MIND STEP

MODELLING INDIVIDUAL DECISIONS TO SUPPORT THE EUROPEAN POLICIES RELATED TO AGRICULTURE

Deliverable D7.1: Prototype of a wrapper to allow a standardized communication channel between the models

AUTHORS	Marc Müller (WR), Alexander Gocht (Thuenen)
APPROVED BY WP MANAGER:	Alexander Gocht (Thuenen)
DATE OF APPROVAL:	06.07.2021
APPROVED BY PROJECT COORDINATOR:	Hans van Meijl (WR)
DATE OF APPROVAL:	09.07.2021
CALL H2020-RUR-2018-2	Rural Renaissance
WORK PROGRAMME Topic RUR-04-2018	Analytical tools and models to support policies related to agriculture and food - RIA Research and Innovation action
PROJECT WEB SITE:	https://mind-step.eu

This document was produced under the terms and conditions of Grant Agreement No. 817566 for the European Commission. It does not necessary reflect the view of the European Union and in no way anticipates the Commission's future policy in this area.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

TABLE OF CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

ACRONYMS

AML	Algebraic Modelling Language
GAMS	General Algebraic Modelling System
GDX	GAMS Data Exchange Format
GGIG	GAMS Graphical Interface Generator
GUI	Graphical User Interface
OpenMI	Open Modelling Interface
SEAMLESS	System for Environmental and Agricultural Modelling; Linking European Science and Society
SEAMLESS-IF	SEAMLESS Integrated Framework

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

EXECUTIVE SUMMARY

The MIND STEP toolbox is a suite of models intended to be a modular framework where functionality can be added with additional models and data and where the models can be executed in flexible combinations. The latter requires to embed the models in the toolbox following protocols developed in other work packages (namely WP2.2, WP3.1 and WP4.1) by using software wrappers which permit the execution of each model from an external environment.

This deliverable 7.1. describes a prototype for a wrapper function targeting a common type of model in the MIND STEP toolbox. Starting points are an overview on typical model workflows and a pragmatic definition and operationalization of the wrapper concept. Experiences from the SEAMLES project, which integrated a set of conceptually different models using wrapper functions are reviewed. An important observation is that the environment from which the wrappers are called should permit the handling of complex data structures. Further, it should be an environment modellers without IT training are familiar with.

The agro-economic simulation models in the MIND STEP toolbox are implemented in an Algebraic Modelling Language (AML) like the General Algebraic Modelling System (GAMS), some of them make use of an external interface that facilitates model execution, data entry, and result exploitation.

Based on these observations, a wrapper function implemented in the R programming language is proposed. It permits the execution of a GAMS-based simulation model, exploiting some of the features of the applied user interface for model set-up and parallel execution, and the exchange of complex data structures between models. The R language facilitates handling of complex data structures and is an environment model developers within MIND TSEP are familiar with, thus the wrapper function shown here can serve as a prototype to facilitate the integration of the MIND STEP toolbox.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1. INTRODUCTION

The MIND STEP suite of models is intended to be a modular framework where functionality can be added with additional models and data and where the models can be executed in flexible combinations. The latter requires to embed the models in the toolbox following protocols developed in other work packages (namely WP2.2, WP3.1 and WP4.1) by using software wrappers which permit the execution of each model from an external environment. The wrapper concept reduces the work to adjust the models for the toolbox and ensures that arguments and data are standardized and well documented. While it would be preferable to design a generic wrapper, the manifold solutions foreseen in this project with respect to programming languages and the number of interlinkages, specific wrappers will be needed. Testing of these wrappers will take place continuously basis and build up the technical basis for validation work in WP6. The test concept forces models to be modularized into testable components, which in turn results in a modular and flexible code architecture.

The implementation of a simulation model usually involves data preparation, model set-up and parameterization, and reporting as depicted by Figure 1, which are usually separated from model equations (separation of code from data). Particularly when relying on statistical sources, data preparation must deal with outliers or missing entries that can impair model execution, performance and more so plausibility of results. This underlines that the generation of the model database is an integral part of the model workflow, particularly because it is instrumental for the model set-up and parametrization in a subsequent step before the model itself is solved.

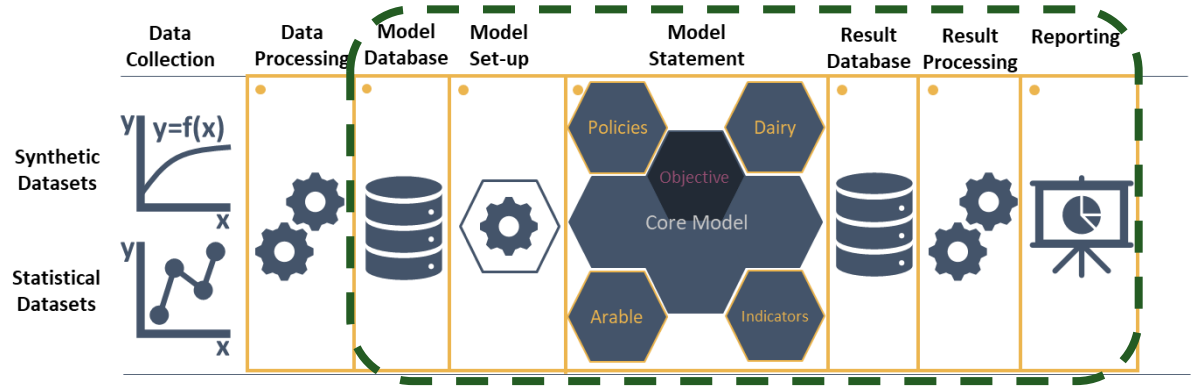

Figure 1 Typical Model Workflow

Source: Britz et al. 2021

Restricting data preparation, parameterization, model solving and reporting to the currently needed farm branches, farming systems or relevant policies greatly eases model application. A block of equations and variables with the related code-blocks for data preparation and reporting, for instance for dairy farming, can be jointly understood as a module if it can be switched off without impairing the use of the core model and other modules. The activation of modules can be data or user-driven. Such a modular design is defined by Russell (2012) as:

"Modularity describes specific relationships between a whole system and its particular components. A modular system consists of smaller parts (modules) that fit together within a predefined system of architecture. Modules feature standardized interfaces, which facilitate their integration with the overarching system architecture. A key feature of each module is that it should encapsulate (or "black box") its messy internal details […] to display only a consistent interface. The designers of modular systems are therefore able to swap modules in a 'plug-and-play' manner, which increases the system's flexibility." (Russell, 2012)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Flexibility in configuring the simulation model is required for a generic model. A modeller may not be interested in applying all aspects of a generic model for a given use case. Instead, modules directly relevant for the research question will be activated and others switched off, for instance by including a specific set of policies or an alternative objective function. Such flexibility in model set-up keeps each instance of the model at manageable size and facilitates the parameterization from a case-study specific database. On the other hand, this flexibility increases the increases the complexity of setting up a particular instance of the model. Several simulation models in the MIND STEP toolbox feature a graphical user interface (GUI), sometimes realized with the „GAMS Graphical Interface Generator" GGIG (Britz 2014), to facilitate, for instance, choosing the included modules or the database to use. In the case of the FARMDYN model, for instance, the GUI permits constructing parts of the model database, the selection of modules, the steering of the solving algorithm, and the exploitation of results (highlighted by the green dashed frame in Figure 1). In addition, the parallel execution of many model instances is also governed by the GUI. To include the models in a workflow that is implemented in an external programming environment, at least some of the functionality of the GUI, like parallelization, has to be made available to the user as well.

The following chapters will address the challenges and solutions regarding the development of wrapper functions that permit the set-up and combined use of models in the MIND STEP toolbox. After a definition and operationalization of the wrapper concept, some examples of wrappers used in previous projects are discussed. Finally, a prototype solution for a wrapper in the MIND STEP toolbox is presented, which permits the set-up and execution of the FARMDYN model from an R environment.

2. THE WRAPPER CONCEPT

In general terms, a wrapper can be understood as a function in a given programming environment that calls another, more complex function, provides the required arguments, and executes it. The complexity of the wrapped function is reduced for the user by providing many of the requiring arguments as defaults, thus limiting the additional user input. A simple example [1] from the R programming language would be a function that uses a basic mean(data, removeNotAvailable) function which takes two arguments: the data for which the mean should be computed and the information how non-available data (NA) points should be treated. For example, they can either be included as zero values in the mean calculation, or they can be excluded entirely from the calculation. If for some reason a program requires a general solution for the treatment of NA values, a wrapper could be a function that calls the original mean function but sets the argument removeNotAvailable to "True". In the R programming language, such a wrapper function would be implemented like this:

```
mean_alwaysremoveNA ← function(data) {
        return(mean(x, removeNotAvialable = True))
}
```

In this case, the user has only to provide the data and needs not to worry about other settings of the wrapped function.

A more complex example is the qplot() function from the ggplot2[2] package in R, which permits the creation of advanced plots. It is a shortcut for the rather complex ggplot() function and is designed for

[1] https://stackoverflow.com/questions/44783295/wrapper-function-in-r

[2] https://ggplot2.tidyverse.org/reference/qplot.html

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

users familiar with more basic plotting functions in R. The function qplot() is therefore a convenience wrapper for creating selected types of plots using a consistent calling scheme. More complex plots will require the user to apply ggplot(), so it is not possible to change all default settings when calling ggplot() through qplot()..

These two examples highlight also the important characteristics of the wrapper concept with regard to the MIND STEP toolbox: The wrapper function has to enable the programmer to send information about the data to be used to the target model, activate or deactivate selected modules, and execute the model, so that retrievable results are generated. These requirements are depicted in Figure 2 for a model like FARMDYN, which is be governed by a GUI that does not only facilitate user inputs, but provides also crucial services for, e.g. model parallelization. The wrapper function would take a list of arguments ("[*,*,*,..]") that executes the relevant functionalities of the GUI, changes at least some parts of the model database, provides information on how the model should be set up, and triggers the relevant modules in the model statement (the "*" in Figure 2 indicate which blocks of the simulation model are affected).

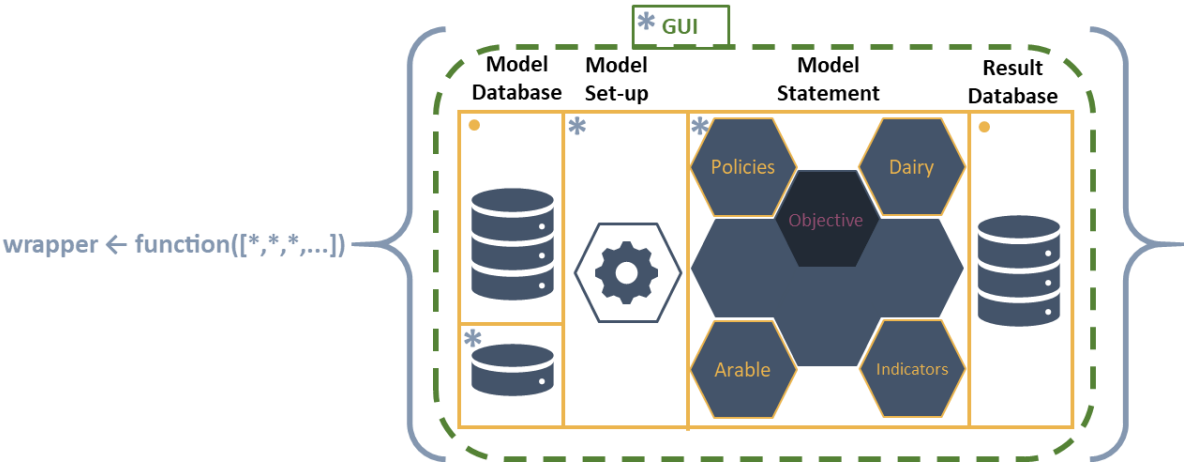

Figure 2 Wrapper as function call to a simulation model

3. WRAPPERS IN AGRO-ECONOMIC MODELLING

3.1. SEAMLESS-IF

The project SEAMLESS (System for Environmental and Agricultural Modelling; Linking European Science and Society)[3] developed a computerized and integrated framework (SEAMLESS-IF) to assess the impacts on environmental and economic sustainability of a wide range of policies and technological improvements across a number of scales (van Ittersum, et al., 2008). Within this integrated framework (SEAMLESS-IF, Rizzoli et al., 2009)), different type of models are linked into model chains, where each model uses the outputs of another model as its inputs and ultimately indicators are calculated. SEAMLESS-IF is based on a layered, client-server architecture. The processing environment facilities, in particular the model chain executor, are deployed on the SEAMLESS server. An important component of this architecture is the SeamFrame modelling framework that was purposely designed to develop integrated assessment tools and offers a series of facilities to encapsulate and wrap existing models for execution by the processing

[3] https://www.seamless-ip.org/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

environment. It allows to deliver model components wrapped by a SeamFrame specific interface, compliant with the Open Modelling Interface (OpenMI) standard (www.openmi.org), so that it can be executed by a processing environment. Although the SeamFrame application server operates in Java environment, the model components can be implemented using other languages as long as they can be integrated. This requirement has been translated into the fact that model components should be OpenMI compliant, by implementing the OpenMI interfaces and allow linking to other components (for data exchange). The model component can take care of this by itself or a wrapper or bridge can be programmed.

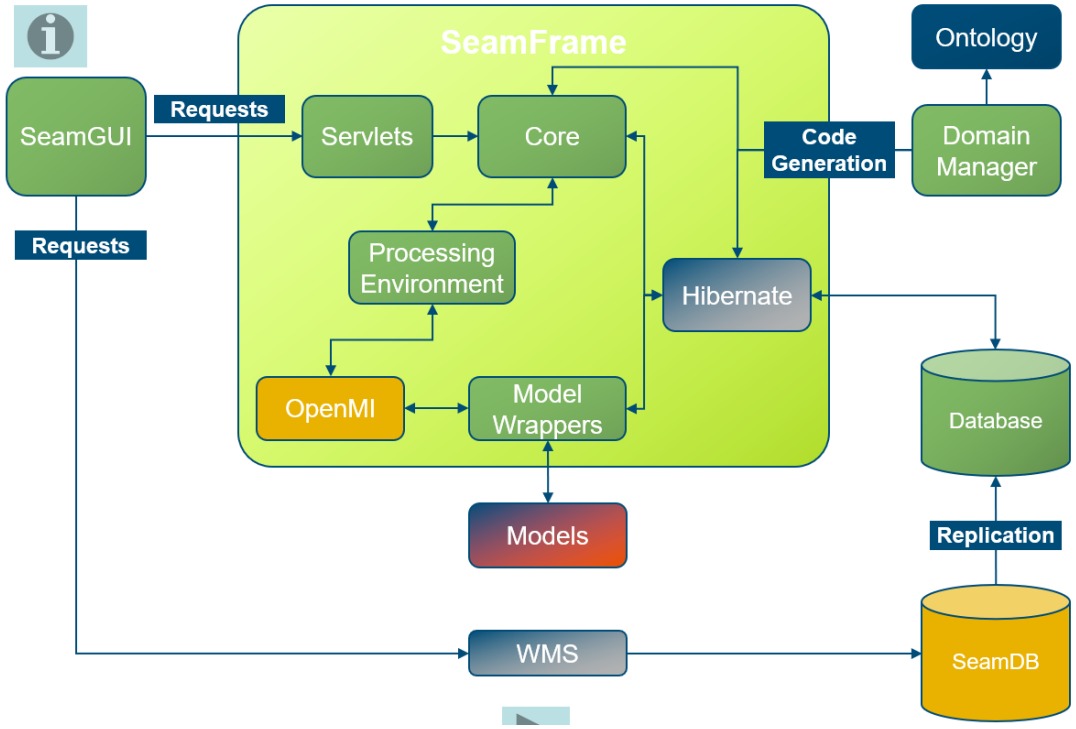

Figure 3 SeamFrame overview

Source: https://www.seamless-ip.org/DVD_Consortium_SEAMLESS/1_Summary_Overview/Features%20SEAMLESS-IF.pps

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

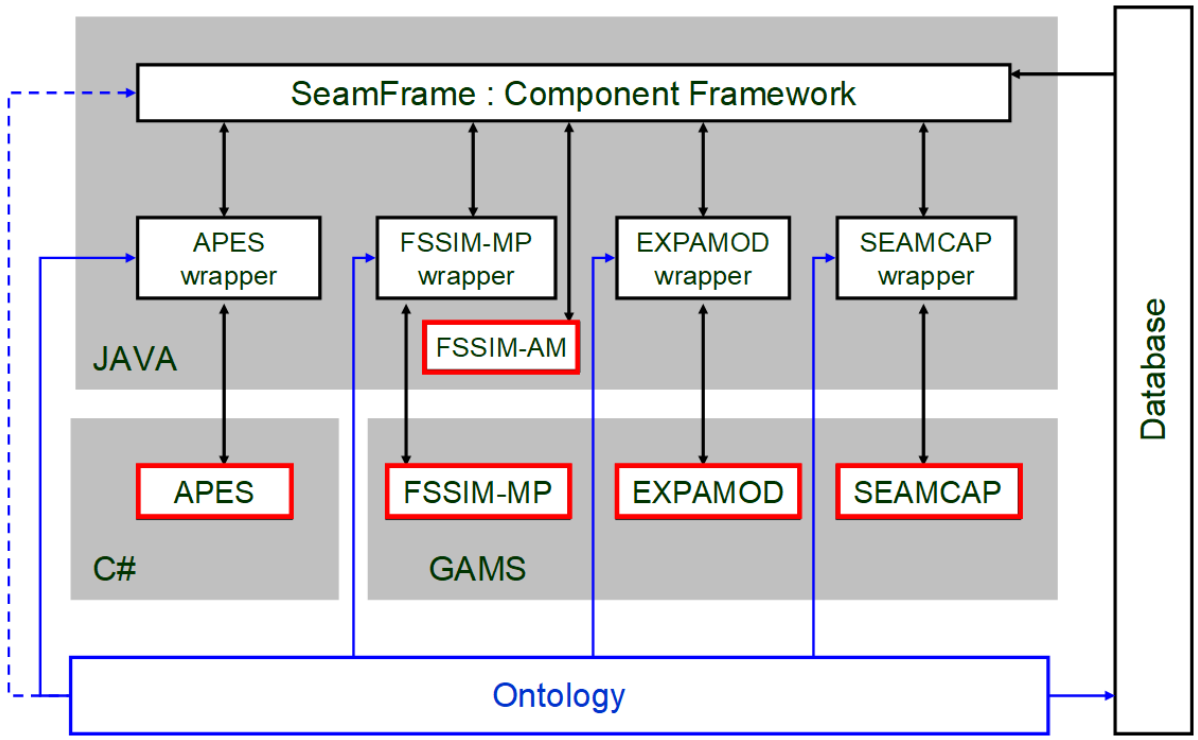

Figure 4 SeamFrame components

Source: https://www.seamless-ip.org/DVD_Consortium_SEAMLESS/1_Summary_Overview/Features%20SEAMLESS-IF.pps

https://doi.org/10.1016/j.agsy.2007.07.009

OpenMI however requires the development of an OpenMi compatible wrapper around the GAMS projects themself. Concepts such as the SEAMLESS-IF are therefore probably only suitable for larger projects focusing on combining components based on different programming languages. Furthermore, SEAMLESS-IF is based on a client/server implementation and requires specific software licences for deployment. Altogether, experiences with SEAMLESS-IF were mixed, which may be caused by the lack of experience among modellers with general purpose programming languages like JAVA (Britz and Kallrath, 2012).

The core of an economic simulation model consists of numerical problem(s) that require a simultaneous solution for all equations. This contrasts with environmental models that often integrate smaller components and are solved recursively in space and time, which asks for a modular design. Compared to economic models, the resolution of environmental models in space and/or time is often higher, but the number of items simulated tends to be smaller.

Simultaneous solution approaches in economic models tend to combine all variables and their relations present in the overall problem in one module. Such simulation models solve for many items such as different types of prices, output generation, primary factor and intermediate input use, trade, demand categories, differentiated by product and space, and eventually time. They are frequently build in a template structure, implemented in an Algebraic Modelling Language (AML). This means that model equations are structurally identical across space and products, and where applicable across periods, while differences are expressed in parameters. AMLs allow efficient coding of such template based models.

Economists rely mostly on equations, and, therefore, favour AMLs, whereas environmental modellers more often additionally use graphical presentations, such as flow charts. They might, therefore, favour object-oriented approaches or even frameworks that allow building models with the help of GUIs. This is

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

in turn a possible overlap between economic and environmental modelling: Models implemented in an AML, consisting of many files for data entry, handling, model set-up, execution, and reporting, tend to become rather complex and difficult to handle. It is for this reason that many economic models, particularly those in MIND STEP, make use of a GUI to govern their models. A prominent example is discussed in the next section.

3.2. GGIG Batch Execution

The GAMS Graphical Interface Generator (GGIG, Britz 2012) is a GUI for the execution of complex GAMS programs. The GUI itself is programmed in JAVA and can be tailored to specific GAMS projects with the help of an XML file. Apart from data entry and exploitation of results, GGIG provides a set of advanced features facilitating the parallel execution, testing, and documentation of GAMS projects. Within the MIND STEP toolbox, the models FARMDYN, IFM-CAP, AGRISPACE, and GLOBIOM make use of GGIG, so they are already structured and implemented in a way that permits taking advantage of the GUI's features. One particular feature of the GUI relevant here is the batch execution facility. This is a tool which:

- Allows executing many different tasks after each other without requiring user input.
- Reports the settings used, any errors and GAMS result codes in a HTML page from which they may queried at a later time.
- Ensures that each new run generates its own listing file, which can be opened from the HTML page.
- Allows storing the output of the different runs in a separate directory, while reading input from unchanged result directories.

The purpose of the batch execution facility is therefore at least twofold. On the one hand, it allows setting up test suits for the GAMS code of a project such as checking for compilation without errors for all tasks and different settings such as with and without market parts etc. Secondly, production runs of e.g. different scenarios can be started automatically. Timer facilities allow starting the batch execution at a pre-scheduled time. Along with functionalities to compare in a more or less automated way differences in results between versions, the batch facility is one important step towards quality control.

The important contribution of GGIG is to mechanize to the largest extent the generation, storage and later inspection of meta data underlying a scenario and the related result set, overcoming an often-encountered weakness in (economic) models because it increases the tractability of data sources and model settings used for specific scenarios. All settings for a particular model run are stored in a file generated by the GUI, which consists of a block of text that can be used for the automated execution by the GUI in a batch-file mode. They can be executed from within the GUI or send to the GUI from another programming environment. As such, they are a efficient entry point for the development of wrappers for simulation models making use of the GUI.

4. THE R LANGUAGE FOR THE MIND STEP TOOLBOX

Input data, parameters, and variables appearing in economic simulation models can have complex structures, consisting of different data types. Support for such data structures was mentioned as a shortcoming of the SEAMLESS-IF framework and the underlying OpenMI standard from a modeller's perspective. Even though more recent versions of OpenMI support more complex data structures and possibly simultaneous equation models, a major drawback is that modellers often have no IT training and experience with general purpose programming languages.

The R programming language (https://www.r-project.org/) is a free software environment for statistical computing and graphics which addresses several of the issues mentioned above: R support functional programming and the creation of distributable packages that can be flexibly integrated in a program. It

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

supports complex data structures, for instance in the form of data-frames, which can contain data of different types (numerical, strings). Most of the economic simulation models in the MIND STEP toolbox are implemented in the AML GAMS (General Algebraic Modelling System), which also features a binary format for data exchange (GAMS Data Exchange, GDX) with fast reading and writing capabilities. Model data stored in GDX files (parameters, variables, sets) can be read and written from an R environment by using a freely available library (gdxrrw) and are directly usable as data-frames. Further, several models feature connection to R, for instance for statistical data analysis, data handling, and visualization. For instance, to interface the GLOBIOM database a R routine has been developed, which allows GDX content to be explored, read, and written. On top of a GLOBIOM visualization interface is provided by the globiomvis R package. It supports analysis and generation of a variety of scenario plots. In addition, globiomvis enables creation of maps for the various regionally and spatially explicit representations of the model. The FarmDyn model has a bridge to R generate sample data for complex experiments and sensitivity analyses. In addition to simulation models with links to R, there are also examples for data processing routines within the MIND STEP toolbox implemented in R, like fadnTools and the current prototype for linkages to the Dutch AgroDataCube (see MIND STEP Deliverable 2.2). This underlines that the use of R has become rather common in the modeller community and that most project partners are familiar with it.

R libraries (or packages) can account for different needs of groups involved in development, maintenance, and application of the interfaces. As example the developers of the interface, needs to have a shared distribution system to commonly develop and extent the interface and deploy. The user group will use the API in a more applicable way by loading the API and applying without any need of changing the function of the interface itself. With the package deployment in R this concept is easily applicable. For both clients a good documentation is required. To support the documentation process for the interface, it shall be to a certain extent generated in an automated way and build up on inline documentation. Besides the documentation of the interface also a use case documentation is of importance. The R-Mark down approach allows easily to compile use cases and make them in different formats available, an example is shown in the next chapter.

5. A PROTOTYPE WRAPPER FOR THE FARMDYN MODEL

Based on the experiences and observations described in the previous chapters, choosing R for the development of model wrappers appears to be a sensible choice. It supports complex data structures, most partners in MIND STEP are familiar with it, the package concepts facilitates exchange of data and methods, and model data, namely in GDX format can be readily processed.

While it is possible to execute GAMS programs directly from an R environment, the wide use of GGIG-based GUIs in the MIND STEP toolbox permits to exploit the advanced and automated functionalities for model set-up and execution provided by the GUI. The prototype wrapper proposed here is an example how these functionalities can be used to execute several instances of the FarmDyn model in parallel.

The user has to provide the wrapper function “runFarmDynfromBatch” with the following information:

- Path to the FarmDyn directory
- Name of an xml file that governs the model-specific appearance of the GUI (e.g. GUIsettings.xml), available by default in the /GUI subdirectory of the FarmDyn directory
- Name of an ini-file that contains user-specific information like the GAMS version to be used (e.g. userSettings.ini), by default in the /GUI subdirectory of the FarmDyn directory
- Name and location of the scenario file used for the batch execution (scenarioBatch.txt) in a location chosen by the user

Creating the latter three files can easily be done my launching the GUI, providing the required settings, and executing the FarmDyn model once. Following the instructions in the GGIG user manual, a marked block

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

of the automatically generated file "runInc.gms" in the /incgen subdirectory of the FarmDyn directory can be copied to the "scenarioBatch.txt" file. Once these preparatory steps are taken, the model can be executed from an R environment by calling the function runFarmDynfromBatch. With the paths to the FarmDyn directory, the two files for the GUI setup (GUISettings.xml, userSettings.ini) and the file for the scenario specification (scenBatch.txt), the wrapper functions generates a temporary command-file containing all the relevant information to execute the model within the GUI. A system command to the GUIExecute.bat file is then issued and the model is run. Figure 5 provides a schematic overview on the steps and files involved in the use of runFarmDynfromBatch.

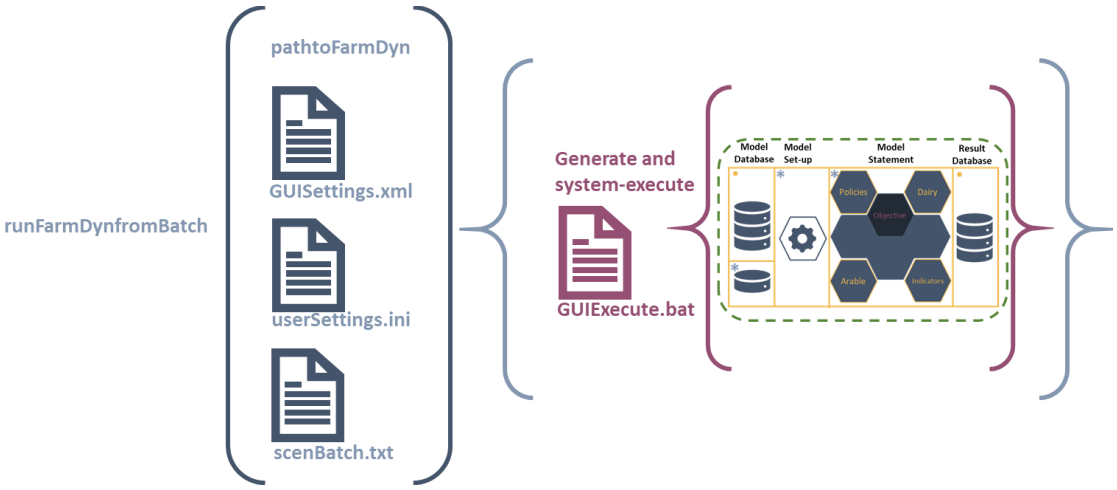

Figure 5 Schematic view of the wrapper "runFarmDynfromBatch"

The R code for the runFarmDynfromBatch wrapper function for the FarmDyn model is shown in Appendix 1. It can also be downloaded from the MIND STEP repository, which requires a password and the additional installation of the FarmDyn model: https://svnesm.iiasa.ac.at/svn/myrepo/wrappers/trunk/FarmDyn

6. CONCLUSIONS

This report develops a concept for a wrapper function that can be used to execute the models in the MIND STEP toolbox from an external environment, such that inputs and outputs can be exchanged between models and results can be combined. For the purposes of this report, a wrapper function was defined as a function in a certain programming environments that provides defaults arguments combined with user inputs to a more complex, possibly external function, and executes it. A review of the SEAMLESS-IF approach (Rizzoli et al., 2009, Knapen et al. 2013) provided important insights in strength and weaknesses of an application of the wrapper concept in agro-economic and environmental modelling:

- Data exchange between models requires handling of complex data structures, like arrays of mixed data types (data-frames, dictionaries)
- Modellers are not automatically IT experts and familiar with general purpose programming languages
- Agro-economic modellers prefer algebraic modelling languages like GAMS
- Modellers are often familiar with statistical or data-handling programming languages

A further observation is that most models used in the MIND STEP toolbox take advantage of the GGIG program (Britz, 2014) for model execution and visualization, which provides advanced model steering functionalities and, most importantly for the purposes discussed here, a batch execution facility.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

MIND Step internal surveys on programming practices and modellers experiences, together with personal communication, showed that most modellers in MIND STEP are at least to some extent familiar with the R programming language. The R environment permits functional programming, efficient handling of complex data structures, and a standardized way to develop program libraries in the form of R-packages. Based on such packages, GAMS-specific data formats can be easily processed.

Based on these observations, a wrapper for the execution of the FarmDyn model exploiting the GGIG facilities was programmed in R. In its current state, it requires the execution of the FarmDyn model (or any other GGIG-driven model) from the GUI to create a set of files that contain a large number of default settings. Once these files are generated, simple changes can be made rather easily and then used to call the proposed wrapper function, which permits the parallel execution of several instances of the FarmDyn model from an R program flow.

The proposed approach is applicable to several models in the MIND STEP toolbox in addition to FarmDyn (IFM-CAP, AGRISPACE, GLOBIOM) as it exploits functionalities provide by R and GGIG. An important subsequent step would be the facilitation to generate the needed files solely from an R environment.

In general, wrapper functions implemented in R can help to integrate the MIND STEP models in a modular and flexible architecture.

7. ACKNOWLEDGEMENTS

This deliverable D7.1: “Prototype of a wrapper to allow a standardized communication channel between the models” is developed as part of the H2020 MIND STEP project which received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

8. REFERENCES

Britz, W., 2014. A new graphical user interface generator for economic models and its comparison to existing approaches. Ger. J. Agric. Econ. 63, 271–285.

Britz, W., Kallrath, J., 2012. Economic simulation models in agricultural economics: the current and possible future role of algebraic modeling languages., in: Kallrath, J. (Ed.), Algebraic Modeling Systems: Modeling and Solving Real World Optimization Problems. Springer, Berlin, Heidelberg, pp. 199–212. https://doi.org/10.1007/978-3-642-23592-4

Britz, W., Lengers, B., Kuhn, T., Schäfer, D., 2016. A highly detailed template model for dynamic optimization of farms-FARMDYN. University of Bonn. Inst. Food Resour. Econ. Version Sept. 147.

Knapen, M.J.R., Janssen, S.J.C., Roosenschoon, O.R., Verweij, P.J.F.M., de Winter, W., Uiterwijk, M., Wien, J.E. 2013. Evaluating OpenMI as a model inte-gration platform across disciplines. Modelling & Software 39: 274-282.

Rizzoli, A.E., Wien, J.J.F., Knapen, R., Ruinelli, L., Athanasiadis, I. Jonsson, B., 2009. Updated version of final design and of the architecture of SEAM-LESS-IF Report No.47, SEAMLESS integrated project, EU 6th Framework Programme, contract no. 010036-2. www.SEAMLESS-IP.org, 31 pp, ISBN no. 978-90-8585-590-3.

van Ittersum, M.K., Ewert, F., Heckelei, T., Wery, J., Olsson, J.A., Andersen, E., Bezlepkina, I., Brouwer, F., Donatelli, M., Flichman, G. and Olsson, L., 2008. Integrated assessment of agricultural systems–A component-based framework for the European Union (SEAMLESS). Agricultural systems, 96(1-3), pp.150-165.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

APPENDIX 1

Define function runFarmDynfromBatch

```
runFarmDynfromBatch <- function(FarmDynDir, IniFile, XMLFile, BATCHDir, BA
TCHFile) {

    # make sub directories
    GUIDir <- paste(FarmDynDir,"GUI",sep="/")
    BATCHFilePath <- paste(BATCHDir, BATCHFile, sep = "\\")

    # General JAVA command
    javacmdstrg <- r"(java -Xmx1G -Xverify:none -XX:+UseParallelGC -XX:Per
mSize=20M -XX:MaxNewSize=32M -XX:NewSize=32M -Djava.library.path=jars -cla
sspath jars\gig.jar de.capri.ggig.BatchExecution)"

    # append specific files to JAVA command
    javacmdparac <- paste(javacmdstrg,IniFile,XMLFile,BATCHFilePath,sep =
" ")

    # create bat file
    runbat = paste0(GUIDir,"/runFarmDyn.bat")
    if (file.exists(runbat)) x=file.remove(runbat)

    b = substr(runbat,1,2)
    b = c(b,paste('cd',gsub("/", "\\\\",GUIDir)))
    b = c(b,c("SET PATH=%PATH%;./jars"))
    b = c(b,javacmdparac)
    writeLines(b,runbat)
    rm(b)

    # execute FarmDyn in batch mode
    system(runbat)

}
```

Execute function runFarmDynfromBatch

```
runFarmDynfromBatch("C:/FARMDYNMANURE","wecr_FarmDyn.ini","wecr_FarmDyn_de
fault.xml","batch","batch16028.txt")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.