**GUILDELINES FOR THE MODULARIZATION OF WILIAM**

Contents

[1. Objectives of modularization: 1](#_Toc132294516)

[2. Modularization 1](#_Toc132294517)

[3. Modularization SWITCHes 2](#_Toc132294518)

[3.1. Types of SWITCHes 2](#_Toc132294519)

[3.2. Providing alternative data for running disconnected from the rest of WILIAM 3](#_Toc132294520)

[4. Programming of SWITCHes 4](#_Toc132294521)

[5. Maintenance of modularization - update of SWITCHES 5](#_Toc132294522)

Modularization refers to structuring the model in smaller parts which are internally coherent (modules and submodules). Given that WILIAM is a very large and complex model, modularization eases model development and validation, allowing the model to be simulated in a customized way isolating modules and submodules from the rest of the model. This task requires as a first step to identify all the links between modules (intermodule links). Modularization is crucial for successful model development and validation, but does not need to be included in the publically released versions of the model.

# Objectives of modularization:

* Development of the (sub)module and validation: due to the large amount of links and 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., those variables where the SWITCHES affect).
* Eases the comparison of results obtained by two different modules performing the same function and integrated in the model (structural uncertainty).
* Eases the comparison of results with and without a link or feedback, i.e., showing the value-added of the integration of different dimensions within the same framework. This feature can be very useful for papers.
* Eases extracting one module from the whole WILIAM and running it in a different .mdl standalone (which can be practical to speed up development, perform calibration, etc.).
* 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.
* Vensim submodels feature (<https://www.vensim.com/documentation/sub-models.html>) was reviewed but assessed not to be practical in the current stage of LOCOMOTION project.

# Modularization

WILIAM modularization is reflected in the following aspects of the model:

1. Folder structure of model parameters: one folder by module
2. Folder structure of model ToDo&Documentation: one folder by module
3. Structure of scenario\_parameters.xlsx file: one tab by module or submodule
4. Specific excel to load the value of the SWITCHES: switches.xlsx, located in the same folder than scenario\_parameters.xlsx (/scenario\_parameters/) since it is closely related with model simulation. Including them in a separate excel will also facilitate eliminating them before publicly releasing the model.
5. .mdl:
   1. the name structure of Vensim views (module-name\_of\_module.category), also reflected in the *data dictionary*.
   2. Introduction of SWITCHES which allow to activate/deactivate different parts of the model.

Loading the values of the SWITCHes through excel instead than from WILIAM has another key advantage: if only the values from SWITCHes are changed, it will not be necessary to compile the code each time, hence improving simulation speed substantially.

The next section specifically focuses on the programming of SWITCHES.

# Modularization SWITCHes

SWITCHES can take only 2 values:

* 0: disconnected (OFF)
* 1: connected (ON)

The by-default value=1 if a module or connection is validated, =0 if not validated.

To systematize the SWITCHes it is proposed to establish the following acronyms for each module:

Table

|  |  |
| --- | --- |
| **Module full name** | **Acronym** |
| ECONOMY | ECO |
| FINANCE | FIN |
| ENERGY | NRG |
| MATERIALS | MAT |
| LAND\_AND\_WATER | LAW |
| CLIMATE | CLI |
| DEMOGRAPHY | DEM |
| SOCIETY | SOC |

## Types of SWITCHes

4 different types of SWITCHes are considered (besides one more type specifically related with policies in order to activate/deactivate policies and which is differentiated by the module SWITCHES presented here by the suffix ’\_SP’):

1. **Module SWITCHes**. Refer to submodules within a module, or internal links inside a module. Name: SWITCH\_FULL-NAME-OF-MODULE (e.g., SWITCH\_ECONOMY):
   * 0: the module runs isolated from the rest of WILIAM, replacing inter-module variables coming from other modules with exogenous parameters.
   * 1: the module runs integrated with the rest of WILIAM, the inter-module variables are endogenous coming from other modules
2. **Transversal SWITCHes.** Name: SWITCH\_NAME-OF-SWITCH. This option is useful when one SWITCH affects transversally different modules across the whole model (e.g., SWITCH\_CLIMATE\_CHANGE\_DAMAGE refers to all represented climate change impacts in different modules)
3. **Submodule SWITCHes**. Name: SWITCH\_MODULE-ACRONYM\_NAME-OF-SUBMODULE (e.g., SWITCH\_ECO\_HOUSEHOLDS)
   * 0: the submodule runs isolated from the rest of WILIAM, replacing inter-submodule variables coming from other submodules with exogenous parameters.
   * 1: the submodule runs integrated with the rest of WILIAM, the inter-submodule variables are endogenous coming from other submodules
4. **Inter-module link SWITCHes**. Name: SWITCH\_MODULE-ACRONYM-ORIGIN-MODULE-2-ACRONYM-RECIPIENT\_NAME-LINK (e.g., SWITCH\_DEM2ECO\_HOUSEHOLDS\_NUMBER).

* 0: link between modules deactivated
* 1: link between modules activated
* If the link involves more than one module, the rule will be to name it SWITCH\_X2-ACRONYM-RECIPIENT\_NAME-LINK (e.g., SWITCH\_X2DEM\_LIFE\_EXPECTANCY\_AT\_BIRTH).

The creation of submodules is optional and up to the modellers, in case they judge useful to structure their module in smaller units.

## Providing alternative data for running disconnected from the rest of WILIAM

It is necessary to provide an alternative to the case when the submodule runs isolated from the rest of WILIAM or the link between the modules is deactivated. 2 options are proposed:

Load specific exogenous data. The exogenous parameters should be named including the prefix “EXO” to the name of the variable (e.g., EXO\_NAME\_OF\_THE\_VARIABLE). These data could be timeseries or static data.

Use the INITIAL(A) function of Vensim (<https://www.vensim.com/documentation/fn_initial.html>). This function returns the value A at initialization (i.e., for the year 2005) and does not change it during a simulation. This option has the advantage to avoid loading external data but cannot however be always used:

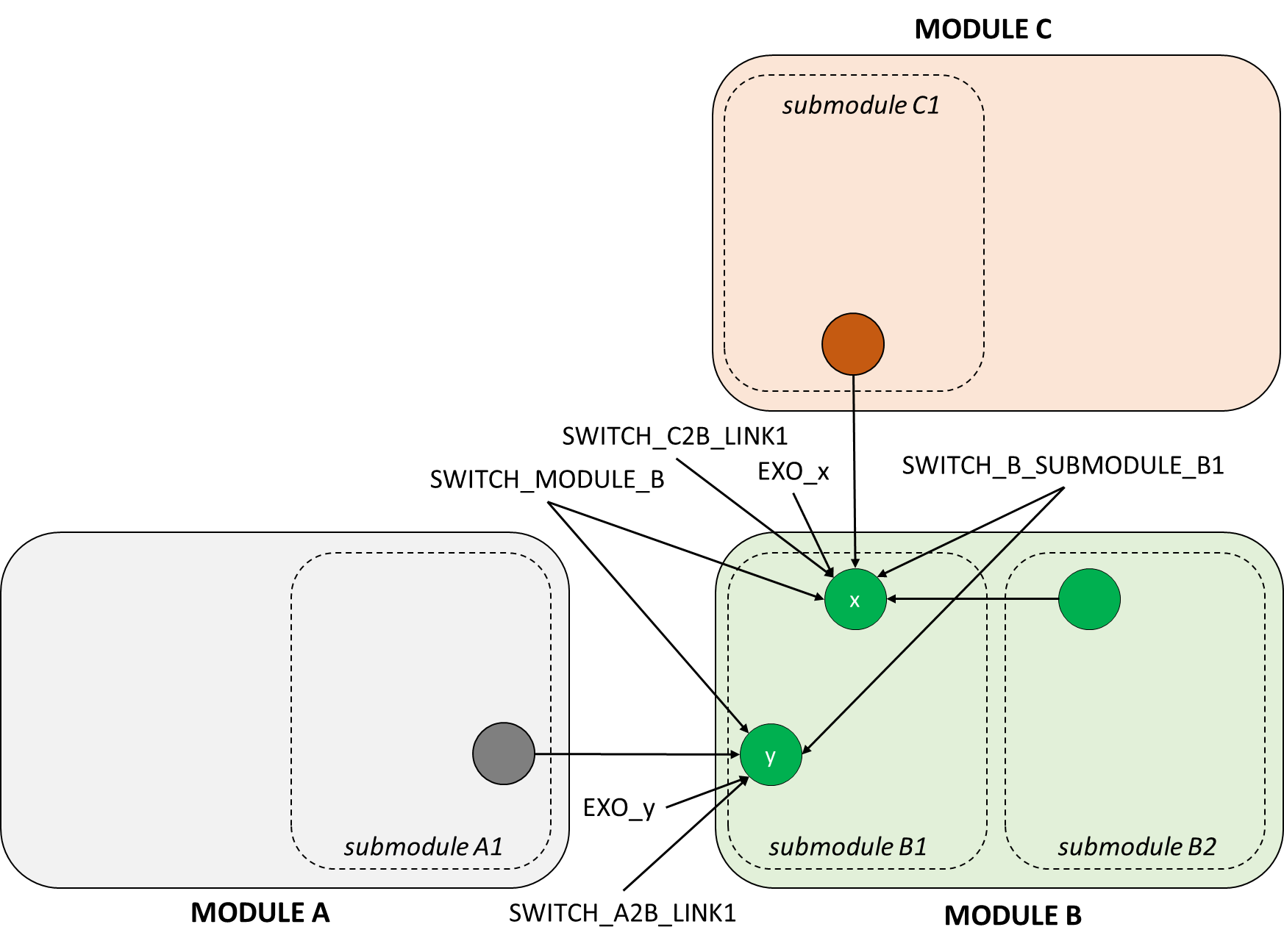
If a developer wants to have full control on the inputs to its module,

If a developer desires to be able to extract the module and run standalone in a separated .mdl.

2005-2015 initialization issue for different modules

when there is a feedback loop broken with a delayed\_TS variable

shows an example of the proposed nomenclature for the different SWITCHES affecting 2 variables from the MODULE B (x and y):



Figure

# Programming of SWITCHes

Each intermodule variable is affected by at least 2 SWITCHES (3 in case it belongs also to a pre-defined submodule). There are different options to program these SWITCHES depending on the hierarchy we want to consider between these options, see below Table 2 (1) nested IF THEN ELSE, (2) OR , (3) AND functions for the case of the variable ‘y’ as defined in Figure 1 above:

(1) IF THEN ELSE (SWITCH\_MODULE\_B = 0, EXO\_y,

IF THEN ELSE (SWITCH\_A2B\_LINK1=0, EXO\_y,

y ))

(2) IF THEN ELSE (SWITCH\_MODULE\_B=0 :OR: SWITCH\_A2B\_LINK1=0, EXO\_y, y ))

(3) IF THEN ELSE (SWITCH\_MODULE\_B=0 :AND: SWITCH\_A2B\_LINK1=0, EXO\_y, y ))

Table 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Value of SWITCHes | SWITCH\_MODULE\_B | 0 | 0 | 1 | 1 |
| SWITCH\_A2B\_LINK1 | 0 | 1 | 0 | 1 |
| Result obtained | (1) nested IF THEN ELSE functions (with priority to SWITCH\_MODULE\_B) | EXO\_y | EXO\_y | EXO\_y | y |
| (2) :OR: | EXO\_y | EXO\_y | EXO\_y | y |
| (3) :AND: | EXO\_y | y  (this would allow testing this individual feedback with the rest of the module isolated) | y  (this would allow testing the module linked with the rest of WILIAM excepting for this link) | y |

As shown in Table 2 the first remark is that the options nested (1) IF THEN ELSE and (2) OR are equivalent. The :OR: option is preferred due to its more compact modelling. Both the :OR: and :AND: options are useful in different ways, but it is proposed to use the :OR: option as a standard given that it matches better with the steps in model development: first validate modules and then feedbacks. However, modellers are invited to do all the checks and tests they consider relevant to validate the model.

A consequence of this programming with :OR:, is that all the conditions are met in order to be active, so a column of SWITCH dependencies has been included in switches.xlsx to inform model developers/users about these dependencies.

# Maintenance of modularization - update of SWITCHES

If a connection or module is validated, then the developer should (1) change the value of the SWTICH from 0 to 1 in switches.xlsx, and mention in the comment that the validation has been done.

If a new connection or module is added, the SWITCH should be created in switches.xlsx.

Model Supervisors should take especial care when integrating checking that the new SWITCHES have been included properly in switches.xlsx.