LOCOMOTION

Report of the common modelling framework

WP 9, Task 9.1, D.9.1 (version 5.0)

31st January 2023

LOW-CARBON SOCIETY: AN ENHANCED MODELLING TOOL FOR THE TRANSITION TO SUSTAINABILITY (LOCOMOTION)

H2020-LC-CLA-2018-2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821105.

DOCUMENT HISTORY

Project Acronym	**LOCOMOTION**	
Project title	Low-carbon society: an enhanced modelling tool for the transition to sustainability	
Project coordination	Universidad de Valladolid (Spain)	
Project duration	1st June 2019 – 30th November 2023	
Deliverable No.	D9.1. Report of the common modelling framework	
Dissemination level	Public (PU)	
Status		In progress
		To be verified by other WPs
	*	Final
Due date of deliverable	31st of May 2020	
Delivery date	29th of May 2020	
Version	v.5.0	
Work package	WP9 – Technical coordination and quality assurance	
Lead beneficiary	UVA	
Author (s)	Yania Crespo (UVa), Ignacio de Blas (UVa), Iñigo Capellán-Pérez (UVa), Jesús Vegas (Uva), Luis Javier Miguel González (UVa), David Álvarez (UVa), Martin Baumann (AEA), Margarita Mediavilla (UVa), Roger Samsó (CREAF), César Llamas (UVa), Carmen Hernández (UVa), Luis Fernando Lobejón (UVa), Iñaki Arto (BC3), Ignacio Cazcarro (BC3), Gonzalo Parrado (UVa), Mohamed Lifi (UVa), Gonzalo Manero (UVa)	

Date	Ver.	Author	Comment
10/10/2019	0.1	Luis Javier Miguel González (UVA)	Table of contents
22/11/2019	0.2	Luis Javier Miguel González (UVA)	Index of contents
10/02/2020	1	Yania Crespo (UVa), David Álvarez (UVa), Martin Baumann (AEA), Margarita Mediavilla (UVa), Roger Samsó (CREAF)	Describes naming conventions, programming rules, data dictionary, protocol and quality assurance tool. Introduces version control.
13/02/2020	1.1	Yania Crespo (UVa)	Refine naming conventions, clarify aspects regarding adjectives and acronyms.

11/03/2020	1.2	Jesús Vegas (UVa). César Llamas (UVa), Carmen Hernández (UVa)	Section about creation and use of input.xlsx
11/04/2020	1.3	Yania Crespo (UVa)	Initial version of the modules proposed. Appendix A. LOCOMOTION Data Dictionary description
14/04/2020	1.4	Ignacio de Blas (UVa)	Minor changes. Review and editing format.
22/04/2020	1.5	Ignacio de Blas (UVa)	Changes after first review
24/04/2020	1.6	Luis Javier Miguel González (UVa), Yania Crespo (UVa), Jesús Vegas (Uva), Ignacio de Blas (Uva), Iñigo Capellán-Pérez (UVa), David Álvarez (UVa), Gonzalo Parrado (UVa), Luis Fernando Lobejón (UVa), Iñaki Arto (BC3), Ignacio Cazcarro (BC3)	Changes after second review. Set up of LOCOMOTION regions and sectors
14/05/2020	2.0	Ignacio de Blas (UVa), Gonzalo Parrado (UVa)	Changes after WP9 Workshop (part 1), update of LOCOMOTION regions
21/05/2020	2.1	Iñigo Capellán-Pérez (UVa)	Changes after WP9 workshop (part 2) and inclusion of the WP8 agreement on definitions and scenario design
28/05/2020	2.2	Jesús Vegas (UVa), Yania Crespo (UVa), Iñigo Capellán-Pérez (UVa)	Restructuring, improvements to the naming convention and excel files management, as well as corrections to the Annexes on LOCOMOTION countries and sectors.
15/07/2020	2.3	Iñigo Capellán-Pérez (UVa)	Update and completion of programming standards, definition of categories, description of WILIAM 0.1 and integration of the contents shared to WP9 in July 2020 (inter-module diagrame, etc.).
3/11/2020	3	Iñigo Capellán-Pérez (UVa), Jesús Vegas (UVa), César Llamas and Yania Crespo (UVa)	Set up of input xlsx data structure and naming convention. Other minor related changes.
3/08/2021	4	Yania Crespo, Iñigo Capellán-Pérez & David Álvarez-Antelo	Update of glossary on definitions in section 7.1 as per WP8 update. Updates in naming conventions & other programming standards (e.g., units consistency) Updates for data dictionary (including new Appendix E: VENSIM PLUGIN FOR SONARQUBE) A new appendix summarizing the installation and user manual of the plugin for SemanticMerge in gmaster,

			which is a tool for making easier the visualization of differences between versions and aiding in merging tasks in gmaster (APPENDIX F: ABOUT THE VENSIM PLUGIN FOR SEMANTICMERGE). A full tutorial has also been prepared, cf Alfresco link. https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/2b449951-96ff-40a6-9d53-3d74754408ff Managing 9-35 regions within the same WILIAM model. Updated input data files management and usage with the issue about the multidimensional arrays >2 dimension and the excels2vensim application. Updated template for input data xlsx structuring. Correction of minor errors. Work with IOT matrixes in Vensim
30/09/2021	4.1	Ignacio de Blas	Units consistency in WILIAM. Minor changes. Review and editing format.
31/01/2023	5.0	Mohamed Lifi & Gonzalo Manero	Updates in naming conventions & other programming standards (e.g., Symbol naming conventions rules). Updates for the use of data dictionary (tutorial link is updated). Figures 4, 7, and C1 are updated. Table 1 & 2 are updated. Updates in developing the Wiliam model (e.g., Intermodule Consistency is removed from Wiliam model).

COPYRIGHT

©2019 LOCOMOTION Consortium Partners. All rights reserved. LOCOMOTION is a HORIZON2020 Project supported by the European Commission under contract No.821105. For more information of the project, its partners, and contributors please see LOCOMOTION website https://www.locomotion-h2020.eu/. You are permitted to copy and distribute verbatim copies of this document, containing this copyright notice, but modifying this document is not allowed. All contents are reserved by default and may not be disclosed to third parties without the written consent of the LOCOMOTION partners, except as mandated by the European Commission contract, for reviewing and dissemination purposes. All trademarks and other rights on third party products mentioned in this document are acknowledged and owned by the respective holders. The information contained in this document represents the views of LOCOMOTION members as of the date they are published. The LOCOMOTION consortium does not guarantee that any information contained herein is error-free, or up to date, nor makes warranties, express, implied, or statutory, by publishing this document.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

ABBREVIATIONS AND ACRONYMS

Acronym	Description
API	Application Program Interface
BAU	Business-as-usual
CARTIF	Fundación CARTIF
CSV	Comma Separated Values
CREAF	Centre for Research on Ecology and Forestry Applications
CRES	Centre for Renewable Energy Sources and Saving
DB	Database
DBMS	Database Management System
DMP	Data Management Plan
EROI	Energy Return on Investment
FAIR	Findable, accessible, interoperable and reusable
FEC	Final energy consumption
FED	Final energy demand
GA	Grant Agreement
GDP(pc)	Gross Domestic Product (per capita)
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
IAM	Integrated Assessment Model
ID	Identification Code
IPCC	Intergovernmental Panel on Climate Change
LDAP	Lightweight Directory Access Protocol
MOOC	Massive
MRIO	Multi-Regional Input Output
NECPs	National Energy and Climate Plans
NGO	Non-governmental organizations
OECD	Organisation for Economic Co-operation and Development
OEP	OpenEnergy Platform
PAV	Partially Aggregated Variable

PSB	Project Stakeholder Board
RCPs	Radiative Concentration Pathways
RDMS	Relational Database Management System
RES	Renewable energy sources
REST	Representational State Transfer: software architectural style that defines a set of constraints to be used for creating client-server apps and services.
RESTful	Web services, APIs, webapps that conform to the REST architectural style.
SAB	Scientific Advisory Board
SDGs	Sustainable Development Goals
SEO	Search Engine Optimisation
SLA	Service-level agreement
SPA	Shared Policy Assumptions
SSP	Shared Socioeconomic Pathways
UC	Use Case
UML	Unified Modelling Language
UTM	Urchin Tracking Module
UVA	University of Valladolid
WILIAM	Within Limits Integrated Assessment Model
XML	Extensible Markup Language

EXECUTIVE SUMMARY

The joint development of an IAM simulation model between various research groups requires a significant effort in technical coordination. The adoption of common rules and procedures for modelling and programming are essential for the effectiveness of work. This document contains a set of guidelines to facilitate the efficient and quality development of the LOCOMOTION IAM model. The name chosen by the consortium is "Within Limits IAM" (WILIAM). Efficient coordination also requires agreement in terms of nomenclature of common transversal terms, a section about common nomenclature for policy analysis and scenario assessment has also been included providing definitions as well as a general overview on how WILIAM model will be simulated.

This document, although it will be a necessary initial reference, will be a living document during the development of the IAM, adapting to new proposals for improvement that arise during the development of software products. This version is the 5th update since D9.1 submission in May 2020.

The objective of this document is to define the common modelling and programming framework of WILIAM, as well as the tools that facilitate efficiency in common work. To do this, the different IAM products are characterized, the general structure of the model is outlined, the standards for programming are defined as well as common definitions for policy analysis and scenario assessment are provided.

1. INTRODUCTION

1.1. SCOPE

The overall objective of LOCOMOTION is to enhance the IAM developed in the MEDEAS European project (https://www.medeas.eu/) in order to provide policy makers and relevant stakeholders a reliable and practical model system to assess the feasibility, effectiveness, costs and impacts of different sustainability policy options, and to identify the most effective transition pathways towards a low-carbon society.

To achieve this objective, the development of the modules of the new IAM (the name chosen by the consortium is "Within Limits IAM" (WILIAM)) is distributed in 4 modelling WPs (WP4, WP5, WP6 and WP7) together with a WP for data management (WP2) and another for developing scenarios and policies (WP8). The coordination and integration of the work of these 6 WPs is undoubtedly necessary, is being done through WP9 and requires at least four working conditions:

1. The design of the general structure of the model is understood by all the working groups so that all groups orient their work aiming at their integration into the general structure.
2. All working groups follow the same rules and criteria in the programming of each module. These rules include aspects of naming of variables, structure and programming style.
3. The use of online computer tools that allow programming work to be shared.
4. A fluid communication between the working groups for all those variables that can be shared between different modules.

The objective of this document is to contribute to the fulfilment of these 4 objectives.

A first proposal was developed by WP9 and shared with all partners participating in the model development in a meeting hold during the 5th and 7th of May 2020 online due to the exceptional conditions generated by the COVID-19 crisis. This document incorporates the improvements and agreements reached during that meeting.

An additional section, jointly worked between WP8 and WP9 deals with setting up a common nomenclature of transversal terms related with policy analysis and scenario assessment and simulation.

1.2. STRUCTURE

The structure of the document is the following, section 2 presents the different WILIAM products that will be developed of the project. Section 3 describes the general structure of WILIAM. Section 4 explains the programming standards about data dictionary, name convention, inputs files and other programming standards while section 5 the software configuration management. Section 6 shows different aspects in the development of WILIAM model and section 7 defines the common nomenclature for policy analysis and scenario assessment. Finally, appendixes provide more information about Vensim software, about LOCOMOTION data dictionary webapp and webservices, and about Vensim plugins for SonarQube and SemanticMerge, as well as tables representing the regional and sectoral disaggregation of WILIAM.

2. PRODUCTS OF WILIAM THAT WILL BE DEVELOPED

2.1. PRODUCTS ACCORDING TO THE PROGRAMMING LANGUAGE

According to the programming language there will be two types of products:

1. The main WILIAM product will be programmed in Vensim (see Appendix A for more details about Vensim Software). This product will be the one used for the main deliverable oriented to policy-makers, and the one used by the Locomotion consortium to report and disseminate results. Simpler releases, also programmed in Vensim, will be those used for the web and the game.
2. The final full release in Vensim will be translated into Python, with the aim of offering a completely open-source model. This release in Python will be oriented to the scientific community so that they can analyse and reuse the equations and structure of the model with total transparency. It will be also useful to make easier integration on webapps.

2.2. PRODUCTS ACCORDING TO THE FINAL USE OF WILIAM

According to the final use of WILIAM there will be four releases, with different interfaces:

1. Main release (Model Analyzer): It will be the most complete version developed in Vensim. The user interface oriented to policy makers and experts. Two scenarios of different complexity (full and simplified) are available to the users.
2. Educational product: oriented to the game and dissemination among students and citizens. This version may be simpler than the main version to gain execution speed. The Interface may have more educational elements and avoid very technical terms.
3. Web product (Model Explorer): It should be the fastest execution. In order to do this, it should be as simple as possible and even it can use a results emulator based on results already stored from a large case collection. The interface must respond to the simplicity of the model and the best appearance on the web.
4. Python product. The main version will be translated into Python. This version is oriented to the scientific and academic world. Regarding the experience of MEDEAS, although many aspects will be improved, it is expected to be slow in execution. In order to improve performance, the geographical or sector disaggregation may be reduced, but all the equations of the model will be represented to allow the complete mathematical understanding of it. It will have a basic interface oriented to the scientific community. Attractive or educational elements are not required for this interface, but it must be rigorous, clear and simple to use. Public graphic libraries will be used.

Table 1. Final WILIAM releases in LOCOMOTION

Product	Programming language	Model complexity	User Interface	Target audience	Purpose of the product.
Main	Vensim	Full model	Complete Professional	Policy makers & Experts (Full scenarios) Policy makers (Simplified scenarios)	Results generation
Educational (game)	Vensim	Medium or Low	Educational	Students and citizens.	Educate and raise awareness
Web	Vensim or emulator	Low (Fast to run)	Web-based (Fast to run)	General audience.	General diffusion of Locomotion. Education and awareness

Product	Programming language	Model complexity	User Interface	Target audience	Purpose of the product.
Python	Python	Full model	Scientific (basic)	Scientific community	Share the Locomotion model with the scientific community in open-source code

2.3. RELEASES ACCORDING TO THE GEOGRAPHICAL LEVEL

From the point of view of the geographical scale, WILIAM will be a single multiregional model. Some variables, such as the concentration of CO_2 in the atmosphere, will be global, while others variables will be also represented at regional level. The full model is planned to be released in two stages according to the geographical level:

1. **Regional level (9 major world regions):** EU-27, United Kingdom, China, EASOC (East-Asia & Oceania), India, LATAM (Latin America excepting Mexico), Russia, USMCA (US, Mexico & Canada) and LROW[1] (Locomotion Rest of the World). (see Table B. 1 and Figure B. 1 in Appendix B for details on the composition of each)
2. **9 major regions + the 27 countries of the EU-27,** which is one of the 9 major regions considered (a total of 36 geographical units).

The objective of this 2-steps release process is two-fold: on the one hand, due to computational challenges, given that the 36 geographical units version is expected to be computationally demanding considering a MRIO analysis of around 60 sectors, and on the other hand since the modelling of relations between 9 units is conceptually similar but operationally easier than for 35 units.

WILIAM models will be composed of 9 global regions (some of them are aggregated of countries and others are a country in it-self): EU-27, UK, China, EASOC (East-Asia & Oceania), India, LATAM (Latin America excepting Mexico), Russia, USMCA (US, Mexico & Canada) and LROW (Locomotion Rest of the World). This are two additional regions than what is stated in the Grant Agreement (7). In fact, the process of selecting this disaggregation has been very complex given that the initial concept was to use the WIOD input-output database (http://www.wiod.org/), also used in MEDEAS. However, in the last months it became clearer that another input-output database, ICIO (OECD Inter-Country Input –Output, http://oe.cd/icio) has a number of advantages and it was finally decided to switch to this database. Among the main advantages are: a higher regional disaggregation representing a total of 63 countries +1 RoW, ensured periodic updates, higher compatibility with other databases such as Eurostat or the IEA balances, the possibility to be combined with EXIOBASE IO database to further disaggregate some key sectors for the LOCOMOTION project such as the energy one, etc. In this context, given that the choice of the IO database is a major step in the development of the economy module with implications for the whole model, a process was conducted by WP4 in consultation with WP9 and the other modelling WPs, in order to reach to the final proposal of 9 global regions. Hence, this division was elaborated taking into account especially, but not only economic criteria.

It has to be noted that the common multiregional disaggregation of LOCOMOTION represents the “lowest common denominator” for all modelling WP and thus it does not prevent the modelling of a particular module, if decided so by the responsible partners, may be built with a more detailed regional disaggregation; however, this should be done consistently with the 9 global regions defined here. For example, it would be possible to model separately Mexico from USMCA, but it would not be possible to create a new region in which LATAM would also include Mexico. Like this, before entering the IO table the results could be aggregated in the 9 global regions.

[1] The acronym LROW is chosen instead of ROW to avoid mistakes when working with the ICIO database given that in this database the rest of the world is named as “ROW”

3. GENERAL STRUCTURE OF WILIAM

The general structure of the model is described in the Figure 1:

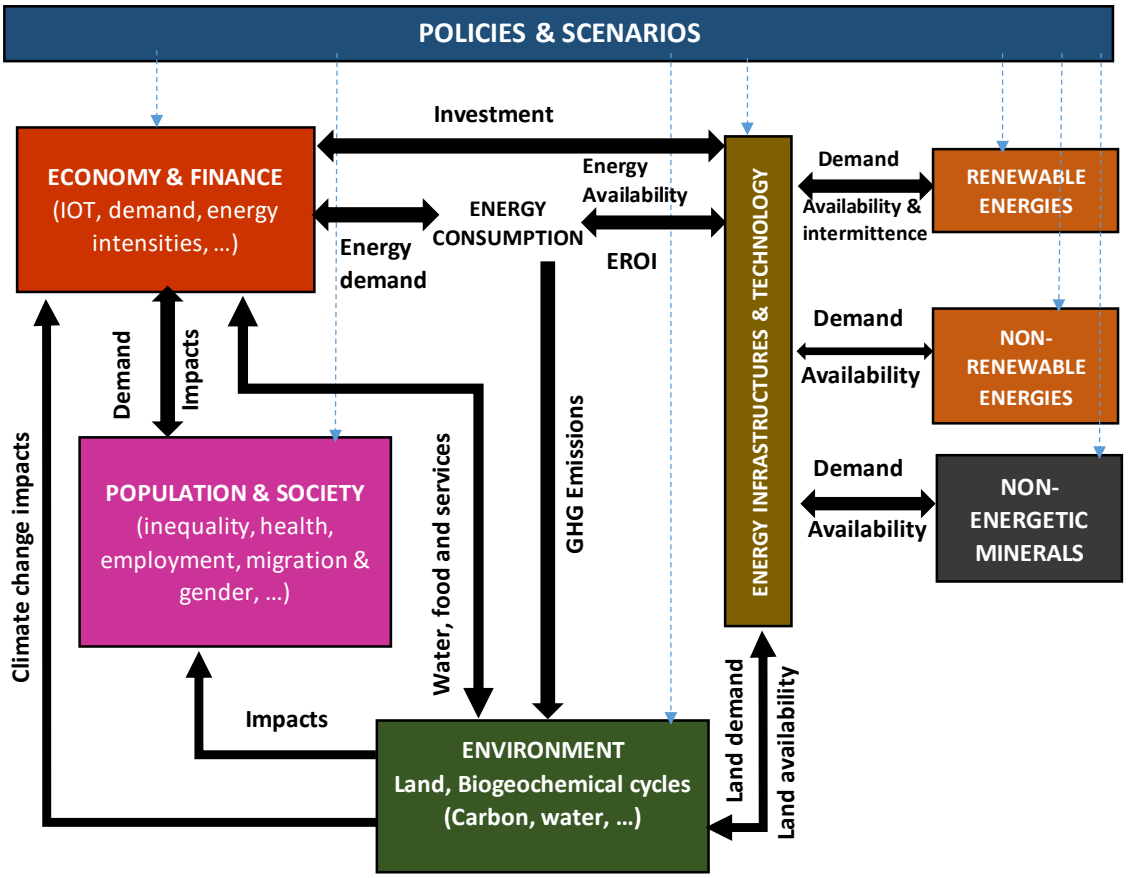


Figure 1: General structure of LOCOMOTION models

From this general structure, in order to organize the modeling work, the modules described in section 6.2.1 will be developed. Each module is the responsibility of each corresponding WP working group. However, there are a large number of relationships between the different modules. These relationships should be addressed through communication between the involved WPs both bilateratlly and through the coordination of WP9. In this document some methodological considerations are collected to take care of the coherence between the variables that connect.

The starting point of the model will be the overall goals pursued and the policies and scenarios that are proposed to achieve these overall goals. The objectives will be expressed in the form of indicators (such as GHG concentration, GDP per capita, other well-being indicators). The values pursued for these indicators will be the exogenous variables that will drive the dynamics of the model, along with the policies and scenarios. By default, the variables of each module will be disaggregated in the regions / countries considered as vector. These regions are defined in Appendix B.

The model execution sequence will follow the following basic steps:

The demand for products and services from households, the public sector and companies' investment will be a function of historical values and of the policies, scenarios and indicators pursued. The economic sectors are defined in Appendix C. These demands will be the entrance to the economic and financial system, which will be conditioned by human, energy and material resources. The outputs of the economy module will be inputs, at

least, of the energy and materials modules. The outputs of these physical modules will in turn determine the dynamics of the rest of the modules, with various feedbacks between them, including feedbacks on the economic and financial modules. In some cases, these feedbacks require the allocation of resources. Some of the functions of resource allocation which will need to be defined are:

- Fossil fuels and uranium consumption allocated between regions (except those protected from international trade via policies)
- Land products between regions (except those protected from international trade).
- Household income, government expenditure and gross fixed capital formation allocated between different economic sectors.
- Primary resources (e.g., fossil fuels, land products, minerals) into final resources for energy, food and raw materials for industry.
- Final energies conversion into other final energies and/or storage.
- Final energies available in every region allocated to different economic sectors and households.
- Land allocation to different land uses.
- Water allocation to different uses.

A process for the common definition of the allocation of resources in LOCOMOTION was set up after the online WP9 workshop in May 2020 and is currently in progress. In this workshop the selection of the regions and sectors was also approved.

4. PROGRAMMING STANDARDS

The design and development of each module must be carried out taking into account that it must then be integrated into the global multiregional model. The first reference of the global model will be the IAM of MEDEAS-World. Therefore, the design and development of each module must be carried out oriented towards its integration in MEDEAS-World (Vensim). That is why it is necessary for all modeller groups to have a good knowledge of the IAM MEDEAS-World in Vensim. We recommend that all modellers read the methodological paper of MEDEAS (Capellán-Pérez et al., 2020). It is also recommended to consult the documentation available on the web https://www.medeas.eu/ and especially the models that can be downloaded at https://www.medeas.eu/model/medeas-model#. We also recommend the completion of the MOOC course: "Simulation models for the design of transition paths towards a sustainable society", which briefly explains the MEDEAS models and can be taken free of charge.

As stated above, the starting point of WILIAM is the existing MEDEAS World IAM. However, before expanding with the new features described in the Locomotion project, some aspects of MEDEAS programming must be improved, specifically, the names and documentation of the variables (section 4.2) and the structure of the input file (section 4.3) In addition, a data dictionary (section 4.1) and a software configuration management must be developed (section 5).

In case of discrepancy, the programming standards described in this document have priority over the stage of WILIAM v0.1 released in Gitlab in July 2020.

4.1. DATA DICTIONARY

In the data dictionary the information on symbols can be collected in two different ways:

a) automatically from Vensim programs by injection using an API service (WP9).

b) through a form as a web interface to the data dictionary, as part of the database (WP2).

The documentation of each symbol will include at least the full name, the description of the symbol, the units, classification according to different criteria (**module, category and or/subcategory, etc**.), the method of obtaining or calculating and / or the original source of the data and level of certainty and reliability.

A **module** includes a broad number of symbols which fall within the same knowledge field (e.g., economy, energy, climate, etc.). The differentiation of modules has also a practical side in order to organize the controlled and coordinated development of the model (cf. section 6.2.1 for the modules set-up in WILIAM)

The project can define **categories and subcategories**, which are 2 levels of further classification of symbols below a module. Categories and subcategories allow to differentiate between groups of symbols which, although belonging to the same module or category, can be further grouped. This is useful for model organization and rationalization. Each module has the freedom and flexibility to organize the categories and subcategories that adapt better to their needs. There are some examples of symbols' categorization in: https://tntcat.iiasa.ac.at/AR5DB/dsd?Action=htmlpage&page=series.

The naming convention for the Vensim views (cf. section 4.4) has been designed to allow the automatization of the classification of each symbol in the data dictionary in the automatically from Vensim way (a), as well as to be compatible with the naming of the input data files (cf. section 4.3).

The conceptual kind of symbols that are going to be considered in Locomotion data dictionary are described in section 4.2.2.

The data dictionary available on the website will also include information on:

- acronyms (for each acronym: letters and meaning),
- adjectives (for each adjective: the word and its usage in the model),
- semantic rules (for each rule: a name of the rule for short and an explanation) and
- unit system (the set of units that should be used in the model).

The acronyms' information will be used in naming conventions and are going to be checked automatically that the acronym is used always in uppercase as part of a longer name in the presence of the data dictionary service.

The adjectives' information is intended to help in naming conventions. Modellers should check the data dictionary website information when selecting a name for a variable in order to use adjectives consistently (examples: total, final, required, …).

The semantic rules information is intended to capture domain knowledge that should be preserved or respected in the IAM. It is suggested that semantic rules should be captured as possible in Vensim using "Reality check".

The unit system of WILIAM model is available in data dictionary. Modellers should check the units of the system before include it in one symbol.

As described in Appendix D, we propose a protocol on the data dictionary based on modules and roles on modules. Each modification automatic by the injection system or manual (insert, update, delete) in the data dictionary should be validated. Users should be authorized to modify the data dictionary according to the following organization:

- All users should be identified on the website that is governing the data dictionary.
- Each user has roles. The role determines the operations the user can accomplish.
- There are three different roles: project general supervisor, module supervisor, module programmer
- **Module programmer:** a user can be designated as module programmer regarding one or more modules. Module programmers can insert and update information in the data dictionary of symbols (variables…) related to the module they authorized on. Each module can have many associated programmers. Any modification accomplished by programmers should be validated by a supervisor. Nevertheless, it is preferable not to accomplish manual insertions or updates but let the injection service do this job automatically from the model.
- **Module supervisor:** a user can be designated as module supervisor regarding a particular module. The module supervisor can (and should) validate any information that is still pending to be validated. The module supervisor can do any modification the programmers can do. In addition, the module supervisor can delete information. There is just one module supervisor for each module. It is recommended that module supervisors delete symbols that should be updated from the model to allow the automatic injection detect from the model that the symbol information is new to the data dictionary and automatically inject it. A utility to detect discrepancies between the model and the data dictionary information is delivered as described in Appendix E.
- **Project general supervisor:** a user can be designated as project general supervisor. It is possible in a project to have more than one general supervisor. General supervisors can do any task that module supervisors can do, they are authorized to all modules.
- The organization should guarantee that there is at least one general supervisor, at least a module supervisor per module, and at least one programmer per module.

The governance of the data dictionary should proceed according to this protocol:

1. Information is introduced usually by automatic injection from the model when pushing changes to the gitlab repo or, occasionally, by manual usage of the data dictionary webapp by module programmers. Any modification remains pending of validation.
2. A module supervisor or a project general supervisor can review the information in the data dictionary and validate it if possible. Supervisors can edit the information in order to complete it, adjust it and improve its description.

In the previous protocol the step 1 is preferred to be accomplish by automatic injection from Vensim programs (.mdl files). This automatic injection, as long as any manual insertion through the data dictionary webapp, remains pending of validation (step 2).

The data dictionary website and related API services developed by CRES are currently available. The link for the data dictionary website is http://www.cres.gr/LOCOMOTION_client/faces/index.xhtml.

The automation of quality assurance tasks will check that each Vensim program is consistent with the already validated information on the data dictionary (see Appendix E).

Appendix D describes more details regarding LOCOMOTION data dictionary.

In addition, a manual is available in Alfresco with a description of the functionality of the data dictionary website:

https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/3602fd72-347c-4608-8366-9fd59aa90609

4.2. NAMING CONVENTIONS

As stated above, the names of the current MEDEAS variables must be improved in WILIAM. The objectives of this process will be:

1. Adopt the most commonly used international nomenclature to be able to communicate the results of the project scientifically.
2. Adopt common criteria among all groups of WILIAM modellers to have a final model with common and consistent names, structures and style.
3. Improve program execution times.
4. Facilitate the integration of all modules.
5. Facilitate the communication with interface programs.
6. Facilitate the translation of Vensim to Python.
7. Obtain a model as understandable as possible for users.

4.2.1. SOURCES

Different sources have been considered to define naming conventions:

- Python naming conventions (in order to make translation from Vensim easier): https://www.python.org/dev/peps/pep-0008/#naming-conventions
- SMILE & XMILE naming conventions (in order to provide interoperability, useful for example for conversion form Stella/WORLD7). *SMILE and XMILE: A Common Language and InterchangeFormat for System Dynamics: http://docs.oasis-open.org/xmile/xmile/v1.0/os/xmile-v1.0-os.html*
- Previous work as part of MEDEAS Project: Internal document: starting from AEA proposal, enriching by debating with CREAF, taking into account recommendations from:
  - *IAMC http://www.globalchange.umd.edu/iamc/*
  - *IIASA https://data.ene.iiasa.ac.at/database/*

- Other sources consulted:
  - "System Dynamics Model Correctness Checklist" (Lai and Wahba, 2001)
  - "Best Practices for System Dynamics Model Design and Construction with Powersim Studio" (Malczynski, 2011)

4.2.2. KIND OF SYMBOLS

In this project we are going to distinguish a "conceptual" kind of symbols and a "programming language" kind of symbols.

The following "conceptual" kind of symbols are considered in Locomotion:[2]

- **Variable**: a symbolic name associated with a value and whose associated value may change.
- **Historical data series**: a scalar parameter estimated from historical data or an array of data points indexed in time order.
- **Model parameter**: a symbolic name associated with a value (typically an estimate) and whose associated value does not change over time neither with scenarios (fixed).
- **Constant**: fixed number which does not vary over time and is independent of the rest of symbols of the model (e.g., unit conversion, physical constants, etc.).
- **Scenario parameter**: parameter which is allowed to change in scenarios (including switches for selecting different options in scenarios).
- **Index**: name of the indexing symbol (corresponds with "subscripts" in Vensim).
- **Index case**: name of each element of an array (corresponds with the "subscript values" or "subscripts elements" in Vensim).
- **Switch**: a symbol which allows to enable/disable submodules or features (without including switches for selecting different options in scenarios).

Follow a couple of example of symbols' naming with relation to vector symbols to further clarity:

"total_regional_energy_consumption[REGIONS_I](t)"

- total_regional_energy_consumption is a **variable** since it varies over time
- "REGIONS_I" is **index**
- Let's assume that there are 2 regions, EUROPE and ASIA, so these would be **index elements**

"A_MATRIX[A_MATRIX_HISTORICAL_YEARS_I](t)"

- A_MATRIX is a **historical data**
- "A_MATRIX_HISTORICAL_YEARS_I" is **index**
- Let's assume that we have eleven historical data points, from 2005 to 2015, i.e., YEAR2005, YEAR2006, …, YEAR2015 being **index elements**

This "conceptual" kind of symbols are implemented with Vensim kind of symbols ("programming language" kind of symbols).

In this project we are going to distinguish, without loss of generality, the following Vensim kind of symbols:

- Variable (no matter if it is auxiliary or not), which can be subscripted or not
- Constant (no matter if it is unchangeable constant or not), which can be subscripted or not
- Subscript (also named in Vensim subscript range)

[2] There is another kind of input in MEDEAS-W which has not been considered here and which we could use more in LOCOMOTION: statistical distributions. See also the LOCOMOTION document "Guidelines to describe the uncertainty of the WILIAM input parameters_sept2020.docx", available at: https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/367ea132-a8f4-4829-8db4-15dbdab8e877.

- Subscript value (also named in Vensim subscript element)
- Lookup (also named in Vensim lookup table)
- Function
- Reality check
- Switches

The "conceptual" type of symbols should be always implemented in Vensim using items from the previous list.

The automatic quality checking tool and the data dictionary automatic injection service detect these types from the model in Vensim file .mdl and inject a programing language type for each symbol with the rest of the characteristics of the symbol. Module supervisors when validating the symbol should check this and should complete the semantic information matching the symbol with the conceptual type of symbol in the model (Project Types of Value). There are some matchings allowed, general supervisors should check and improve the allowed correspondence between programming language symbol types and project type of value.

4.2.3. SYMBOL NAMING CONVENTIONS RULES

Each one of the following naming conventions rules is marked with (A), (S) or (M). Having rules marked as (A) means that they are checked automatically, while (M) stands for manual. In the case of manual rules, module and general supervisors should check these conventions and programmers should be especially careful with respecting these rules when defining the name in Vensim programs or (rarely) in the data dictionary website. In between are semiautomatic (S) rules. In this case, a part of the rule is checked automatically, and the rest should be carefully applied by module programmers and validated by supervisors.

- Only British English terms should be used. (M)
- AND, OR, NOT, IF, THEN, ELSE, etc. the names of all built-in functions in Vensim and XMILE cannot be used as names (note that this does not prevent from using these words as part of a name, e.g., *CH4_and_N2O_radiative_forcing*). (A)
- Allowed characters in names: English alphabet letters [A-Z][a-z], numbers [0-9] and the character '_'. (A)
- Symbol names should be composed by any number of words separated by the character underbar "_" (cf. required Vensim configuration for properly working with underbars in section 4.4.1). (A)
- Symbols names must not start with a number. (A)
- All symbols should be defined for the group .wiliam (and not .control). .control symbols are only defined for those related with simulation control.
- Variable names should be lowercase, with words separated by underscores as necessary, (*e.g. cumulated_coal_extraction).* (A)
- Constants should be written in all capital letters, with underscores separating words, and this implies that the operation of two constants is also a constant and should also be written in capital letters (*e.g. EFFICIENCY_IMPROVEMENT_GAS_FOR_ELECTRICITY).* (A)
- Symbols names should not include the unit of the variable. Unit definition as part of the Vensim variable declaration should be used instead following LOCOMOTION unit standard (the unit system is defined in the data dictionary) and also should be included in the metadata description contained in the data dictionary. An exception will be made when symbols are converted also to other units (e.g., to other units which are customary such as for example oil extraction in Mb/d) should be written as the original symbol and the new units in lower case (*e.g., real_extraction_oil* and *real_extraction_oil_mbd* (M)

- Avoid magic numbers[3], define constants instead. Constants should be defined in the program as well as in the data dictionary. Exceptions are made for usual numbers which repetition is normal in a model, for powers, roots, percentages, etc. By default the exceptions are: 0;1;2;3;4;5;6;7;8;9;10;-1;100. (A)
- Numbers must be stored as numeric values in the excel files. Please do not store them as text, if you store them as text, Excel shows a warning about that. They can be converted to a numeric value by clicking in the warning.
- When symbols are representing values that define a particular scenario, they correspond to a scenario parameter as defined in section 4.2.2. Hence, their names should have "_SP" as suffix, and the rest of the name should be expressed according to constant name conventions (*e.g.: POPULATION_EVOLUTION_SP).* (S)
- Symbols used for unit conversion should be considered as constants and should start with "UNIT_CONVERSION" (*e.g. UNIT_CONVERSION_EJ_TWH").* (S)
- Acronyms should be used only for well-known jargon (e.g. GDP, EROI, RES, PV, TPES, etc.) or conventions such as chemical formulas (e.g., CO2, CH4, etc.). Acronyms should be defined in data dictionary (see section 4.1). In the presence of the data dictionary service, the acronyms can be capitalized in the middle of the name (*e.g. required_FED_all_sectors_by_fuel)*. When acronym is not present in data dictionary service, the rule for naming variable in all lowercase applies (*e.g. required_final_energy_demand_all_sectors_by_fuel)*. (A)
- Words should not be abbreviated (e.g. conv, min, max, dem...). (M)
- If possible, make the relevant word the first in the symbol name. For instance, the first word of the name of each variable should indicate the type of physical magnitude that they are, for example: energy, minerals, land, infrastructure, vehicle, share, policy, etc. (M)
- Words such as "final", "total", "real", "required", "potential", "available", etc. are adjectives that should be used consistently across all modules, should take conventions into account and should be explained in the data dictionary as it was previously mentioned (see section 4.1). (M)
- The name of the symbols might include other adjectives that show the role of the variable in the allocation of the model. For example, the final energy could be *final_energy_demanded* or *final_energy_consumed*. These adjectives should also be explained in the data dictionary as it was previously mentioned (see section 4.1). (M)
- Flows ("rates" in Vensim) will follow the general naming rules, being recommendable to build their name by juxtaposing "increase" or "decrease" to the name of their corresponding stock. This does not apply when the clarity in the physical meaning of the flow would be hampered, in the case that several flows enter or exit the same stock (as in *Evol_final_energy_intensity*), or the flow is an intermediary between two stocks (as in *land_arable_to_urban*).(M)
- Years should not be in the variable name, but in the description. The years for historical data should be indicated in the index elements (e.g., *"A_MATRIX_HISTORICAL_YEARS_I"* defined e.g., from *"YEAR2005"* to *"YEAR2015"*). Any exception (e.g., mention to a specific paper "AUTHOR_ET_AL_YEAR") should be defined in the data dictionary. The delay time must be defined from the index "*DELAY_TIME_I*". (M)
- Indexes (subscripts in Vensim software) should be clearly distinguished by naming convention. The arrayed symbol should be also distinguished from the index elements. (A)
    - Both should be considered as constants. Hence naming conventions for constants is applied.
    - All subscripts defined should include "_I" as suffix in order to show the Index concept (*e.g.:*

[3] The use of unnamed *magic numbers* in code obscures the developers' intent in choosing that number, increases opportunities for subtle errors (e.g. is every digit correct in 3.14159265358979323846 and is this equal to 3.14159?) and makes it more difficult for the program to be adapted and extended in the future. Replacing all significant magic numbers with named constants makes programs easier to read, understand and maintain.

  - a) *COUNTRIES_I,* index elements should be named as constants, for instance *FRANCE, SPAIN, …,*
  - b) *SCENARIOS_I*, index elements should be named as *MOST_LIKELY_SCENARIO, BUSINESS_AS_USUAL, …,*
  - c) *FE_I*, index elements *SOLIDS, LIQUIDS, ELECTRICITY…*).
- Working with arrays in Vensim can pose problems when the same index (subrange in Vensim) appears in more than one dimension of that array. This is the case for example in a squared matrix of dimension COUNTRY_I x COUNTRY_I. The solution for this is to create a new index with the same elements as a subrange of the original index. This way we have 2 indexes which only differ in their name. In practical terms, this is done by creating a subrange which has as elements the same subscript than the original index to be replicated and type "Subscript" and subtype "Equivalence" (cf. "Mapping Subscripts": https://www.vensim.com/documentation/21260.html). The copies should be named with the suffix _MAP_ before of the original suffix _I and after the name of the original, e.g.,: REGIONS_MAP_I <-> REGIONS_I.[4] The compliance with this rule avoids problems with automatic reordering by alphabetical order of indexes in the subscript menu. (A)
- Multidimensional variables/constants can be based on one or more indexes. Each index is a dimension in the vector variable/constant and should lead to the definition of a subscript and its subscripts elements as mentioned before. (S)
- For delayed variables, the delay time (e.g. 1 year, etc.) should not be included in the variable name but in the description. There are two different kinds of delayed variables: (a) those uses for econometric regressions, etc. typically delayed 1 year or 2 years; (b) those delayed on TIME STEP, typically used in system dynamics models to avoid algebraic loops. The name convention should be the usage of (a) delayed_ b) delayed_TS_ prefix to the variable name *e.g. delayed_name_of_the_variable, delayed_TS_name_of_other_variable.* (S)
- Naming of symbols initializing each stock will be built by adding "INITIAL_" to the name of the stock: "INITIAL_NAME_OF_THE_STOCK". (A) E.g.

```
INITIAL_POPULATION= 2000
  ~       people
  ~       Initial value from WorldBank in 1995.
  |

population= INTEG (
  population_variation,
          INITIAL_POPULATION)
  ~       people
  ~       Population projection.
  |
```

- Add prefix "IMV_" (inter-module variable) to those variables which identify the variables which are planned to connect different modules in the future, but while the autonomous development of a branch goes on are considered as exogenous for modelling convenience. In the final version of the model these variables must disappear (they should have all been endogenized during the model development process). The naming would be IMV_NAME_OF_VARIABLE (everything in caps lock since these variables will typically be exogenous while each branch is developed).(M)
- In Vensim programs: new functions should be named according to variable rules with some special cases:
  - *Functions as lookups: when defining a lookup table, it should be distinguished with suffix "lt" (e.g.: function "effect energy" defined as a lookup table should be named as effect_energy_lt).* (A)
- "Reality checks" (e.g.: *Population*>0) should be encouraged in the Vensim programs as validation for:

[4] Cf. dummy model for A matrixes for a practial example: https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/b392a8a6-98f0-4d6f-b803-a242536b02a6.

  - the inputs
  - the simulation domain/business rules
  - The names associated with "Reality checks" should be considered as function names with prefix "RC_" (*e.g.: RC_energy_production_input, RC_energy_conservation).* This should be explained in the data dictionary as part of what we call above in this document as Semantic Rules. (A)

- In the case that model structures are integrally taken (or with just minor modifications) from another existing model, the name of the original symbol will be reported in the comment of the symbol in Vensim code together with the model version used as reference to encourage traceability. (M)
- Add prefix "check_" to make a comparison between endogenous variables of your own module, or with some exogenous parameters (e.g., HISTORICAL_DATA), or with variables (endogenous or exogenous) from other modules. In addition, make sure that the name of the new variable "check_xxxx" (e.g., check_comparison_between_calculated_land_products_demanded_for_food) must be written in red because it must be removed before the public release of Wiliam model. (M)
- There are 3 types of configurable scenarios in the Data Dictionary: two for the Model Analyzer and one for the Model Explorer:
  Model Analyzer scenario: Accepts any variables and scenario parameters, regardless of their dimensions and size. That being said, to simplify the user experience, adding big matrices as scenario parameters is discouraged ---> Example from Economy module: LABOUR_PRODUCTIVITY_VARIATION_SP[REGIONS_35_I, SECTORS_I].
  Simplified Model Analyzer scenario: Accepts any variables and scenario parameters, but they should ideally have smaller dimensions (e.g., aggregated) than those of the Model Analyzer (scalar, vector or small matrix) ---> Example from Economy module: LABOUR_PRODUCTIVITY_VARIATION_1S_SP[REGIONS_35_I].
  Model Explorer scenario: only accepts 0-dimensional scenario parameters (scalars) ---> Example from Economy module: LABOUR_PRODUCTIVITY_VARIATION_1R_1S_SP.

  If a Scenario parameter is already a small matrix, vector or a scalar, it can be defined as a Scenario parameter directly in the Data Dictionary, without any additional required change in the model.

  If a Scenario parameter with many dimensions (large matrix) is to be included in more than one of the 3 scenarios, it may only maintain its original shape (dimensions) for the "Model Analyzer scenario", while some (or all) of its dimensions will need to be aggregated to be used in the "Simplified Model Analyzer scenario" and "Model Explorer scenario". To do that, include the "SIMPLIFIED_MODEL_NAME_OF_POLICY_SP" parameter in the equations of the specific scenario parameters as explained hereafter (e.g., SIMPLIFIED_MODEL_CAPITAL_PRODUCTIVITY_VARIATION_SP).

  The SIMPLIFIED_MODEL_NAME_OF_POLICY_SP parameter in the model (Vensim) can take the values of 0, 1, or 2, which have the following effect:

  SIMPLIFIED_MODEL_NAME_OF_POLICY_SP = 0 -> Scenario parameter may be included in the Model Analyzer scenario (if the Model Analyzer checkbox is checked in the Data Client).
  SIMPLIFIED_MODEL_NAME_OF_POLICY_SP = 1 -> Scenario parameter will be included in the Simplified Model Analyzer scenario (if the Simplified Model Analyzer checkbox is checked in the Data Client).
  SIMPLIFIED_MODEL_NAME_OF_POLICY_SP = 2 -> Scenario parameter may be included in the Model Explorer scenario (if the Simplified Model Analyzer checkbox is checked in the Data Client).

  The SIMPLIFIED_MODEL_NAME_OF_POLICY_SP has the following specific use cases:

* **If the scenario parameter is already a scalar** (without dimensions), it can already be used in the three scenarios (Explorer, Analyzer and Simplified Analyzer), so there is no need to add the SIMPLIFIED_MODEL_NAME_OF_POLICY_SP parameter in the expression.
*** If the scenario parameter must be simplified just for the Model explorer scenario** (the Model Analyzer and the Simplified Model Analyzer scenarios will use the default parameter dimensions), then you must use Vensim's IF_THEN_ELSE function, with the following two branches:
IF SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=0 OR SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=1, read the Scenario parameter with default dimensions.
ELSE (case SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=2), read the scalar (remember that the Model Explorer scenario only accepts 0-dimensional parameters) scenario parameter from a different location (using GET DIRECT CONSTANT).
*** If the scenario parameter is a large matrix, and we wish to include this parameter in the 3 different scenarios**, then use the IF_THEN_ELSE function, with the following 3 branches:
IF SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=0, then read the Scenario parameter with default dimensions,
ELSE, IF SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=1, then read the Scenario parameter with one or more aggregated dimensions (this will make it available for the Simplified Model analyzer scenario),
ELSE (SIMPLIFIED_MODEL_NAME_OF_POLICY_SP=2) then read the Scenario parameter with 0 dimensions (this will make it available to the Model Explorer scenario).

An example of this workflow is available in the WILIAM model for variable "annual_capital_productivity_variation".

Note that the parameters that need to be included in the scenarios of the Data Dictionary for are those that contain the GET_DIRECT_CONSTANT, and not the main variable. So, in the "annual_capital_productivity_variation" example, in the Data Dictionary we would:

1. Add "CAPITAL_PRODUCTIVITY_VARIATION_HISTORIC" as Scenario parameter and check the Model Analyzer checkbox (Note that this example was not the simplest one to get the concept, because it could also be "CAPITAL_PRODUCTIVITY_VARIATION_SP, if the default value for SELECT_CAPITAL_PRODUCTIVITY_VARIATION_SP=2.)
2. Add "CAPITAL_PRODUCTIVITY_VARIATION_1R_1S_SP" and check only the Simplified Model Analyzer checkbox for parameter (note from the parameter name that regions and sectors are aggregated).

- Add prefix "SELECT_" when its necessary to choose between several choices, for example:
The parameter "SELECT_POLICY_DIET_PATTERNS_SP" from "diets and land products demand submodule", in this scenario parameter we have to choose between four choices, IF SELECT_POLICY_DIET_PATTERNS_SP = 0 ---> APPLICATION OF DIET PATTERN OF POLICY DIETS 1, IF SELECT_POLICY_DIET_PATTERNS_SP = 1 ---> APPLICATION OF DIET PATTERN OF POLICY DIETS 2, IF SELECT_POLICY_DIET_PATTERNS_SP = 2 ---> APPLICATION OF DIET PATTERN OF POLICY DIETS 3, and IF SELECT_POLICY_DIET_PATTERNS_SP = 3 ---> APPLICATION OF DIET PATTERN OF POLICY DIETS 4.
*Note that there is a big difference between "SELECT_" AND "SWITCH", the first syntax (SELECT_) just to choose one of the available choices, but the second syntax (Switch) just to activate or deactivate some policy parameters.
- Symbols which are shares: name them as share_xxxx and the units "dmnl" (remember there is a rule to avoid percentages).
- Add prefix "HISTORIC_XXXX" when parameters store historical data (*e.g.:* HISTORIC_DATA_OF_DIET_PATTERNS).
- Avoid using functions that are not programmed in the PSYD (for more details about the PSYD functions check the following link: https://pysd.readthedocs.io/en/master/python_api/functions.html).

4.2.4. AUTOMATIC CHECKING OF NAMING CONVENTION

Automatic checking of naming convention rules is achieved by means of SonarQube. SonarQube is a platform for automatic code review for quality assurance purposes. It can integrate with the organization workflow to enable continuous code inspection. SonarQube allows to analyse different languages and it can be enriched via plugins to include a new language. Quality assurance team have developed a plugin for SonarQube to analyse Vensim programs in its textual representation (as .mdl files). In order to have the Locomotion SonarQube with this Vensim code analysis plugin available, we have installed, configured and deployed an on-premises instance of SonarQube at:

https://sonarqube-locomotion.infor.uva.es

As part of the rules that are going to be checked automatically by this SonarQube-based Vensim automatic code reviewer, a connection to the project's data dictionary website is available. The tool checks whether the symbols in Vensim programs (variables, constants, …) are defined and validated in the data dictionary as well if the symbol's characteristics in the Vensim program are not consistent with its definition in the data dictionary. Quality control system works better if a Check Model is done first (Model>Check Model in Vensim.)

Quality issues are created in SonarQube server. The Figure 2 shows an example of the SonarQube interface. Each issue is categorized according to the violated rule. A severity value is given to each rule. The interface allows to filter issues by severity and violated rule. Clicking on the issue in the left-hand side menu, the interface will show the position on the Vensim program where the violation is. Clicking on the tag "See rule", an explanation of the violated rule is given as well as Noncompliant and Compliant code examples.

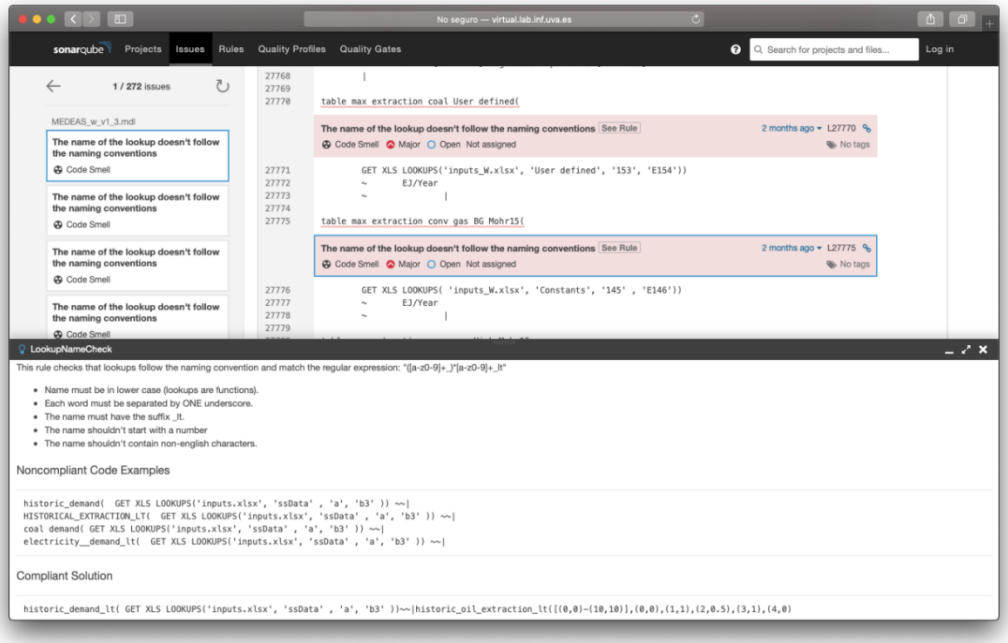


Figure 2: Screenshot of SonarQube interface.

More about Vensim plugin for SonarQube in Appendix E.

4.3. INPUT DATA FILES MANAGEMENT AND USAGE

MEDEAS models loaded all required data and scenario parameters for model running from a unique excel file (i.e., inputs_W.xlsx for MEDEAS-W and inputs_EU.xlsx for MEDEAS-EU). Working with excel files as data storage

offers to the modellers a convenient interface to easily add, modify and remove data. Organized in tabs and colour-coded, the experience of MEDEAS has also proved to serve as an intuitive and relatively simple interface for non-technical general users of the models. However, the scheme from MEDEAS had to be adapted to the LOCOMOTION context taking into account the following aspects:

- The evolution from single-region to multiregional model substantially increases the amount of data to be handled simultaneously (MEDEAS regional models were run in a one-way process, not multiregional).
- The increase in the number of modules.
- The need of coordination of modellers within each module and among the different modules to share the stored data.
- The importation of data from Excel to Vensim programs by field name instead by cell reference, as it was done until now. This enhances substantially the flexibility of the organization of data in the excel tabs and helps minimizing accidental errors when setting the data ranges. When moving data to a different tab or to a different excel check the "Name Manager" in order to avoid potential problems.
- A more systematic approach towards vectorization of the variables allowing a more compact and rational organization of the data stored in the excels.
- The need to maintaining the naming convention of the symbols and its inclusion in the data dictionary.

Because of all this, we must define a procedure to create, modify and use the data contained in the excel files in a coordinated manner. Hence, the current unique "inputs.xlxs" file has been split in different .xlsx input files. The proposed approach is flexible and takes into account the specificities of each module in terms of data volume and type. The data of all the modules will be stored during the phase of model development as follows:

1. One folder by module, and inside it, preferably one excel file by module and one tab per category.[5] However, each module will be free to skip this rule in the case of not being functional and include several Excel files in their corresponding module. These module folders are inside a general folder named "model_parameters" and inside it each module folder will be named by the same name of the module
2. One excel file for scenario parameters to run a simulation allowing to activate/consider different hypotheses and policies/policy targets. This excel file have different tabs, the first tab "ReadMe" is to have a general idea on the excel, the second tab "list_policies_hypotheses" where you can find a list of all policies and hypothesis used in each module of Wiliam model, the rest of the tabs are per module (demography, society, economy, finance, energy, energy-transport, land_and_water, climate, and materials) to better structure data and facilitate version control during model development. This file is located in a separate folder "scenario_parameters" in order to facilitate the storing of different scenario configuration files.
3. The general folder "model_parameters" also includes one common excel file including constants and conversion factors.

This file management and usage method has been designed to ease coordination and collaborative work during the model development phase. However, different possibilities for the final released version –to be decided in

[5] The following reasons to organize excel input files one tab per category (instead of for example one tab per region) were identified:

- Minimize data manipulation: data are typically downloaded from databases/manipulated for all regions at the same time
- Minimize the number of Vensim importing functions:
  - data for the 35 regions can be loaded at the same time with 1 unique Vensim function profiting of the index "REGIONS_I"
  - data which are the same for all regions are loaded just once
- Facilitates distributed programming and version control of the excel files: given that each branch will be more likely focused in 1 or a reduced number of categories but always all regions.
- Facilitates the expansion from 9 to 35 regions

the future- may exist targeting rather the usability of the model by final users, such as for example reading from a structured database or creating input files with parameters by region and one tab per module.

This mean that, e.g., assuming 9 modules (see section 6.2.1), a minimum of 11 input .xlsx files should be handled in the model development process. This way, the expansion from 9 to 36 geographical units would not increase the number of input .xlsx files.

Notes:

- There is an additional xlsx input file "sable.xlsx" which stores dummy variables in order to communicate Vensim with the SABLE developer software which allows to build interfaces. This excel should be ignored during WILIAM development.
- Indexes should be loaded directly in Vensim avoiding to use the tool "GET_DIRECT_SUBSCRIPT" (cf. https://www.vensim.com/documentation/fn_get_direct_subscript.html) given that it makes more complex and less efficient the automatic control (SonarQube) and because the Data Dictionary already stores that information, automatically injected from models, validated and shared with all modellers using the web client developed at CRES http://www.cres.gr/LOCOMOTION_client/.

Figure 3 shows an example of uploading of data from an inputs .xlsx file into Vensim following a vectorial structure.

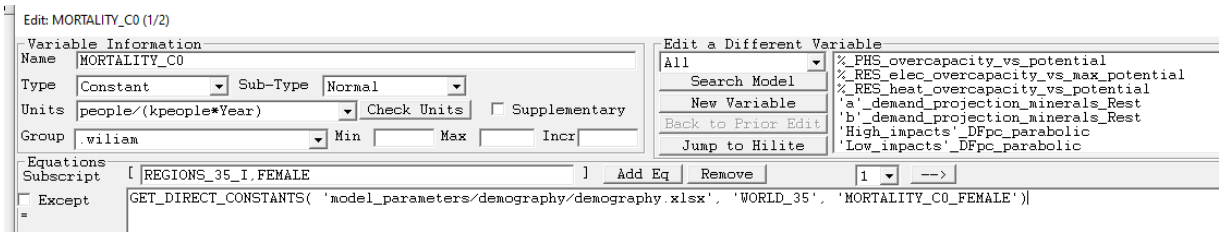


Figure 3: Example of uploading of data from an inputs .xlsx file into Vensim following a vectorial structure.

4.3.1. TYPES OF INPUT DATA FILES

Three types of input data files are used: for constants and conversion factors common to all modules, for module input data and for scenario parameters. Rules for the naming of these files have also been defined.

4.3.1.1. CONSTANTS AND CONVERSION FACTORS EXCEL

This file includes a table with unit base conversion factors between different multiples of the same unit of measurement (e.g., conversion from tera to kilo or exa to mega) as well as equivalence conversion factors between different units of measurement (e.g., conversion from TWh to EJ).

This common file for all modules allows to standardize the constants and conversion factors used in the whole model, and as such it will be coordinated by WP2 and WP9. This file is named "constants.xlsx".

4.3.1.2. MODULE INPUT DATA EXCELS

There is a folder by module and all the input data files specifically used by this module will have to be located in that folder. By default, there will be an excel file per module (see section 6.2.1 for modules to be considered in WILIAM). However, each module will be free to skip this rule in the case of not being functional and include several Excel files in their corresponding module. As expressed in the online WP9 workshop hold in May 2020, this could happen for at least 3 reasons:

- Large data size (in the case excel cannot handle such an amount of data in one unique file)

- Global data without regional disaggregation (e.g., Climate module)
- Common data shared by all regions (e.g., data on international trade)

Each module excel file will have a tab per category in order to better structure the information region and maybe some others needed for a better understanding of the rest of the file (index, content explanation, contact data of creators, etc.). Regional data will be collated all together in vectorial and/or matricial data in order to ease to loading of data during model development. In the case that all the parameters from a module cannot be located in the same .xlsx input file, then it is allowed to create new .xlsx files in the module folder (one per category, being each tab a sub-category).

These files will be named taking as reference the name of the module "moduleX.xlsx" (e.g., economy.xlsx or energy.xlsx), and the excel tabs will be structured following the name of categories of each module. If there is a need to store the data of a module in different input data .xlsx files, the data will be structured by categories as follows "moduleX-categoryY.xlsx" and the tabs will be structured following the name of subcategories. This naming is compatible with the naming of Vensim views in order to facilitate the automatic processing of model parameters in the Vensim model to assist in the Software Control Management.

4.3.1.3. SCENARIO PARAMETERS EXCEL

This excel file includes all the hypotheses and policies/policy targets available to perform a simulation with WILIAM. Each module developing team is responsible for feeding it with the relevant hypotheses and represented policies/policy targets related to their module. This excel file will be coordinated by WP8 but each modeler will have the right to edit the tab of "list_policies_hypotheses" and also the tab corresponding to the policies related their respective module.

This file will be named "scenario_parameters.xlsx".

4.3.2. METHOD FOR ADDING NEW DATA TO THE INPUT DATA FILES

Each excel file will be under the responsibility of the module developers which have created it. An excel file will be readable by all modellers of all modules, but writable only by the team of the owner module who will be the only ones that will create, modify and remove data on it. With two exceptions: constants.xlsx and scenario_parameters.xlsx will be writable for all modules, while for the latter each module will be responsible for its respective tab. General coordinators will need to ultimately assure the consistency of both files.

The data in the excel spreadsheets will be referenced by the same name they have both in the Vensim code and the data dictionary (see section 4.1). Naming convention rules set in section 4.2 have been developed taking into account the possibility to easily name multiple cells in excel by their row and column headings through the option "Create Names from Selection". Note that adding the symbol * to the field name in Vensim allows also to read data from excel in columns:

GET_DIRECT_CONSTANTS('model_parameters/materials/materials.xlsx','World', 'current_resources')*

This way, the same name should be used as reference in 3 places: name of the symbol in Vensim, name in the input data excel and name of the field name (cf. Figure 3). This approach will allow to significantly automatize, speed up and reduce errors in the process of loading data from the input xlsx files to the code in .mdl.

The last version of all excel files are stored in gitlab repository and from there the modellers can download a local copy of them to work with. The integration of the data files should be done following the same process defined as the rest of the model in section 5.

In the case that a modeller needs a data that at the moment is not in any excel file of the modules he contributes to, two possibilities exist. The simplest case is that this data belongs to one of the modules he contributes, so he can create it following the stated procedures. However, in case this data belongs to another module and is used

as an “input” in his module, he will have to (1) identify which module is responsible for its generation (consult the centralized data dictionary and ask WP9 in case of doubts) and (2) request and discuss with the corresponding module supervisor the creation of a new symbol in the DB related to this data. As this could take some time, to not stop the modelling process temporary data can be created and used while the request is approved. This local and temporary variable will be overwritten when the new version of files would be download from the repository.

All the main aspects of this process are shown in Figure 4.

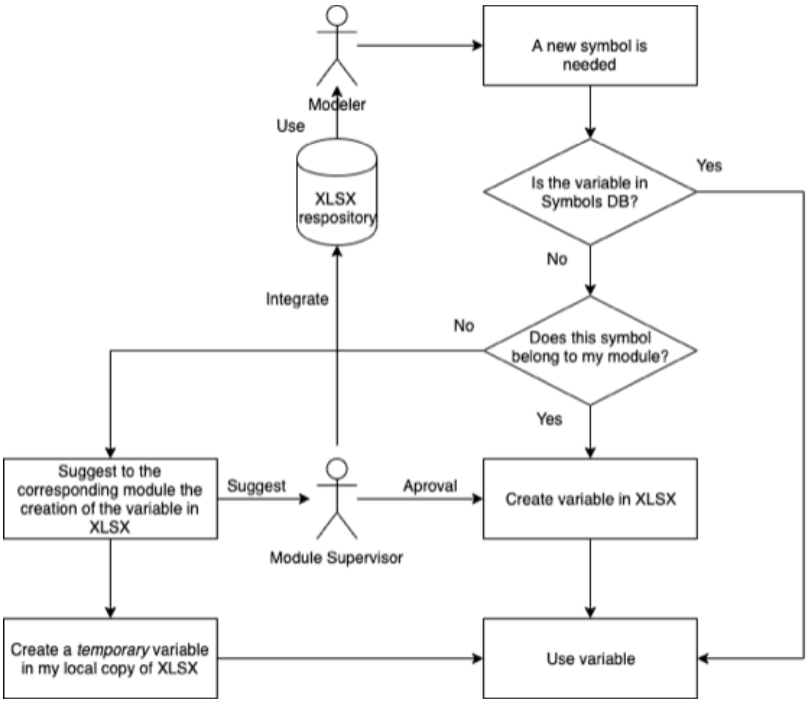


Figure 4: Flow diagram of new data and associated symbol creation and usage process

4.3.3. STRUCTURE OF DATA IN INPUT .XLSX FILES

Some standards are required to structure data in the input .xlsx files in order to have functional and homogenous data files across modules. The following aspects have been especially taken into account in their design considering both model developers and future users: compactness of information, simplicity for the data pre-processing and loading in Vensim, flexibility of the proposed structure and clear layout of the data stored. Different layouts have been considered considering the diversity of types of parameters to be used in the WILIAM model (switch, scalar, vector, function, matrix, etc.). Colours are proposed to be used in order to make the .xlsx files more visually intuitive. Note that during model development, particular cases for data storing may appear not explicitly having been considered here. In these cases, model developers should follow the proposed rules in order to achieve the same aforementioned goals:

- Always indicate the units of the parameter.
- Nomenclature:
  - the symbol "_" before a tag indicates that this tag cannot be modified, it should remain the same
  - each data table includes in the top-left-high hand corner a tag SELECT, SCALAR, VECTOR, FUNCTION or MATRIX to identify the type of data stored.

  - after the definition of the type of data, the symbol ":" indicates the name associated to that data stored. It is not compulsory to include a name (e.g., different parameters may be stored together)
  - between parentheses "()" is included the tag referring to the lines and columns (first columns and after lines)
  - Cell range names in the excel file must be well defined, they cannot start before the data, leave empty cells or cells that include units
- Layout: with a large amount of parameters the layout can be changed and the vectors/matrixes can be transposed.
- Colors (optional). Code: green for indexes and index elements, light orange for data and grey for switch options.

Find below the resulting proposed structures for: select, scalar, vector, function and arrays of ≥ 2 dimensions described in a generic way. An excel file with these same generic illustrated with examples is attached to this report ("input data xlsx structuring WILIAM.xlsx", also available in Alfresco D9.1 folder) including notes. Note that the numbers included in that tables are fictitious and included just for illustrative purposes.

Select between ≥2 different options:

SELECT_:NAME_OF_SELECT	
_VALUE_OPTIONS|Dmnl	_OPTIONS
0	Name of option 1
1	Name of option 2
i	Name of option i
n	Name of option n
_VALUE_SELECTED_OPTION	2

Parameter(s) common to all indexes and belonging to the same category:

_SCALAR(PARAMETER)	PARAMETER_a	PARAMETER_b	PARAMETER_j	PARAMETER_z
UNIT	EJ	$	m^2	%
_VALUE	25	10	…	15

Parameter(s) vectorized for the same dimension

_VECTOR(PARAMETER,INDEX)	PARAMETER_a	PARAMETER_b	PARAMETER_j	PARAMETER_z
INDEX|UNIT	EJ	$	m^2	%
INDEX_ELEMENT_1	25	10	…	15
INDEX_ELEMENT_2	25	10	…	15
INDEX_ELEMENT_i	…	…	…	…
INDEX_ELEMENT_n	25	10	…	15

Array of 2 dimensions:

_MATRIX:NAME_VECTOR(INDEX_COLUMNS,INDEX_LINES)				
INDEX_ROWS|UNIT	INDEX_ELEMENT_COLUMN_1	INDEX_ELEMENT_COLUMN_2	INDEX_ELEMENT_COLUMN_j	INDEX_ELEMENT_COLUMN_m
INDEX_ELEMENT_ROW_1	10	11	…	10
INDEX_ELEMENT_ROW_2	20	21	…	20
INDEX_ELEMENT_ROW_i	…	…	…	…
INDEX_ELEMENT_ROW_n	40	41	…	40

Vectorization (function of two variables). Case for y=f(x):

_FUNCTION:NAME_OF_FUNCTION(name_of_variable_y=f(name_of_variable_x))	_UNIT												
name_of_variable_x	UNITS_x	50	51	52	53	54	55	56	57	58	59	60	61
name_of_variable_y	UNITS_y	60	61	62	63	64	65	66	67	68	69	70	71

Matrix of 3 dimensions:

_MATRIX:NAME_VECTOR(INDEX_ROWS1,INDEX_ROWS2,INDEX_COLUMNS)					
UNIT					
INDEX_ROWS1	INDEX_ROWS2	INDEX_ELEMENT_COLUMN_1	INDEX_ELEMENT_COLUMN_2	INDEX_ELEMENT_COLUMN_i	INDEX_ELEMENT_COLUMN_n
INDEX_ELEMENT_ROW1_1	INDEX_ELEMENT_ROW2_1	10	11	…	10
INDEX_ELEMENT_ROW1_2	INDEX_ELEMENT_ROW2_2	20	21	…	20
INDEX_ELEMENT_ROW1_i	INDEX_ELEMENT_ROW2_i	…	…	…	…
INDEX_ELEMENT_ROW1_n	INDEX_ELEMENT_ROW2_n	40	41	…	40

4.3.4. LOADING MULTIDIMENSIONAL ARRAYS WITH >2 DIMENSIONS IN VENSIM

Vensim functions to directly load .xlsx data doesn´t allow to load at once data with more than 2 dimensions. This is an issue since programming with many subscripts will make likely the need to load multidimensional data, e.g., size of some arrays already implemented in the model: REGIONS_I: 9/35; SECTORS_I: 62; PROCESS_TRANSFORMATION: 43; AGE_COHORTS: 18; FINAL_ENERGY_I: 8, etc. Without a proper management, this would require the loading of data by parts, requiring a very intensive manual work for both defining cellranges in Excel and setting Vensim functions.

In this context a research was launched in order to identify potential solutions. Two main options were identified: maintaining Excel reading or switching to .vdf (Vensim Data Files) files. Proposals for loading data in WILIAM. Addressing identified issues:

4.3.4.1. PROPOSALS FOR LOADING DATA IN WILIAM. ADRESSING IDENTIFIED ISSUES:

4.3.4.1.1. MAINTAINING EXCEL READING

a. Vensim excel functions are limited to load up to 2-dimension data, and therefore data with higher dimensions have to be chopped into as many equations as the total number of combinations generated by the added extra dimensions, which involves a lot of manual work, possible solution:
    i. Automatize the generation of Vensim code (in .mdl) from the definition of the subscripts, the name of the variable, the Excel file and the name of the Sheet to generate the required set of equations. Automation can be done with extra columns inserted into excel that get the necessary data from excel data and generate the functions that can then be copied directly into the Vensim code, or with an excel macro you can further automate this process by adding a button. An excel example has already been made with the extra columns with the data reading functions and a Vensim model that simulates with these functions, both of them will be attached with this document. This will require the modification of the current .xlsx template data loading file (“input data xlsx structuring WILIAM.xlsx”, available in Alfresco D9.1 folder)
b. There are different choices to define the dimensions that will appear in the two-dimensional arrays and the dimensions that will generate the additional functions:
    i. Each case of loading of arrayed data could be studied in a case analysis, and then the subscripts with the shorter length would always be chosen to be the last one of the arrays e.g., Given 4 subscripts: Sub1, Sub2, Sub3, Sub4:
        1. We identify the variables with the largest number of index elements, e.g., let’s assume that they are Sub3 with 50 index elements and Sub4 with 70.
        2. If Sub1 has 9 and Sub2 has 5 index elements, then we would require 9x5=45 equations
    ii. Some index elements shall always appear in the first positions, e.g., REGIONS_I and SECTORS_I, and the rest of the index elements, according to their size, shall be placed in the following positions of the multidimensional variables.
c. In the case of economic matrix data, Matrix A for example is a particular case with an already proposed solution: https://ecm.cartif.es/share/page/site/locomotion/folder-details?nodeRef=workspace://SpacesStore/47870dba-3777-45f2-8d45-2764eaf4d48d
d. A lot of manual work for defining the cellranges in excel and to define the Vensim loading data functions.

4.3.4.1.2. SWITCHING TO .VDF (VENSIM DATA FILES) FILES

a. All data must be stored in excel files, while a choice should be made for .vdf:

i. Convert all data to .vdf format
ii. Convert to .vdf only data with dimension>2

b. The vdf file needs a previous conversion from the excel input data.
   i. It is necessary to modify the excel input data in order to facilitate its conversion to vdf format
   ii. The conversion need some manual work with the Vensim conversion tool. With a script that includes de "XLS2VDF()" function this process can be automated to build the .vdf from the Excel data, example done with this function runs without problems directly with the Vensim File-> Open model -> Command script (*.cmd) functionality. E.g., MENU>XLS2VDF|input_file.xlsx|input_file.vdf|conversion.frm|, the conversion.frm is a file that store the steps needed to the conversion, this file is made one time in the first conversion and no anymore.

c. Need additional work to read the data in Python since it cannot read the propietary format .vdf
   i. Use of automatic translation with the help of an improved version of the pySD library.
   ii. Automate the function generation needed to switch from reading from vdf to reading from Excel before translating to python.
      1. Functions to load vdf data : GET_VDF_DATA('file','varname',start,slope) https://www.vensim.com/documentation/fn_get_vdf_data.html?q=GET+VDF+DATA
      2. Functions to load xls data: GET_DIRECT_DATA('file','tab','time_index', 'range_varname') https://www.vensim.com/documentation/fn_get_direct_data.html?q=GET+DIRECT+DATA
      3. The automated script need to transform the GET_VDF_DATA() functions to GET_DIRECT_DATA() functions to allow Python code to load the excel input data.

4.3.4.2. ADVANTAGES AND DISADVANTAGES OF PREVIOUS SOLUCTIONS

b. Excel input data (*.xlsx file)
   i. Advantages
      1. Open format
      2. Current software version control system is linked with the excel files (*.xlsx)
      3. Compatible with current Python conversion method (pySD)
      4. Legible and easy to use format by both model developers and future users
   ii. Disadvantages
      1. Data limited to 2 dimensions, any extra dimension implies an increase in the number of arrays and therefore data with more than 2 dimensions have to be chopped into as many equations as the total number of combinations generated by the added extra dimensions (if an array has dimensions a, b, c, d, … and e, then the total of equations would be dim(c) x dim(d) x … x dim(e),
      2. Implies a lot of manual work in the excels (definition of excel range names)
      3. Implies a lot of work in the Vensim code functions.

c. Vensim data format input (*.vdf file)
   i. Advantages
      1. Up to 8-dimension data can be loaded with a unique function (minimize input errors and number of code lines)

2. No need of defining excel range names
3. Potential confusion to model users due to data being located in 2 places (but only be read by the model in 1 place) is minimized because. vdf files is a Vensim proprietary format that directed to store input and output data but cannot be edited.

ii. Disadvantages

1. Needs a previous conversion from the excel input data to the vdf data file.
2. Need additional work to read the data in Python since it cannot read the proprietary format .vdf
3. .vdf files cannot be checked by the version control.
4. Doesn´t work properly with the Vensim 8.X.X versions

4.3.4.3. ADOPTED SOLUTION: EXCELS2VENSIM PYTHON APPLICATION

The .vdf approach was discarded due to the number of disadvantages and the lack of experience with its use. In order to overcome the issues with the excel option, namely the fact that defining the cellranges in excel and the Vensim loading data functions require a lot of manual work, a Python application named Excels2Vensim was developed by CREAF with the purpose to automatize these tasks: https://excels2vensim.readthedocs.io/en/latest/.

To use the Excels2Vensim application follow instructions and see example videos at: https://ecm.cartif.es/share/page/site/locomotion/folder-details?nodeRef=workspace://SpacesStore/c07ab2ca-acb4-492b-b229-aba123c7b49e.

4.4. OTHER PROGRAMMING STANDARDS

As naming convention rules in symbols, each one of the following rules is marked with (A), (S) or (M). (see section 4.2.3)

4.4.1. VENSIM CONFIGURATION STANDARDS

- Due to the size of the model, it is necessary to use a Vensim version higher than 8.0 (Vensim versions 7. have memory issues with the new versions of the WILIAM model). We recommend from now to use a version that is higher than Vensim 9.2.
- Model -> Settings configuration (M):
    - INITIAL TIME: 2005
    - FINAL TIME: 2100 (any developer is free to test with any FINAL TIME horizon however when submitting it should be validated that the model runs properly and it is stable by 2100)
    - TIME STEP: 0.25
- The integration method is Euler.
- Symbols that are used for control should not be declared in the no control zone (group) and vice versa. The default control symbols are: TIME STEP, INITIAL TIME, FINAL TIME, SAVEPER. (A)
- Do not modify the name of the .mdl file in order to the SCM to work properly and find the differences between the same file before and after the changes. (M)
- Separation of words in symbol names by underbar ("_"): Vensim always automatically converts spaces in symbol names into "_"; the programmer can introduce the underbars manually to produce the same result. To avoid unintended errors, the programmer has to ensure that the following option in not selected in Vensim: (M)

  Model->Setting->Sketch->Use hard underbar (do not convert to space)

  Note that before doing a commit in the Gitlab (cf. section 5 and SCM tutorial), it is necessary to check that the following option is activated (save the model after any change):

  Tools -> Options -> Settings -> Variable display format: Show_underbar

- Loading of external data from .xlsx input files: the functions “GET DIRECT” (GET DIRECT DATA, GET DIRECT CONSTANT and GET DIRECT LOOKUPS, cf. https://www.vensim.com/documentation/ for further information) should be used in order instead “GET XLS” to avoid that all .xlsx input files are automatically opened by Vensim when simulating the model. This also allows to run the model in a OA which does not have Microsoft Excel installed. It has to be taken into account that the model cannot run if any of the .xlsx input files are open when executing it. (M)
- All variables loaded from Excel files through “GET_DIRECT” functions must be defined in the views corresponding to exogenous variables, which are located in the first views of the model and are classified according to the module, e.g. demography-exogenous_inputs.
- Using GET DIRECT functions do not include a '/' before the path, e.g.: /model_parameters/climate/climate.xlsx. This is quite confusing as in UNIX-based systems '/' is the root of the filesystem. Therefore, the leading '/' should be removed to represent the relative path, e.g.: model_parameters/climate/climate.xlsx.
- Each Vensim view name should indicate the module, category and subcategory to which the variables represented there belong. The structure will be “module-category.name_of_the_view”, with “name_of_the_view” corresponding to the subcategory, e.g., energy-electricity.demand. This standard is consistent with the naming of the data input .xlsx files, and will also help systematizing the classification of information in the data dictionary and also the SCM. (A)
- Categories and subcategories are unique, there cannot be two subcategories with the same name even though they belong to different categories. (A)
- The following new views must be added to each module views in Vensim: name_of_module-outputs (include all the most relevant variables from the module) and name_of_module-validation (include the validation process and also all the "check_" variables).
- Respect the existing inter-module (input/output) variables in WILIAM v0.1 communicating different modules. These relationships can be updated/modified but never deleted in order to preserve the internal coherence of the model. For the convenience of the modellers, these variables are marked in red colour in the Vensim code.
- WILIAM modularization (related with SWITCHES): all the modelers have to make an effort to modularize the model, so the model is able to run in a customized way without some submodules.
- The symbols in the Vensim model (.mdl file) that have a reference to an Excel file should have the same reference information stored in the data dictionary. The reference to the Excel file is automatically injected to the data dictionary. The symbols predefined in Vensim such as FINAL TIME, TIME STEP, etc. and functions are ignored (excepting lookups tables) (A).
- Do not include data which are not used by the model in the data loading excels. They take space and Vensim reads the full files. Include this as documentation or auxiliary files.
- Lookups tables data should not be embedded in the model but read from an Excel file (A), i.e. avoid

```
name_lt([(0,0)-
    (10,10)],(1985,1378.15),(1986,1422.43),(1987,1422.43),(1988,1422.43),(1989,1422.
    43))
```

use instead

```
name_lt(GET_DIRECT_LOOKUPS('file.xlsx', 'World', 'index', 'name'))
```

4.4.2. MODULARIZE WILIAM

We should make an effort to modularize the model, so the model is able to run in a customized way without some submodules.

WILIAM is a very large and complex model, composed of different modules and submodules related by many feedbacks. We already have a modularization from the semantic point of view given by the name structure of Vensim views (module-name_of_module.category) also reflected in the *data dictionary*. But there are several practical advantages to hard-code modularize the model:

- Development of the (sub)module and validation: due to the large amount of feedbacks in the full model, it can be very complex or even impossible to validate each (sub)module receiving dynamic information from the rest of the model, at least during first phases. For this, it is recommended to run and validate the (sub)module standalone.
- Eases the replacement of a (sub)module by an updated/different version, given that the variables linking the (sub)module with the rest of the module (inputs and outputs) are clearly identified when modularized (i.e., where the SWITCHES affect).
- Eases the comparison of results obtained by two different modules performing the same function and integrated in the model.
- Although Vensim has an in-built function for "Partial simulation" (https://www.vensim.com/documentation/ref_partial_simulation.html), this option is only valid when want to only simulate variables which are in the same view.

The solution is to implement SWITCHES across the module which allow to activate/deactivate different parts of the model. All the variables linking the (sub)module with the rest of the module (inputs and outputs) need to be identified. For these, the different These SWITCHES are implemented through IF THEN ELSE function which the following 2 options:

0: the (sub)module runs isolated from the rest of WILIAM, replacing inter(sub)module variables with exogenous parameters.

1: the (sub)module runs integrated with the rest of WILIAM.

For those variables which are inputs to the (sub)module, it is necessary to load data exogenously to be used when the SWITHCH takes the value 0. This data could be historic timeseries or static data.

In some cases, it may be useful to have two-levels of SWITCHES, i.e., one SWITCH can disconnect/connect at once several SWITCHES. This can be very useful in some cases, e.g., for different types of climate change impacts to be able to discriminate between one specific damage or all at the same time. The solution would be to create an endogenous SWITCH which will be named SWITCH_all.

4.4.3. CODE OPTIMIZATION IN VENSIM

Code optimization is any method of code modification to improve code quality and efficiency. A program may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer input/output operations.

Simulation time is a critical issue of WILIAM given the large size of the model. Follow a list of general and Vensim specific guidelines, as well as other options/ideas for further research identified to the moment.

- GENERAL GUIDELINES
  - Delete obsolete parts of code and unused data.
  - Vectorise all operations as much as possible.
  - Minimize the number of matrix operations (especially if the arrays are of large size and/or have more than 3 dimensions).

    - Load data in final units to avoid performing unit conversions in the model.
    - Avoid any unnecessary operation.
- SPECIFIC GUIDELINES FOR VENSIM
    - Use Vensim pre-defined functions instead of "manual" operations (i.e., reproducing the full sequence of program instructions explicitly in Vensim).
    - Minimize the use of the SUM () function.
    - Minimize the number of matrix operations (especially if the arrays are of large size and/or have more than 3 dimensions).
    - Minimize the size of the arrays matrices operations' (e.g., if the required output need to be aggregated, if possible do the aggregation before the operation).
    - Do not include data which are not used by the model in the data loading excels. They take space and Vensim reads the full files. Include this as documentation or auxiliary files.

4.4.4. UNITS CONSISTENCY IN WILIAM

- All symbols should be modelled with their corresponding units. By default an standard WILIAM units must to be used (see section 6.2.4 for unit agreement process in WILIAM). The data dictionary includes a part that defines this unit system. (A)
- Do not use the Vensim utility on synonyms of units that allows that in some places of the model different words are used for the same conceptual unit. It is proposed to be consistent with the unit declared for a symbol in the Vensim program as well as in the data dictionary. However, the Vensim utility for synonyms can be used for singular/plural of some units such as year/years or vehicle/vehicles (M)
- Always declare symbols that do not have an associated unit of measurement as dimensionless with Dmnl acronym, i.e. any symbol should have a declared unit from the unit system or "Dmnl". (A)
- Avoid the use of percentages and use shares instead (e.g., use 0.5 and not 50%) to avoid confusion with units. Percentages can be used for output variables. (M)
- Each unit conversion of different units of measurement will be done in 3 steps: (1) use "MATRIX_UNIT_PREFIXES" variable (this variable is located in constants view of the model) to adjust units in the base units, then (2) use the equivalence conversion factor (must be defined in constants.xlsx), and finally (3) use again "MATRIX_UNIT_PREFIXES" variable to adjust to the desired magnitude. E.g., to convert from EJ to TWh, first convert EJ in J ($1EJ=10^{18}J$), then J to Wh (1 Wh=3600 J) and then from Wh to TWh ($1TWh=10^{12}Wh$).
- Unit consistency must be always maintained during the modelling process. Module supervisors will be in charge of making respect this rule. The use of the Vensim tool "Units Check" (https://www.vensim.com/documentation/ref_units_check.html) is extremely recommended. Some typical cases:

    - Flows will always have as units the one of the stock they are connected to divided by year, e.g., if the stock has *kg* units the flows connected to it will be *kg/year*.

- When “MATRIX_UNIT_PREFIXES” variable is used in WILIAM, to maintain consistency in the units two new variables must be added for each conversion. E.g. if “MATRIX_UNIT_PREFIXES is using to convert J per EJ, the variables “UNIT_CONVERSION_J_EJ” and “J_per_EJ_units” must be added to the model in constants view (see Figure 5 below).

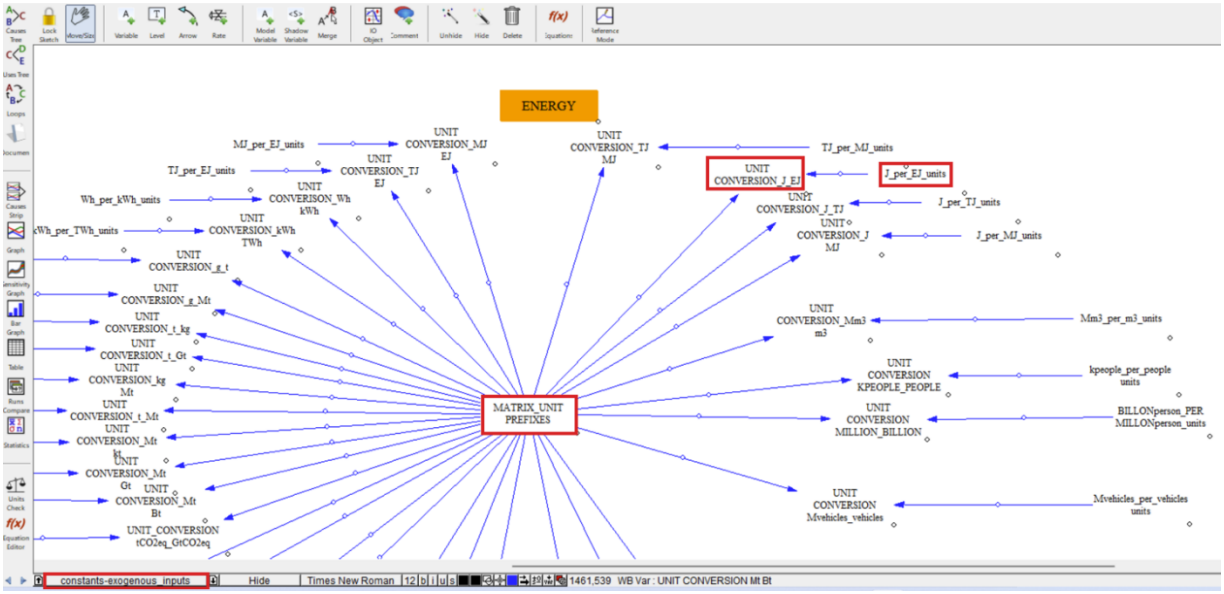


Figure 5: Constants-exogenous-inputs view in Wiliam model

4.4.5. COLOR LEGEND

There is a great heterogeneity which is confusing for the model developers (and future users). And to understand in a simple and effective way all types of variables used in each (sub)module, we have proposed the following color legend coding:

- Blue: Most important variables of each submodule (a figure should be included for each of these variables in each view).
- Red: Errors or things to finish (to be deleted before public release of the model).
- Orange: General or informative comments.
- Green: Variables from other modules.
- Pink: Policy variables.
- Brown: SELECT variables.
- Gray: Shadow variables.
- Purple Hexagon-shape: to (de)activate (sub)module switches which allow to run in an isolated way that (sub)module. These switches are activated with "1" and deactivated with "0". Only leave activated if the interconnections of the module have been validated. Nomenclature: SWITCH_NAME_OF_SUBMODULE

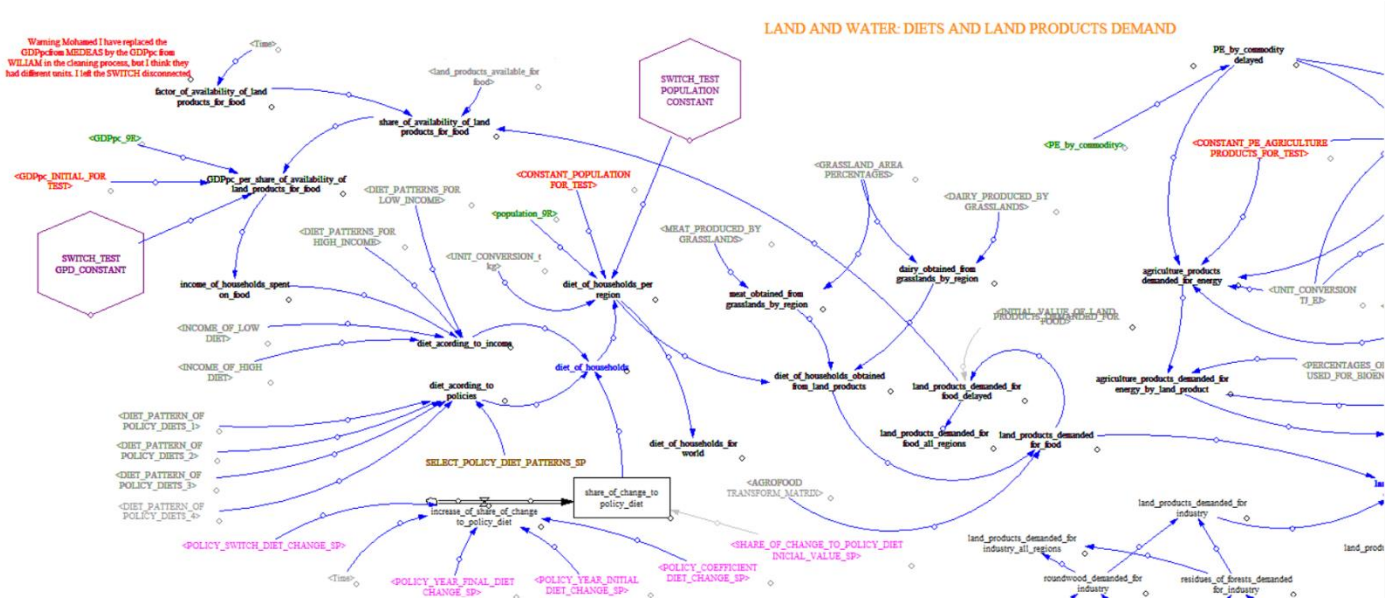


Figure 6 Diets and land products demand view in Wiliam model

Basic principles:

- Relate main variables and introduction of general comments with an easily identifiable color and shape code.
- The squared shapes are not allowed given that they could be confused with stocks.
- Include in each view figures showing the most relevant outputs of that. The most relevant outputs of the module should be located in name_of_module.outputs (Create figures also there).

5. SOFTWARE CONFIGURATION MANAGEMENT

Software Configuration Management (SCM) is defined as a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in Software Engineering. Several subprocesses and tasks are part of the SCM process such as version control, government of changes, project progress control, backup storages tasks, managing roles and responsibilities, audit and reviews, etc.

SCM processes and tasks require of tool support. Locomotion adopts Gitlab on-premises as SCM tool: https://gitlab-locomotion.infor.uva.es/develop

Login name, full name and email are necessary in order to create users of the SCM tool. Characters allowed are delimited to the English alphabet. Same users have been created simultaneously at sonarqube-locomotion.infor.uva.es. An updated list of the roles of users can be found here:

Note that if anybody wishes to modify his/her role in any module please mark it on the table and inform yania@infor.uva.es to perform the change in GitLab.

For each LOCOMOTION module (see section 6.2.1) a Gitlab project has been developed. Projects have been created with “internal” visibility level which means that users should be identified in the system in order to “see”. All users have Reporter permission on any project created at gitlab-locomotion server. Some users have permission to “modify” code. These users are designated with Developer role. Module and General supervisors will be designated with Maintainer role. Role permissions are summarized in the table available on: https://docs.gitlab.com/ee/user/permissions.html

A correspondence between gitlab roles and data dictionary roles is stablished:

a- Module Programmer as Developer role in a gitlab project for a module
b- Module Supervisor as Maintainer role in a gitlab project for a module
c- General Supervisor as Maintainer role in all gitlab projects and in the special project for wiliam integration.

Each project will have a Board or List of Issues. https://docs.gitlab.com/ee/user/project/issues/. Issues will describe a goal to be accomplish in the project and will have tasks. Issues can be assigned to project members. Project members can register each time the time they spend working on the tasks. Associated with each issue a branch in the version control will be created. Changes in branches can be merged in the main branch, called master, by approval using a Merge request (when requesting Merge request, the model should run in the standard settings (cf. 4.4.1. VENSIM CONFIGURATION STANDARDS), i.e., same TIME STEP, FINAL YEAR and integration method than in the master branch). In order to keep the repository simple and controlled, the proposed structure is to have:

1- A master branch or main trunk
2- A branch per issue and per developer from master, created from the issue tracker.

We recommend focusing on views in Vensim System when working in the changes tracked in each branch not used in other branches at the same time, in order to simplify merges from one branch to the other.

An additional project has been created solely as issue tracker to report potential questions and issues with the integrated data web dictionary (managed by CRES).

https://gitlab-locomotion.infor.uva.es/locomotion/data-dictionary

Not all branches related to issues will be merged in master. Developers can create issues and associated branches in order to research and tests some ideas. Whether the result should be merged with master is an important decision and should require colleagues and supervisors' reviews and approval. Master branch should contain a clean and functional version.

Finally, the projects of different modules will merge in a release of WILIAM model.

Releases should be tagged in the repository in order to easily find the releases made: https://docs.gitlab.com/ee/user/project/releases/

Releases can be associated with project Milestones:
https://docs.gitlab.com/ee/user/project/releases/#releases-associated-with-milestones

In order to keep track of changes in modules, .mdl text and Excel files should be submitted to Gitlab repository, check-in at the proper branch.

As a result of evaluating multiple tools, we propose the use of "gmaster" tool (https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/6c34fa86-f93c-4b1a-ae91-afa29885b096) as the local version control. However, vensim .mdl files have sections regarding visual representation. At this moment the problem of make easier differences visualization and making decisions on merges regarding visual representation is not fully solved. That is why we recommend working in different views per branch.

Nevertheless, a tool for making easier the visualization of differences between versions and aiding in merging tasks in gmaster has been developed. It is a Vensim plugin for SemanticMerge. SemanticMerge is a product developed by Codice Software also the gmaster creators, as part of their products suite, to endow version control with semantic information. Appendix F describes the installation and user manual of this plugin in gmaster.

Tracking changes regarding Excel files will be initially performed with the tool spreadsheetcompare but a further development of gmaster to integrate Excel files control is currently ongoing.

A tutorial describing step by step the use of the control version software and methods has been elaborated and distributed to all modelling partners (cf. https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/ad01b668-9d94-4f5f-9870-4b849e968049), as well as, a four parts video tutorial has been recorded for helping in the control version software usage (cf. https://ecm.cartif.es/share/page/site/locomotion/documentlibrary#filter=path%7C%2FWP09%2FCommon%2520modelling%2520framework%2520WILIAM%2FSCM%2520tools%2520and%2520Workflow%2520tutorials%2FVideo-tutorials%2520SCM%2520tools%7C&page=1). Moreover, online workshops have been organized in order to clarify points and solve questions.

Key aspects for the correct use of Vensim software before save a commit:

1. Check that the following option is selected:

   Tools > Options > Settings > Variable display format: Show_underbar.

(Vensim always automatically converts spaces in symbol names into "_". To avoid unintended errors, the programmer needs to ensure that always the following option is NOT selected in Vensim: ***Model->Setting->Sketch->Use hard underbar (do not convert to space)***.)

2. Check that the model is correct **Model>Check Model** (In the initial versions of the model WILIAM, a series of warnings appears when you perform Check Model. These refer to the fact that there are unused variables. In this phase, you must ignore the warnings and close the window with Close.
3. Order the equations using the option "Alphabetical by group" Depending on your Vensim version:
   Vensim 7. : **Check model->Reform and Clean->Alphabetical by Group**
   Vensim 8. : **Model-> Settings->File format->Eq. order: Alphabetical by Group**

   "The Model>Reform/Clean command allows you to reformat and reorder model equations (see Figure 7*). This is very useful if you are moving between the Equation Editor and Text Editor and want to put equations in alphabetical order or order them by group. It can also be a good way to document a model for archival purposes.*
   If you are storing a model in binary (.vmf) format and end up with a long list of units you are no longer using, or run into unexpected problems working with the model, you can use Reform/Clean to clean up any internal problems. If you are working with models in text (.mdl) format, this process (with no reordering or character set translation) is automatically performed each time you close and reopen the model.
   Equation Order determines the order of the equations in the Text Editor, and when saved in .mdl format."

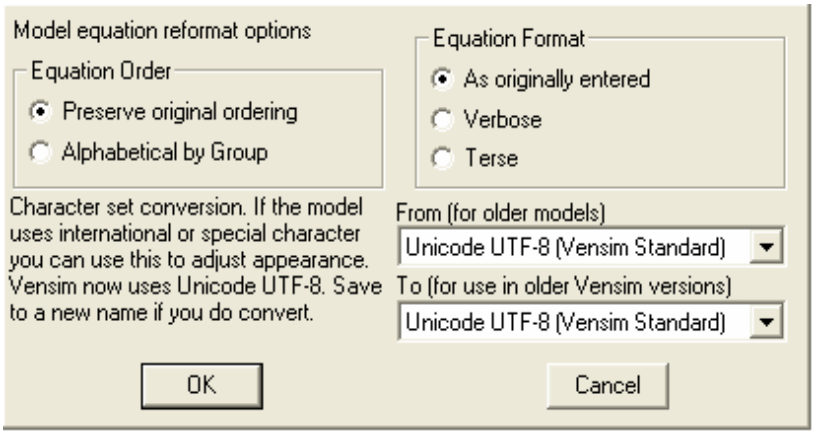


Figure 7: Screenshot of Vensim Model>Reform/Clean command.

5.1. SCM WORKFLOW SUMMARY

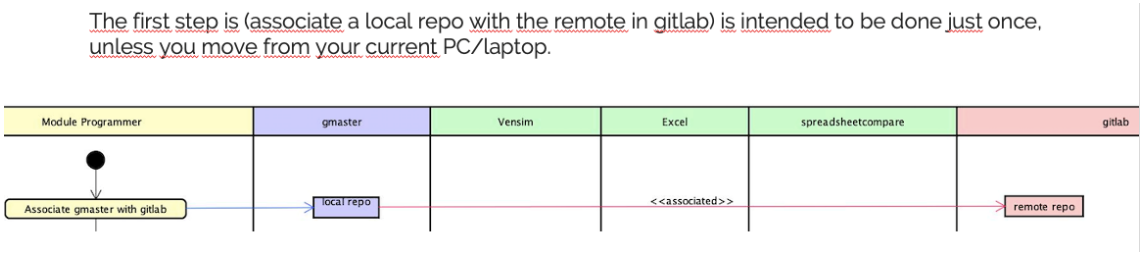
Every organization working with SCM version control should follow a workflow

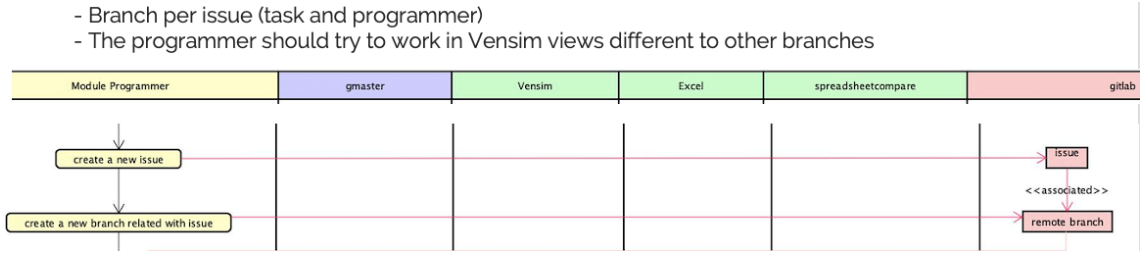
- There are very well known workflows (https://www.atlassian.com/git/tutorials/comparing-workflows):
  - Gitflow
  - Centralized
  - Feature per branch
  - Forking

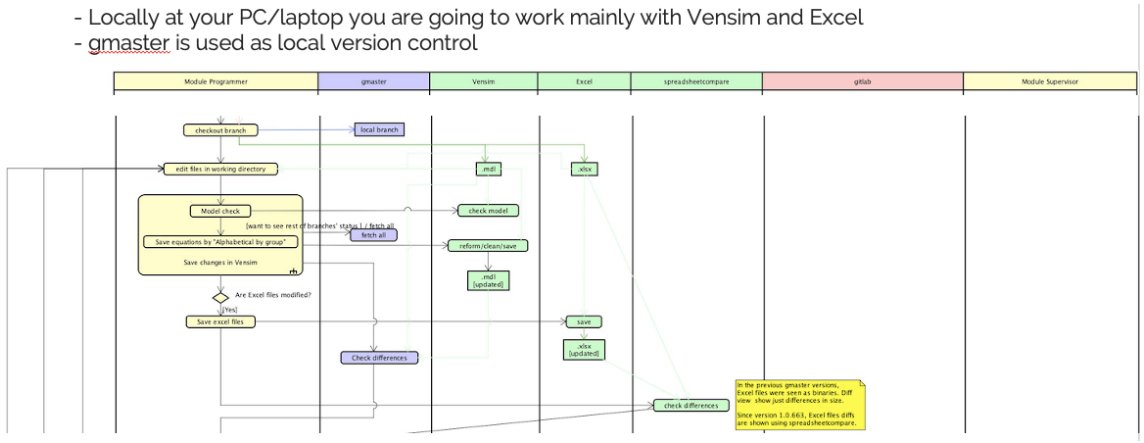
    - Task workflow (https://www.plasticscm.com/branch-per-task-guide/intro/the-task-workflow)
- Given our context we should
    - simplify
    - take into account the problem with Vensim views
- Having a gitlab repository (project in gitlab terminology) for each module
- Each repository (per module) starts from the WILIAM skeleton and excel files
- Each repository (per module) has a master branch (protected), main trunk or **the baseline**
    - Branch per issue (task and programmer)
    - The task and the programmer should try to work in Vensim views different to other branch per issue (task and programmer)
    - After the task is finished, merge request to master

Our workflow is described here: https://docu.cartif.es/share/s/bD4R7lxJTi-mtcYx_HfwkQ

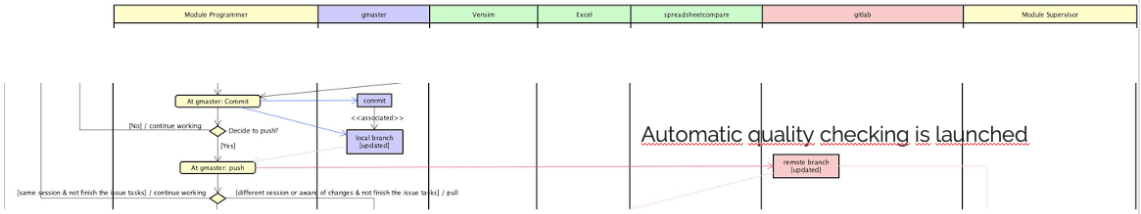
Next a graphical summary is shown:





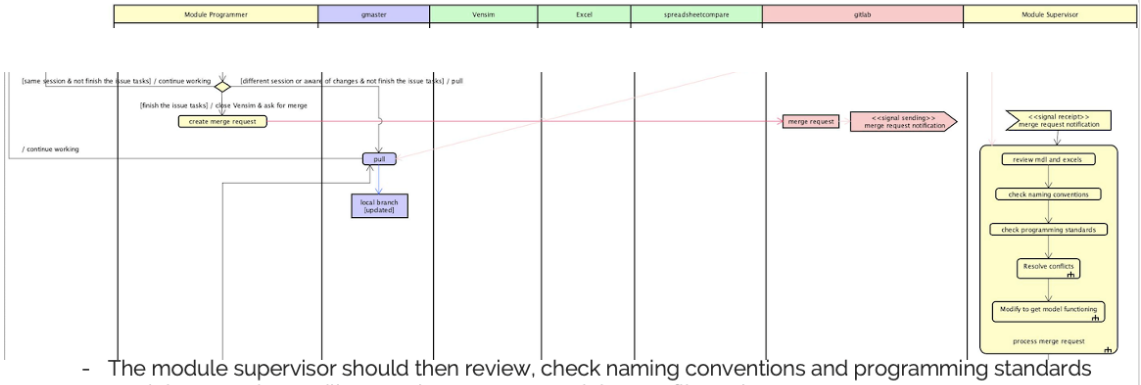


- Locally at your PC/laptop you are going to commit your changes in gmaster
- Occasionally you push your changes to the remote repo (gitlab).



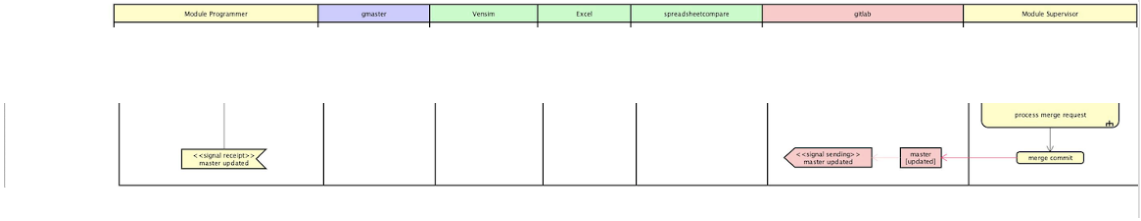
- A push to gitlab automatically launch the automatic quality checking
- All symbols that fulfill the automatically checked rules are injected to the Data Dictionary
- Symbols automatically injected to the Data Dictionary are declared as "Pending" of validation.
- Module Supervisors should complete/edit their information and validate the symbols.
- If any editing is required, programmers should be notified to change the Vensim program accordingly

- When you have a piece of work in your branch that can be integrated in master, create a merge request
- Merge requests are created in gitlab.

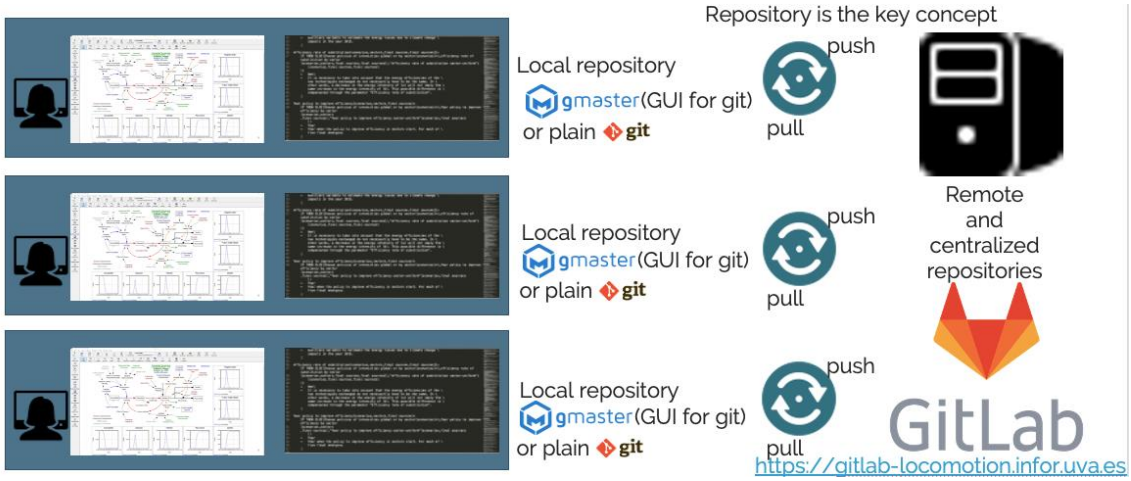


- The module supervisor should then review, check naming conventions and programming standards
- Module supervisors will merge into master, resolving conflicts when necessary.

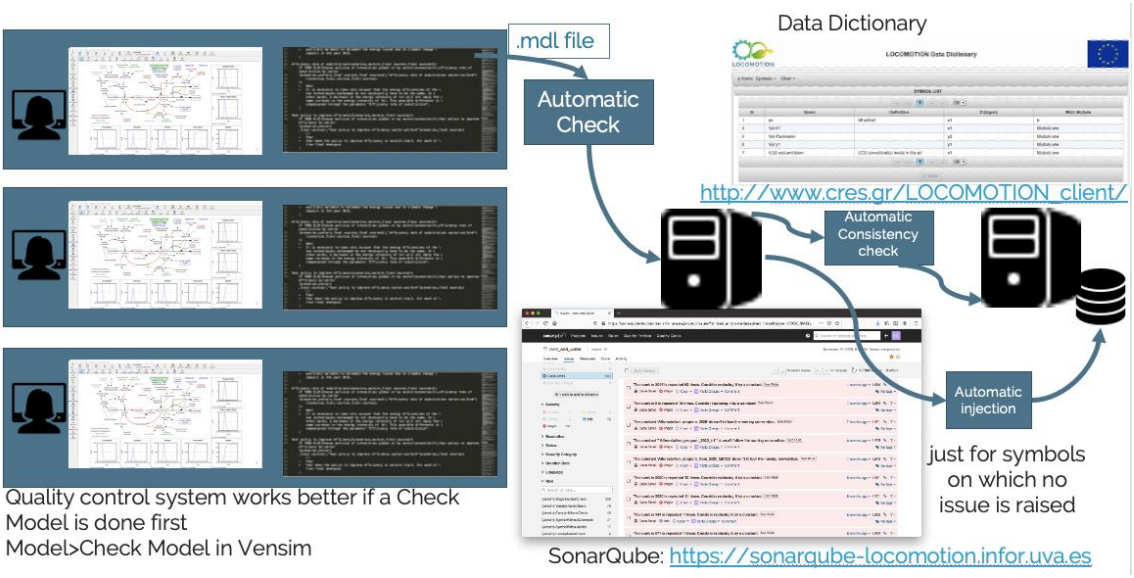
- Module programmers should be notified that a merge to master has been accomplished.
- Start the workflow again, as it is shown in the first image



According to our workflow, modelers are working locally with Vensim and git/gmaster, push to gitlab to share their local changes, viceversa pull from gitlab to see locally the changes shared by the rest of the team.



A push to gitlab triggers the automatic quality checking. The automatic quality checking also accomplishes an automatic consistency checking with the data dictionary and an automatic injection of new symbols and their information (see Appendix E).



It is important to remark that manual checking and validation by supervisors of symbols automatically injected to the data dictionary is very important.

6. DEVELOPING THE WILIAM MODEL

WILIAM is being developed in a sequential way, from modules to full model, and from simpler to more complex structures. This sequential approach is adopted in order to strive to achieve at least three objectives:

1. To be able to validate the integration of each module (simple or reduced versions) with the rest of the IAM before it reaches its most complete version.
2. To have simple versions of the modules to be able to use in the easiest and fastest releases of WILIAM (Educational and Web).
3. To get experience in Python translation with simple versions of WILIAM, and the connection with end user interfaces. Attempts will also be made to test the connection to end-user interfaces with simple versions of WILIAM.

An initial proposal for the stages and rules of programming was developed by UVa as leader of the WP9 and shared with the rest of partners contributing to model development and coordination in a workshop of 3 days (5-7 May 2020) hold online due to the exceptional COVID situation. The main objective was to agree on a basic set of rules and procedures so all programmers, contributing to different modelling WP and modules, could start programming together in a coordinated and controlled way. This basic set of rules and procedures should be understood as a common starting point which will be adapted through the life of the project following currently unforeseen requirements and issues.

In this chapter, section 6.1 focus on the stages in programming WILIAM and section 6.2 presents the agreements achieved during the developing of the model as the modules in WILIAM or the coordination between having different regional disaggregation in WILIAM.

6.1. STAGES IN PROGRAMMING WILIAM

Although the final development of the WP4, WP5, WP6 and WP7 is expected to end in the 24th month, the following stages in each one of the modules must be taken into account to the validation and in integration in the full WILIAM model (see Figure 8):

1. **Module design** (work to be developed in each modelling WP, i.e., WP4-WP7). Design of the structure and the relationships between variables in the module, in Systems Dynamics terminology corresponds to the influence diagrams.
2. **Data collection** (connected with the WP2).
3. **Modelling of relationships between variables**. Development of the equations that relate the variables and where appropriate obtain the parameters that define these relationships. In some cases, a first validation of the modelling may be performed at this stage. This modelling task in the form of mathematical equations can be carried out, a priori, with any computational tool that is considered appropriate according to the case (MATLAB, SPSS, Eviews, GAMS, …). Inputs will also be received from WP8 to inform the modelling of policies/policy targets.
4. **Development and programming of the module.** The model developed in the previous stage in the auxiliary computational tool will be now programmed in Vensim software following the programming standards of this deliverable. Estimation of the parameters for each causal relationship. Information and data could be collated taking into account parametric and structural uncertainty to facilitate uncertainty and sensitivity analyses to be performed in a later stage in WP8 (Pastor et al., 2020). A guideline to describe the uncertainty of the WILIAM inputs parameters has been set by CREAF and BC3 as can be seen in Alfresco:

https://ecm.cartif.es/share/page/site/locomotion/documentlibrary#filter=path%7C%2FWP09%2FUncertainty%2520analysis%7C&page=1

5. **Validation and test of the module**. This will be a first validation of the module still without integrating with the rest of the IAM. Different possibilities exist: uncertainty, sensitivity, robustness, stability analyses, etc. (cf. (Barlas, 1996; Sterman, 2000, sec. 4))
6. When possible, **the exploitation and dissemination of the partial results** obtained in the module will be undertaken, even before its integration into the full IAM.
7. **Merge of each module** (release number) in the WILIAM master branch through an integration project (merge projects in GitLab), this will result in periodical releases of the full model.
8. **Module (release number) validation** and preliminary results.
9. **Results exploitation**, diffusion and publishing.

From the results obtained in each version Nº (0, 1, 2), it will **return to point 2) or 3), until the final version** (3) of the module is reached.

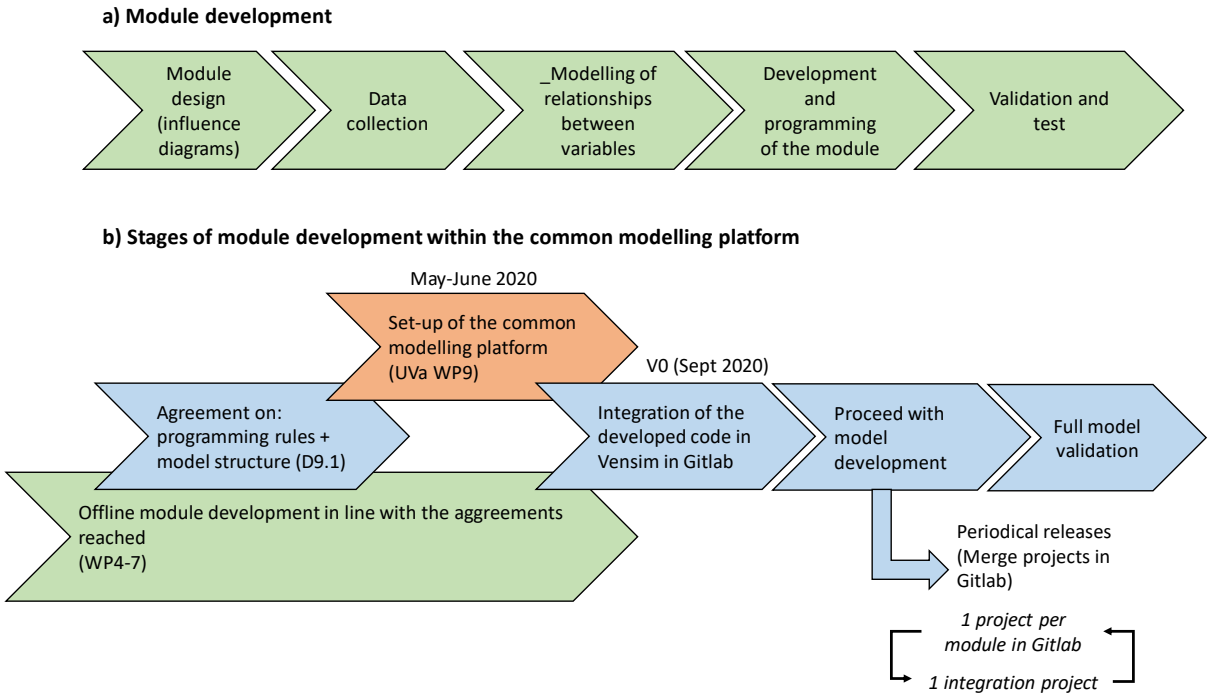


Figure 8: Planned stages in the development of each LOCOMOTION module.

In parallel, regular bilateral meetings between developers of modules having interconnections (i.e., all) should be organized periodically. These could be organized from the initiative of a module to contact another module and through general model coordination (WP9). Due to the high level of dimensions represented and feedbacks present in WILIAM, issues will need to be solved iteratively. In particular, (1) the consistency between monetary and physical variables, and (2) the homogenization and consistency of criteria in allocation functions.

The definition of "first simple version" of each module will depend on the working groups of each WP. A general suggestion may be to initially develop the module without geographical disaggregation (global data) and then disaggregate, first, in 9 regions and, later, in the 35 regions / countries. In other cases, the complexity of the structures, equations or variables used may be the criterion of greater or lesser complexity.

It is important to keep in mind that in each modelling WP it will be necessary to dedicate approximately 2 months for the validation of the module when it is integrated with the rest of the IAMs. Due to feedback between modules, this integration may produce unexpected results and help discovering errors which may require some reprogramming.

6.2. DEVELOPING THE WILIAM MODEL

6.2.1. MODULES IN WILIAM MODEL

The definition of the modules in which a model can be divided is very important from two points of view. First, it allows to conceptually depict the interrelations between the modules which together give a panoramic view of the whole model (see e.g., Figure 2 in (Capellán-Pérez et al., 2020) for MEDEAS-W model). Second, from a practical point of view, it allows to split modelling work in a consistent, controlled and coordinated way (software configuration management, see section 5).

Table 2 includes a proposal of 8 modules for WILIAM taking into account the following criteria:

- Ideally, each module should be developed within one unique WP,
- Each module should be developed in a "sufficiently autonomous" way with relation to the rest of the modules (this is of course relative since ultimately all modules are dependent on the others)
- Each module should have a "sufficiently large" size with relation to the rest of modules (i.e., it does not make sense to create a distinct module which just 5 variables)
- Avoid having a too large number of modules which would hamper coordination instead of facilitating it.
- A balance between the number of model programmers and the number of modules: to be defined, a module should be planned to be programmed by at least 2 programmers.
- The programmers of the WILIAM modules must check a propose series of basic requirements in order to ensure the quality of the model as described here: https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/405a11b7-9143-4443-a864-322083673d24

Table 2: List of modules in WILIAM by WP

WP*	Modules		Comments
WP4	1	economy	
	2	finance	
WP5	3	demography	
	4	society	Including social & environmental indicators
WP6	5	land_and_water	
	6	climate	Simple climate model.
WP7	7	materials	Including minerals as well as non-energy uses of materials with potential energy
	8	energy	Including bottom-up modelling of Energy supply, Transport, Buildings & Industry

*Policies to be developed in WP8 will belong to each of the respective modules.

Each module can be modelled in an unlimited number of views in Vensim. Each Vensim view will belong to only one module. This structure is expected to facilitate the controlled and coordinated development of the model.

The definition of modules can vary as the project evolves depending on model development evolution and the identification of eventual needs and limitations not have been foreseen to date. Also, modules can conceptually be divided in sub-modules for the purpose of model conceptual description and communication. Whenever a submodule is created and used in the software configuration management, it should be introduced in the data dictionary.

6.2.2. REFERENCES FOR THE DEVELOPMENT OF WILIAM

The development of the WILIAM model takes as reference two points which are of different nature:

1. A conceptual starting point which is determined by the objectives set in the GA
2. A “tangible” starting point which is represented by the WILIAM v0.1 available in GitLab

On the one hand, the conceptual starting point determined by the objectives set in the GA can be seen in the inter-module diagram showed in Figure 9 below. This diagram contains the basic relations between the WILIAM modules. The variables represented are only those shared between modules. Variables inside the box of a module are calculated in it and are the exclusive responsibility of each module. Arrows correspond to the variable inside the box close to the name and indicate that this variable is an input to another module.

A code of colors is used differentiating each module generated inputs to other modules. Arrows that do not go to other modules are outputs of the model. Exogenous variables (policies) are represented in pink color.

The modelers responsible for developing each module will have to make a simplified representation of the main dynamics calculated inside each respective module.

The INTER-MODULE DIAGRAM is a proposal of WP9 that we hope is useful for communication between WP’s. It is not a fixed structure and can be modified, but variables and relations can only be changed under the supervision of WP9 and with the agreement of all the modules involved. For the coordination of suggestions and changes, please, contact Marga (marga@eii.uva.es).

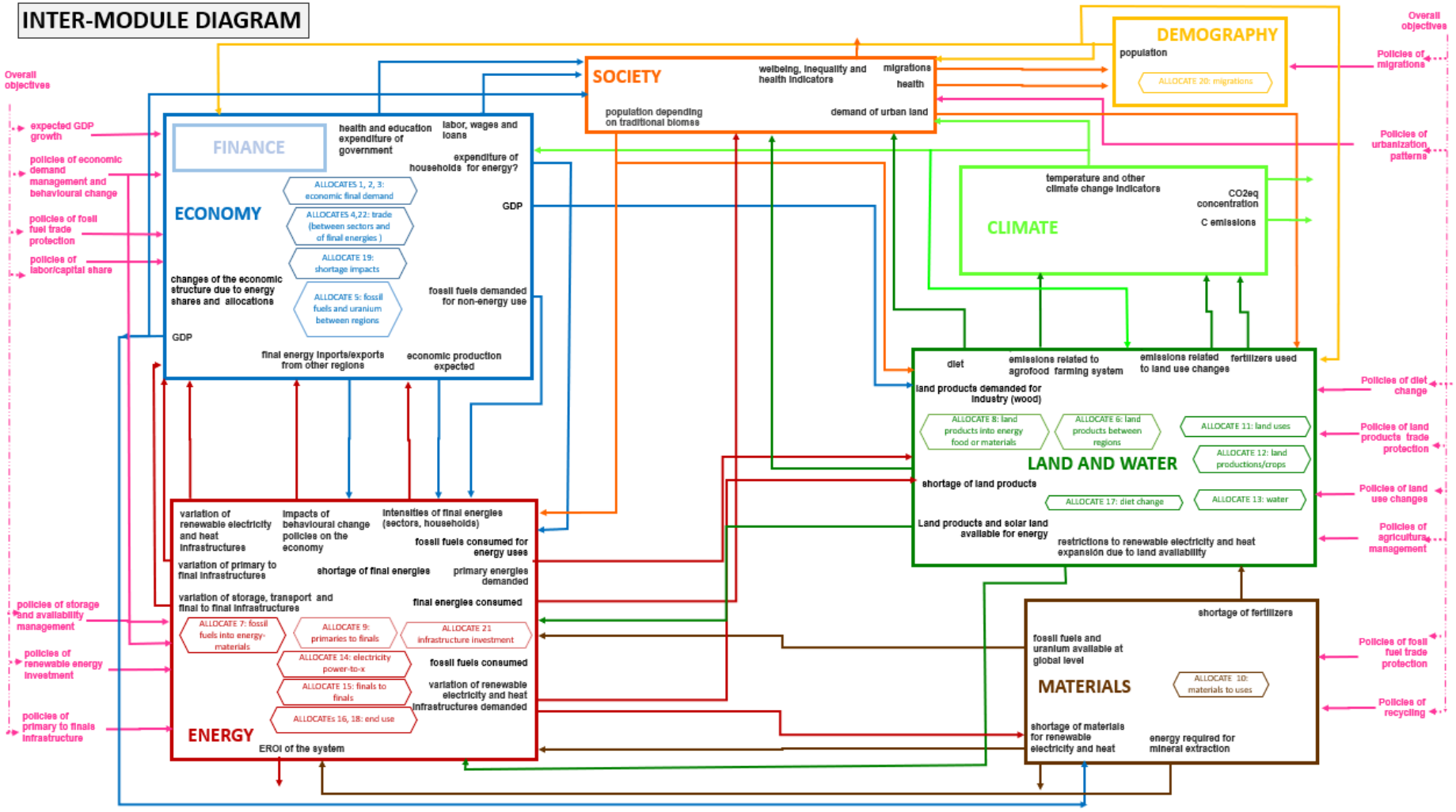


Figure 9: Inter-module diagram of WILIAM following GA and bilateral ongoing modelling WP discussions. Allocations are represented in the modules inside hexagons. Finance module is an exception: it is located inside economy because finance can be considered a part of economy.

On the other hand, the "tangible" starting point which is represented by the WILIAM v0.1 was uploaded to GitLab in July 2020. Taking MEDEAS-W as a reference represents a number of advantages with relation to the controlled and coordinated development of WILIAM comparing with starting from new "empty" projects for each module in GitLab:

- LOCOMOTION project main aim is the improvement of MEDEAS so it makes sense to take it as reference. Still, it remains uncertain "how much" MEDEAS-W is going to remain at the end of the LOCOMOTION project. However, it is clear that there are no constraints to replace existing structures and feedbacks if the changes are considered in the GA and there is consensus that they represent an improvement to the existing modelling.
- It is easier to model (1) with a model that already runs and (2) having clear the boundaries of each module (as set in Figure 9) than starting from "empty projects".
- Having clear the boundaries of each module facilitates maintaining the internal coherence of the model while module development progresses.

Following the guidelines set in D9.1, MEDEAS-W was adapted to be uploaded in GitLab for the start of the controled and coordinated development of WILIAM. The result of this adaptation is what we have named WILIAM v.0.1 and it included the following main changes from MEDEAS-W (note that WILIAM development since then has made that some of the below points have changed):

- Split of the original inputs_W.xlsx file in 3 different types: 1 xlsx per module within a folder per module, 1 xlsx for common constants and units change, 1 xlsx for scenario parameters in a specific folder (with 1 tab per module). This structure may evolve on demand (cf. 4.3).
- All variables are read from the .xlsx files by field names instead of by cell number. "Magic numbers" have been removed.
- Start in 2005 instead of 1995 to adapt to ICIO available historic timeseries data
- Remove vector [scenarios] present in MEDEAS models which allowed the simulatenous simulation of 6 scenarios. This vector has been removed in order to improve the speed of the model. Model usability is barely affected since it in order to compare different runs it is just necessary to prepare different scenario_parameters.xlsx files.
- Regionalization: a common vector to all modules has been created (REGIONS_I) with 9 elements corresponding to the 9 WILIAM regions defined in D9.1 (some of them are aggregated of countries and others are a country in it-self): EU-27, United Kingdom, China, EASOC (East-Asia & Oceania), India, LATAM (Latin America excepting Mexico), Russia, USMCA (US, Mexico & Canada) and LROW (Locomotion Rest of the World). It has to be highlighted that the model has not been calibrated to the historical data and parameters of these regions, but in fact each region runs as 1/9 of the world. Hence, data in the input xlsx files are all global and have been adapted in the .mdl when necessary. This approach has been taken for the sake of simplicity given the effort of calibrating for the 9 WILIAM regions would have been extremely time consuming at this stage and it is the task of each module (and would be useless in those parts of the model which are going to be fully replaced). The parameter "NUMBER_OF_REGIONS" is set at 9 and extensive variables at world level are divided by 9.
  (Update August 2021: given the different module development, two subscripts for the number of regions were implemented in gitlab in June 2021 to allow that different modules may be developed for 9 or 35 regions within the same model. This was solved by creating one subscript REGIONS_I of 36 elements, the 35 regions + EU aggregated. To facilitate the programming 4 subranges where created: EU_27R_I, REGIONS_35R_I, REGIONS_9R_I, ROW_8R_I. See section 6.2.3).
- Nomenclature: it was finally not possible in the available timeframe to adapt the name of all symbols to the naming convention rules reflected in the D9.1. Let us recall that MEDEAS models have over 3,000 variables each. For illustrative purposes, the symbols of just a module, the demography module, have been renamed following this convention. Still, some cleaning has been performed on the naming of MEDEAS to avoid some known confusions and mistakes.

Hence, the names of the variables in Figure 9 do not necessary coincide with the ones currently existing in WILIAM v0.1. In some occasions, there is no equivalent variable. The relations between variables of de diagram and variables of WILIAM 0.1 are detailed in file "Inter-module WILIAM variables.xlsx": https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/1c71a910-8c37-495f-9772-2b32e2867530.

Hence, the development of each module will have to take into account the following rules:

1. Use all the inputs from other modules and preserve all the outputs produced by the module already existing in WILIAM v0.1. This ensures that the whole model runs and that each module is consistent with the rest of the model. Of course, during WILIAM model development it will be possible to modify/improve an existing feedback which could also increase the number of variables involved. This control will be facilitated by the symbol data base dictionary.
2. Progressively include all the new relationships planned in the GA of LOCOMOTION (bilateral module meetings with WP9 coordination). The modules in charge of creating input variables to other modules should report to these modules when a new variable is available so they can be used after MERGE projects.
3. The preservation of current modelling structures of MEDEAS is optional and subject to the assessment of each module.

In the case a modeller needs a variable which in that moment is not generated by the module who should generate it (e.g., due to different development stages in different modules), he/she will have to (1) identify which module is responsible for its generation (consult the centralized data dictionary and ask WP9 in case of doubts), and (2) request and discuss with the corresponding module supervisor the creation of a new variable (see section 4.3.2). As this could take some time, to not stop the modelling process temporary the modeller can create a local and temporary variable which will be overwritten when the module is able to compute and share the requested variable.

6.2.3. COORDINATION BETWEEN MODULES HAVING DIFFERENT REGIONAL DISAGGREGATION IN WILIAM

The difference between 9 and 35 global regions in the short-term relies in the disaggregation (or not) of EU in 27 member states (MS) depending on the module. In the first versions:

- Economy module: 35 regions (EU disaggregated)
- Rest of modules: 9 regions (EU aggregated)

The issue lies in the communication from the rest of modules → economy (or other module with 35 regions), since in the other direction aggregation would easily work.

We assess that not linking the rest of modules (9 regions) with the ones of Economy (35 regions) would incur in larger errors than to approximate the value of the inter-module variables for each EU MS taking as reference the endogenous behaviour of the aggregated EU region simulated in the modules working for 9 global regions. Proposal from WP9 coordination after discussion in the last WP9 meeting:

- Each WILIAM module working at 9 global regions identifies the inter-module variables which communicate that module with the economy module (in the direction EU aggregated → 27 EU MS).

- The economy module should receive the information disaggregated for 35 regions (8 global + EU disaggregated in 27 MS) from the rest of modules (this allows to spread the additional data search effort among different modules).

- The economy module should produce the variables at 9 regions by aggregating all the EU MS into one unique EU region.

- Each module (with possible assistance from WP2) collects the historical data for at least the full historical period of WILIAM (i.e., from 2005) of the inter-module variables for which there is the need to set the share weights for each EU MS. As a first approximation, the last available historical data share weights for each inter-module variable for each EU MS could be maintained constant when performing simulations into the future.

- If a module assesses that the number of inter-module variables to be sent to the economy module is too high (i.e., too high effort of data collection), a prioritization should be performed by this module to select the most important inter-module variables which are going to be disaggregated from 1 region (EU) to 27 EU MS. The remaining inter-module variables can be disaggregated in further work or just been left for the 35 region version. Take into consideration that it is usually easier to find data for EU member states than for other WILIAM regions, and that WP2 may assist in this task.

  Figure 10 shows a graphic representation of the method to deal with different regional disaggregations in WILIAM.

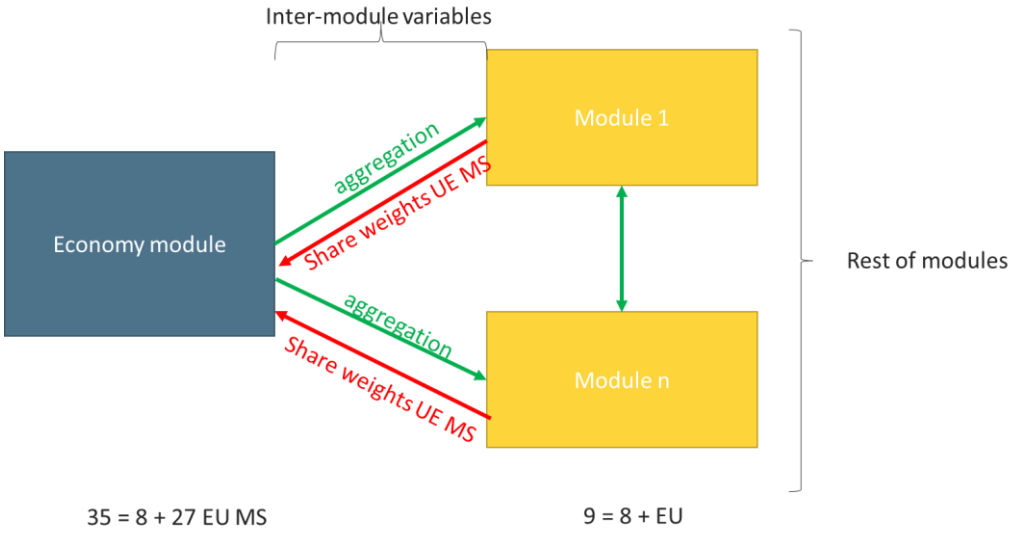


Figure 10: Diagram representing the method to deal with different regional disaggregations in WILIAM

- Since these inter-module variables are in fact endogenous, these historical data share weights will also be able to be used for model validation.

As the modules shift from 9 to 35 regions these inter-module variable weights will be progressively deleted as they are replaced by the 35 → 35 regions in all module linkages.

Example Vensim program (*Example 35-9 regions subscripts WP9.mdl*)

Download example program in: https://ecm.cartif.es/share/page/site/locomotion/folder-details?nodeRef=workspace://SpacesStore/94257113-3cee-4da7-a1bd-0bad30025bdf

In practical terms, we will have 1 unique subscript [REGIONS_I] with 36 elements and 4 subranges (https://www.vensim.com/documentation/subscriptcontrol.html), cf. Figure 11:

- [REGIONS_35R_I]: 35 index elements corresponding with the 35 WILIAM regions

- [REGIONS_9R_I]: 9 index elements corresponding with the 9 WILIAM regions (EU is aggregated in one unique region)
- [EU_27R_I]: 27 index elements corresponding to the 27 EU MS
- [REGIONS_8R_I]: 8 elements corresponding with all the global regions excepting EU

No variable of the model will be defined for the subscript REGIONS_I, each variable will be defined either for 35 regions [REGIONS_35R_I] or 9 regions [REGIONS_9R_I]. The subscripts [EU_27R_I] and [REGIONS_8R_I] will only be used for the interface variables between modules which use 35 or 9 regions.

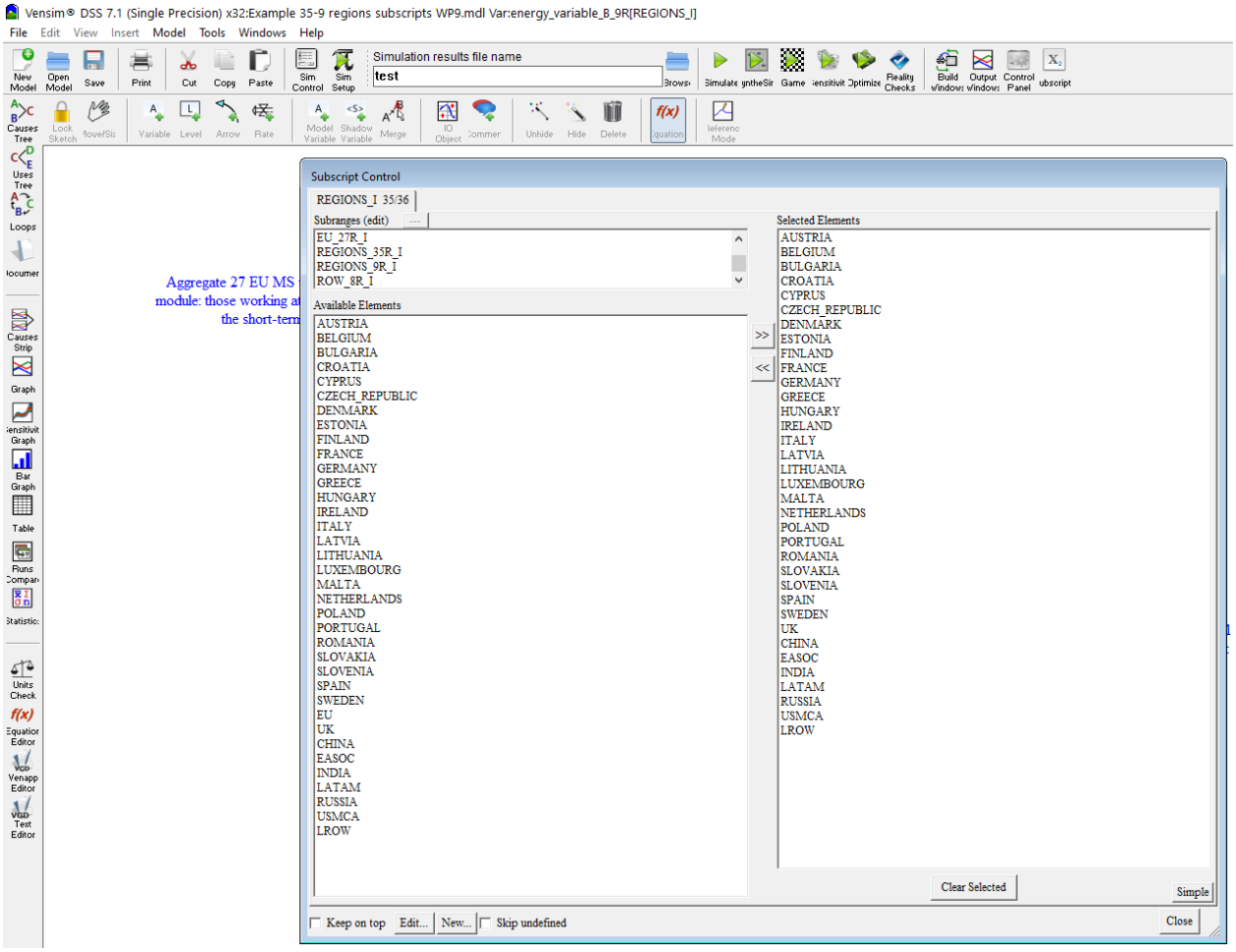


Figure 11: Screenshot of the subranges created within the REGIONS_I subscript.

Comments:

- Absolute values for each of the EU MS inter-module variables should be loaded from .xslx files to have already the data available for validation purposes, and internally compute in Vensim the respective EU MS share weights.
- Given that there will be 2 variables representing the same concept (but just with different regional disaggregation), in the Vensim example model e.g., “economy_variable_A” and “economy_variable_A_9R”. The proposal is to name with the suffix “_9R” the variable for 9 regions.

The order of the elements in the index REGIONS_I follows the following logic:

- EU27 countries are located first
- UK follows
- Follow the rest of global regions by alphabetic order (CHINA, EASOC, INDIA, LATAM, RUSSIA, USMCA)
- The last region is the "LOCOMOTION Rest of the World" LROW

How to make the transition from 9 to 35 regions in a given module in the future

In case of switching to 35 regions, a module should delete all the share weights for each of the 27 EU MS with the economy module, but should include new share weights for those inter-module variables of this module with the rest of modules working at a different regional disaggregation. Due to this reason, coordination among modules is key to jointly shift regional disaggregation.

In practical terms:

- Replace the subscript [REGIONS_9_I] in all the affected variables of the module by the subscript [REGIONS_35_I]. For this, the Vensim option *Edit->Set subscripts* (https://www.vensim.com/documentation/21220.html) could be useful.
- Variables converted from 35 → 9 regions:
    - The variable "economy_variable_A" would be equivalent to "economy_variable_A_9R". We should merge both, keeping the equation in "economy_variable_A" but also the links of "economy_variable_A_9R" with the rest of modules. This can be easily be done with the Merge Vensim tool (https://www.vensim.com/documentation/22950.html).
- Variables converted from 9 → 35 regions:
    - The EU MS share-weights would be unnecessary for the modelling: delete the variable "energy_variable_B_EU_27R"
    - The variable "energy_variable_B" would be equivalent to "energy_variable_B_9R". As in the preceding case, we should merge both of them, keeping the equation in "energy_variable_B_9R" but also the links of "energy_variable_B".
- It will always be interesting to keep these 4 regional subranges structure in order to generate output variables at 9 region level which may be useful when preparing results for dissemination, publications, etc.

6.2.4. COORDINATION BETWEEN MODULES FOR UNITS CONSISTENCY IN WILIAM

Unit consistency must be always maintained during the modelling process therefore a correct coordination between all modellers is necessary. For this, the standard units for WILIAM have to be agreed in the consortium. These standard units will have to follow the International System of Units. This list is available in Alfresco:

https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/21a9774f-5844-4a8b-a8d6-8856abfdf8b2

After that first list of units, each WP can add new units to the model previous communication with WP9.

An example of WILIAM standard units can be seen in the **Table 3**.

Table 3: Provisional WILIAM standard units of the model

List of standard units in WILIAM			
	Type of variables	Unit	Concepts
1	Dimensionless (e.g., shares, ratios, etc.)	DMNL	Dimensionless
2	Time	Years	Time (Years)
3	Economic stocks (e.g., capital stock) at current prices/nominal terms	dollars	Current monetary units (dollars)
		Mdollars	Current monetary units (Mdollars)
	Economic stocks (e.g., capital stock) at constant prices/real terms	dollars_2015	Constant monetary units (dollars_2015)
		Mdollars_2015	Constant monetary units (Mdollars_2015)
	Economic flows (e.g., production, demand, etc.) at current prices/nominal terms	dollars/year	Current monetary units per year (dollars)
		Mdollars/year	Current monetary units per year (Mdollars)
	Economic flows (e.g., production, demand, etc.) at constant prices/real terms	dollars_2015/year	Constant monetary units per year (dollars_2015)
		Mdollars_2015/year	Constant monetary units per year (Mdollars_2015)
4	Power stock (e.g., capacity installed)	W	Power stock (Watt)
	Power flows (e.g., construction, decomissioning, etc.)	W/year	Power flow (Watt)
6	Energy stock	J	Energy stock (Joule)
	Energy flows (e.g., consumption or demand)	J/year	Energy flow (Joule)

7. COMMON NOMENCLATURE FOR POLICY ANALYSIS AND SCENARIO ASSESSMENT. SCENARIO DESIGN IN LOCOMOTION

Policies, policy targets, storylines, scenarios, etc.: it has been realized that for different LOCOMOTION partners these words may have different meanings which is due on the one hand to their extensive and popular use, and on the other hand to the distinct background and methods which each of us are used to apply in our daily work.

Hence, it was detected the need to establish a common definition for each of these concepts so all partners are in the same page when developing the work in WP8 specifically and in all the project broadly since these are cross-cutting concepts. These definitions must follow as much as possible typical conventions as well as be functional to their use and interpretation in the Grant Agreement to minimize confusion among partners. For the sake of consistency/clarity among us, we need to accept that for some of us some of these concepts would be defined differently or that we would name the ideas described behind them differently.

7.1. DEFINITIONS

Table 4 includes a list of definitions (and synonyms) of each concept key for WP8 taking into account how the Project proposal was written, including examples and specific information on which LOCOMOTION tasks/WPs deal with each of them. Still, we warn that the definitions and key terms suggested in this document imply some differences with relation to the original GA which were introduced to enhance the clarity and comprehensiveness of the concepts and avoid some confusions which have been detected in the meantime.

The order of the items defined in Table 4 follows the order to perform a simulation with WILIAM. Two concepts are central in this framework (see Figure 12): (1) the overall goals, which refer to broad, general objectives at societal level, and (2) policy measures which are specific interventions in different parts of the system which can be specified through quantifiable targets (policy targets) and are implemented through instruments (policy instruments).

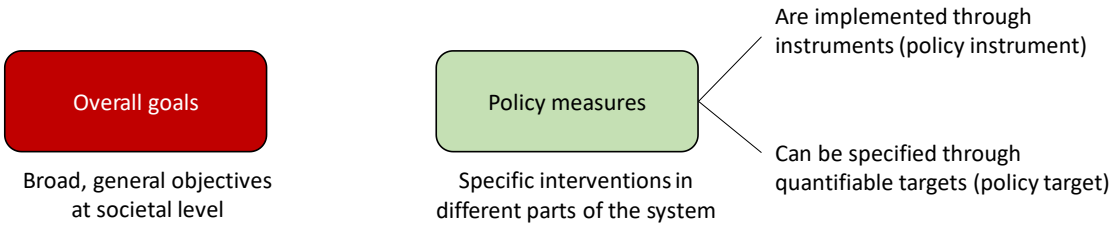


Figure 12: Schematic definition of overall goals and policy measures.

For example, an overall goal could be the full net decarbonisation by 2050 or maintain the global average temperature below 1.5^{o}C. A policy measure could be to promote renewables in the electricity sector which could have a target of let's say 50% by 2035 and which could be implemented through a series of policy instruments such as feed-in-tariffs, mandatory installation of RES in new/renovated buildings, carbon tax, etc. Of course, a same policy instrument may contribute to implement different policy measures and reach different targets.

Table 4: List of definitions, examples and tasks within LOCOMOTION for key concepts related with policy analysis and scenario assessment

	Definition	*Examples*	Task in LOCOMOTION
OVERALL GOAL (aka overall objective)	An overall goal is by definition a broad, general objective at societal level (not to be confounded with a policy measure which correspond to a more specific objective).	*Full net decarbonisation by 2050; global average temperature < 1.5^{o}C; achieve full employment; maintain a certain level of societal equity; annual GDPpc growth of 3%; SDGs, etc.*	The achievement of all the overall goals indicates that a simulation has been successful into proposing solutions to the identified problems, and constitutes the basis for developing policy recommendations. In LOCOMOTION it is planned to develop a list of desired "overall goals" taking into account different criteria (social, economic, environmental, etc.). WP5 is developing well-being indicators (e.g., SDGs (UN, 2020)) and Task 8.2.3 will also focus on this. It is planned that the demand functions to be developed within WILIAM take into account these overall goals so that the model naturally pursuits its attainment (WP4 and WP9). Two kinds of "overall goals" can be differentiated: those identified as driving a given storyline (storyline goals) and those which the model users may want to check if a given storyline are able to fulfil (expanded overall goals). In any case, it is required that the model has measurable variables that can be used as indicators to verify the fulfilment of these overall goals.
STORYLINE (aka narrative)	General narrative about how the future might unfold. Qualitative description of the trajectories of economic, social, technological and environmental evolution that the world (including heterogeneous disaggregation) may follow in the near future (hence, it implicitly includes a certain number of policy measures).	*Business as usual, Green Growth, Degrowth, etc.* *SSPs represent a relevant set of storylines since they are standard in the climate science community (this does not mean that we cannot simulate other storylines which are not considered within the SSPs).*	Task 8.3.1 will identify a minimal set of relevant storylines to simulate with the model (SSPs as relevant for IAM/climate community + other relevant storylines from literature, stakeholders, etc.). These storylines may depict heterogeneous regional developments since WILIAM will be a multi-regional model.

	Definition	*Examples*	Task in LOCOMOTION
POLICY	A set of ideas or a plan of what to do in particular situations that has been agreed to officially by a group of people, a business organization, a government, or a political party.[6] Different types of policies can be defined depending on their scope: Energy Policy, Economic Policy, Trade Policy, Health policy, Environmental policy, etc., not to be confounded with policy measures which are more specific.	*Decarbonization, transition to renewables, reduce unemployment, reduce inequalities, GDPpc growth, increase exports, etc.*	The concept of "policy" is very broad, general and even loose and for these reasons it will not be separately worked in the project, but rather considered jointly with the overall goals and storyline. An important conceptual distinction is made between assumptions included in the model that are considered "the result of policies" and those considered "hypotheses". Assumptions that refer to human behaviour, resp. anything that can be influenced by humans are considered the result of (often multiple) policy measures. One example are changes in people's mobility behaviour (e.g. choice of mode, number of person-km, etc.) or demographic development. Hypothesis refers to assumption that are conditioned by biophysical realities, but subject to uncertainty. One example is the reserves of minerals or oil.
POLICY OBJECTIVE (aka policy goal)	Generally formulated desired outcome of a policy. Not to be confounded with policy target (see below), which refers to a specific quantified level or rate set for the chosen objective.		
POLICY MEASURE	Intervention in a part of the existing system (economy, environment, social welfare, etc.) typically promoted by institutions such as governments and regulatory institutions to drive a technological, behavioural, infrastructure, etc. change with relation to current trends. As shown in Figure 12, policy measures can be specified through quantifiable targets (policy targets) and are implemented through instruments (policy instruments).	*Feed-in-tariffs for renewables, carbon tax, border adjustment tax, raising awareness to consume sustainable products, new laws limiting or prohibiting the consumption or production of polluting products, etc.*	Task 8.1 will collate a structured database of policy measures (from IAMs, from NECPs, scientific literature review + stakeholders). Task 8.2 will make a selection of those: (1) viable to be implemented in the model and (2) more relevant based on a diversity of criteria (including stakeholders' opinions) in cooperation with the WP5 which may for example include a selection of targets among the SDG targets. It will be often not possible to model the effect of a policy measure directly in WILIAM due to the inherent complexity of modelling very heterogeneous measures in the absence of sufficient granularity (geographical, economic agents, etc.). A practical alternative in these cases is to represent the intended effect (=policy target) that such a policy measure would have on a particular feature or sector (be it endogenously or by a exogenous assumption that is consistent with the storyline).
POLICY TARGET	Quantifiable target (=intended effect) of a given policy measure.	*32% of renewable energy in total energy by 2030, increasing public transport passengers by 20%, reducing the same 20% of private*	

[6] Adapted from: https://dictionary.cambridge.org/es-LA/dictionary/english/policy.

	Definition	*Examples*	Task in LOCOMOTION
		transport passengers, reduce meat consumption in the diet by 10%, targets from the SDGs, etc.	The selected policy measures to be modelled will be communicated to each modelling WP for their implementation. The effectiveness, unintended effects, impact on equity, as well as cost, feasibility and acceptability of implementation, etc. of each measure are different, and WILIAM must be able to show which combination of policy targets and measures is best in achieving the overall goals (this will be part of the LOCOMOTION policy recommendations to be done in Task 8.5). In any case, it is required that the model has measurable variables that can be used as indicators to verify the fulfilment of these policy targets.
POLICY INSTRUMENT	Mechanisms and tools to implement policy measures and eventually reach policy targets. It should also be considered that a same policy instrument may contribute to implement different policy measures and reach different targets. In LOCOMOTION 12 types of policy instruments are considered (cf. D8.1 (Böck et al., 2020)): regulatory, financial incentives, taxes, behavioural, improved infrastructure, education and training, obligation schemes, public procurement, research and development, roadmaps and plans and other.	*Laws, regulations, codes, subsidies, taxes, campaigns to raise awareness, new infrastructure projects for transport or urban planning, focused education and training for professionals, obligation schemes, public procurement, science funding, etc.*	The classification of policy instruments performed in Task 8.1 might help to indicate if, how and in what LOCOMOTION module policy measures might be modelled.
HYPOTHESIS	Hypotheses are assumptions made in the model that refer to biophysical realities that are not controllable by human societies. Hypotheses are subject to uncertainty.	*Oil reserves, mineral reserves, tipping points* *How much oil is available at X extraction cost in region Z?*	Hypotheses are exogenous inputs in the model.

	Definition	*Examples*	Task in LOCOMOTION
SCENARIO	A consistent quantification of a storyline. It is composed by a set of quantitative inputs to which together allow to simulate a storyline with the model and get results. These inputs can be very different, e.g,. hypotheses, policy measures or policy targets.	*(Same as for storylines)*	Task 8.3.2. In collaboration with modellers the quantitative inputs corresponding to each storyline will be set. Scenario parameters: all the quantitative information to perform a simulation will be gathered in a dedicated excel file which will have 1 tab per LOCOMOTION region/country (work under development in WP9).
ADAPTIVE SCENARIO	The trajectory of the initially defined scenario can be modified due to the internal constraints present in the model.	*A Green Growth/Degrowth/etc. which does not attain its predefined overall goals*	It is a result of the simulation (endogenous), hence no specific tasks are envisaged to specifically tackle this. They will be obtained one the model is simulated in Tasks 8.4, 8.5 and 9.2.

The next section describes in more detail how these concepts should be understood and set in order to perform simulations of WILIAM.

7.2. RUNNING A SCENARIO WITH WILIAM

7.2.1. SCENARIO METHODOLOGY

WILIAM will be an IAM which will represent many dimensions: economic, social, land-use, mineral, energy, land-use, etc. in the long term (~2100). This makes that the simulations performed with the model are inherently subject to a great degree of uncertainty issued from these and other dimensions not included in the model (just think about COVID or 2020 USA elections). Moreover, as it is well known, the further we look into the future, the less we can say with confidence.

Performing simulations with models can be a cumbersome task when the models have many parameters and hypotheses of uncertain evolution as well as an almost infinite possible combinations of policy measures which can be varied at the same time. "Scenario methodology" offers an approach to deal with the limited knowledge, uncertainty and complexity of natural and social sciences and, applied to this kind of models, can be used to group the different future possibilities into coherent and meaningful scenarios. Each scenario is based on a storyline which represents a (qualitative) archetypal and coherent vision of the future which may be viewed positively by some people and negatively by others. Each of these storylines already includes implicitly some hypotheses and policies (e.g., the BAU scenario from the World Energy Outlook from the IEA is typically termed "Current Policies"). Moreover, each of these storylines are driven by certain logics and motivations which we have termed here "overall goals driving the storyline". A classic example within IAMs would be the pursuit of 2-3%/yr GDPpc growth with regional convergence. In a second step, policy scenarios are designed on top of the baseline scenarios in order to add or enhance policy targets and measures in order to reach a pre-defined expanded set of overall goals beyond those driving the storyline. In climate IAMs, a classical pre-defined expanded overall goal is the staying below 2^{o}C by 2100. This scenario methodology has been widely used in Global Environmental Assessments such as the Millennium Ecosystem Assessment, the UNEP's Global Environmental Outlook or the successive IPCC reports -SRES, SSPS- (e.g., (MEA, 2005; Riahi et al., 2017; SSP db, 2018; van Vuuren et al., 2012)). However we should also take into account that in these institutional frameworks the governments have a say (e.g., veto right in IPCC) and the scenarios issued from these institutional processes cannot hence be considered as unbiased "purely" scientific outcomes (e.g., (Girod et al., 2009)). Hence, this should also be taken into account when selecting the set of relevant storylines to be modelled with WILIAM. Moreover, WILIAM will have a very specific and particular structure and approach, so other sets of scenarios may not be fully equipped to simulate scenarios with this tool, due for example to the absence of dimensions present in WILIAM which are not present in other IAMs (e.g., mineral requirements) or variables which are exogenous in other IAMs which are endogenous in WILIAM (e.g., population and GDPpc).

It is key to understand that the power of IAMs is not predictive power but rather in the intercomparison of applying different policies in different plausible contexts and assess which ones would be the most beneficial in a succession of "what-if" exercises. MEDEAS and WILIAM models work under the logic of "adaptive scenarios": at the beginning of the simulation there are some current trends + policies which are expected to happen within a given general context (storyline), but then feedbacks may or may not allow these current trends and policies to really happen in the future. In this sense, MEDEAS/WILIAM models especially focus on biophysical (e.g., land, water, mineral and energy availability) and socioeconomic constraints (e.g., economic structure, available income) (for more information on this see (Capellán-Pérez et al., 2020)). Of course, it is impossible to develop a model with full predictive power neither we intend to achieve that, and also in the real world if the narrative happens to be unfeasible in

a given point of the future there will be changes in the socio-cultural dimension (which our model is neither intended to capture) which will deviate the storyline from the one initially set. However, this kind of changes are beyond the scope of our project and in general beyond the scope of IAM modelling and belong to the discussion about the limits of modelling for policy advice.

7.2.2. BASELINE VS. POLICY SCENARIOS

Based on the aforementioned, two main types of simulations will be able to be performed with WILIAM:

> **(1) Baseline scenarios**, in which a given storyline will be implemented in the model. The efficacy of this scenario will depend on the level of fulfilment of a set of predefined overall goals (which include those driving the storyline but can also be expanded freely by the model user to include other dimensions). In the case that these overall goals are not attained, the design of policy scenarios will be then required:
>
> **(2) Policy scenarios:** these scenarios include the additional or enhanced policy measures which, consistently with the storyline, allow to attain the pre-defined overall goals.

Hence, 2 kind of inputs are required to perform a baseline simulation with the MEDEAS or WILIAM, and one additional input (additional/enhanced policies) to run policy scenarios (see more detailed definitions in the Table 4 above):

- **Hypotheses**: assumptions related to features affected by uncertainty and which are assumed to be independent from human behaviour and decision within the model (e.g., fossil fuel availability, RES technical potentials, climate change impacts given a certain rise of temperature, etc.).
- **Policy measures** implicit/consistent with the storyline, including the ones derived from those overall goals driving the storyline in each case.
- **Additional/enhanced policies** to fulfil the pre-defined overall goals in the case of not having been attained in the baseline scenario.

Note that depending on the specific storyline to be implemented, a same input could be considered of a different type. For example: GDPpc. While in a Green Growth storyline the increase of GDPpc is a policy target, in post-growth narratives the GDPpc will be just an output without any specific a priori expectations. Hence, each scenario will have to be implemented in a case by case analysis also taking into account the approach and architecture of the model. This way of dealing with these issues is conceptually similar to the approach taken in the IAM community, see SSP/RCP climate scenarios architecture through the Shared Policy Assumptions (SPA) see (Kriegler et al., 2014) (in fact our scheme is simpler and more robust since we consider all kind of policies/policy targets together, while the SPA focus solely on climate-focused policies; this avoid problems with borderline cases due to the fact that policies are often motivated by multiple objectives).

Additionally, since WILIAM will be a multi-regional model it will be possible to conceive storylines, policies and overall goals which differ at regional level. However, in this document for the sake of simplicity we will refer to just one region.

7.2.3. MODEL RUNNING MODES

Following the GA and the implementation plan of WP8, WILIAM is planned to be run in 3 different modes:

a- Simulation mode,
b- Multi-objective parameter optimization mode (which allows to automatize runs performed in simulation mode),

c- Uncertainty analysis.

We can also think of a fourth mode (d/ Hybrid mode) resulting of the combination of these modes.

Follows a short description of the required steps to run the models in each of these modes.

7.2.3.1. SIMULATION MODE

Follows a list of steps to simulate a baseline scenario which are conceptualized in Figure 13.

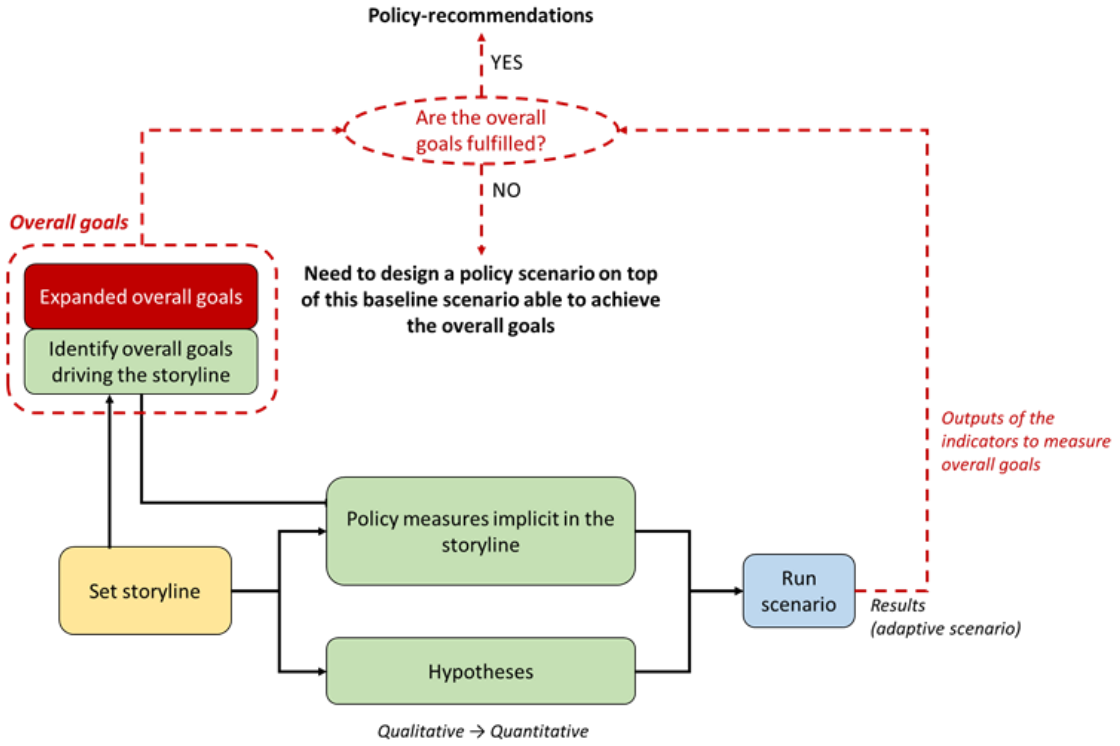


Figure 13: Conceptual representation of the steps to simulate a baseline scenario with the LOCOMOTION model.

1. Set a storyline about how the future world might unfold (including potential regional heterogeneities). This includes setting qualitative information for each storyline on the main underlying hypotheses, the implicit policy targets and measures as well as the identification of the overall goals driving the storyline which will be pursuit by the model

2. A number of policy measures may be consistent with each storyline (of course, one same policy measure or target can be part of different storylines). This qualitative information should be categorized by scenario parameters of the model through "input tables" (see Table 5 extracted from the supplementary material of (Riahi et al., 2017)). As a starting point, current trends are taken as reference which allows to build a BAU scenario. Varying these trends to above or below current trends (e.g., Low, Med and High in Table 5) we get exogenous variations on current trends.

Table 5: Example of input table from (Riahi et al., 2017) for qualitative assumptions for energy demand across SSPs.

Table A.1: Qualitative assumptions for energy demand across SSPs

	SSP 1			SSP 2			SSP 3			SSP 4			SSP 5		
	Country Income Groupings														
SSP Element	**Low**	**Med**	**High**	**Low**	**Med**	**High**	**Low**	**Med**	**High**	**Low**	**Med**	**High**	**Low**	**Med**	**High**
Non-climate Policies															
Traditional Fuel Use	fast phase-out, driven by policies and economic development			intermediate phase-out, regionally diverse speed			continued realiance on traditional fuels			continued traditional fuel use	some traditional fuel use among low income housholds		fast phase-out, driven by development priority		
Energy Demand Side															
Lifestyles	modest service demands (less material intensive)			medium service demands (generally material intensive)			medium service demands (material intensive)			low service demands	modest service demands		high service demands (very material intensive)		
Environmental Awareness	high			medium			low			low	high		medium (low for global level/high for local level)		
Energy Intensity of Services															
Industry	low			medium			high			high	low		medium		
Buildings	low			medium			high			medium	low/medium		medium		
Transportation	low			medium			medium		high	low/medium	low		high		
General Comments				some regional diversity retained											

3. The qualitative tags “Low”, “Med” and ”High” are quantified for each dimension and region. This means re-doing the previous Table 5 replacing the qualitative information by numbers.

4. The model user can also identify some additional overall goals just to check if the resulting adaptive scenario also fulfils them.

5. Simulation: the efficacy of each scenario will be done by comparing the obtained value of the model outputs which correspond to the overall goals with the pre-defined set of overall goals. These overall goals may or may not be achieved given that since WILIAM (in the same spirit than MEDEAS) will include a number of feedbacks representing physical and social constraints, it may happen that some storylines are ultimately not feasible. This is not in contradiction with the whole framework given that we should think that the result are “adaptive scenarios”.

For the sake of being clearer, the following Figure 14 is the same than Figure 13 but giving some examples applied to the specification of a Green Growth scenario with MEDEAS about the different types of inputs to characterize a scenario (panel a), and an ad hoc result depicting how an “adaptive scenario” looks like when one of the overall goals are not reached (panel b):

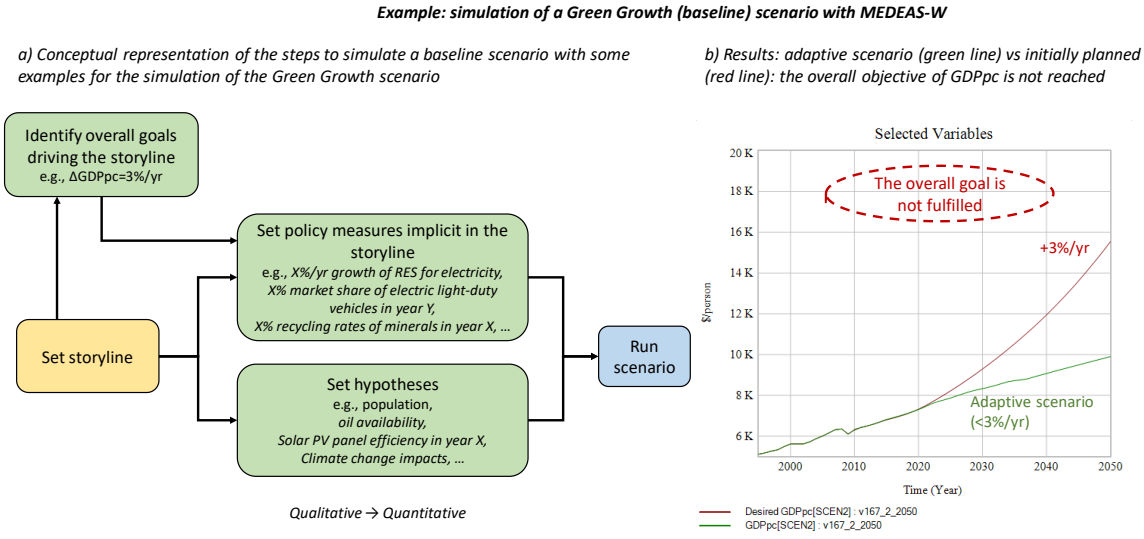


Figure 14: Example of scenario implementation and simulation for a Green Growth scenario within MEDEAS.

If the overall goals have not been reached, the model user can then decide to design a policy scenario following the steps described in Figure 15 by introducing additional/enhanced policy measures beyond those just strictly implicit in the storyline:

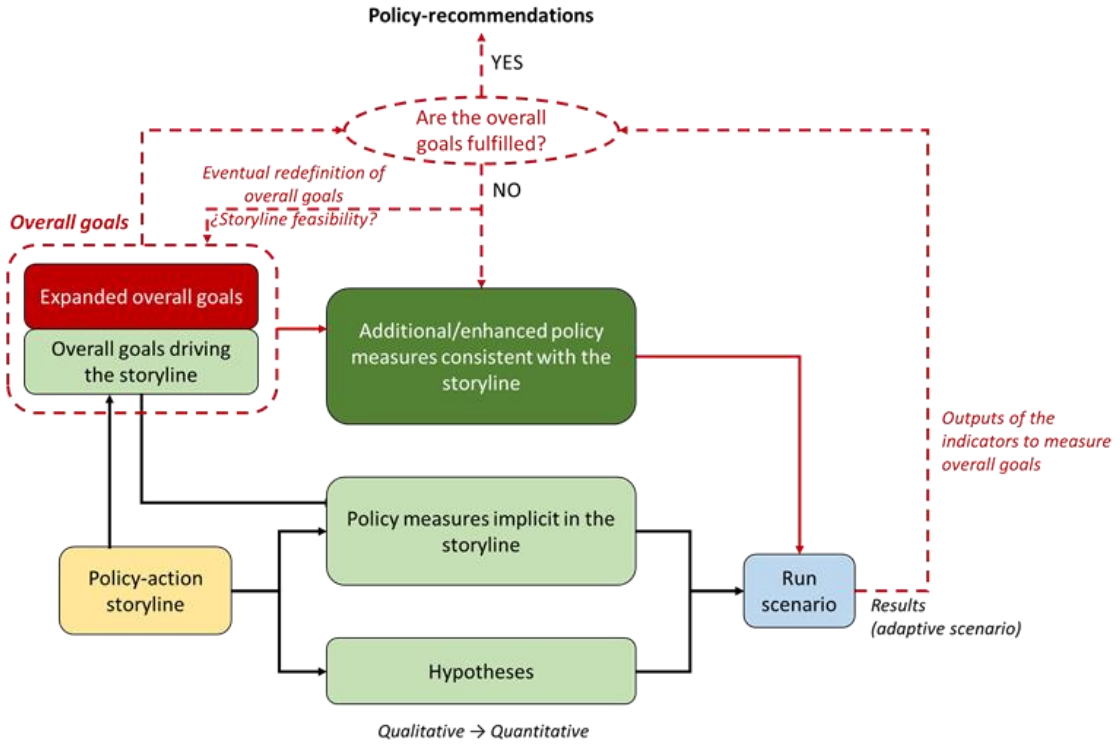


Figure 15: Conceptual representation of the steps to simulate policy scenarios with the LOCOMOTION model.

The user can try different combinations of enhanced policy measures consistent with the storyline. In the case of not fulfilling the overall goals these could be even redefined in order to define them in a more realistic way. The set of policy measures which fulfil the overall goals can be thus framed as policy recommendations. However, it should not be discarded that a storyline proves unable to fulfil the overall goals, in this case the policy recommendations would focus into warning against the pursuit of a given storyline.

Since doing this process manually is expected to be highly inefficient, an automatic procedure could be developed to cover the full range of possibilities through multi-objective parameter optimization as shown in the next section.

7.2.3.2. MULTI-OBJECTIVE PARAMETER OPTIMIZATION MODE:

The optimization mode makes sense when the objectives are clearly quantifiable and the system to be optimized is well-known and under control (which may be the case for example in energy system models (Bjelić and Rajaković, 2015)). With IAMs we have precisely the opposite situation due to the many interrelated multi-dimensional uncertainties, which hinders the effectiveness of this type of analysis. Moreover, the real world of course does not work in "optimization mode".

However, this type of analysis offers the possibility to explore the existing operation space towards reaching some goals and help informing policy recommendations, especially if combined with sensitivity analysis. Hence, it is planned to apply the optimization mode on top of the scenarios already defined in

simulation mode to systematically and automatically explore all the possible combination of pre-selected inputs (bounded by plausible bounds) in order to select the combination which provides better outcomes with relation to the overall goals and inform policy-recommendations (see Figure 16). Some work has been already performed with MEDEAS with this method (e.g., (Martelloni et al., 2019)) but it still remains largely unexplored. The definition of the objective function (weighted importance of the different overall goals) represents one of the most complex issues to define these kind of problems, especially relevant when all the overall goals are not achievable simultaneously as it is the case with adaptive scenarios.

Another option would be to perform scenario discovery analysis (Bryant and Lempert, 2010; Gerst et al., 2013; Kwakkel et al., 2013). In this case we could let policies be activated or not randomly in each simulation and ex-post we could check which policies are associated to each type of energy transition. Given the trade-offs and complementarities between different policy measures and instruments, this approach would allow us to uncover unexpected interactions between policies that could lead to new routes towards low-carbon transition.

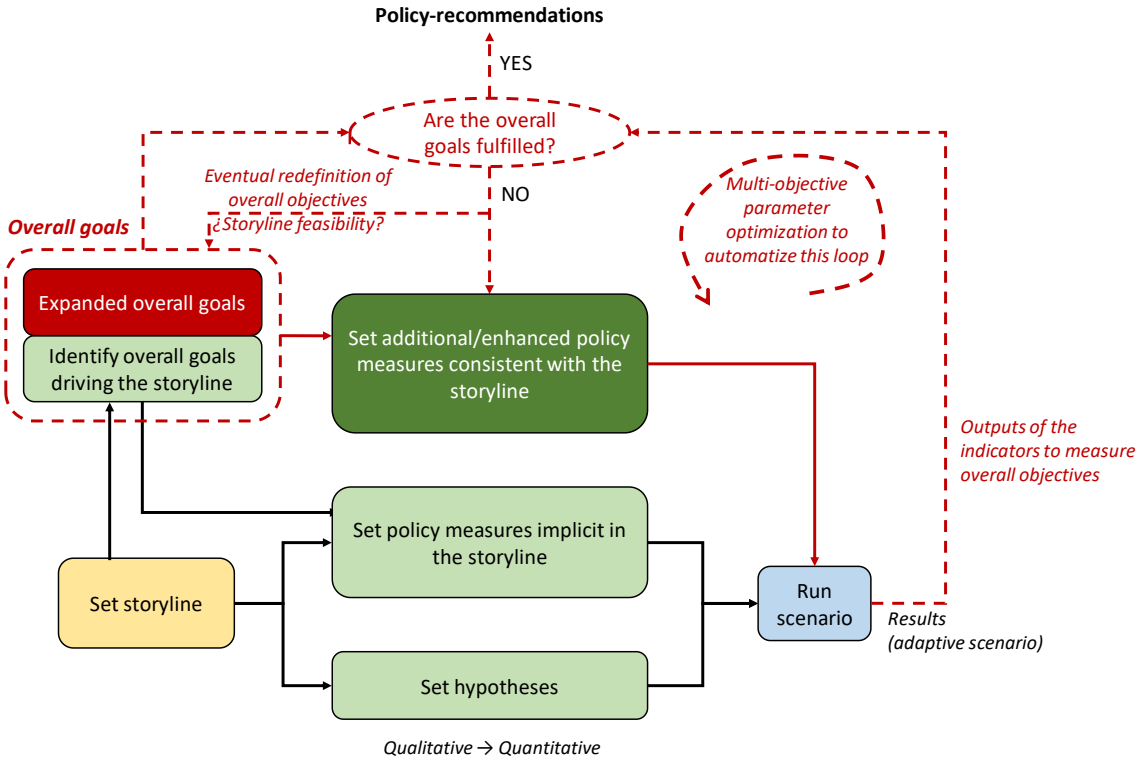


Figure 16: Conceptual representation of the steps to simulate policy scenarios applying an automatic multi-objective parameter optimization with WILIAM.

7.2.3.3. UNCERTAINTY ANALYSIS

Uncertainty analysis can be performed on top of an already defined scenario by applying plausible distributions around the central value of uncertain inputs, see e.g., this applied to a 4 BAU cases applying the MEDEAS model in the Supplementary Material in (Capellán-Pérez et al., 2020). Another example performed with IMAGE is (van Vuuren et al., 2008) which applied conditional probabilistic scenario analysis consisting on the combination of statistical methods of uncertainty analysis at parameter level with storylines in baseline mode (i.e., in the absence of additional climate policies), although this approach is rather rare within IAMs.

Uncertainty analysis can be complemented with sensitivity analysis in order to determine the extent to what the uncertain inputs are responsible for producing uncertainty in the outputs (Saltelli et al., 2004) (e.g., (Samsó et al., 2020)).

Robustness analysis which is part of the validation process can similarly be performed by selection broad ranges for the distributions of the inputs in order to check extreme scenarios.

7.2.3.4. HYBRID MODES

Hybrid modes allow to combine the strengthens of each of the aforementioned running modes. A simulation can be developed e.g., fully deterministically, and later some inputs which are identified to be significantly affected by uncertainty may be run stochastically.

Hybrid modes are even recommended for the formulation of policy recommendations as it is the case of (Morgan and Keith, 2008) which recommend that "variables over which the specific decision makers have no direct or indirect control should be treated probabilistically. However, variables over which those decision makers do have control should typically be treated parametrically so that the analysis provides insight about the implications of decisions or policy choices that could be made."

(D'Alessandro et al., 2020) represent another example of work performing an hybrid mode. In their simulations with the EUROGREENs model the parameters are in general fixed while for some of them for which a relevant uncertainty is identified the model is run stochastically. In their case, they performed n=500 runs for each scenario with different random processes for the extraction of new technologies.

Hybrid approaches need to be applied with care in order to ensure the consistency between all the possible variations of inputs. It might not be coherent to consider some possibilities simultaneously within the same narrative. For example, in (D'Alessandro et al., 2020) some of the policies considered in the Green Growth scenario are not considered in the alternative scenarios because they would be contrary to their narrative. So, it would be unwise to run the model with all the degrees of freedom.

CONCLUSIONS

The overall objective of LOCOMOTION is to enhance the IAM developed in the MEDEAS European project in order to provide policy makers and relevant stakeholders with a reliable and practical model system to assess the feasibility, effectiveness, costs and impacts of different sustainability policy options, and to identify the most effective transition pathways towards a low-carbon society. To achieve this objective, the development of the modules of the new Locomotion IAM (the name chosen by the consortium is "Within Limits IAM" (WILIAM)) is distributed in 4 modelling WPs (WP4, WP5, WP6 and WP7) together with a WP for data management (WP2) and another for developing scenarios and policies (WP8). The technical coordination and integration of the work in these 6 WPs, done through WP9, is hence indispensable for the effectiveness of work.

This deliverable focuses on the common rules and procedures for modelling and programming within the LOCOMOTION project. The different IAM products are characterized, the general structure of the model is outlined and the standards for programming are defined. Efficient coordination also requires agreement in terms of nomenclature of common transversal terms, a section about common nomenclature for policy analysis and scenario assessment has also been included providing definitions as well as a general overview on how WILIAM model will be simulated.

This document, although it will be a necessary initial reference, will be a living document during the development of WILIAM, adapting to new proposals for improvement that arise during the development of software products.

.

APPENDIX

APPENDIX A: ABOUT THE VENSIM SOFTWARE

A.1. THE VENSIM SOFTWARE

Vensim is an industrial-strength simulation software for improving the performance of real systems. Vensim's rich feature set emphasizes model quality, connections to data, flexible distribution, and advanced algorithms. Configurations for everyone - from students to professionals.

Vensim is used for developing, analysing, and packaging dynamic feedback models; providing:

- High quality, with dimensional consistency and Reality Checks™
- Connections to data and sophisticated calibration methods
- Instant output with continuous simulation in SyntheSim
- Flexible model publication
- Model analysis, including optimization and Monte Carlo simulation

Vensim contains many industry-leading technical advances in simulation technology.

- Causal Tracing™
- Subscripting
- Optimization
- Resource Allocation algorithms

A.2. THE VENSIM SOFTWARE FAMILY

Vensim is available in several configurations to fit different modelling needs:

Vensim PLE (Personal Learning Edition) helps you get started with building system dynamics and systems thinking models. Vensim PLE is free for educational or personal use and can be downloaded from our web site.

Vensim Professional allows you to use subscripting for easy handling of detail complexity, contains a text editor, and has optimization capabilities including model calibration and policy optimization.

Vensim DSS enables you to create management flight simulators for models, to customize Vensim by defining macros or external functions, and to link to other programming software through the Vensim DLLs.

There is a set of sample models, as well as an extensive User Guide and a Reference Manual included with Vensim. These demonstrate everything from simple physical systems to large and complex business and social systems. All configurations ship with a Model Reader program, which can be distributed free of charge allowing others to review and simulate models.

The Vensim family of software runs on Windows 7, 8 & 10 and Mac OS10. It has a small memory and disk footprint for a full installation.

A.3. THE MODEL DEVELOPMENT

For the development of LOCOMOTION framework, the Vensim DSS version comes with full Vensim features. For the license (Commercial, Public research, Academic) since it is a public research project it is possible to use the academic version that comes with many advantages in relation to price discounts https://vensim.com/volume-discounts-and-site-license-pricing/

A.4. THE MODEL DISTRIBUTION

The Vensim models can be distributed on desktop with the free Model Reader. The Vensim Model Reader is a free software which allows you to publish models constructed with Vensim and distribute them to other people. Your model and the Vensim model reader can be copied and passed to as many people as you want, giving people access to your model without their needing to purchase Vensim. https://vensim.com/free-download/

APPENDIX B: GEOGRAPHICAL SCOPE IN LOCOMOTION

WILIAM is composed of 9 global regions (some of them are aggregated of countries and others are a country in it-self): EU-27, United Kingdom, China, EASOC (East-Asia & Oceania), India, LATAM (Latin America excepting Mexico), Russia, USMCA (US, Mexico & Canada) and LROW (Locomotion Rest of the World). EU-27 is disaggregated by the 27 EU members.

Table B. 1 the composition of each region with their corresponding ISO code (ISO 3166 is a standard published by the International Organization for Standardization (ISO) that defines codes for the names of countries, dependent territories, special areas of geographical interest, and their principal subdivisions (e.g., provinces or states). Figure B. 1 shows the world map with the LOCOMOTION regions.

Table B. 1: Global LOCOMOTION regions and its different codes.

	LOCOMOTION global regions	LOCOMOTION acronym	Countries	ICIO Code (matches with ALPHA-3 excepting for ROW)	ALPHA-2 code
1	European Union	EU27	Austria	AUT	AT
			Belgium	BEL	BE
			Bulgaria	BGR	BG
			Croatia	HRV	HR
			Cyprus	CYP	CY
			Czech Republic	CZE	CZ
			Denmark	DNK	DK
			Estonia	EST	EE
			Finland	FIN	FI
			France	FRA	FR
			Germany	DEU	DE
			Greece	GRC	GR
			Hungary	HUN	HU
			Ireland	IRL	IE
			Italy	ITA	IT
			Latvia	LVA	LV
			Lithuania	LTU	LT
			Luxembourg	LUX	LU
			Malta	MLT	MT
			Netherlands	NLD	NL
			Poland	POL	PL
			Portugal	PRT	PT
			Romania	ROU	RO
			Slovakia	SVK	SK
			Slovenia	SVN	SI
			Spain	ESP	ES
			Sweden	SWE	SE
2	United Kingdom	UK	United Kingdom	GBR	GB

3	China	China	China (People's Republic of)	CHN	CN
			Hong Kong SAR	HKG	HK
4	East Asia and Oceania	EASOC	Australia	AUS	AU
			Brunei Darussalam	BRN	BN
			Cambodia	KHM	KH
			Chinese Taipei	TWN	TW
			Indonesia	IDN	ID
			Japan	JPN	JP
			Korea	KOR	KR
			Malaysia	MYS	MY
			New Zealand	NZL	NZ
			Philippines	PHL	PH
			Singapore	SGP	SG
			Thailand	THA	TH
			Viet Nam	VNM	VN
5	India	India	India	IND	IN
6	Latin America	LATAM	Argentina	ARG	AR
			Brazil	BRA	BR
			Chile	CHL	CL
			Colombia	COL	CO
			Costa Rica	CRI	CR
			Peru	PER	PE
7	Russia	Russia	Russian Federation	RUS	RU
8	United States, Mexico and Canada	USMCA	Canada	CAN	CA
			Mexico	MEX	MX
			United States	USA	US
9	Rest of the world	LROW	Rest of the world	ROW	-
			Switzerland	CHE	CH
			Iceland	ISL	IS
			Israel	ISR	IL
			Kazakhstan	KAZ	KZ
			Morocco	MAR	MA
			Norway	NOR	NO
			Saudi Arabia	SAU	SA
			Tunisia	TUN	TN
			Turkey	TUR	TR
			South Africa	ZAF	ZA

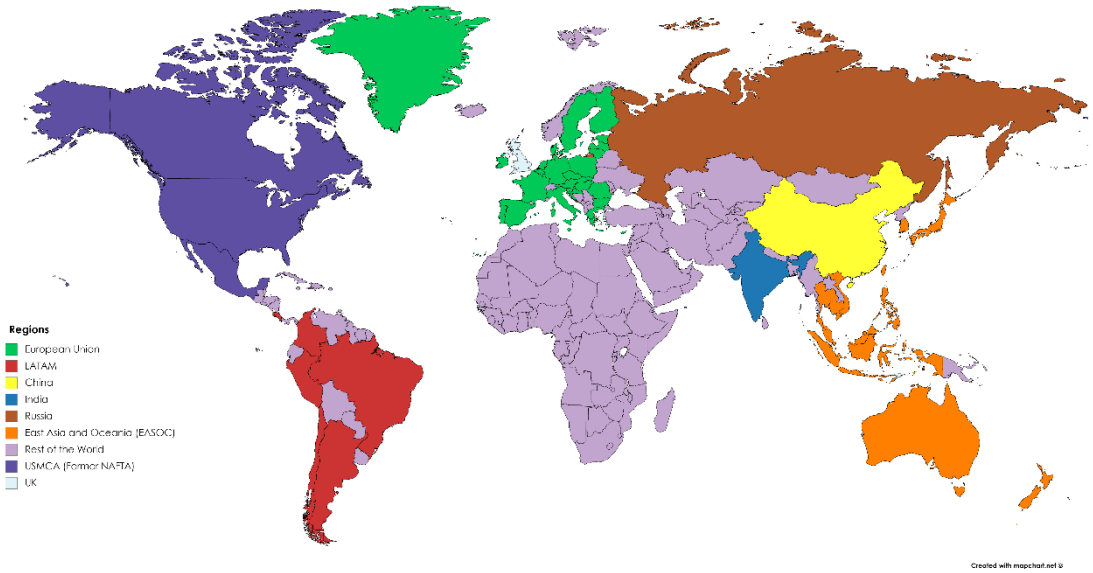


Figure B. 1: World map with the LOCOMOTION global regions.

APPENDIX C: ECONOMIC SECTORS IN WILIAM

A process was launched lead by WP4 and WP9 in order to decide about the economic sectoral disaggregation of WILIAM. This process has experienced the same delays and complexities than the selection of regions given that it is dependent on the IO databases used. On the one hand, the greatest disaggregation level the best since it allows to have more information. On the other hand, the greatest disaggregation level brings closer potential computational issues as well as makes more complex the modelling since more factors need to be taken into account (e.g., more demand functions, endogenization of the coefficients of the matrix A, etc.). Hence, a balance needs to be reached so that the representation of the economic structure is the most functional possible to the objectives of the whole model. Hence, during this process two additional criteria were taken into account:

- Relative importance of each sectors at economic (monetary) level (e.g., in the total aggregated demand),
- Aspects which are planned to be modelled in more detail in the physical part of the model (e.g., energy, minerals, transportation, land, social)

Table C. 1 shows the disaggregation chosen (including the equivalence with NACE Rev. 2 (EUROSTAT, 2008) for further detail on what is included in each sector) with a total of 62 sectors. This disaggregation was accepted in the WP9 workshop in May 2020.

Table C. 1: : Sectoral proposed sectoral disaggregation for WILIAM and equivalence with NACE rev.2 (EUROSTAT, 2008)

LOCOMOTION sector	Equivalence with NACE Rev.2	Description
1	01.1 + 01.2 + 01.3	Crops
2	01.4 + 01.5 + 01.6 + 01.7	Animal production, hunting and related service activities.
3	02	Forestry, logging and related service activities
4	03	Fishing, operating of fish hatcheries and fish farms; service activities incidental to fishing
5	05	Mining of coal and lignite; extraction of peat
6	06.1	Extraction of crude petroleum and services related to crude oil extraction, excluding surveying
7	06.2	Extraction of natural gas and services related to natural gas extraction, excluding surveying
8	09.1	Extraction, liquefaction, and regasification of other petroleum and gaseous materials
9	07.21	Mining of uranium and thorium ores
10	07.1	Mining of iron ores
11	Part of 07.29	Mining of copper ores and concentrates
12	Part of 07.29	Mining of nickel ores and concentrates
13	Part of 07.29	Mining of aluminium ores and concentrates
14	Part of 07.29	Mining of precious metal ores and concentrates
15	Part of 07.29	Mining of lead, zinc and tin ores and concentrates
16	Part of 07.29	Mining of other non-ferrous metal ores and concentrates

LOCOMOTION sector	Equivalence with NACE Rev.2	Description
17	08.	Quarrying on sand, stone and clay; mining of chemical and fertilizer minerals, production of salt, other mining and quarrying n.e.c.
18	10 + 11 + 12	Food products, beverages and tobacco
19	16 + 17 + 18	Wood and products of wood and cork; paper products and printing
20	19.1	Manufacture of coke oven products
21	19.2	Petroleum Refinery
22	20 + 21	Chemicals and pharmaceutical products
23	22	Rubber and plastic products
24	23	Other non-metallic mineral products
25	24	Hydrogen
26	25	Fabricated metal products
27	26	Computer, electronic and optical products
28	27	Electrical equipment
29	28	Machinery and equipment, n.e.c.
30	29 + 30	Motor vehicles, trailers and semi-trailers and other transport equipment
31	13 + 14 + 15 + 31 + 32 + 33	Other manufacturing, including manufacture of furniture, textiles, wearing apparel, leather and related products; repair and installation of machinery and equipment.
32	Part of 35.11	Production of electricity by coal
33	Part of 35.11	Production of electricity by gas
34	Part of 35.11	Production of electricity by nuclear
35	Part of 35.11	Production of electricity by hydro
36	Part of 35.11	Production of electricity by wind
37	Part of 35.11	Production of electricity by petroleum and other oil derivatives
38	Part of 35.11	Production of electricity by solar photovoltaic
39	Part of 35.11	Production of electricity by solar thermal
40	Part of 35.11	Production of electricity n.e.c., including production of electricity by biomass and waste, production of electricity by tide, wave, ocean and production of electricity by Geothermal
41	35.13 + 35.14	Transmission, distribution and trade of electricity
42	35.2	Manufacture of gas; distribution of gaseous fuels through mains
43	35.3	Steam and hot water supply
44	36 + 37 + 38 + 39	Collection, purification and distribution of water; sewerage; waste collection, treatment and disposal activities; materials recovery; remediation activities and other waste management services
45	41 + 42 + 43	Construction
46	45 + 46 + 47	Wholesale and retail trade; repair of motor vehicles
47	49.1 + 49.2	Transport via railways

LOCOMOTION sector	Equivalence with NACE Rev.2	Description
48	49.3 + 49.4	Other land transport
49	49.5	Transport via pipelines
50	50.1 + 50.2	Sea and coastal water transport
51	50.3 + 50.4	Inland water transport
52	51	Air transport
53	55 + 56	Accommodation and food services
54	61 + 62 + 63	Telecommunications, IT and other information services
55	64 + 65 + 66	Financial and insurance activities
56	68	Real estate activities
57	09 + 52 + 53 + 69 + 70 + 71 + 72 + 73 + 74 + 75 + 77 + 78 + 79 + 80 + 81 + 82	Other business sector services, including mining support services and supporting and auxiliary transport activities.
58	84	Public admin. and defence; compulsory social security
59	85	Education
60	86 + 87 + 88	Human health and social work
61	58 + 59 + 60 + 90 + 91 + 92 + 93 + 94 + 95 + 96	Arts, entertainment, recreation and other service activities, including publishing, audiovisual and broadcasting activities
62	97 + 98	Private households with employed persons

"n.e.c": not elsewhere classified
Equivalence with ISIC classification available in (EUROSTAT, 2008).

APPENDIX D: LOCOMOTION DATA DICTIONARY WEBAPP AND WEBSERVICE

This appendix contains the requirement specification (functional, non-functional and information requirements) limited to the purposes of a project centralized data dictionary, its governance (protocol), controlled vocabulary of the project and rules and project dissemination as well. Information requirements, and the conceptual domain built as a result of analysing these requirements, should be enriched with other requirements that are necessary for data management purposes (sources, reliability, etc.), obtaining at the end an integrated database with twofold objective.

In this appendix, a normalized approach to write user stories according to Mike Cohn (Cohn, 2004) is applied. User stories are used to express either functional, non-functional or information requirements.

The user story template used is: **As** a (role/stakeholder) I **want** (something) **so that** (benefit).

Expressing "**so that** (benefit)" is used to prioritize the user stories in the product backlog and/or to determine the value of implementing the user story as well as to keep track of the goals of the different stakeholders.

Definition of the context:

- An IAM development project
- A geographically distributed team, with different persons working in different parts of the code
- A large set of symbols (variables, constants, parameters, etc.) to manage

Definition of the Software Project stakeholders[7] in this context:

- the project coordinator
- the quality assurance team
- users who are not part of the project
- project members with different responsibilities and tasks defined in the protocol described in next section.

D.1. DATA DICTIONARY GOVERNANCE PROTOCOL

The quality assurance team has the mission of defining rules and protocols in a project with the aim of improving the process should improve the quality of the product. Rules and protocols should also help in coordination which should benefit efficiency and efficacy in the project. It is also their mission to supervise how the protocols are followed and to develop or implant tools to help in these tasks.

The data dictionary governance protocol is based on modules and roles on modules. Each modification (insert, update, delete) in the data dictionary should be validated. Users should be authorized to modify the data dictionary according to the following organization:

- All users should be identified in the website that is governing the data dictionary.
- Each user has roles. The role determines the operations the user can accomplish.
- There are three different roles: project general supervisor, module supervisor, module programmer

[7] Stakeholders in this context is referring to the term **Software Project Stakeholder** as "a person, group or company that is directly or indirectly involved in the project and who may affect or get affected by the outcome of the project". What is Stakeholder Identification? It is the process of identifying a person, group or a company which can affect or get affected by a decision, activity or the outcome of the **software project**. It is important in order to identify the exact requirements of the project and what various stakeholders are expecting from the project outcome.

- Module programmer: a user can be designated as module programmer regarding one or more modules. Module programmers can insert and update information in the data dictionary of symbols (variables…) related to the module they authorized on. Each module can have many associated programmers. Any modification accomplished by programmers should be validated by a supervisor.
- Module supervisor: a user can be designated as module supervisor regarding a particular module. The module supervisor can (and should) validate any information that is still pending to be validated. The module supervisor can do any modification the programmers can do. In addition, the module supervisor can delete information. There is just one module supervisor for each module.
- Project general supervisor: a user can be designated as project general supervisor. It is possible in a project to have more than one general supervisor. General supervisors can do any task that module supervisors can do, they are authorized to all modules.

The governance of the data dictionary should proceed according to this protocol:

1- Information is introduced usually by module programmers and it remains pending of validation.
2- A module supervisor or a project general supervisor can review the information and validate it if possible. Supervisors can edit the information in order to adjust it and improve its description.

In the previous protocol the step 1 can be substituted by an automatic injection from Vensim programs (.mdl files). This automatic injection, as manual insertion through the data dictionary webapp remains pending of validation (step 2).

The automation of quality assurance tasks will check that each Vensim program is consistent with the already validated information on the data dictionary (see Appendix E).

D.2. DEFINITION OF SYMBOLS IN THE DATA DICTIONARY OF LOCOMOTION

The conceptual kind of symbols that are going to be considered in Locomotion data dictionary are described in section 4.2.2.

D.3. REQUIREMENTS SPECIFICATION

D.3.1. FUNCTIONAL REQUIREMENTS AS USER STORIES

FR-1. **As** the project coordinator, I **want** that the users registered in the system have three types of roles: module programmer, module supervisor and general supervisor **so that** the centralized data dictionary aids in implantation of the protocol established in the project (see Section 3) for the definition and validation of symbols (see IR-03).

FR-2. **As** a module programmer, I **want** to be able to introduce metadata into the data dictionary **so that** it makes easier to share information about the symbols defined in the module with other project members, and additionally to facilitate coordination as well as quality assurance, model dissemination, and comprehension. The metadata registered by a module programmer will remain pending of review and validation by the module supervisor or a general supervisor to keep the data dictionary controlled according the quality assurance protocol.

FR-3. **As** a module programmer, I **want** to modify the metadata previously introduced into the data dictionary **so that** decisions made throughout the project development are up to date. The metadata that a module programmer modifies becomes pending of review and validation by the module supervisor or a general supervisor to keep the data dictionary controlled according to the quality assurance protocol.

FR-4. **As** a module supervisor, I **want** to delete the symbol and its metadata **so that** decisions made throughout the project development are up to date.

FR-5. **As** a module supervisor, I **want** to review entered metadata and validate it **so that** the integrity, coordination, and quality of the information in the data dictionary can be assured.

FR-6. **As** the project coordinator, I **want** that the metadata that a module programmer can introduce or modify must be relative to the module or modules in which the programmer participates **so that** the data dictionary remains controlled within the quality assurance protocol.

FR-7. **As** the project coordinator, I **want** that the metadata that module supervisors can validate and/or modify must be relative to the module under their responsibility **so that** the data dictionary remains controlled within the quality assurance protocol.

FR-8. **As** the module supervisor, I **want** to receive alerts when metadata are introduced or modified **so that** I can review and validate them. These alerts will be shown in the webapp. In order not to send invasive emails with every modification, a summary email will be sent, by default, every day only in case new alerts exist. “Every day” frequency can be modified (“every week, every 3 days”).

FR-9. **As** the module supervisor, I **want** to run all the tasks that a module programmer can do, referred to the module of which I am responsible **so that** I can act in case of unavailability or overload of work of the programmers, or in case of modification/correction before validating is required.

FR-10. **As** a module supervisor, I **want** to create acronyms, adjectives and semantic rules in the data dictionary **so that** the work of naming symbols, and the tasks of modelling and programming are facilitated to project members, as well as the understanding of the project by people outside it.

FR-11. **As** a module supervisor, I **want** to modify or delete acronyms, adjectives and semantic rules in the data dictionary **so that** the decisions taken throughout the project development are kept updated in the data dictionary.

FR-12. **As** a module supervisor, I **want** to be able to obtain the data dictionary public information as a Word file in a human-readable format **so that** sharing information with different stakeholders and generating deliverable documentation are facilitated.

FR-13. **As** a general supervisor, I **want** to run all the tasks that a module supervisor can do (referred to all modules) **so that** I can act in case of unavailability or overload of work of the module supervisor, or in case of modification before validating is required.

FR-14. **As** a general supervisor, I **want** to add modules to the project **so that** the project structure is outlined, and responsibilities related to these modules can be assigned.

FR-15. **As** a general supervisor, I **want** to modify the module information **so that** the decisions taken throughout the project development are kept updated in the data dictionary.

FR-16. **As** a general supervisor, I **want** to delete a module **so that** the decisions taken throughout the project development are kept updated in the data dictionary. In the case the module to be deleted has users with assigned roles and symbols associated with it as its main module, the system will enable to reassign these user roles or symbols to another module or to delete them one by one in a controlled way. Cascade deletions[8] of user roles or symbols will not be allowed.

FR-17. **As** a general supervisor, I **want** to add a user assigning the user at least one role **so that** the project members can run actions in the data dictionary according to the tasks defined for each role. The module supervisor's role requires indicating the module the user is responsible for. The module programmer's role requires indicating the modules in which the programmer participates (one or more). The general supervisor's role is not associated with any module or modules in particular but with all modules.

FR-18. **As** a general supervisor, I **want** to add a module to the user as a programmer so that he/she can participate in various modules.

FR-19. **As** a general supervisor, I **want** to add a new role to a user so that the user can participate in one or various modules as a programmer and as a supervisor of a certain module at the same time.

FR-20. **As** a general supervisor, I **want** to be able to delete a user **so that** any change of the staff in the project can keep up to date in the data dictionary.

[8] Cascade delete is a technical term in database terminology to describe a characteristic that cause deletion of a row to automatically trigger the deletion of related rows, or delete data from child tables automatically when you delete the data from the parent table.

FR-21. **As** a general supervisor, I **want** to modify or delete a user's role **so that** any change of the staff in the project can keep up to date in the data dictionary or risks regarding staff unavailability can be mitigated.

FR-22. **As** a general supervisor, I **want** to add/modify/delete the values for ProjectTypeOfValue, ProgrammingLanguageSymbolType and their association **so that** I can adapt the data dictionary to the context of the project (see IR-4, IR-11 and IR-12).

FR-23. **As** a registered user of the system, I **want** to consult all the existing information in the data dictionary regarding modules, symbols and their characterization, categories (and subcategories; therefore, Symbols (see IR-03) are categorized) , acronyms, adjectives, and semantic rules **so that** the project tasks are facilitated. The information should be shown filtered according to it is already validated or not.

FR-24. **As** the project coordinator, I **want** that people outside the project can consult all the information in the data dictionary without having to register as a user, avoiding personal data storage **so that** the transparency and dissemination of the project are guaranteed, at the same time that GDPR is fulfilled. The information visible to unregistered users excludes all the aspects related to users and their roles in the project.

Other requirements that are still pending of precise definition:

FR-25. **As** a user, I **want** to obtain the data dictionary public information as an Excel file following the format specified in XXXX *(definition of the format is not ready yet*) **so that** information sharing is facilitated, as well as comparisons with other similar projects, according to the indications of IAMC, IPCC Data Distribution Center (see AR5).

D.3.2. NON-FUNCTIONAL REQUIREMENTS AS USER STORIES

NFR-01. **As** the project coordinator, I **want** the data dictionary interface to be a webapp **so that** access, as well as maintenance (system updates), is facilitated.

NFR-02. **As** the project coordinator, I **want** the webapp interface to have a responsive design **so that** it adapts well to different sizes of devices' screens.

NFR-03. **As** the project coordinator, I **want** users with roles in the project must show their credentials to perform the tasks associated with their roles **so that** the security of the information contained in the data dictionary is guaranteed.

NFR-04. **As** the project coordinator, I **want** users cannot run other tasks for which they are not authorized **so that** the protocol for defining and validating symbols in the project is fulfilled.

NFR-05. **As** the project coordinator, I **want** to make available the data dictionary functionalities through a RESTful API **so that** the automation of tasks is facilitated.

NFR-06. **As** the quality assurance team, we **want** the functionality for adding symbols and their information via RESTful API is a priority. It should be available at the time the data dictionary service is running so that this information can be automatically injected from the existing Vensim programs.

NFR-07. **As** the quality assurance team, we **want** available a RESTful API service that, giving a list of symbol names, returns (as a JSON response which format is presented in Section 5) all the information regarding the so-named symbols, if they are stored and already validated in the data dictionary **so that** the Vensim programs can be automatically checked.

NFR-08. **As** the project coordinator, I **want** to have a token-based-authentication system between services **so that** the security of the RESTful API operations is ensured and hence the integrity of the data dictionary.

NFR-09. **As** the project coordinator, I **want** both registered users and unregistered users external to the project can filter the information in the data dictionary either by exact match or by similarity (substring-based) **so that** usability is improved.

NFR-10. **As** the project coordinator, I **want** the system to be available at least 340-350 days a year, 24 hours a day, **so that** its usage by users in different time zones is facilitated, and at the same time to allow performing maintenance tasks that require stopping the system's services.

NFR-11. **As** the project coordinator, I **want** the whole system (source code, interfaces, notifications, messages and RESTful API as well) to be written in English **so that** communication between researchers and other stakeholders in an international environment in general and European environment, in particular, is facilitated.

D.3.3. INFORMATION REQUIREMENTS

The system should store information on:

IR-01. User: all users will have a username, full name, and email address. A user has at least one role in the data dictionary governance protocol. A user can have one or two different roles. A user can be a programmer in one or several modules and can be a supervisor of a module. A user that is a general supervisor will have just one role (general supervisor) since the authorization levels of this one includes the others.

IR-02. Module: a module will have a name that will be unique in the project. The symbols defined in the module and the users authorized on the module (as programmers or as a module supervisor) will be associated.

IR-03. Symbol: symbols are the main elements of the data dictionary. Each symbol will have a name, which will be unique in the project, an explanation, the units and the type of symbol (its role in the project). When symbols do not have units, they will be indicated as Dimensionless.

IR-04. ProjectTypeOfValue: The type of the symbol in the project indicates what role the symbol plays: variable, historical data, parameter, constant, scenario parameter, index, index element, switch (as stated in 4.2.2).

IR-05. Symbols, indexes and their cases, as well as categories, must be validated and their state must reflect if they have already been validated.

IR-06. All symbols have a category associated with them. A category hierarchy of 2 levels maximum, can be defined. In the case of a category-subcategory hierarchy, the symbol must reflect which subcategory it belongs to.

IR-07. When the symbols are multidimensional (disaggregated variables) or historical series, they will have one or more indexes associated with them.

IR-08. Each index has associated the different values or cases that the said index represents. For example, REGIONS_I would be an index, and its values or cases would be ASIA, USA, EU, ...

IR-09. In the aim of completing the definition of the project controlled vocabulary, acronyms (letters and meaning) such as "EROI": "Energy Return on Investment" and adjectives (the word and its form of use in the project) such as "total": "cumulative value ….For example …" are stored in the data dictionary. These acronyms and to be used in the symbols' names.

IR-10. In the aim of better explaining project domain constraints, semantic rules (a name for short and explanation) are stored. For example, "conservation of energy": "the total energy of an isolated system remains constant; it is said to be conserved over time. This law means that energy can neither be created nor destroyed; rather, it can only be transformed or transferred from one form to another.".

IR-11. ProgrammingLanguageSymbolType: The modules of the project are programmed using some language. The data dictionary stores the types of symbols that can be used in the programming language. Vensim is the language used in the Locomotion project. The kind of symbol in Vensim stored in this data dictionary are: Variable, Constant, Subscript, Subscript element, Variable subscripted, Constant subscripted, lookup, reality check, and function.

IR-12. The data dictionary stores a relationship between the types of symbols in the project (ProjectTypeOfValue) and the types of symbols in the programming language (ProgrammingLanguageSymbolType). For example, having Vensim as programming language, an index is defined through a subscript, a series of historical data can be defined with a lookup table or a subscripted variable, a parameter can be defined with a switch (a variable with, for instance, 0, 1,

2, 3 as possible values which is used in constructions of type IF ... THEN ... ELSE IF ... THEN ... ELSE ...). This relationship is used as a guide for symbol validation.

A task force group focused on the assessment of uncertainty in the WILIAM is currently developing a method to identify the level of uncertainty of the most relevant model and scenario parameters which will likely derive in additional information requirements to be provided to the data dictionary.

D.4. REQUIREMENT ANALYSIS: CONCEPTUAL MODELLING

Figure C.1 shows a conceptual model as an UML class diagram as a result of analysing the requirements. The diagram highlights in red the changes introduced compared with the previous version of the conceptual model.

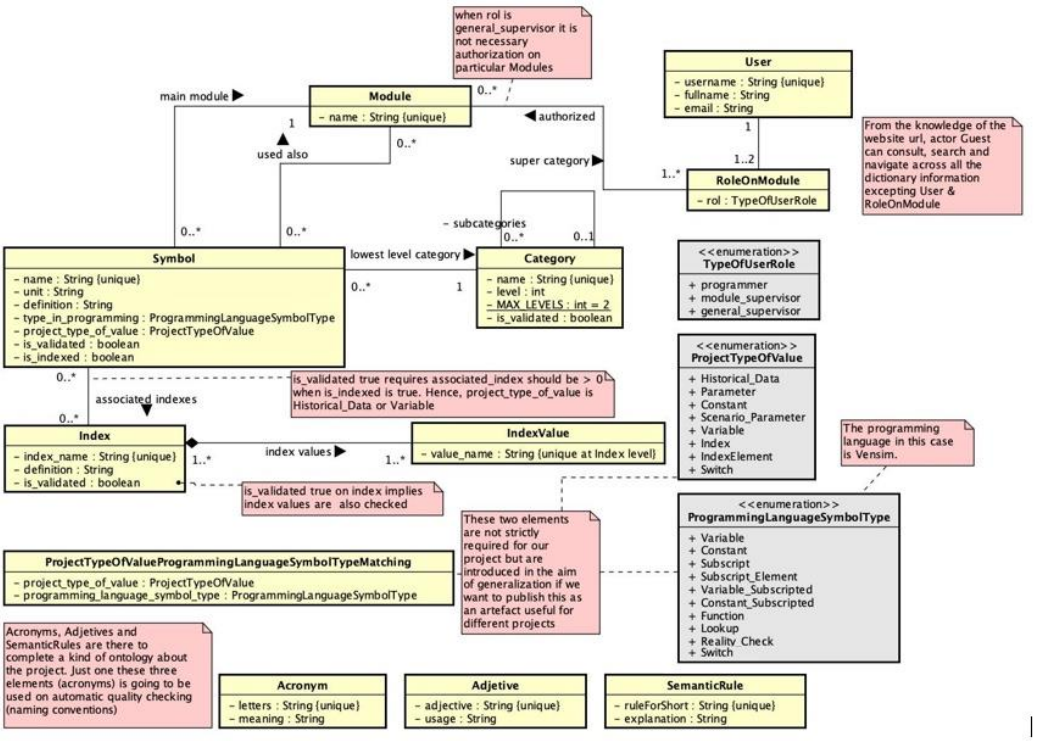


Figure C. 1: Conceptual model as an UML class diagram

D.5. DESIGN: DATABASE SCHEMA

Based on the conceptual model obtained as a result of the requirements analysis, we propose the following scheme (Figure C. 2) for the database that must store the information regarding the project centralized data dictionary, its governance (protocol) and the controlled vocabulary of the project:

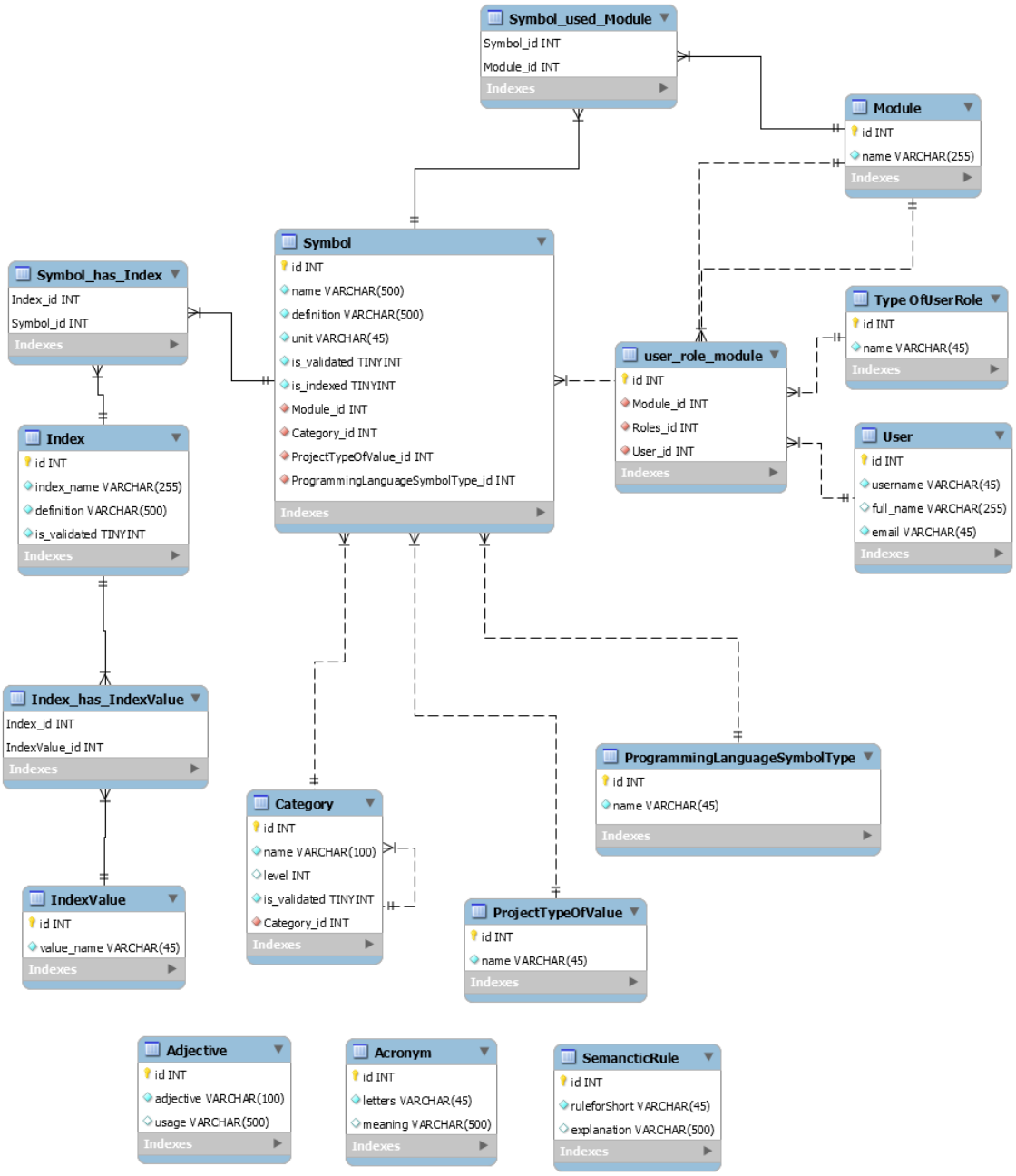


Figure C. 2:Relational design

D.6. RESTFUL API SERVICES AND DATA EXCHANGE FORMAT DEFINITION

Service name	Input	Output
qaGetSymbolsDefinition	{"symbols": ["SymbolExample1", "SymbolExample2"]}'	{"symbols": [{"name": "SymbolExample1", "definition": "definition example for symbol 1 example", "unit": "Kg", "indexes": [], "modules": [{"main": "IAM Module One", "secondary": []}], "category": "CategoryExampleTopLevel", "project type of value": "Scenario_Parameter", "programming symbol type":"Constant" }, {"name": "SymbolExample2", "definition": "definition example for symbol 2 example", "unit": "N", "indexes": ["IndexExample1", "IndexExample2"], "modules": [{"main": "IAM Module One", "secondary": ["Testing_modules"] }], "category": "CategoryExampleBottomLevel", "project type of value": "Scenario_Parameter", "programming symbol type":"Constant" }], "modules": ["IAM Module One", "Testing_modules"], "indexes": [{"name": "IndexExample1", "definition": "definition example 1", "values": ["ValueExample1"] }, {"name": "IndexExample2", "definition": "definition example 2", "values": ["ValueExample2", "ValueExample3"] }], "categories": [{"name": "CategoryExampleTopLevel", "level": 1, "super_category": "null"}, {"name": "CategoryExampleTopLevel", "level": 1, "super_category": "null"}

Service name	Input	Output
		] } or Refused request (security token incorrect or insufficient privileges, incorrect input json format)

qaAddSymbolsDefinition	{"symbols":
	[
	{
	"name":"SymbolExample1",
	"definition":"Example definition",
	"unit": "N",
	"category": "category1",
	"is indexed": "true",
	programming symbol type
	
	},
	{
	"name":"SymbolExample2",
	"definition":"Another example definition",
	"unit": "Kg",
	"category": "category1",
	"is indexed": "false"
	}
	],
	"module": "IAM Module One",
	"indexes": [
	{
	"index_name":"SymbolExample1",
	"definition":"Index definition",
	"values":
	[
	"value1",
	"value2"
	]
	},
	{
	"index_name":"SymbolExample2",
	"definition":"Index definition number two",
	"values":
	[
	"value1",
	"value3"
	]
	}
	]
	}
		NONE
		or
		Refused request (security token incorrect or insufficient privileges, incorrect input json format)

Usage examples:

```
curl -X GET
'https://url.of.datadictionary/qaGetSymbolsDefinition/PGUcZgkN_y1DpWuo
dzzz' -H "content-type: application/json" -d '{"symbols":
["SymbolExample1", "SymbolExample2"]}'
```

```
curl -X POST
'https://url.of.datadictionary/qaAddSymbolsDefinition/PGUcZgkN_y1DpWuo
dzzz' -H "content-type: application/json" -d '{"symbols":
[{"name":"SymbolExample1", "definition":"Example definition", "unit":
"N"}, {"name":"SymbolExample2", "definition":"Another example
definition", "unit": "Kg"}], "module": "IAM Module One", "indexes":
[{"index_name":"SymbolExample1", "definition":"Index definition",
"values": ["value1", "value2"]}, {"index_name":"SymbolExample2",
```

```
"definition":"Index definition number two", "values": ["value1",
"value3"]}]}'
```

The SonarQube plugin developed for the .mdl files quality checking will use both services in order to i) check consistency between symbols in Vensim models and the centralized data dictionary, and ii) automatically inject to the data dictionary new symbols from Vensim models which remain pending of validation.

APPENDIX E: ABOUT VENSIM PLUGIN FOR SONARQUBE

A Vensim plugin for SonarQube containing 23 rules has been deployed. These 23 rules can be consulted on the server itself by following the link below:

https://sonarqube-locomotion.infor.uva.es/coding_rules?activation=true&qprofile=AXOfhF8X2CxLD-ZIAslW

There is an automatic way to run the quality check that is configured in gitlab-locomotion.infor.uva.es:

When a modeler pushes the local changes to the repository in gitlab-locomotion.infor.uva.es, the check automatically starts. Each module is configured to check only what is produced in that module. The results can always be reviewed at https://sonarqube-locomotion.infor.uva.es/. The wiliam project is configured to check the entire .mdl file.

Each project set up for a module follows a naming convention about the module's views. If any view does not start with the module name, nothing that is defined in that view will be checked by the sonarqube plugin in that module/gitlab project. Hence, it is a priority to apply the naming convention for the views because otherwise there are issues that are only going to come out when the full wiliam will be fully checked (see section 4.4).

It is important to remember that the rules implemented are the ones that can be automatically checked. There are some other rules and conventions that should be checked manually. When all the rules for a given symbol are manually checked, it can be validated by supervisors in the Data Dictionary.

The automatic injection to the shared Data Dictionary is also fully functional. Only symbols without any violation of the programming rules and naming conventions are automatically injected. Several rules programmed in the SonarQube are prepared for the moment when we have symbols validated at the centralized data dictionary and a mismatch occurs between the version in the .mdl file and the information in the data dictionary.

Some other utilities have also been implemented in the Vensim plugin for SonarQube. To configure the activation/deactivation of the utilities the sonar-project.properties file should be used:

- A file with the differences found between the symbols in .mdl and the symbols in the data dictionary can be obtained. To use this utility in the sonar-project.properties file we must activate the property: *vensim.dictionary.getDiff=true*
- The injection to the data dictionary can be enabled or disabled using the property: *vensim.dictionary.inject.*
- A log with all messages exchanged with the SonarQube server when performing the quality control and the communication with the data dictionary can be obtained using the property *vensim.log.serverMessages=true*
- Where the diff summary file and other auxiliary files such as a detailed information of all symbols and the file with the message log mentioned before are stored can be configured using the *properties vensim.log.file* and *vensim.auxiliaryFiles.directoryName* as shown:

  #-Set the name of the file where the log generated will be saved.

  #-If unset, log will be shown in the terminal.

  vensim.log.file=log.txt

#-Set the folder name where all generated files will be stored.

#-(Default: auxiliary_files). If you want a different folder name such as other_folder_name use:

vensim.auxiliaryFiles.directoryName=other_folder_name

A documented version of the sonar-project.properties file have been updated to the current branches in the repository.

Utility to detect discrepancies between the model and the data dictionary information

In collaboration of UVa with Panos Stratis from CRES, we have defined a service that allows the deletion of a list of symbols from the data dictionary at once. It is explained here:

https://gitlab-locomotion.infor.uva.es/locomotion/data-dictionary/-/issues/22

This feature, together with the *getDiff* utility, would help to decide at some point of the model development to detect the case when several symbols are no longer in the .mdl file but are still in the data dictionary and we want to delete them all at once. This would also help in the case where we have several validated symbols in the data dictionary that are inflicting mismatch with the current state of the symbols in the mdl file.

In that case, there are two options,

a) you want to fix what is in the mdl to meet what the data dictionary says or

b) you want to fix the data dictionary to be like the current version of the mdl.

In case of b), although it can be done directly in the web interface, it is better to inject when there are many of them. The defined service can be used to delete the list of symbols in that case and let the automatic reinjection of the symbols do the work. You can also delete the symbols one by one directly in the web interface.

Finally, there is a way to use the quality check locally in your computer, without using the automated way prepared to be launched when pushing to gitlab. The interested person has to install an application called sonar-scanner. It can be obtained from https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/ (for Windows, Linux and Mac). If you want to use it, please speak with yania@infor.uva.es. You have to be careful with the definition of properties in the sonar-project.properties file according to the tests you want to do.

APPENDIX F: ABOUT THE VENSIM PLUGIN FOR SEMANTICMERGE

Details on installation and usage of the developed Vensim plugin for SemanticMerge integrated in gmaster can be found at the online manual:

https://ecm.cartif.es/share/page/site/locomotion/document-details?nodeRef=workspace://SpacesStore/2b449951-96ff-40a6-9d53-3d74754408ff

Figure F 1 shows the gmaster interface including the SemanticMerge marked with red and blue numbered rectangles in order to facilitate the below explanation.

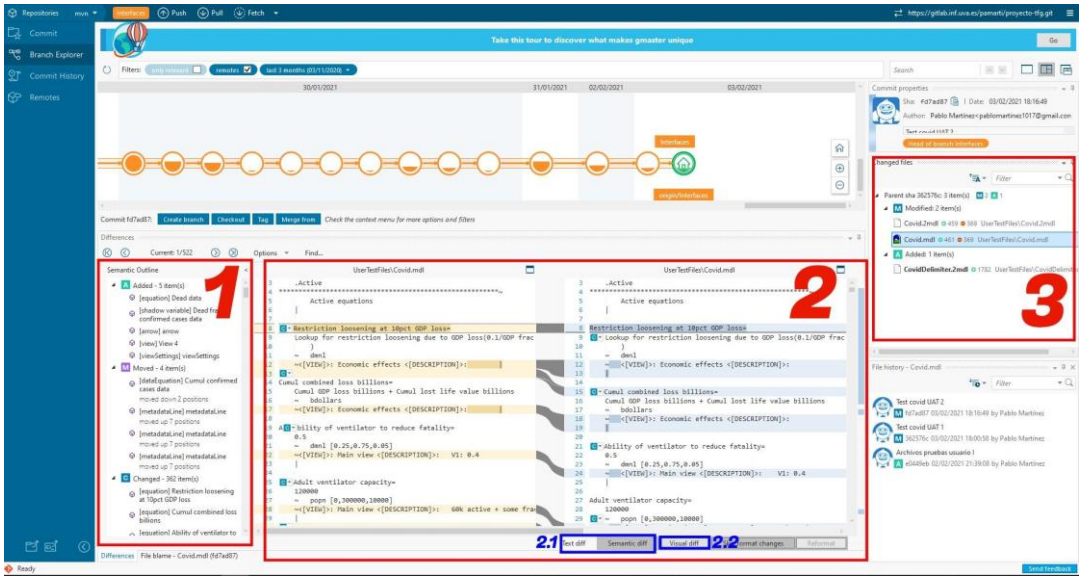


Figure F 1: gmaster interface with the Vensim pluing SemanticMerge.

In the first section marked with a red box and the number 1, we can observe the first new feature added by the tool. In this section you can see the modification count between two versions of the same file. As can be seen in the figure, the modifications are divided in five main sections. Within each of these categories, in turn, we can distinguish the kind of modified element as it can be an equation and a view, as well as its name. The different kinds of modification are explained below:

- **Added**. A new element has been added to the file. It is represented by a green-coloured **A**.
- **Changed**. An element of the file has been modified. It is represented by a blue-coloured **C**.
- **Deleted**. An element of the file has been removed. It is represented by a red-coloured **D**.
- **Moved**. An element of the file has been moved within its text format. It is represented by a purple-coloured **M**.
- **Renamed**. An element of the file has been renamed. It is represented by a purple-coloured **R**.

Figure F 2 shows the graphic symbols used to more easily distinguish each kind of modification.

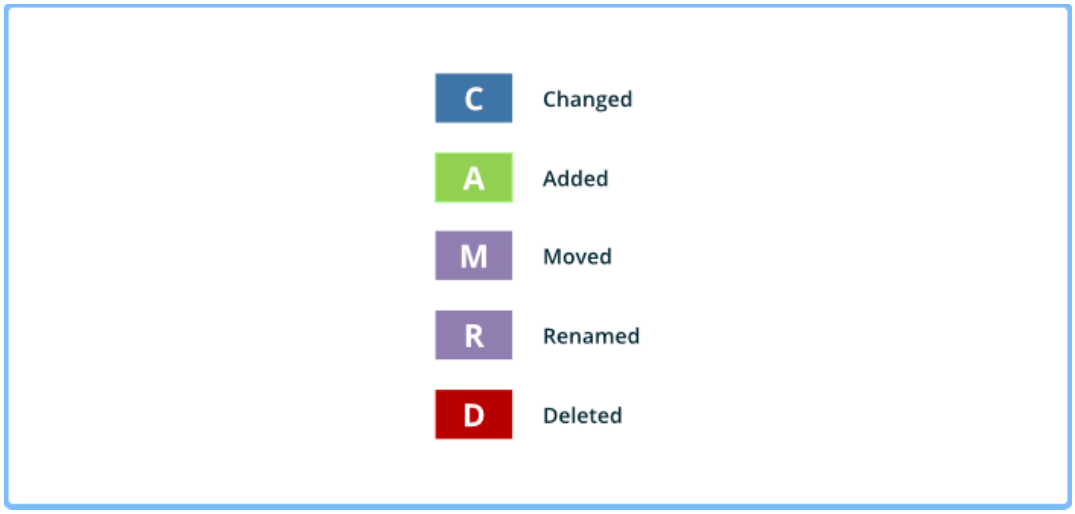


Figure F 2: Graphic symbols used to more easily distinguish each kind of modification

In the second section marked with a red box and the number 2, the two versions of the file are being analysed, showing the oldest version on the left and the current version on the right (in the case of a *merge*, the version of the source branch and the version of the destination branch would be shown). There are coloured strips that visually relate the changes between both versions of the file. An initial appears next to each change, which indicates the category of modification from among the five possible ones explained above (C, A, M, R and D).

The blue box labelled 2.1 (at the bottom of section 2), is used to switch between the conventional mode of differences between versions with plain text, and SemanticMerge's own semantic mode.

The second box in blue, labelled 2.2, showcases the changes in a more compact and visual way, using only the graphics associated with the five possible categories of modification. Figure 2.7 in the online tutorial shows the result of selecting this way of visualising the differences.

Finally, the third section shows the modified files in the version of the selected project or *commit*.

For further detailds, see the full tutorial "Development of a SemanticMerge plugin to facilitate the integration of Vensim files versions" available online.

REFERENCES

Barlas, Y., 1996. Formal aspects of model validity and validation in system dynamics. System Dynamics Review 12, 183–210. https://doi.org/10.1002/(SICI)1099-1727(199623)12:3<183::AID-SDR103>3.0.CO;2-4

Bjelić, I.B., Rajaković, N., 2015. Simulation-based optimization of sustainable national energy systems. Energy 91, 1087–1098.

Böck, E., Eggler, L., Roher, M., Papagianni, S., Christodoulaki, R., Taxeri, E., 2020. Review of policy options to drive societies towards sustainability (LOCOMOTION DELIVERABLE https://www.locomotion-h2020.eu/ No. D8.1). LOCOMOTION h2020, Valladolid, Spain.

Bryant, B.P., Lempert, R.J., 2010. Thinking inside the box: A participatory, computer-assisted approach to scenario discovery. Technological Forecasting and Social Change 77, 34–49. https://doi.org/10.1016/j.techfore.2009.08.002

Capellán-Pérez, I., Blas, I. de, Nieto, J., Castro, C. de, Miguel, L.J., Carpintero, Ó., Mediavilla, M., Lobejón, L.F., Ferreras-Alonso, N., Rodrigo, P., Frechoso, F., Álvarez-Antelo, D., 2020. MEDEAS: a new modeling framework integrating global biophysical and socioeconomic constraints. Energy Environ. Sci. https://doi.org/10.1039/C9EE02627D

Cohn, M., 2004. User Stories Applied: For Agile Software Development. Addison-Wesley.

D'Alessandro, S., Cieplinski, A., Distefano, T., Dittmer, K., 2020. Feasible alternatives to green growth. Nature Sustainability 1–7. https://doi.org/10.1038/s41893-020-0484-y

EUROSTAT, 2008. NACE Rev. 2 - Statistical classification of economic activities in the European Community [WWW Document]. URL https://ec.europa.eu/eurostat/web/products-manuals-and-guidelines/-/KS-RA-07-015 (accessed 4.24.20).

Gerst, M.D., Wang, P., Borsuk, M.E., 2013. Discovering plausible energy and economic futures under global change using multidimensional scenario discovery. Environmental Modelling & Software, Thematic Issue on Innovative Approaches to Global Change Modelling 44, 76–86. https://doi.org/10.1016/j.envsoft.2012.09.001

Girod, B., Wiek, A., Mieg, H., Hulme, M., 2009. The evolution of the IPCC's emissions scenarios. Environmental Science & Policy 12, 103–118. https://doi.org/10.1016/j.envsci.2008.12.006

Kriegler, E., Edmonds, J., Hallegatte, S., Ebi, K.L., Kram, T., Riahi, K., Winkler, H., van Vuuren, D.P., 2014. A new scenario framework for climate change research: the concept of shared climate policy assumptions. Climatic Change 122, 401–414. https://doi.org/10.1007/s10584-013-0971-5

Kwakkel, J.H., Auping, W.L., Pruyt, E., 2013. Dynamic scenario discovery under deep uncertainty: The future of copper. Technological Forecasting and Social Change, Scenario Method: Current developments in theory and practice 80, 789–800. https://doi.org/10.1016/j.techfore.2012.09.012

Lai, D., Wahba, R., 2001. System Dynamics Model Correctness Checklist, SYSTEM DYNAMICS IN EDUCATION PROJECT SLOAN SCHOOL OF MANAGEMENT MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

Malczynski, L.A., 2011. Best practices for System Dynamics model design construction with Powersim Studio. Sandia National Laboratories.

Martelloni, G., Di Patti, F., Perissi, I., Falsini, S., Bardi, U., 2019. MEDEAS-World model calibration for the study of the energy transition. arXiv:1906.01997 [physics].

MEA, 2005. Millennium Ecosystem Assessment. Ecosystems and Human Well-being: Scenarios, Global Assessment Reports. Island Press, Washington DC (USA).

Morgan, M.G., Keith, D.W., 2008. Improving the way we think about projecting future energy use and emissions of carbon dioxide. Climatic Change 90, 189–215. https://doi.org/10.1007/s10584-008-9458-1

Pastor, A.V., Vieira, D.C.S., Soudijn, F.H., Edelenbosch, O.Y., 2020. How uncertainties are tackled in multi-disciplinary science? A review of integrated assessments under global change. CATENA 186, 104305. https://doi.org/10.1016/j.catena.2019.104305

Riahi, K., van Vuuren, D.P., Kriegler, E., Edmonds, J., O'Neill, B.C., Fujimori, S., Bauer, N., Calvin, K., Dellink, R., Fricko, O., Lutz, W., Popp, A., Cuaresma, J.C., Kc, S., Leimbach, M., Jiang, L., Kram, T., Rao, S., Emmerling, J., Ebi, K., Hasegawa, T., Havlik, P., Humpenöder, F., Da Silva, L.A., Smith, S., Stehfest, E., Bosetti, V., Eom, J., Gernaat, D., Masui, T., Rogelj, J., Strefler, J., Drouet, L., Krey, V., Luderer, G., Harmsen, M., Takahashi, K., Baumstark, L., Doelman, J.C., Kainuma, M., Klimont, Z., Marangoni, G., Lotze-Campen, H., Obersteiner, M., Tabeau, A., Tavoni, M., 2017. The Shared Socioeconomic Pathways and their energy, land use, and greenhouse gas emissions implications: An overview. Global Environmental Change 42, 153–168. https://doi.org/10.1016/j.gloenvcha.2016.05.009

Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M., 2004. Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. John Wiley & Sons.

Samsó, R., de Blas, I., Perissi, I., Martelloni, G., Solé, J., 2020. Scenario analysis and sensitivity exploration of the MEDEAS Europe energy-economy-environment model. Energy Strategy Reviews 100582. https://doi.org/10.1016/j.esr.2020.100582

SSP db, 2018. SSP Database (Shared Socioeconomic Pathways) - Version 2.0 (December 2018). Available at: https://tntcat.iiasa.ac.at/SspDb.

Sterman, J.D., 2000. Business dynamics: systems thinking and modeling for a complex world. Irwin/McGraw-Hill Boston.

UN, 2020. Global indicator framework adopted by the General Assembly (A/RES/71/313), annual refinements contained in E/CN.3/2018/2 (Annex II), E/CN.3/2019/2 (Annex II), and 2020 Comprehensive Review changes (Annex II) and annual refinements (Annex III) contained in E/CN.3/2020/2 (No. A/RES/71/313, E/CN.3/2018/2, E/CN.3/2019/2, E/CN.3/2020/2).

van Vuuren, D.P., de Vries, B., Beusen, A., Heuberger, P.S.C., 2008. Conditional probabilistic estimates of 21st century greenhouse gas emissions based on the storylines of the IPCC-SRES scenarios. Global Environmental Change, Local evidence on vulnerabilities and adaptations to global environmental change 18, 635–654. https://doi.org/10.1016/j.gloenvcha.2008.06.001

van Vuuren, D.P., Kok, M.T.J., Girod, B., Lucas, P.L., de Vries, B., 2012. Scenarios in Global Environmental Assessments: Key characteristics and lessons for future use. Global Environmental Change 22, 884–895. https://doi.org/10.1016/j.gloenvcha.2012.06.001