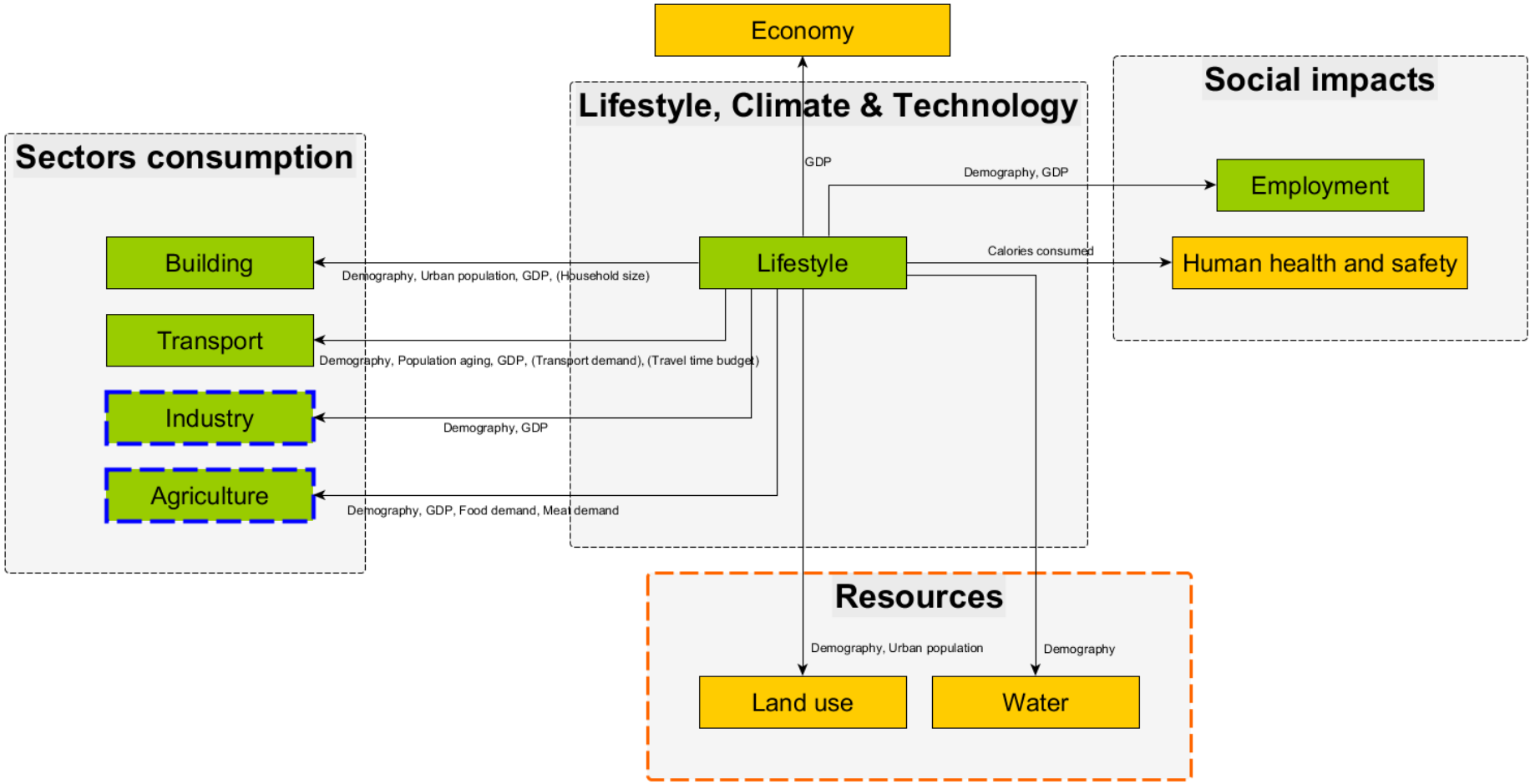


Figure 13. Example of interaction between Lifestyle and each of the other modules

3.3.2.2 Depth 2 interfaces: Google sheet interaction document

The purpose of this 2nd depth level is to provide a readable yet detailed view of each interaction each module has. Also, to ensure consistency between modules providing and those receiving variables.

The depth 2 is based on a series of collaborative sheets (see transport example here). This allows the partners to clearly define what are the exchange variables between the modules. See Figure 14 for an example of interaction between Buildings and Industry.

	Legend			compare to buildings interface definition			
		Core					
		Non-Core					
Naming convention	**Alignment with Building**	**Inputs (to be received from Building module)**	**Type**	**Description**	**Unit**	**Country**	**Year**
fts_bld_res-build[m2]	Yes	Building (gross floor) area	Residential	gross floor new demand	m2	Member state level	yearly
fts_bld_non-res-build[m2]	Yes		Non-residential	gross floor new demand	m2	Member state level	yearly
fts_bld_reno-res-build[m2]	Yes	Insulation area (renovation)	Residential	surface area new demand	m2	Member state level	yearly
fts_bld_reno-non-res-build[m2]	Yes		Non-residential	surface area new demand	m2	Member state level	yearly
fts_bld_fridges[num]	Yes	Appliances	fridge/ freezer	new demand	km	Member state level	yearly
fts_bld_wash-machines[num]	Yes		washing mashine	new demand	# new	Member state level	yearly
fts_bld_dishwashers[num]	Yes		dishwasher	new demand	# new	Member state level	yearly
fts_bld_pipes[km]	Yes	pipes for district heating		new demand	km new	Member state level	yearly
Naming convention	**Alignment with Building**	**Outputs from Industry module**		**Vector**	**Unit**	**Country**	**Year**
	Yes	waste heat			TWh		

Figure 14. Depth 2 interface: Google sheet interaction document

3.3.2.3 Depth 3 interfaces: KNIME interaction

The last depth of interface definition is in the modelling itself. It corresponds to the implementation of the level 2 (Google sheet) interface inside the visualization program.

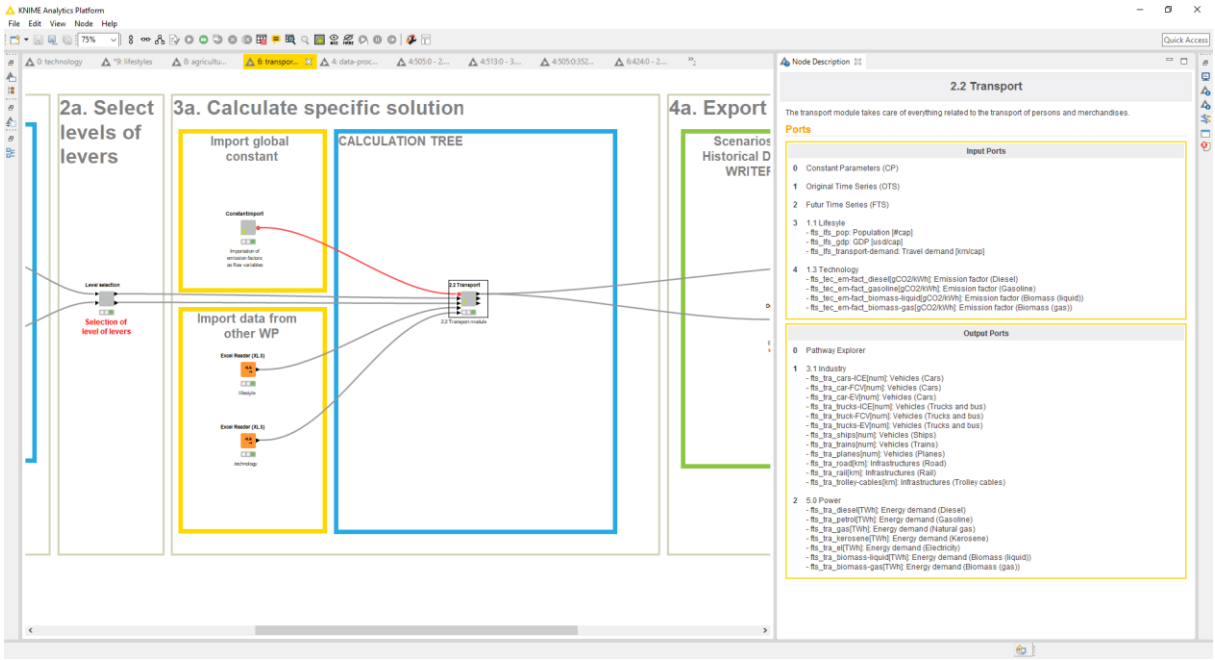


Figure 15. Example of interface (on the right) of buildings node with the other sectors.

3.3.3 Interfaces between countries

This rationale is being assessed by UCPH in the Work package 7 'Transboundary Effects, Trade & flows'. By default, we think of proceeding as follows:

The model will perform the calculations first at national level. Each country will specify quantities, imports and exports along the following dimensions: Food, Energy, GHG emissions, materials & resources, and economic impacts. The imports and exports will be segmented between within the EU or outside the EU. Adding this all up, the model will assess the quantities, imports and exports of the EU block to the rest of the world. As a sanity check, in parallel, a calculation will be performed with total EU block values.

To reduce calculation and modelling complexity, the countries will be independent one of another and the model will not specify the origin and destination country for the flows.

3.3.4 Interfaces with the Pathway Explorer

There are different possible architectural options regarding the interactions between the model and the Pathway explorer (the web interface). The team is currently assessing the benefits, issues, effort requirements of each in collaboration with WP 9.

Option	Description	Benefit	Issues	Effort	Our conclusion
1 **Web client issues direct API calls to modelling server**	- Model is deployed on a separate server and provides an API to get results - Global Calc web client code is modified to issue API call to model	- Limited effort to modify - Uses current prototype of model API	- Cross-scripting prevents this from being used in production	- Modify JS/HTML client code	This is only possible for testing purposes (e.g. Sarajevo presentation)
2 **Web client issues API calls to web server, which calls the model API**	- Model is deployed on separate server and presents an API to provide results - Web client issues API calls to web server - Web server validates and issues API calls to modelling server	- Flexible, "microservice" type of architecture, securing the model server - Clear separation of model vs. website (e.g. easier updates to model) - Easier to scale (e.g. use the same model server for multiple websites)	- Implementation of API calls in Ruby (or switch to Python?)	- Modify web server backend to process API calls to model	This is the preferred option for production due to modularity and easy identification of responsibilities
3 **Web server in Python on same server as model**	- Single server running a Python webserver (e.g. Flask) - Server handles web requests and either natively calls the model's Python code, or a compiled version	- Uses only Python for backend (no more Ruby) - Can be converted to a microservices architecture later	- Need clear responsibility of the web server configuration (e.g. security needs)	- Transfer front-end code to Python server	This is the simplest technical option but separating website / model responsibilities is harder
4 **Python model is compiled to C and web server backend is modified**	- This approach mimics as closely as possible the Global Calc approach: o Python compiled to C library o Ruby code bindings modified to use C library	- Same architecture as Global Calc	- Complexity of creating a new Ruby/C binding	- Python/C compilation - Ruby/C bindings	This should only be used if we wanted to stick exactly to the Global Calc architecture

Table 6. Possible interactions between the Model and the Pathway explorer

Those options are illustrated in the Appendix 7.2.

4 Modelling choices

This section describes the technical choices and the complexity management choices.

4.1Technical choices

The technical choices range from the choice of using visual programming, to selecting the programming language and optimizing the interaction between both.

In addition, an assessment is performed on how to best share the visual programming language assumptions to the stakeholders

4.1.1 Visual Programming

We consider three approaches to "modelling" tools. 1) Excel, 2) Visual programming, 3) Direct coding programming.

The Global calculator[3] was modelled in Excel, then converted in a programming language (C). For EUCalc, we perform the modelling in a visual programming language.

A comparison of the pros & cons of each is summarized in the table below:

Modelling tool approaches	**Pros**	**Cons**
Excel	• Most people have it on their computer • Many built in functions • Execution speed: Very slow • You see all the calculation results	• Hard to see the calculation logic • Not open source • Only sheets of 2 dimensions (rows and columns) • Many programming concepts cannot be programmed
Matlab	• High recognition in universities • Many built in function • Execution speed: Fast to very fast • Can import and export in several formats	• Not stable for production environment
Visual programming	• Calculation logic easiest to understand • Execution speed: fast • Can import and export in many formats	• Another additional technology • Some programming concepts cannot be programmed
Direct coding programming	• Most programming concepts can be programmed • Execution speed: fast to very fast	• Hardest to understand for non-programmers

[3] The Global Calculator is a model of the world's energy, land and food systems to 2050 (http://tool.globalcalculator.org/globcalc).

	• Can import and export in most formats	

Table 7. Comparison of modelling tools approaches

By using visual programming, the consortium considers EUCalc can become more

Modular	• The model easier to customize • The model is easier to update, with clear buildings blocks and interfaces between them
User friendly	• A visual GUI (Graphical User Interface) clearly illustrates the calculation flow, having an easier overview of the workflow creates transparency • Implementation is faster
Easy integration	• Nodes in the visual programming tool can easily be converted in direct programming code • The model is easier to debug
Powerful	• Visual programming is used by leading institutions worldwide • Visual programming tools facilitate the integration to external sources (Google, DB …)
Collaborative	• Visual programming tools provide an environment to work in parallel in teams using Git repositories.

As an illustration, in Excel, the data corresponding to a lever for one technology typically resembles Figure 16. On the horizontal axis, we have the time periods. On the vertical, we have the 4 ambition levels. In the table, we have the lever data for each ambition level. In the bottom line, we have the chosen ambition data.

Wave generation, installed capacity											GW
Trajectory	Description	Notes	2008	2010	2015	2020	2025	2030	2035	2040	2045	2050
1			0	0	0	0	0	0	0	0	0	0
2			0	0	0	0	0	0	0	0	0	0
3			0	0	0	0	0	0	0	0	0	0
4			0	0	0	0	0	0	0	0	0	0
Chosen			-	-	-	-	-	-	-	-	-	-

Figure 16. Data corresponding to one lever (illustrative)

In visual programming, this table would be represented by a node. The approach enables to visualize a combination of these levers in flow view similar to a diagram. They are less error prone than Excel, and usually very user friendly. They also enable a better and easier conversion to other modelling languages such as Python than Excel.

4.1.2 KNIME

We chose KNIME as the visual programming tool. KNIME is an opensource (free) tool widely used and recognized[4] in the scientific community.

[4] KNIME leads the Gartner Magic Quadrant for Data Science: https://www.kdnuggets.com/2018/02/gartner-2018-mq-data-science-machine-learning-changes.html

A visual representation in KNIME of a calculation flow typically looks like Figure 17. The window to the right illustrates a calculation workflow, where each element is a node.

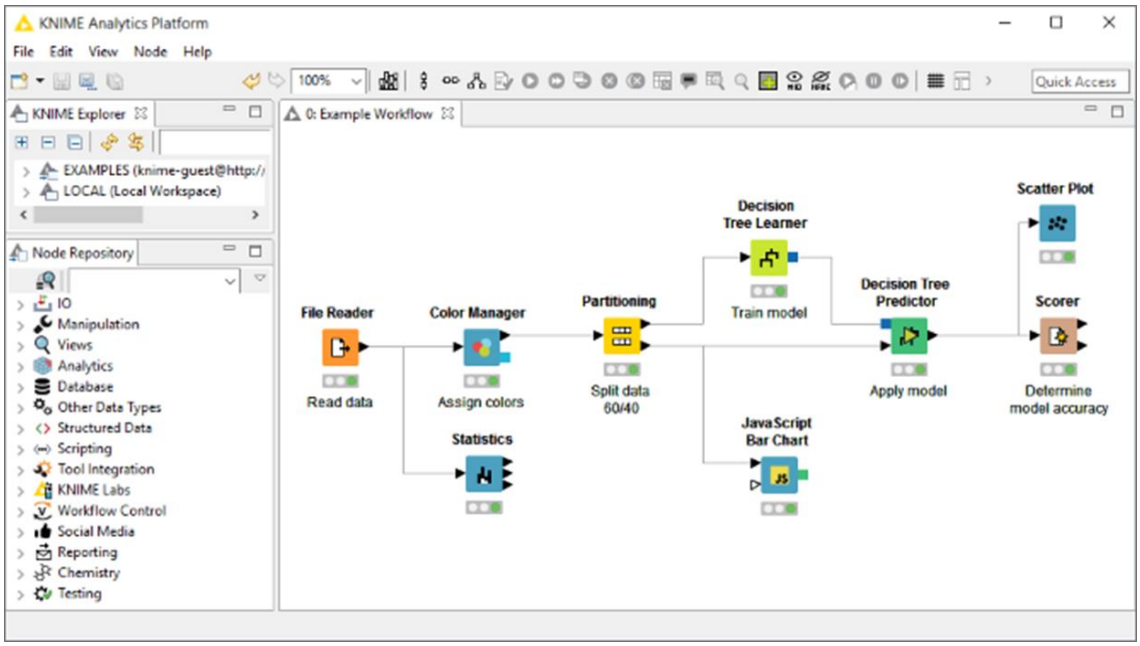


Figure 17. KNIME screenshot

4.1.3 Online version for web application

We need to connect the Pathway explorer to a solution which is modular and runs fast. We have 2 options: to use the KNIME server solution proposed by KNIME, or to convert the KNIME to a direct programming code and run the code on a server. We opted for choosing the KNIME converter because it can be more modular, it runs faster, and is free of costs.

We performed some tests with an overall model (merging the KNIME flows of the different work packages) and our conclusion is that with KNIME server, the resulting file is unlikely to run fast enough to satisfy the online requirements. Therefore, we convert the overall KNIME model to a direct programming code, which will allow speeding up the running time.

As the KNIME model is evolving constantly, it is key to push the KNIME to direct programming code converter far enough to have a robust converter that allow to test the different versions of the model before the deployment.

4.1.4 Python

The choice of Python among the different programming code languages was made for his user-friendliness. It is a well-known language in the scientific community with a lot of available packages. Python is developed under an OSI -approved open source license, making it freely usable and distributable, even for commercial use.

We are using Python for two aspects in the model:

1. For complex calculations in KNIME as we can use Python nodes directly inside the software;
2. For the interface with the web-application: API definition.

4.1.5 EUCalc nodes

In addition to the use of the standard nodes in KNIME, Climact also defines dedicated calculation nodes especially made for the EUCalc. Those nodes allow the modelers to use code directly tailored to EUCalc functionalities. This brings two benefits. First, these EUCalc nodes will correspond to a large amount of KNIME node, simplifying the modelling. Second, as these nodes are already performed in Python, it ensures the Python conversion is optimised for these nodes. This will however require a significant modelling investment from Work Package 8.

4.1.6 Visualization of KNIME workflow online

To ensure the transparency of the model, we are assessing different technologies to view the workflows directly through a web browser, without installing any software:

- The KNIME server provides an online access to the workflows, however this solution is not free;
- yEd[5] provides a web interface. Either the KNIME, either the Python code could be converted to yEd;
- other.

[5] Level 1 interfaces in this document are generated through yEd

4.2 Complexity management of the model operations

Let's look again at the joint model characteristics (Table 1) together with their technical implications in the following table (Table 8).

Answers questions in real time	The time constrain does not place a condition on the technology used to implement the model. To keep the model calculation time under control, our perspective is that only the required complexity should be integrated in the model. Visual programming techniques facilitate the reach of this goal because they provide a clear view on what the model does, where the inter-sector connections are and where there are calculation loops. This enables better trade-off discussions when handling complexity management. Calculation time requirements can sometimes be replaced by space requirements (to store pre-calculated results), these two notions are described in the next section. The space constrain, of a reasonably limited amount of space storage, does not place a condition on the technology used to implement the model. We will opt for the use of a database to store the model inputs, intermediary results and outputs.
Covers a wide array of topic	These three constrains together force us to add more complexity and to switch from an excel model to a programming language such as Python. Combining KNIME and Python ensures the model remains easy to understand, modular, and therefore allows the addition of further complexity. Using Python enables to leverage a significant number of libraries of work.
Is granular enough to provide specific answers	
Goes deep enough to provide added value	
Is collaboratively build	With Git, it is possible to work in parallel on separate KNIME workflows and Python codes. Git facilitates the handling of multiple versions and the consolidation into one overall KNIME model. Each work package is leading the creation of the KNIME workflows for the different sectors keeping in mind that we are building a global model. WP 8 is leading the Python code is two ways. Through the creation of KNIME nodes and the compilation of the KNIME model in Python. The other work packages can but are not expected to build the Python code but are encouraged to understand its overall structure.
Is transparent enough to enhance stakeholder buy in	With KNIME, the model assumptions are easier to access and understand. Because KNIME provides an opensource editor, making the model accessible to other stakeholders. We are working on a way to make the model assumptions accessible through a web browser and without requiring the installation of any software. Python, being a programming language, is less understandable for the general public. On the other hand, the Python code is public and there are open source Python editors.

Table 8. Model characteristics with complexity analysis

4.2.1 Time and space constrains

4.2.1.1 Time

As already mentioned, there is a difference between the time complexity of the algorithms and the response-time of the web-tool. While the complexity of the algorithms could take hours/days to compile for pre-calculations, it is a must, in the Pathway Explorer, to be able to give an answer to the user in less than 2 seconds.

4.2.1.2 Space

We propose to limit the space storage requirement to 10 tera bytes. The space requirements for input data are expected to be well below this boundary. However, this places a significant cap on how much we can leverage pre-calculation (see below).

4.2.1.3 Pre-calculations and its impact on time and space

To reduce the calculation time, one solution is to pre-calculate results for predefined inputs. Let's assess the space required to pre-calculate the whole model in box 1.

> We would need to calculate the model for each lever position. With 50 levers and 30 possible positions per lever, we would need to calculate the results for 30^50 possibilities. Let's assume each model calculation requires 1 second. To pre-calculate the results, we would need to perform 30^50 calculations of 1 second, which corresponds to 2*10^66 years. So likely not the best alternative.
>
> For each combination, the model provides results on 100 variables in 30 countries and 10-time periods. Let's assume each variable requires 8 bits of storage. We would need 2*10^70 Gb of space.

Box 1. Pre-calculation time and space requirements for the whole model

This means we cannot pre-calculate the whole model because we have too many possible inputs.

The calculation time can be reduced through 2 techniques. Parallel calculations and by limiting the pre-calculations to subsets of less levers.

4.2.1.4 Parallel computation

Part of the calculation can be performed in parallel. This can be applied twice.

- Within a model run, several sectors can be computed in parallel;
- In the case of multiple model runs, for example in the case of an optimisation on the model levers, several runs can be computed in parallel.

The difficulty will come from segmenting the calculations that need to be performed sequentially from the ones which can be performed in parallel. Once this segmentation is performed, cloud computing services (e.g. Amazon) can distribute the calculations across a large number of processors calculating in parallel. PIK also has large parallel computing capacity which may be useful.

4.2.1.5 Limiting pre-calculations to a limited scope

We assume that most of the questions the model needs to answer can be modelled with maximum 7 variables per sector.

The variables include both the lever positions and the interconnections from the other sectors.

Let's assess the space required to pre-calculate passenger transport model in box 2.

The following lever positions are required:

- Mode choice (e.g. car, train, bike…)
- Vehicle occupancy (i.e. number of people per vehicle)
- Proportion of car ownership
- Technology efficiency
- Technology choice (e.g. electric, ICE, hybrid, fuel cell)

The following inputs from other models are required:

- # persons (from demography section)
- Distance/person (from behavioural section)

We would need to calculate the model for each lever position. With 7 inputs and 30 input positions per input, we would need to calculate the results for 30^7 possibilities. Let's assume each model calculation requires 1 second. To pre-calculate the results, we would need to perform 30^7 calculations of 1 second, which corresponds to 700 years.

For each combination, the model provides results on 40 variables in 30 countries and 10-time periods. Let's assume each variable requires 8 bits of storage. We would need 2*10^6 Gb of space.

Box 2. Pre-calculation time and space requirements for passenger transport

This is still not acceptable but already much lower. Now let's reduce the input positions to 4 instead of 30. And assume we can interpolate afterwards. Let's assess the space required to pre-calculate passenger transport model in box 3.

The same lever positions and inputs are required:

We would need to calculate the model for each lever position. With 7 inputs and 4 input positions per input, we would need to calculate the results for 4^7 possibilities (16 000). Let's assume each model calculation requires 1 second. We would need to perform 4^7 calculations of 1 second, which corresponds to 0,2 days.

For each combination, the model provides results on 40 variables in 30 countries and 10-time periods. Let's assume each variable requires 8 bits of storage. We would need 1,5 Gb of space.

Box 3. Pre-calculation time and space requirements for passenger transport with low input granularity

This works, but with a limited margin. Interpolations are required because we will need to provide end users with more than 4 positions per lever. The team is currently assessing to which extent these interpolations are feasible (see 3.2.1). The results of this analysis will determine to which extent we can leverage the pre-calculated results.

4.2.2 Feedback loops

In real life, the interactions between the sectors are tightly intertwined and lead to feedback loops. As mentioned above, these interactions between sectors are required but generate complexity. While the general rationale on complexity management is illustrated in the

complexity management section above, we illustrate here several examples enabling to reduce complexity by reducing the number of interactions between sectors.

For example, let us look at the effects of a population increase in Box 4:

1. More people eating leads to more chicken consumption
2. More chicken livestock breeding leads to more cereals consumption (the cap on cereals availability is influenced by a warming climate)
3. More cereals production leads to more fertilizers consumption (the yield of cereals production is influenced by a warming climate)
4. More fertilizers production leads to more ammonia consumption
5. More ammonia production leads to more electricity demand
6. More electricity demand leads to more windmills demand
7. More windmills demand leads to more steel demand
8. More steel demand leads to more electricity demand

In the example above, we arrive at a situation with a feedback loop from 8 to 6. These feedback loops are typically rerun many times and significantly contribute to the calculation time. For this reason, we will avoid them whenever possible.

Box 4

The example above can be illustrated as in the Figure 18.

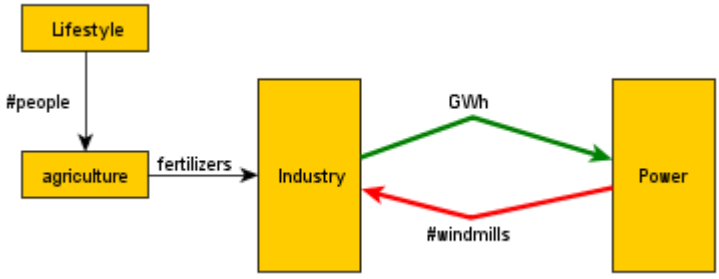


Figure 18.Identification of a loop between Industry and Power

After identifying the loop, we need to consider multiple options as illustrated in Figures below:

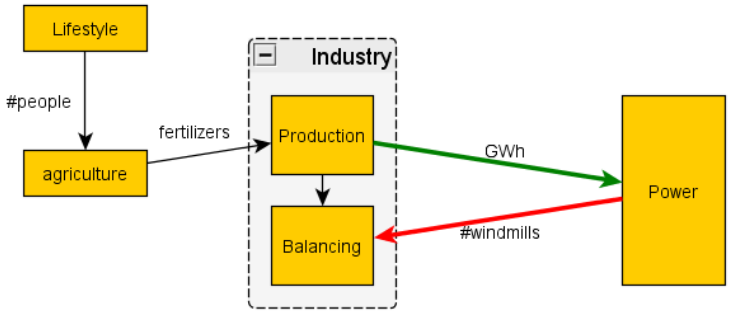


Figure 19. The power to build the windmills is negligible, the weak link is the red link.

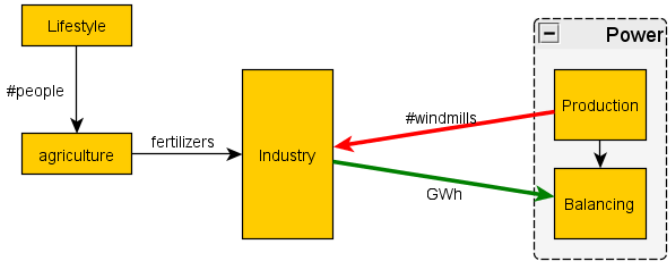


Figure 20. The GWh from industry is negligible and could be handled by a balancing strategy.

The sector leaders of the WPs are then responsible of choosing the better solution between the multiple options. Let us look at another example (and loop) with the effects of an increase steel demand in Box 5.

1. More steel demand leads to more freight demand
2. More freight demand leads to more demand for ships (and ships building materials)
3. Building & using more ships requires more electricity
4. Building more electricity requires the deployment of new power plants
5. Building power plants requires steel

In the example above, several links do not make a material difference because they do not explain a significant portion of the influenced variable. At a global level, ships represent only 2% of steel and power plants represent only a minor contribution to steel demand, which means the necessity of keeping links 3 & 5 should be challenged.

Furthermore, like in the previous example, we can also assess modelling freight demand through a lever, which breaks the link 1.

Box 5

Prior breaking loops, another concept needs to be taken into account, the loop stability. We want to ensure that a broken loop does not have a substantial effect at each calculation step. Adding a link between sectors will be considered as adding a model enriching feature. We will apply the complexity management rules to assess whether to add it.

In the initial version of the model, we left the sectors relatively independent (only connect major links like energy demand to supply, or biomass potential). This enables to keep the model simpler at first. Another advantage of keeping the sectors independent in the first stages is that potential bugs in one sector do not unbalance model inputs in the other sectors. To keep a global perspective, the model development process (described in the next chapter below) forces sectors to specify from the beginning which interactions they envisage.

5 Programming development

5.1Team progress monitoring

We use a dashboard (see Figure 21) to show the advancement of the model over the multiple sectors.

	Module	Leader	CORE	Calculation Trees/Report (I		Code/Knime					Data Input		Interface				Baseline	Lever Values
				CORE modules	Non-CORE Modules	Initial	Cleaned by Climact	Covers CORE complexity	Covers Non-Core complexity	Documentation quality	1 Country	28 Country	Depth 1 (Tree)	Depth 2 CORE (xls)	Depth 3 xls file validation	Depth 3 knime generation extraction	Rough/Exact	Exist/Robust
1.1	Lifestyle	PIK	Core	100%	50%	(a)	(a)	(a)	(a)	50%	75%	75%	80%	75%	μ	(b)	?	75%
1.3	Technology	OGUT	Core	(b)	(a)	(b)	(b)	(b)	(b)	0%	50%	?	80%	25%	(b)	(b)	?	25%
2.1	Building	BPIE	Core	100%	(a)	100%	100%	50%	0%	25%	80%	50%	80%	75%	75%	0%	?	25%
2.2	Transport	Climact	Core	100%	(a)	100%	100%	75%	0%	50%	80%	40%	80%	75%	30%	0%	?	25%
3.1	Industry	OGUT	Core	50%	(a)	100%	100%	25%	0%	10%	50%	0%	80%	75%	75%	0%	?	25%
4.1	Land use	Imperial	Core	100%	(a)	100%	100%	25%	0%	10%	75%	50%	80%	75%	0%	0%	?	?
4.3	Agriculture	Imperial	Core	100%	(a)	100%	100%	25%	0%	10%	75%	50%	80%	75%	0%	0%	?	?
5	Power (5.1 Electricity)	PANNON	Core	75%	(a)	100%	100%	25%	0%	10%	?	?	80%	75%	0%	0%	?	50%
	5.2 Fossil Fuel	PANNON	Core	50%	(a)	0%	0%	0%	0%	10%	?	?	80%	75%	0%	0%	?	100%
5.3	Biomass	Imperial	Core	75%	(a)	0%	0%	0%	0%	10%	?	?	80%	50%	0%	0%	?	75%
1.2	Climate	UEA	Non-Core	(a)	(b)	(b)	(b)	(b)	(b)		?	?	0%	0%	(b)	(b)	?	25%
4.2	Minerals	Imperial	Non-Core	100%	(a)	100%	100%	25%	0%	10%	75%	50%	80%	0%	0%	0%	?	?
4.4	Water	UEA	Non-Core	(a)	?	(b)	(b)	(b)	(b)		?	?	0%	0%	0%	0%	?	?
4.5	Biodiversity	UEA	Non-Core	(a)	?	(b)	(b)	(b)	(b)		?	?	0%	0%	0%	0%	?	?
5.4	Water-energy nexus	EPFL	Non-Core	(a)	0%	0%	0%	0%	0%		0%	0%	0%	0%	0%	0%	?	(b)
6.1	Energy security	TU Delft	Non-Core	(a)	0%	0%	0%	0%	0%		?	?	0%	0%	0%	0%	?	(b)
6.2	Education	TU Delft	Non-Core	(a)	0%	0%	0%	0%	0%		?	?	?	0%	0%	0%	?	(b)
6.3	Human health and safety	TU Delft	Non-Core	(a)	0%	0%	0%	0%	0%		?	?	?	0%	0%	0%	?	(b)
6.4	Employment	EPFL	Non-Core	(a)	50%	100%	0%	75%	50%		50%	50%	100%	75%	50%	50%	?	(b)
6.5	Working conditions	TU Delft	Non-Core	(a)	0%	0%	0%				?	?	0%	0%	0%	0%	?	(b)
7.1	baseline projection	UCPH	Non-Core	(a)	(b)	(b)	(b)	(b)	(b)		(b)	(b)	(b)	(b)	(b)	(b)	(b)	(b)
	Economy		Non-Core	(a)	0%	0%	0%	0%	0%		0%	0%	0%	0%	0%	0%	0%	0%
	EU & Rest of World		Non-Core	(a)	0%	0%	0%	0%	0%		0%	0%	0%	0%	0%	0%	0%	0%

Calculation Trees/Report	Core	*covers complexity of core modules*
	Non-Core	*covers complexity of non-core modules*
Code/Knime	Initial	*first version representative of module calculation rationale*
	Cleaned by Climact	*version cleaned to use the same template accros all the sectors*
	Core	*Core Calculation trees implemented*
	Non-Core	*Non-Core Calculation trees implemented*
Data	1 country	*all data regarding Germany*
	28 country	*all data regarding all the countries, taking into account the filling of missing data*
Interface	Depths 1, 2, 3	*see slides illustrations (1 is Tree visualisation of links, 2 is Xsl definition of dimension and granularity, 3 is Interface definition in knime)*
Baseline	Rough/exact	*baseline of future scenarios*
Lever values	Exist/robust	*definition of level of levers*

Legend

(a)	not relevant
(b)	may be relevant, to be discussed
?	no info
100%	ready for release
75%	phase of finalizing
50%	ongoing
25%	the task has begun
0%	nothing has been done

Figure 21. Dashboard of model status

This allow all the members of the consortium to be aware of the advancement of the project. It is frequently updated and shared with consortium members.

5.2KNIME expertise

Once the calculation trees and the data collection are done, implementing this in KNIME becomes relatively simple. KNIME is an open source platform that links the tree leaves to the data and performs the various operations required. The visual nature of the KNIME tool makes the modelling easier to communicate to stakeholders as well.

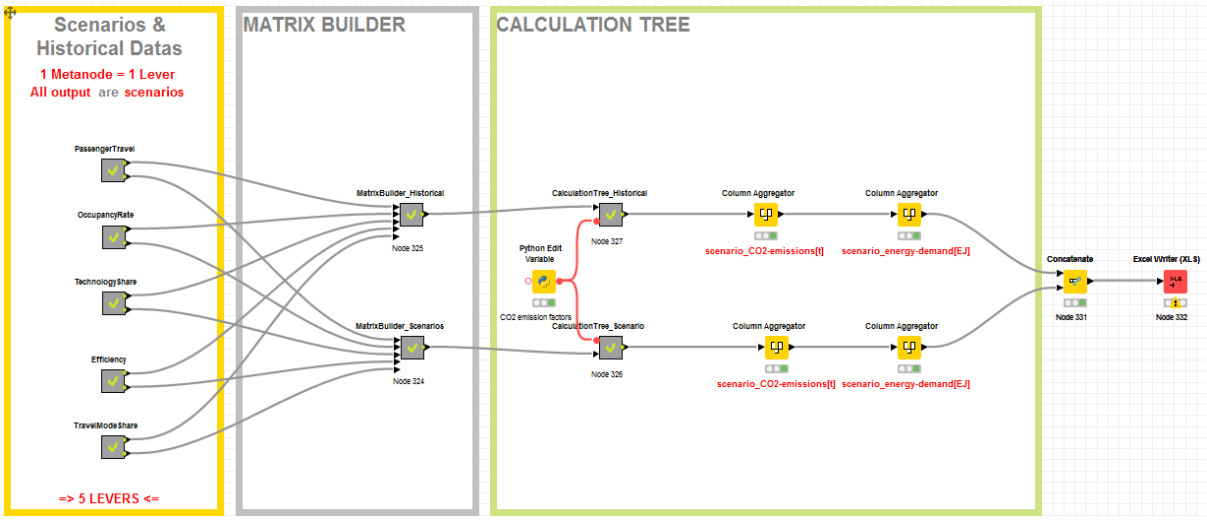


Figure 22. Knime Screenshot

This visual programming is quite intuitive, and Climact is clearly available to support each sector team.

5.2.1 KNIME training

To give every member of the consortium the capacities to use the tool, we have built a training kit to help team members adopting the tool.

Here are a few screenshots (Figure 23 and Figure 24) of the training sessions:

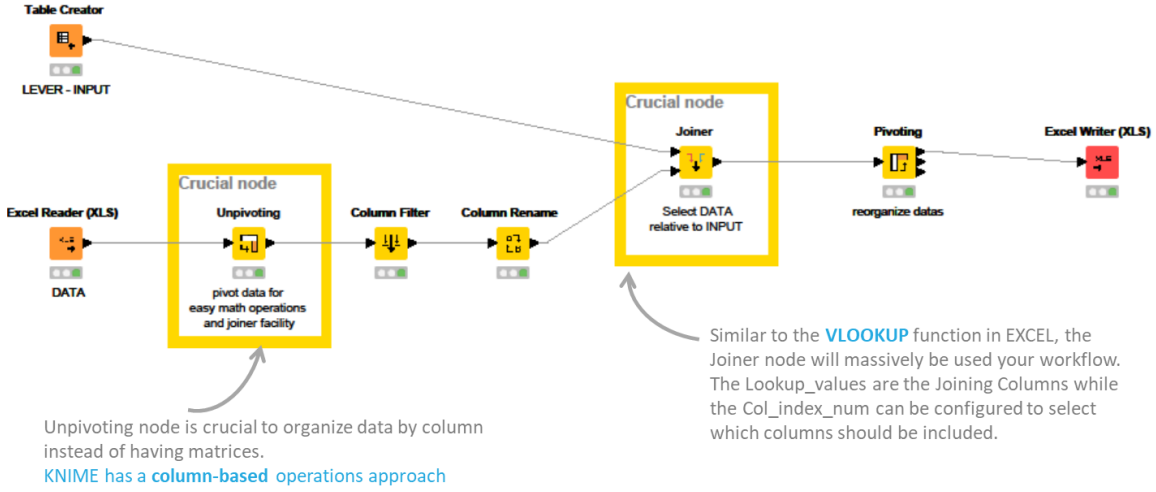


Figure 23. Training session 1: how to read a simple KNIME workflow

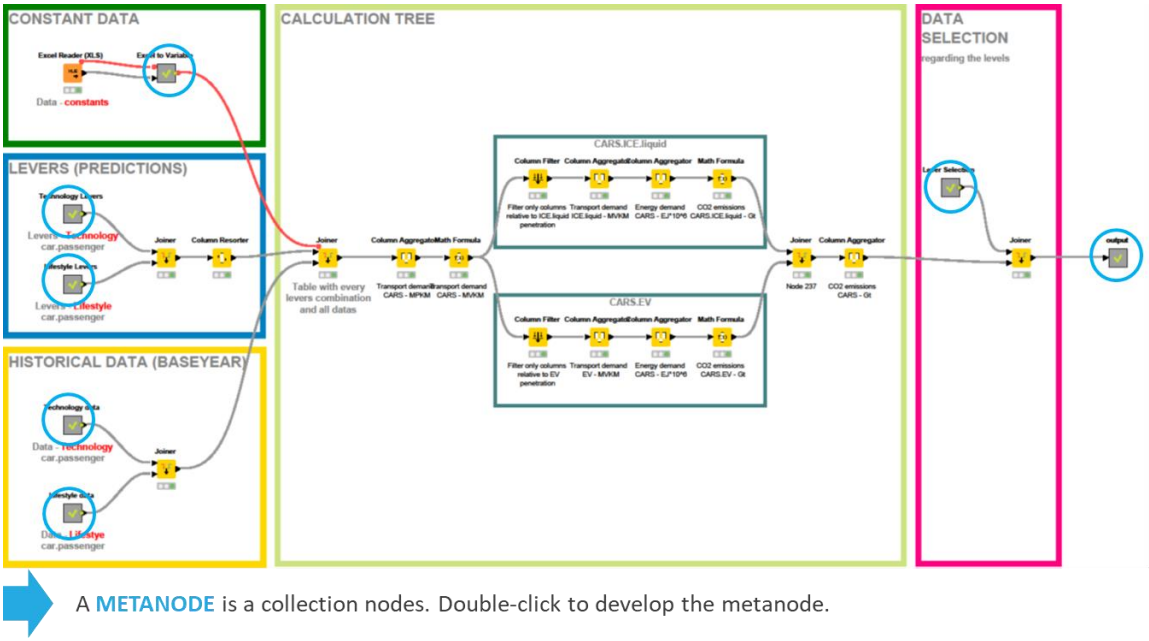


Figure 24. Training session 2: definition of a metanode

5.2.2 How to code with KNIME

Climact provides also the all consortium with some best practices regarding the code itself. The main best practices are the following:

1. Clearly separate data: Observed Time Series (OTS), Future Time Series (FTS), Levels of Levers (LL) and Constant Parameters (CP).
2. Hard code as less as possible (use configuration files instead).
3. Use the metadata to document each file. Create a metadata file for each input file/
4. Build modular and flexible code (e.g. create separate modules for data pre-processing and modelling itself)
5. Embed tests, validation, documentation and data cleaning right from the start, to make sure hypotheses on data format are explicit and documented

5.3KNIME workflows layout and documentation

5.3.1 Generic module-flow

The way of modeling the modules follows the same structure for each sector. This philosophy is illustrated in the Appendix 7.3 with the following steps:

1. Inputs of the data (OTS and LL) to the model (Figure 35)
2. Data are reorganized in a column way to be used by KNIME (Figure 36)
3. OTS and LL tables are merged to form a sector-table (Figure 37)
4. OTS and LL columns are compiled to for the FTS table (Figure 38)
5. The different levers are merged before the calculation trees (Figure 39)

5.3.2 Visual structure of a workflow

The workflow is organized to fit the human understanding: from left to right with a specific colour coding scheme for documentation and explanations.

An example of workflow is illustrated by the Figure 25 with the following colour coding for functional meaning:

- Yellow: Input and cleaning (R=255, G=216, B=0)
- Green: Output/reporting/formatting (R=141, G=198, B=66)
- Red: Error/warning processing (R=255, G=217, B=217)
- Blue: Separate methodological processing steps that are not inputs, outputs or errors (R=37, G=170, B=225)

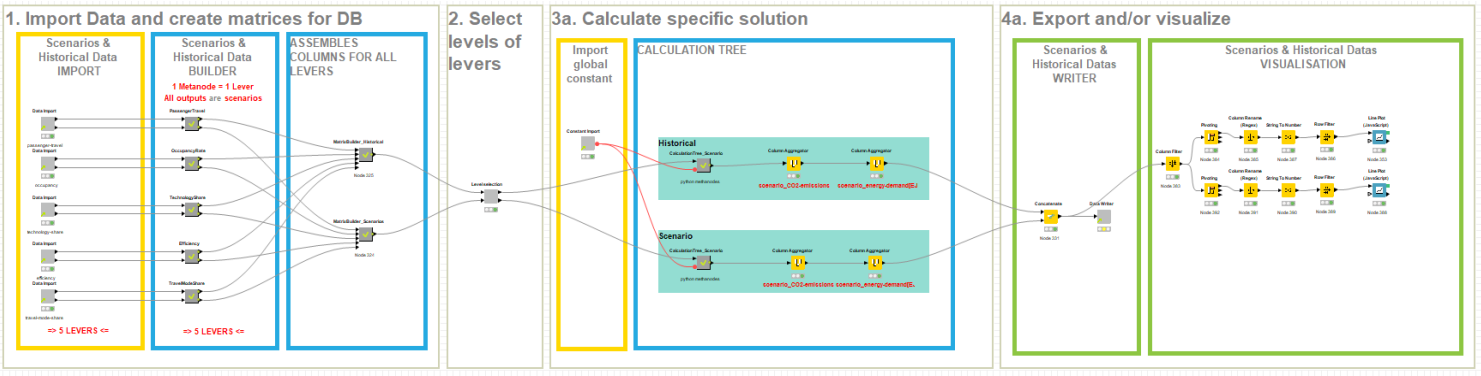


Figure 25. KNIME: Left to right movement to match human reading pattern

The in-workflow documentation is crucial for the transparency of the model. The main points to be considered when documenting the workflow are the following:

- "**How**" the workflow does it shouldn't be described in writing but by using clear logical steps in the workflows. People interested in the last level of detail will need to open nodes configuration.

- Each workflow should have **one supporting methodology document** that follows the same logic as the workflow. The document answers the "**why?**" questions (i.e. justifications)
- **Workflow documentation** only answers the question of **"What the workflow does?"**. It uses annotations and nodes labels and **do not refer to documentation:**
    - Use "workflow annotations" to group elements in boxes (right click box background to change colour). Each box links to **a logical element of the** methodology (see the boxes in Figure 25 for examples)
    - Include a **name for all nodes** (see Figure 26)
    - Boxes and nodes **describe what is happening** e.g. "Calculate heat side demand for hot water"
    - It should **not describe how it is happening (i.e. a KNIME or data processing step)** e.g. "Transpose table and apply join". People interested in this will have to be trained on KNIME and look at the nodes themselves.

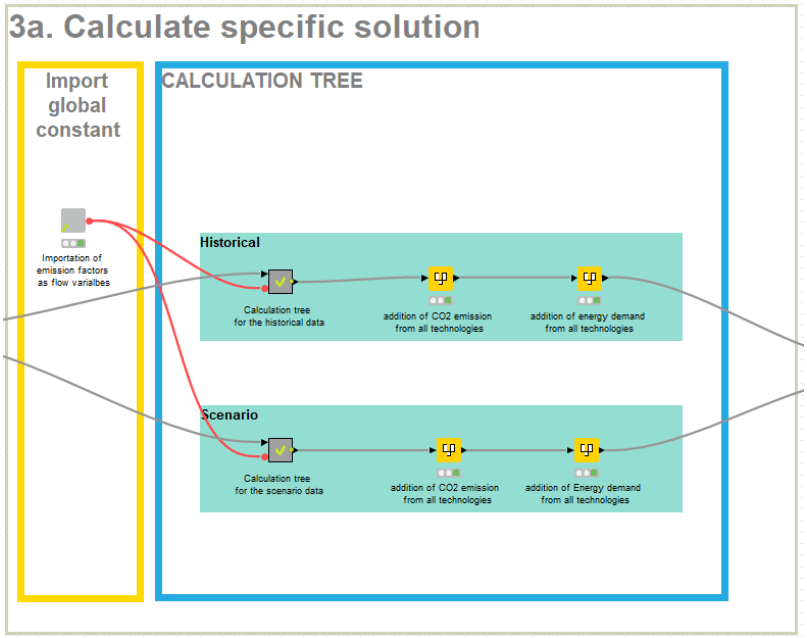


Figure 26. Zoom on the KNIME workflow to show the details and the nodes description

5.3.3 Specific nodes

5.3.3.1 Metanodes

To add more structure to the workflow, the modeler can also create metanodes. A metanode is a way to visually remove lengthy sections of workflows that do not add to the general understanding of the workflow. See on Figure 26 examples of metanodes (grey boxes with a green "V" on the left).

However, metanodes tend to hide complexity and make workflow less "transparent", so the modeler should not overuse them and create metanodes inside metanodes when possible.

The inside of the metanode follows the same documentation standards. It should not be a way to hide away the details

5.3.3.2 Python Nodes

In the context of the EUCalc, the Python nodes are useful. They allow the modeler to use the power of the Python language inside of a visual programming workflow. Moreover, while doing the conversion to the Python model, the Python nodes can be optimized to be more efficient than standard KNIME nodes.

Python nodes have trade-offs. On the plus side, they make the workflow easier to understand and ensure the final python code (after the converter) is optimized. For this reason, we encourage this practice through the use of CalcNodes (see above).

On the minus side, they could hide insightful assumptions. They should only be used when operations using standard KNIME nodes are really difficult, for instance iterative loops on rows/columns or use of specific functions. Or when they enable to bundle a frequent operation with an optimized Python code.

If the modeler uses in-node Python code, he should minimize the amount of operation in the node and document the steps the code implements in the KNIME workflow itself. This typically requires an additional annotation box with more text (see Figure 27).

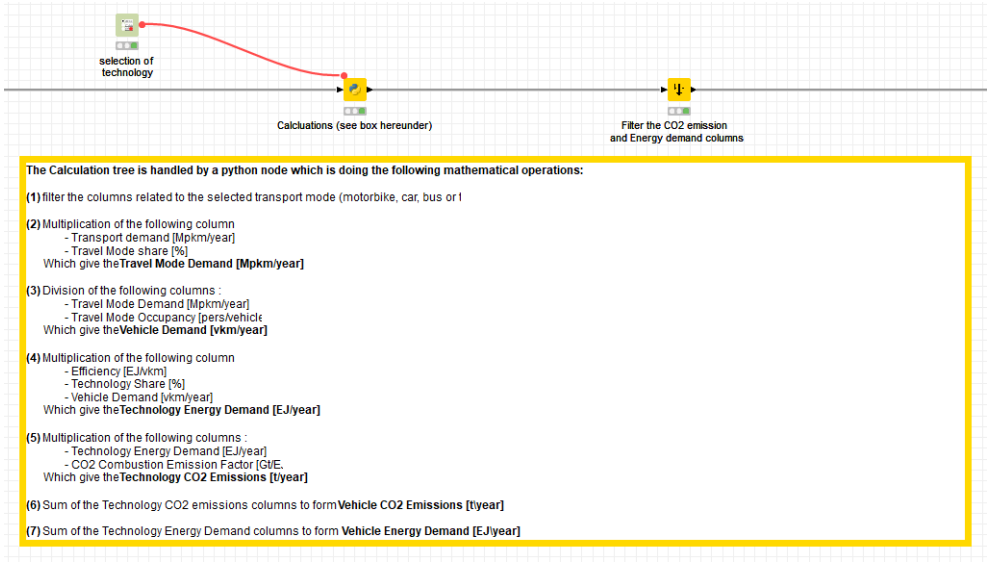


Figure 27. Example of Python nodes with documentation

5.4 Quality review

5.4.1 In-workflow quality check

To ensure the quality of the workflow, Climact introduced the template metanodes. By using the input and output metanodes provided by Climact, the following checks are automatically performed:

- File name and format (inputs);
- Mandatory columns presence (e.g. country for LL and OTS, level for LL) for inputs;

- Data types;
- Column naming (for outputs).

The quality of a workflow stays the responsibility of each sector leader. To ensure this quality, as best practice, the modeler should take a moment to identify assumptions that may be wrong when the input data changes. For instance:

- Values that should be in a certain range (e.g. from 0 to 1 for a proportion);
- Values that should match a list (e.g. a "base year" constant that would be one of values in the "years" list);
- Values hard coded that could actually change.

All these assumptions result in a test performed in the workflow. While consuming time initially, this significantly reduces the effort when something goes wrong.

5.4.2 Between sectors

To ensure the validity of the data in term of values and nomenclature, the modeler should respect the following rules: each column

- have a UNIQUE name;
- have a STANDARD structure (see templates);
- is easily understandable by a human or a machine;
- is related to the metadata of the source.

The naming structure used for all the sector is the following (Figure 28):

Figure 28. Nomenclature of the column names

The standardization of names and file structure allows the validity check between the sectors and the use of standard metanodes.

Note that the list of *data-types* and *module-trigrams* is defined in the Data Management Plan (D11.2).

5.4.3 At the output of the model: to ensure the quality of the calculations performed by the model

To ensure the quality of the output from each model, Climact intends to perform a check by comparing the historical data calculated by the model with historical data found in the literature or in official databases (e.g. EUROSTAT). This will allow us to validate the calculations performed by the model and to calibrate the results.

5.5 From development to production environment

So far, the development is built around 3 layers (see Figure 29): a development, a connection and a production layer. While the development layer is based on KNIME and built by every WP, the connection and production layers are managed by Climact and developed in Python.

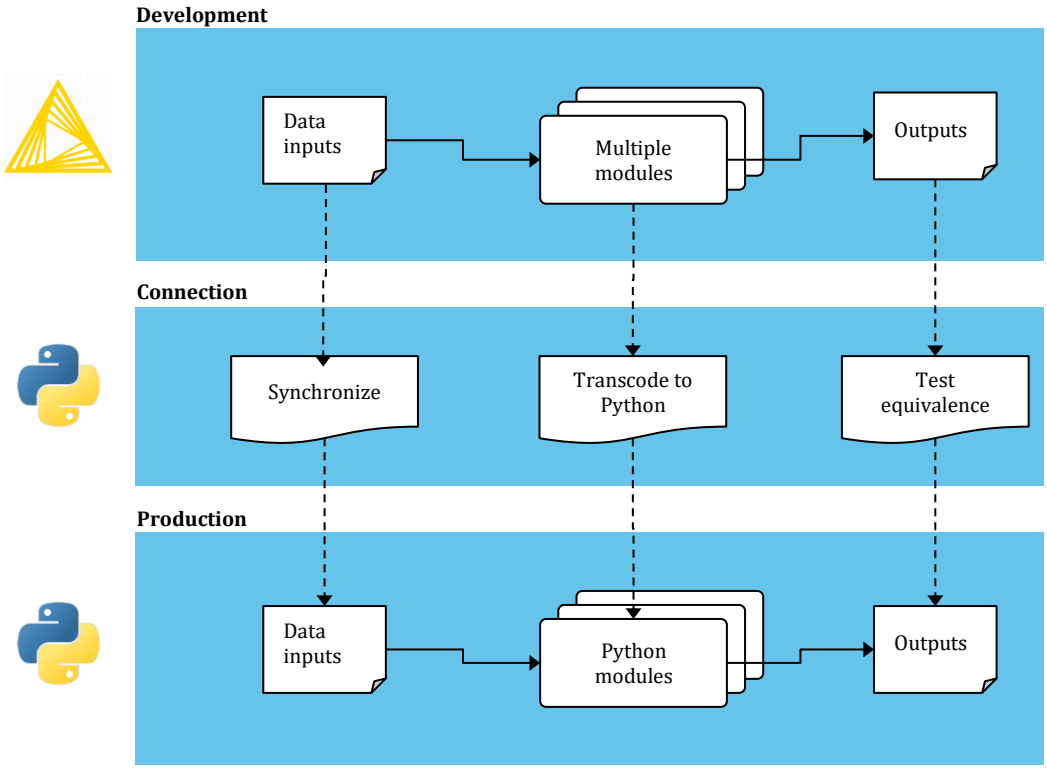


Figure 29. Layers of the modelling process

5.5.1 Development environment

The architecture of the development environment is presented in the Figure 30. It shows the interaction between two modules and the way of importing the data to the model. The raw data inputs specifics to each WP are managed by each module builder. Some of the inputs can be live connections to webserver (e.g. Eurostat, OECD).

dev\

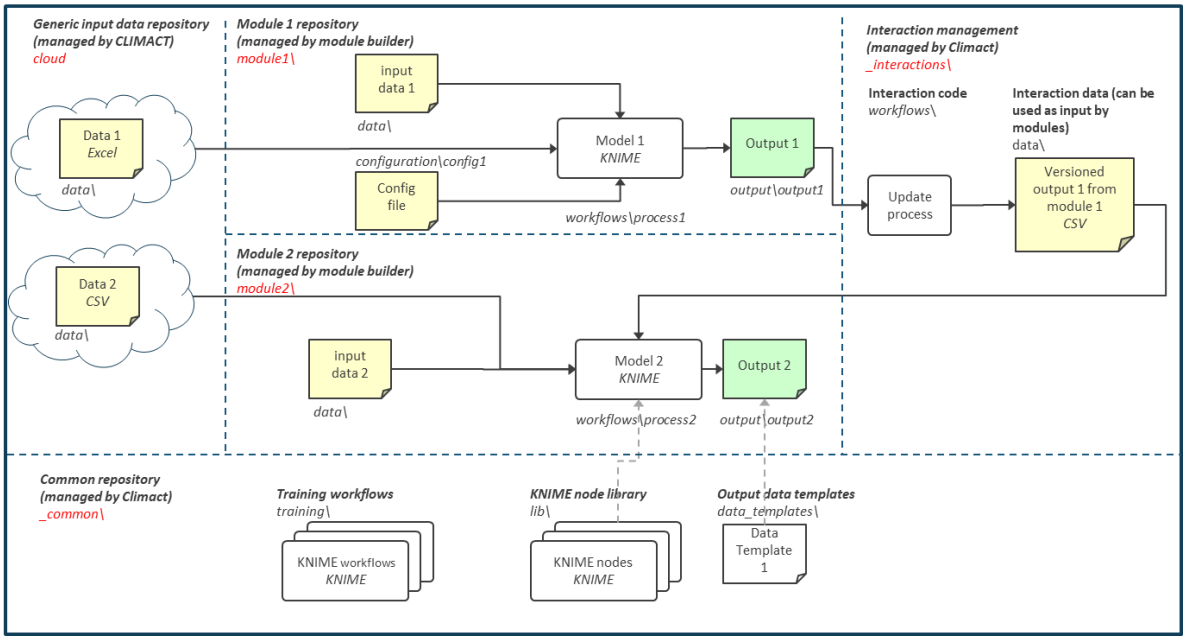


Figure 30. Architecture of the development environment

This architecture is based on a file structure that is represented in the following figure:

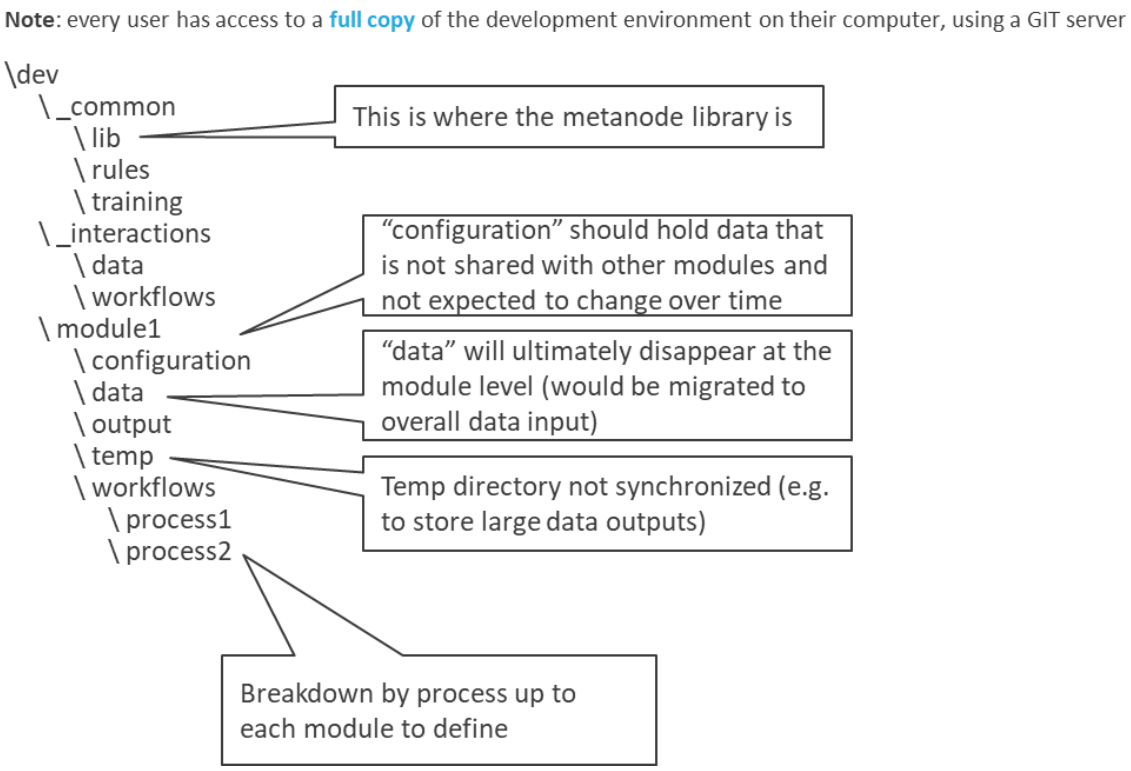


Figure 31. Structure of the development environment folders

This means that every user has access to a full copy of the development environment on their computer and that they have the full compilation capacities to run the model any time they want.

Each module is developed separately. It is the role of Climact to bring all the pieces together to build the model. The current model is presented in the Figure 32.

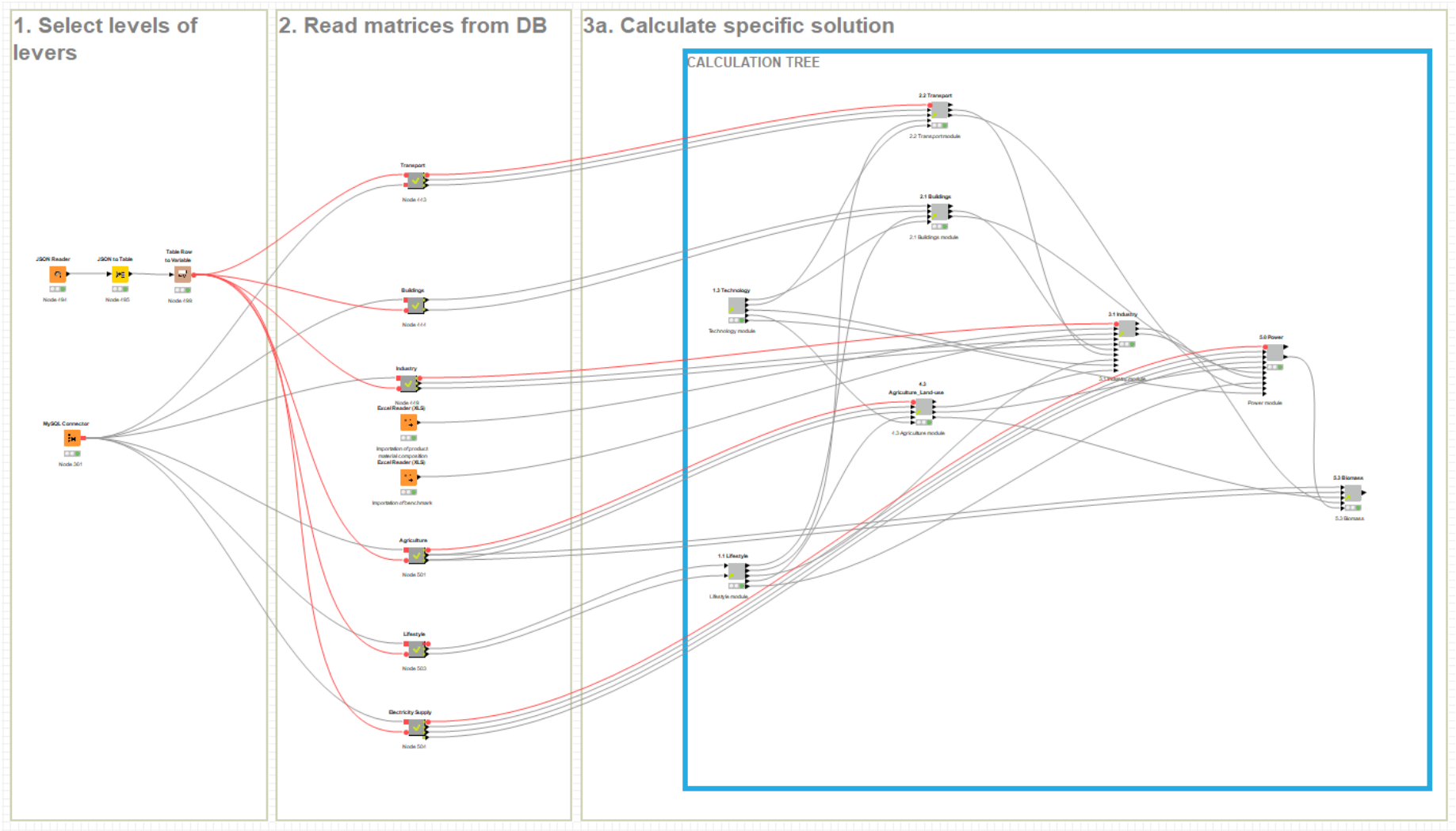


Figure 32. Current version of the model

5.5.2 KNIME to Python converter

The converter takes as input a KNIME workflow and generates a Python code. We ambition it to automatically perform the entire conversion in order to keep the modelling only in KNIME or inside Python nodes.

5.5.2.1 Architecture of the converter

The converter is built to understand the KNIME workflow. It runs in 2 steps:

1. Build a directed graph: using the power of the NetworkX[6] package, the converter is building a directed graph using the input and output ports of the KNIME model as nodes of the python graph (see Figure 33 for a representation and Figure 34 for output from Python). To do so, the python converter is looking at each node and each edge of the KNIME workflow before translating them into python language.
2. Run the directed graph: when the directed graph is ready, we just need to run each node, one by one, respecting the topological order of the graph.

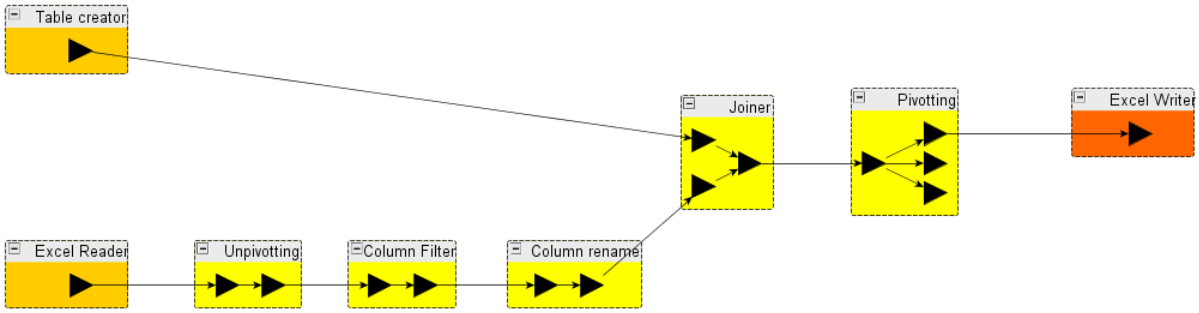


Figure 33. Python representation of the KNIME training session 1 (Figure 23)

[6] NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks: https://networkx.github.io/

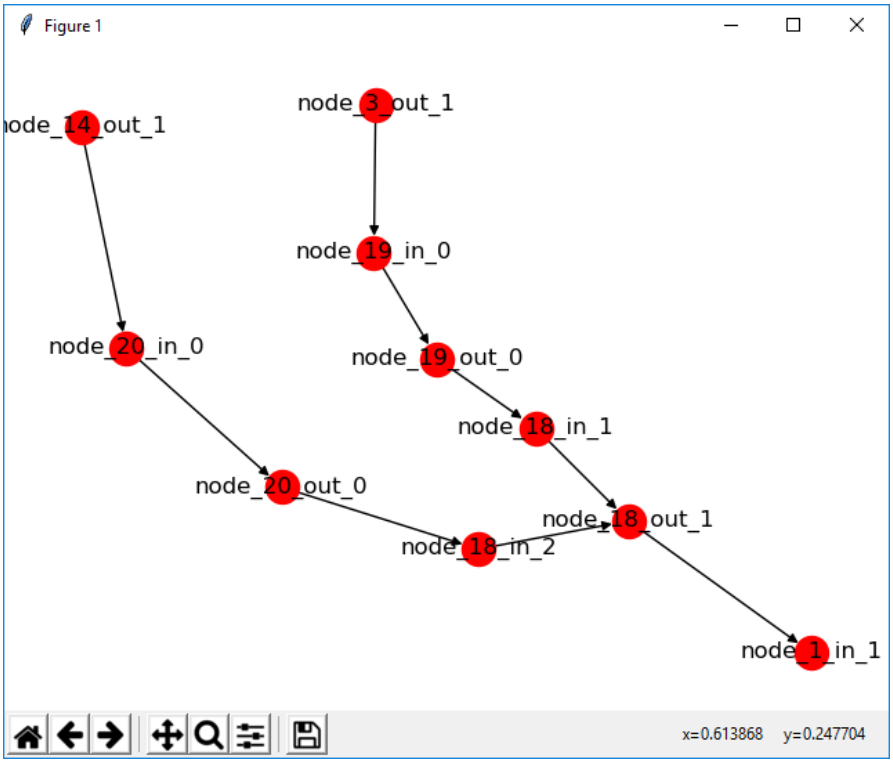


Figure 34. Example of Python directed graph from the converter

5.5.2.2 Implemented nodes

The current available nodes that are under development at Climact are the following:

- column_aggregator_node
- column_filter_node
- column_rename_node
- column_rename_regex_node
- connection
- excel_reader_node
- excel_writer_node
- joiner_node
- math_formula_node
- meta_node
- pivoting_node
- python_1_1_node
- python_1_2_node
- python_2_1_node
- python_2_2_node
- row_filter_node
- table_creator_node
- unpivoting_node
- wrapped_meta_node
- wrapped_node_input
- wrapped_node_output

6 References

[Climact, 2018] Climact (2018). Carbon Transparency Initiative – Prospective analysis to 2050. European Climate Foundation

[Climact& VITO, 2013] Climact, VITO (2013). Scenarios for a low carbon Belgium by 2050.Federal Public Services of Belgium.

[Alexandre Strapasson et al] (2018) Modelling Carbon Mitigation Pathways, An Assessment of the Global Calculator. DRAFT for review.

[KNIME] Documentation, Retrieved April 26, 2018, from https://www.knime.com/documentation

[yEd] Documentation, Retrieved April 26, 2018, from http://yed.yworks.com/support/manual/index.html

7 Appendix

7.1 Current list of levers[7]

WP#	Module	Section	Node	*Categories*	Lever	Trajectory	Limited resource	unit
WP1	1.1 Lifestyle							
		Demography						
			Population			x		people
			Urban population			x		%
		Economy						
			GDP			x		€/year
		Diet						
			Diet		x			kcal/person/day
	1.2 Climate							
			Temperature			x		°C
			Solar radiation			x		W/m2
			Wind			x		m/s
			GHG and aerosol concentration			x		ppm
	1.3 Technology							
			Learning rate		x			%
			Capex		x			€/kW
			Opex		x			€/kW
			Efficiency factor		x			%
			Decommissioning		x			%
WP2	2.1 Buildings			Residential-Commercial				
		Technology						
			Envelop quality		x			GW/(M ha*°C)
			Technology share		x			%
			Appliance efficiency		x			%
		Lifestyle						

[7] Status on 30/04/2018

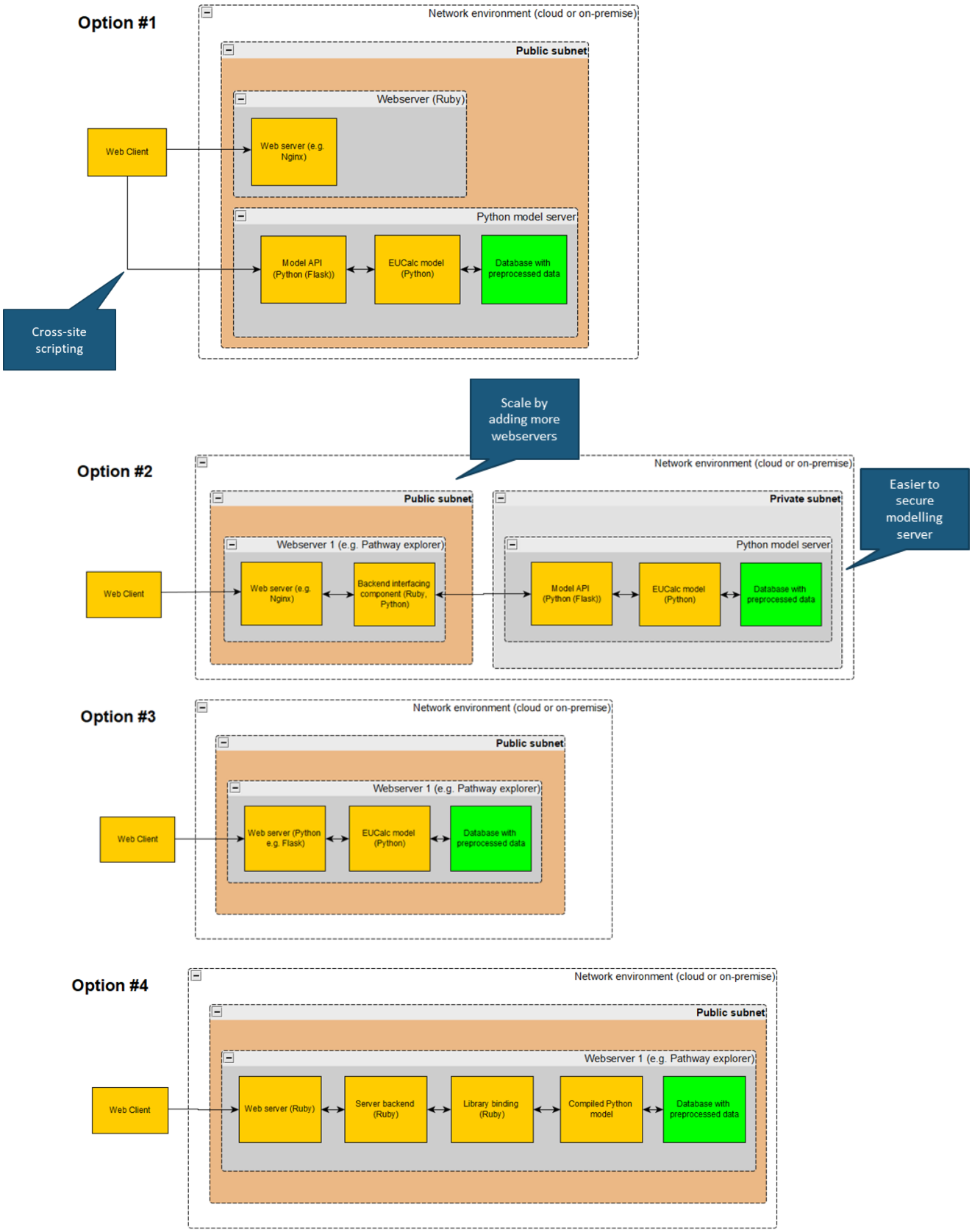
			Building size		x			m²
			Temperature & hot water use		x			°C
			Lighting & appliance use		x			appliances
			Renovation rate & depth		x			
			(households size)		x			
			(use intensity)		x			
			(cooling behavior)		x			
	2.2 Transport			Passenger-Commercial				
		Technology						
			Efficiency		x			l/km
			Technology share		x			%
		Lifestyle						
			Passenger distance (transport demand)		x			pkm
			Freight distance		x			tkm
			Mode		x			%
			Occupancy & Load		x			people
			Car own or hire		x			vkm
WP3	3.1 Industry							
		Technology						
			Design, materials & recycling		x			%
			Energy efficiency in energy-intensive industry		x			%
			Product lifetime		x			yrs
			Raw material demand		x			t
			Fuel switch		x			%
	3.2 Industry							
		Technology						
			Carbon Capture Use & Storage		x			%
WP4	4.1 Land use							
			Allocation		x			%
			Usage Efficiency		x			%
	4.2 Minerals							
			Fossil fuel				x	l

			Minerals				x	kg
	4.3 Agriculture							
		Production						
			Crop yields		x			kcal/m²
			Livestock		x			
		Waste & residues						
			Waste & residues		x			%
	4.4 Water							
			Water				x	l
	4.5 Biodiversity							
			Biomass				x	
			Biodiversity				x	
WP5	**5.1 Electricity**							
		Renewables		Electricity-Heat				
			Wind capacity		x			GW
			Biomass capacity		x			GW
			Solar capacity		x			GW
			Other renewables (Marine, Geothermal…)		x			GW
			(Efficiency)			x		%
		Fossil						
			Fossil fuel (coal, oil) capacity		x			GW
			Ratio of CCS		x			%
			(Efficiency)			x		%
		Nuclear		Electricity-Heat				
			Nuclear capacity		x			GW
		Balancing strategy		*Electricity*				
			Natural gas, Storage & demand shifting, export-import		x			GW, Gwh
	5.2 Fossil Fuel							

		Economy						
			(Fuel Economy)			x		€/l
			(Carbon Price)			x		€/kg
		Technology						
			(Efficiency)		x			%
			(Coal / oil / gas)		x			%
	5.3 Biomass							
			Bionenergy yields		x			W/m²
			Solid or liquid		x			%

Table 9. Current lever List

7.2 Interface with Pathway Explorer options



7.3 Generic module-flow

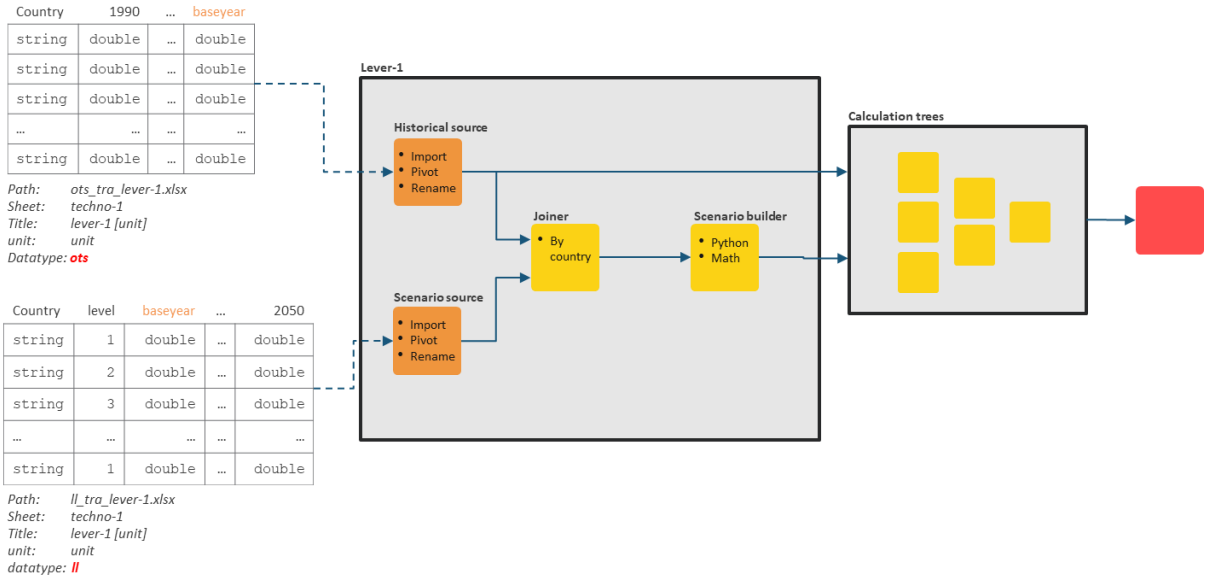


Figure 35. Step 1: input of the data (OTS and LL) to the model.

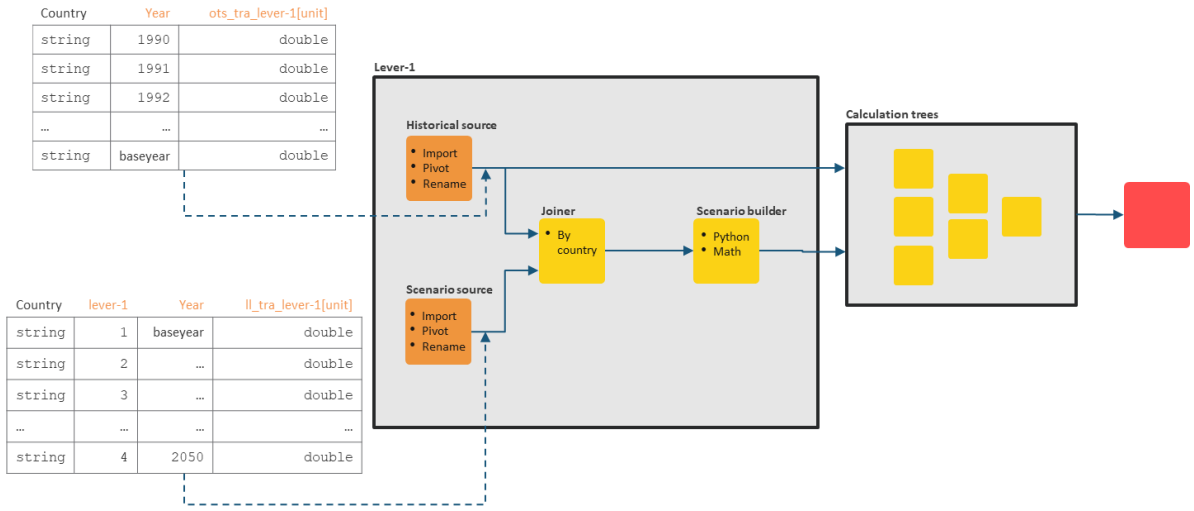


Figure 36. Step 2: data are reorganized in a column way to be used by KNIME

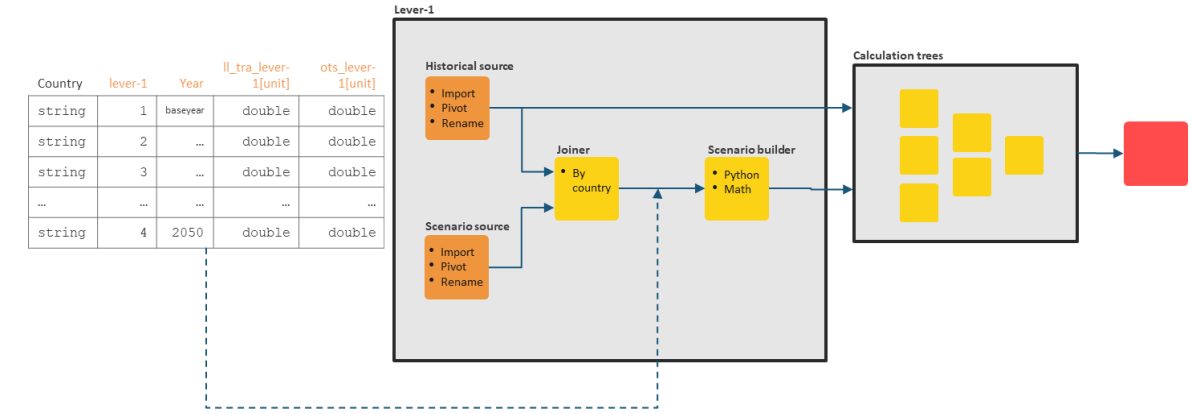


Figure 37. Step 3: OTS and LL tables are merged to form a sector-table

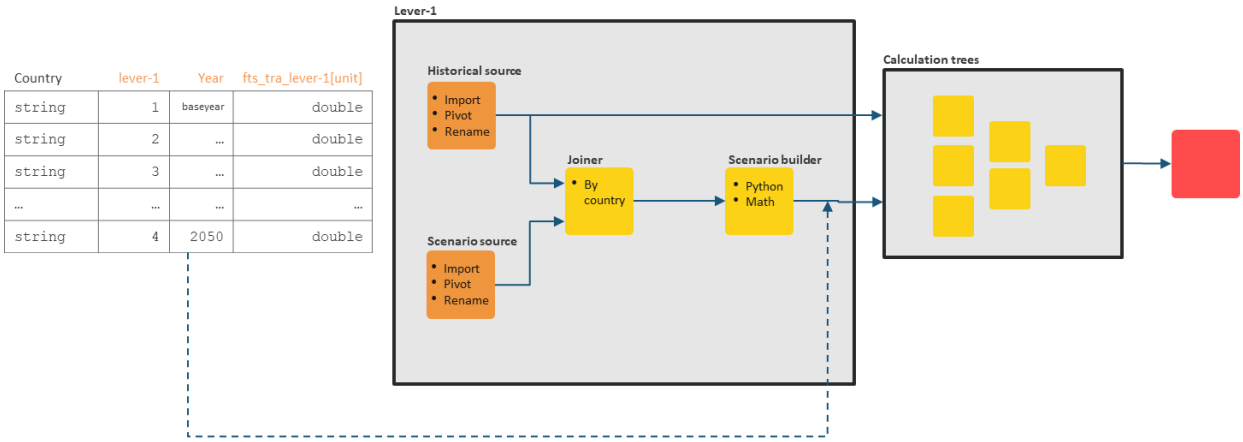


Figure 38. Step 4: OTS and LL columns are compiled to for the FTS table

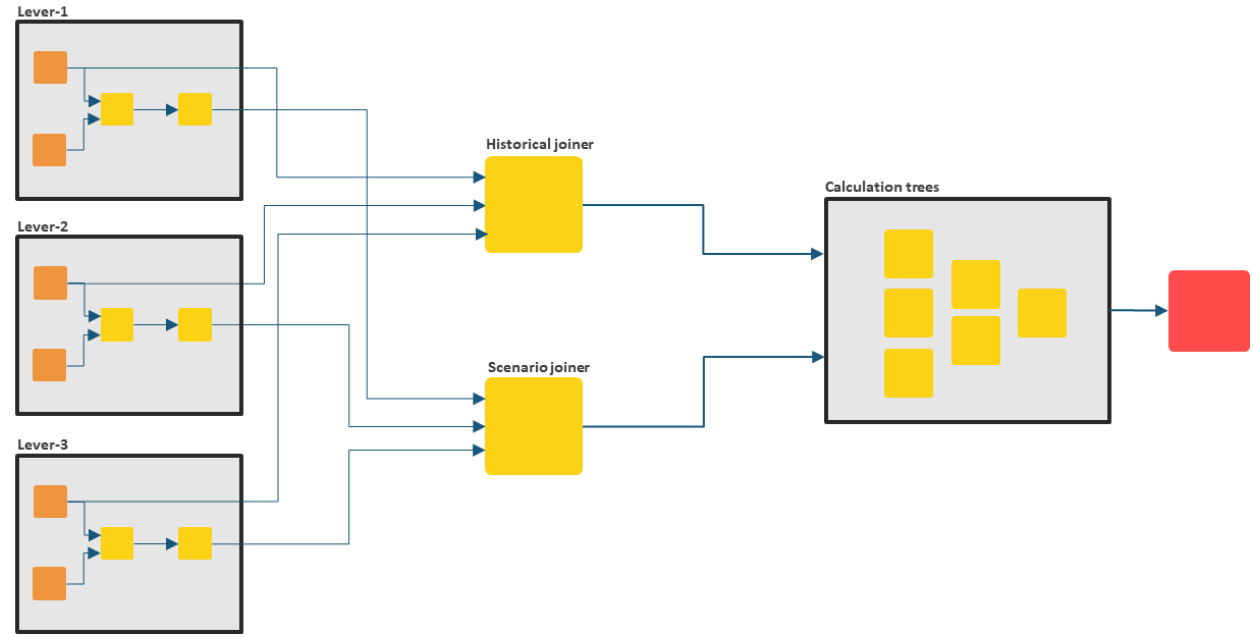


Figure 39. Step 5: The different levers are merged before the calculation trees