D3.3 System Dynamics Tool Technical Specifications

Project Acronym:	PoliRural	
Project title:	Future Oriented Collaborative Policy Development for Rural Areas and People	
Grant Agreement No.	818496	
Website:	www.polirural.eu	
Contact:	info@polirural.eu	
Version:	1.0	
Date:	1 October 2020	
Responsible Partner:	22SISTEMA	
Contributing	Avinet	
Reviewers:	Denis Kolokol (KAJO) Patrick Crehan (CKA)	
Dissemination Level:	Public	X
	Confidential - only consortium members and European Commission	
Keywords	System dynamics modelling, software, Stella, open source, system requirements, solution architecture.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 818496

Revision History

Revision	Date	Author	Organization	Description
0.1	30/09/2020	Patrick Crehan	CKA	Several modifications were introduced following CKA comments.
0.2	01/10/2020	Stein Runar Bergheim	Avinet	Several modifications were introduced following Avinet comments.
0.9	01/10/2020	Antoni Oliva	22SISTEMA	Final revisions & formatting
1.0	01/10/2020	Miloš Ulman	CULS	Formatting corrections

Table of Contents

List of Tables

List of Figures

Glossary

Term	Definition
API	Application Programing Interface
BOT	Behaviour Over Time
BQM	Base qualitative model
C	A programming language
CLD	Causal Loop Diagram
CPU	Central processing unit
Docker	A platform for containerized applications easing installation and deployment of applications in cloud environments
DSS	Decision Support System
Flask	A web service framework for Python
Git	Source code management system
GUI	Graphic User Interface
Http(s)	(Secure) Hyper Text Transfer protocol
KPI	Key Performance Indicator
LCM	Local customized models
OS	1 Operating System, i.e. Windows, Linux etc. 2 Open Source, i.e. a permissive license
Python	A programming language
Repository	A shared, managed storage and change management system used for sharing application code and XMILE models
SD - SDM	System Dynamics - System Dynamics Modelling
SSL	Secure socket layer
TRL	Technology Readiness Levels
UX	Interaction design (end-user centric application design)
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines
WP	Work package
XMILE	OASIS XML Interchange Language (XMILE) for System Dynamics

Executive Summary

System dynamics modelling applied to rural spatial and societal development and planning is one of the focus of the PoliRural project. Work Package 3 focuses on the Innovation Hub, in which the system dynamics solution is a central piece. Task 3.3 tackles the introductory works to design a final solution within the framework of the Innovation Hub, considering the whole process, from the more general qualitative models to the more accurate local dynamics.

The document begins with a brief introduction of the system dynamics concepts and technique, as well as the limits and bottlenecks of modelling. Then an overview of the software packages available for modelling is covered, including open source and proprietary software. The section finishes with a comparison matrix and a first valuation of the most feasible choice at this stage.

Chapters 4 defines the structure of the model and its authoring, starting from the PoliRural Base Qualitative Model, and then the Local Customized model. A special focus has been put on the feedback process of enriching the models and the community generated around it.

To complete the explanation of the process, chapter 5 deals with the model execution, both of the Base Qualitative Model and the Local Customized ones; to end up with the possibility of embedding the models within externals applications.

Once the modelling process has been described, chapter 6 specifies the recommended solution, including the system requirements, architecture and components.

The deliverable ends up with a conclusion containing the recommended components and next steps to complete the first version of the SD solution.

1 Introduction

The concept of system dynamics modeling and simulation is not presently widely adopted within rural spatial and societal development and planning. Indeed, the same could be said for about many other sectors, system dynamics remains an expert discipline and the number of practical implementations fall far short of the potential as seen by system dynamics experts.

It is a fundamental assumption of the PoliRural project that the ability to simulate societal impact of potential policies at the development stage will be valuable decision support for policy developers and decision makers alike.

Where applied, system dynamics has had considerable impact on practical issues in the areas of management, urban planning and environmental change, but its practical impact has been less than might be expected, particularly in addressing social concerns.

System dynamics modelling is at present an expert discipline that is the domain of people with subject matter expertise. This is partly due to the skills and knowledge required to use system dynamics tools, partly due to significant licensing costs of software and, finally, partly due to lack of awareness.

Taking into account the antecedents exposed, PoliRural faces the challenge to test and demonstrate the potential usefulness of SD and facilitate the access to SD techniques and tools, especially around rural communities and policy makers, so that they better appreciate the complex dynamics of natural and human systems, and then, hopefully, use those insights in a constructive way to create better rural living environments.

This deliverable must therefore address the following issues:

- Identify how SD can fit into existing planning workflows
- Identify tools that are suitable to satisfy SD use cases within rural development and planning.
- Identify and propose mitigating measures related to skills and knowledge gaps.
- Identify and propose mitigating measures barriers related to licensing.
- Ensure that the proposed technology platform is affordable, available and accessible.

1.1 Scope as per the Grant Agreement Description of Work

From the grant agreement, we find the following two paragraphs describing respectively work package 3, task 3.3 and deliverable 3.3

“The task will define the rationale for choosing a particular type of system dynamics software. A comparison between Stella and OS competitors will be made (e.g. NetLogo, Simantics, InsightMaker). As Stella is proprietary software, most efforts will focus on the OS solution, and how to build an effective and appealing framework for the post-project exploitation.”

"The deliverable describes the building blocks of the OS solution to be used as a long-term exploitation tool."

This document is therefore structured in a manner so as to address the above and additional context that has evolved since the project was started.

The stated purpose of this document from the Grant Agreement is somewhat at odds with the fact that Stella Architect was selected as the authoring platform for system dynamics models at the contract stage. We have thus reframed the description of tools with the intent of identifying candidate technologies that may be used interchangeably through open, interoperable standard formats like XMILE.

Thus, this specification does not dictate one specific tool to be used for authoring – it merely identifies the capabilities the tool must provide in terms of interoperability. Section four below provides an overview of the candidate technologies in the field and may be used by third parties to make their own implementation of this specification. Particularly at the modelling stage there are viable candidates to be used interchangeably.

1.2 Objectives, purpose

The objective of this deliverable is two-fold, namely to:

- Define the best tools for authoring and execution of system dynamics models in a way that is inclusive with regards to all envisaged stakeholders and that does not exclude the envisaged exploitation scenarios.
- Set out a clear and tangible path towards implementation, integration and deployment of the proposed solutions from the entry TRL-level of 3-4 to the output TRL-level of 6-7 for the project results.

1.3 Stakeholders and their corresponding user groups

Stakeholders and their requirements for SD are identified and described in detail in other PoliRural deliverables. This deliverable is not concerned with the content of each model but with the features that must be present to be able to build and execute all those models. For the purpose of developing solution specification, we therefore group these stakeholders into four groups:

- Model authors: These are the users who are creating and customizing SD models. This would typically be done in cooperation between subject matter experts in the domains of rural planning and system dynamics
- Model executors: These are the users who are specifying input parameters for and running prepared models in order to test different scenarios in order to test the potential impact of policies
- Model viewers: These are the users who will benefit from seeing visualizations of model executions. I.e. visualizations of data series showing development of multiple variables over time given input parameters A, B, C and so on.

- System integrators: These are the users who will integrate the SD API into end-user applications that will allow SD functionality to sit alongside other decision support tools such as web map portals and similar

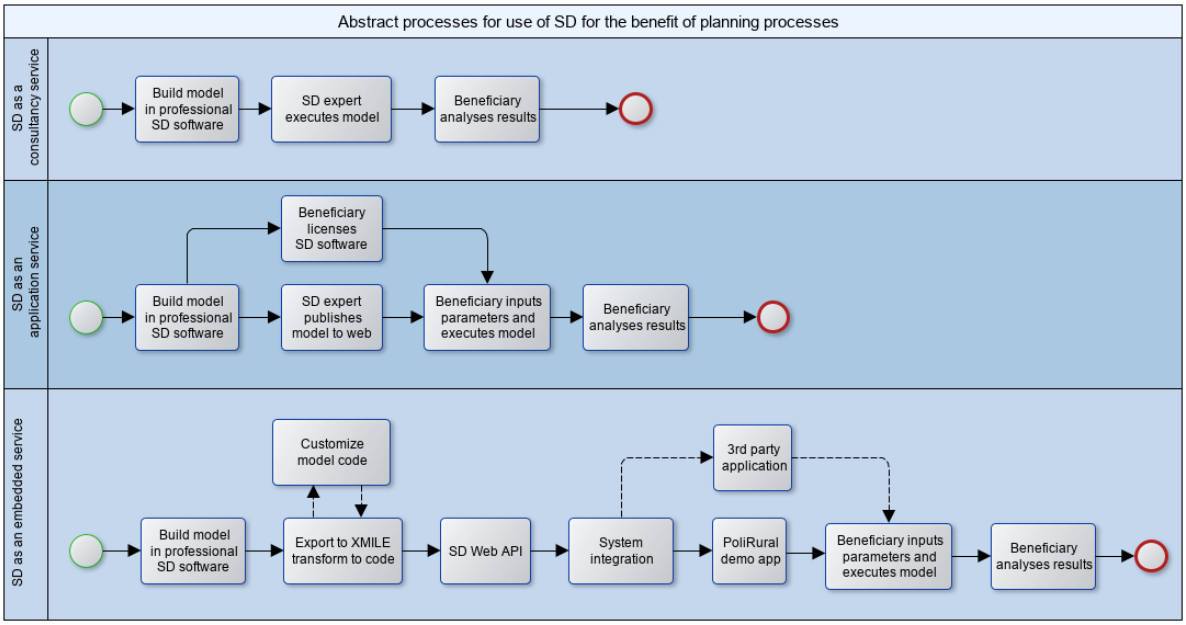
1.4 Exploitation scenarios

PoliRural assumes that system dynamics is useful both as a supporting tool for foresight processes as well as for policy formulation.

The detailed business case for the use of SD will however vary greatly from case to case and the nature of the qualitative and quantitative factors that form the basis of each case. The basic design goal of the solution specified in this deliverable, is that the subject matter expert in the foresight, planning or policy development discipline (referred to below as the "beneficiary") can execute SD models unaided and can tweak input parameters to test conditional development scenarios without incurring cost or time delays due to third parties.

The diagram below shows three alternative exploitation scenarios for deploying SD as part of decision support processes in planning. It does not absorb the specifics of each process planning process, something that will be dealt with through interaction with stakeholders, including PoliRural pilots, as part of the implementation process of this solution.

The scenarios below are described *exclusively* from the solution provider perspective – rather than portraying the benefits of SD within planning as this is dealt with in other deliverables.



The topmost process is based on offering SD as a consultancy service where the work is done by the SD expert. This is the task that require the least amount of *technical* effort on the part of the "beneficiary" of the SD service. This saving does however come at the cost of increased need for communication and the chance of misunderstandings due to differences in knowledge background and varying professional vocabularies between the two disciplines. This is the most common exploitation scenario today.

The middle process allows the "beneficiary" to execute models unaided and thus increases the flexibility. This comes at either the cost of reduced ability to modify the model (i.e. exporting it to a web execution environment) or at the cost of licensing professional SD software (and learning how to use it) in order to execute the model without limitations. This is likely to be a common scenario in the long term as the toolbox and methods applied are increasingly more sophisticated in terms of technology.

The last process, is the one that PoliRural wishes to validate, i.e. enable SD to be executed transparently alongside existing decision support such as web map portals, data and statistics portals etc. It will rely in simplified models and user interfaces to introduce non-SD-expert beneficiaries to model simulations without requiring them to leave their familiar tool and without having to adopt and understand the full technical terminology of SD. This may be a bridge between the two processes.

2 Concepts and limits of System Dynamics

To better understand the approach proposed by SD and also its potential, find below a brief resume of the concepts applying to the technique (2.1) as well as the limits and bottlenecks that can be found across the process of modelling and exploitation of results (2.2).

2.1 Concepts applying System Dynamics

System dynamics is a branch of control theory which deals with socio-economic systems, and that branch of management science which deals with problem of controllability[1].

The objective of System Dynamics is not prediction but understanding and forecast. SD is very useful in building scenarios and anticipating the effects of an action or a policy, and also to detect sensitivity of a complex social system to variations of selected variables.

SD is based on feedback loops, and the premise that the system's behaviour can be explained by its own structure (i.e. the relation between its parts).

Modeling is not so much about creating systems but about solving problems. The system is generated to solve the problem, but not on something pre-existing. Thus, the first task is to clearly define the problem. To help us on this task we identify trends that are indicative of the problem to tackle. We call these trends Behaviour Over Time (BOT) graphs.

BOT is a graph showing the trend we want to address. It usually covers a period of minimum 10 years in the past. It doesn't need to be an accurate graph. BOT will allow us to formulate the first dynamic hypothesis. See two examples of BOT in the graphs below.

[1] A. Ford, Modelling the Environment, Island Press, 2009.

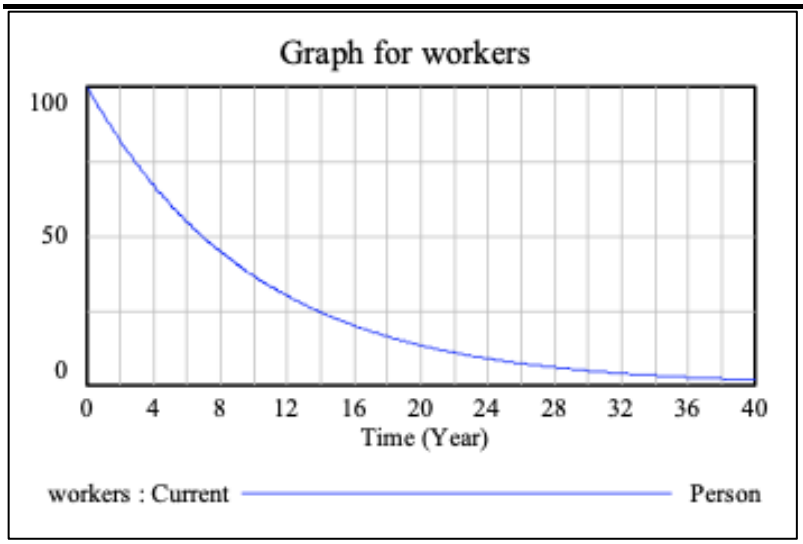


Figure 1 Example 1 of BOT

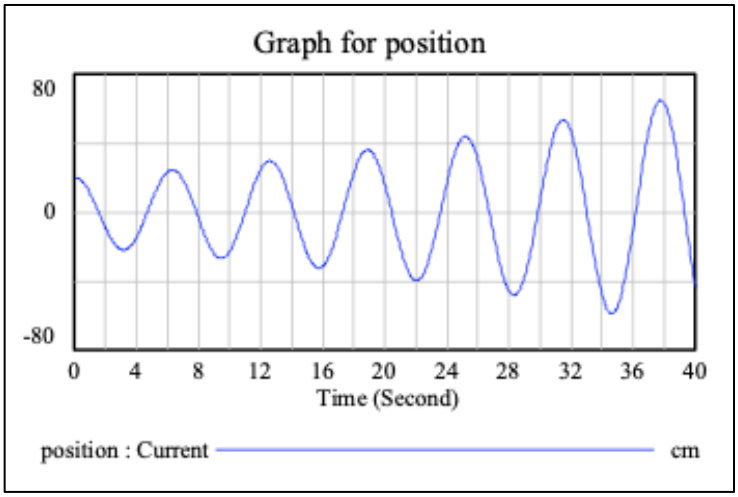


Figure 2 Example 2 of BOT

Causal Loop Diagrams are maps showing the causal links among variables. CLDs are good for capturing dynamic hypotheses; eliciting and capturing the mental models of individuals or teams; and communicating the important feedbacks that are believed to be responsible for a problem.

The links between variables have a polarity defining the relation, the polarity can be positive or negative, like it is shown below.

	**Positive relation**. If A increases (decreases), then B increases (decreases) above (below) what it would have been.
	**Negative relation.** If B increases (decreases), then C decreases (increases) below (above) what it would have been.

Table 1 Polarity in Causal Loop Diagrams

By linking relations between variables loops are defined. A loop can also be positive or negative. Positive loops are called Reinforcing loops and the behaviour they show is exponential growth. Negative loops are called Balancing loops and the behaviour they show is goal-seeking.

Below an image of a positive loop running as far as the limiting condition allows, then the negative loop starts to run.

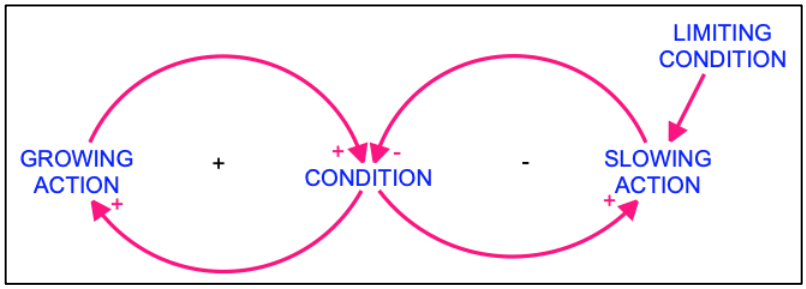


Figure 3 Positive and Negative feedback loops combining

Stocks and flows is the material translation of CLD diagrams. Models are built with stocks and flows. These are the main elements of S&F diagrams:

- **Stocks** are represented by rectangles. Stocks are accumulations, and they give the systems inertia and provide them with memory.
- **Flows** (inflows and outflows) are represented by a pipe pointing into (inflows) or out of (outflows) the stock.
- **Valves** control the flows. They can be modified by variables or stocks themselves.

Clouds represent the sources and sinks for the flows. A source represents the stock from which a flow originating outside the boundary of the model arises; and a sink represents the stock into which a flow leaves the model. Clouds are then defining the boundaries of the model considered.

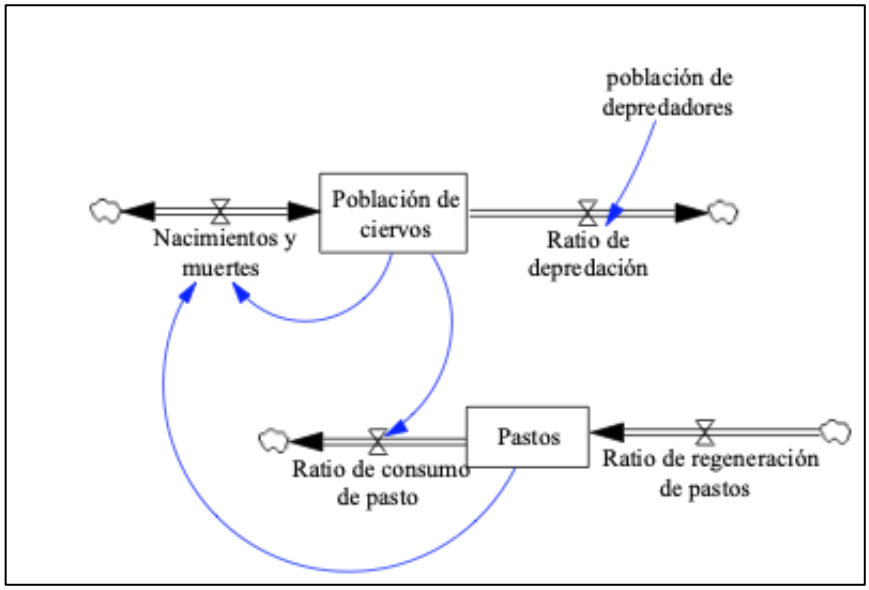


Figure 4 Example of a stocks and flows diagram

2.2 Limits and Bottlenecks in developing models

As Norbert Wiener put it, “the best material model of a cat is another, or preferably the same, cat". A model, like a map, is a representation, and therefore a simplification, of reality. Also like in maps, what you have to ask for a model to be useful is an equilibrium between simplification and usefulness (both in technical and content terms), and this is to be applied in a case by case base.

Main factors affecting the above mentioned equilibrium are:

- Efficiency, to solve the problem the model tackles.
- Simplicity of tools, to allow a larger number of people to interact and extract knowledge from the model.
- Ease of migration, to allow different levels of intervention, from specialist modellers to local technicians or policy makers.
- Capacity for processing and execution, in order for the models to be run in standard computers or web based applications.

2.2 Lessons learned during the project

More than ten models have been elaborated so far in the PoliRural project, with different objectives and scope. This gives the team an experience beyond the more technical aspects. Thus, considering the points highlighted in the previous section, and also considering the experience gained so far, some conclusions can be drawn:

- The inherent complexity of a model to be useful and relevant. It is not easy to reflect local dynamics and keep it simple and understandable, without giving the impression that everything is decided in a black box.

- The difficulty of combining the complexity of the model and a user-friendly interface in the Stella software environment (including the model sharing platform ‘Exchange’).

These points add to the overall challenge as expressed in the introduction, and the lessons learned will be of great help designing the SD final solution.

3 Overview of available software implementations

There are multiple software solutions to build maps and models using System Dynamics software and principles. In this point we are going to describe and classify the most used, considering the following variables:

- Processing capacity

- Graphic User Interface
- Storytelling
- XMILE Support
- Environment
- Licensing

While the stated purpose of this deliverable is to identify the “right tools”, a significant premise was established in the contract by stating that Stella would be used for model authoring in PoliRural.

This does however not mean that the solution described here requires Stella to be the starting point for any implementation. The solution described could be realized using alternative components at different steps along the way from model authoring, execution and visualization/analysis.

3.1 Interoperability in system dynamics

A fundamental assumption of the SD components in PoliRural is that it must be possible to execute models independently of authoring environments — and that it must be possible to exchange models between professionals working within different software platforms.
For this reason, we must look to what standardization efforts and widely used formats could enable effective interchange. There is only one prominent open interchange format, XMILE.

3.1.1 OASIS XML Interchange Language (XMILE) for System Dynamics

https://docs.oasis-open.org/xmile/ns/XMILE/v1.0

XMILE is an open XML protocol for sharing interoperable SD models and simulations. The work builds upon SMILE, a descriptive, DYNAMO-based language; the XMILE specification itself encodes SMILE in XML.

XMILE enables System Dynamics models to be reused with Big Data sets to show how different policies produce different outcomes in complex environments. Models may be stored in cloud-based libraries, shared within and between organizations, and used to communicate different outcomes with common vocabulary. XMILE will also support the integration of system dynamics models and simulations into mainstream analytics software.
Using XMILE, online repositories may be created to model common business decisions. XMILE also enables the development of standards-based applications for mobile technologies and social media.

3.1.2 Proprietary formats

While XMILE is the acknowledged interchange format, the internal format(s) used by Vensim are also supported by some applications including Pysd and Stella that both are capable of interpreting Vensim models.

3.2 Description of individual tools

Having established the importance of interoperability, it is time to take a look at the individual software packages that exist in order to evaluate which are most suitable for the use cases of PoliRural.

3.2.1 Commercial system dynamics and simulation software

First, we take a look at the contenders within the commercial/proprietary domain where we will find the majority of SD subject matter experts. Much of the description is quoted here is based on the providers own online marketing material with links to each product provided for in-depth reference.

3.2.1.1 Vensim

http://vensim.com/

Ventana Systems is one of the first software companies for modelling System Dynamics, starting operations in 1985. The brand of the company is Vensim, a modelling software now in its version 8.1.

Vensim is completed with Ventity, which uses a friendly interface, and a platform for models at Forio.

Vensim allows models to be exported to a web interface via Python, and then improve the design with HTML 5 programming.

There is also a programming language developed by Ventana Systems UK, Sable, that allows developers to create user interfaces for the models.

Vensim and all the family products are proprietary software. However, there is a basic version of the software free for students and inexpensive for commercial use. There is also a model's reader version for free.

Vensim is considered to be the better and faster software for professional complex models. The price of a single license of Vensim DSS (the top software version) is 1,995$.

3.2.1.2 Stella

https://www.iseesystems.com/

ISEE Systems was also born in 1985. STELLA is the main brand of the company for modelling software, and it was the first software to introduce an icon-based model building and simulation tool.

Stella has always been more user friendly, and in the latest versions it has deepened this capacity. Storytelling is a strength in STELLA, and this gives an opportunity for a better understanding and sharing ideas and points of view.

Like in the case of Vensim, STELLA allows models to be exported and build a web interface to make them interactive. The platform ISEE Exchange is the Forio equivalent but with more designing capacities.

ISEE Systems does also have a free reader edition. The price of a STELLA Architect Software is 2.999$.

3.2.1.3 Anylogic

https://www.anylogic.com/

Anylogic is a modelling software focused on industrial operations. It combines Discrete Event, Agent Based and System Dynamics simulation methods, and it is indeed the only software introducing multimethod simulation modelling.

Its niche is limited to industrial operations and logistics, and it has libraries focused in supply chain, transportation, rail logistics, etc.

The first product of the company came in 2000, AnyLogic 4.0. It is also a proprietary software but in comparison with the previous products, Anylogic is more closed software, with a clear focus on industrial operations and logistics.

3.2.1.4 PowerSim

https://powersim.com/

PowerSim is an integrated environment for building and running business simulation models in the. The application features stock-and-flow modeling including multiple diagrams for organizing models, drag-and-drop variables into input/output objects to create user interfaces etc.

Less a standards-compliant authoring environment and more of a targeted application for simulation driven decision support in government there are few ways into or out of PowerSim that can be exploited for the purposes of PoliRural. However, concepts used in the modelling and visualization interfaces may serve as inspiration for features in the demonstrator client that will be made part of the PoliRural innovation hub.

3.2.2 Free and/or open source system dynamics and simulation software

3.2.2.1 Insight maker

https://insightmaker.com/

Insight Maker is a free web based modelling and simulation environment. It has been developed in the last years by Gene Bellinger. It runs on Javascript and it can support big models. Insight Maker combines agent based and system dynamic modelling, and it is ideal for educational purposes.

The interface is easy to use but it lacks the professional outlook of the previous softwares. Besides, the modeller needs some coding skills to define equations and macros.

There is an extensive library of built-in functions and a large API to script and take control of a model.

3.2.2.2 Simantics System Dynamics

http://sysdyn.simantics.org/

Free and Open Source software suite for system dynamics model editing and execution. Simantics System Dynamics is by the provider's own description a "ready-to-use system dynamics modelling and simulation software application for understanding different organizations, markets and other complex systems and their dynamic behavior".

Simantics System Dynamics is used for modeling and simulating large hierarchical models with multidimensional variables. The models are created in a traditional way with stock & flow diagrams and causal loop diagrams. Simulation results and the model structure can be analyzed with different visual tools

The core of the application is a model browser that includes the the model configuration itself as well as a number of supporting components including:

- Spreadsheets allow storing and maintaining values in a familiar format
- Experiments are the way to simulate the model
- Modules enable structural modeling
- Functions contains built-in and user-defined functions
- Modelica provides a convenient way to use functions in your models. You can create your own functions, export and import complete function libraries and share function libraries to be used in all of your models.
- Charts are user-defined displays of simulation result data

At first glance, this would seem a good candidate for use in PoliRural - but, though complete and operational, it is not presently actively developed.

3.2.2.3 Pysd

https://github.com/JamesPHoughton/pysd/wiki

By the provider's description, Pysd is "a simple library for running System Dynamics models in python, with the purpose of improving integration of Big Data and Machine Learning into the SD workflow". PySD translates Vensim or XMILE model files into python modules, and provides methods to modify, simulate, and observe those translated models.

Compared to the other tools listed here it is a small and obscure piece of software, yet it provides a key enabling feature that PoliRural requires and has been included due to its unique value proposition. It is not a stand-alone editing or authoring environment, instead it takes pre-authored models and translates them into executable Python code.

While the developer of Pysd built the software with the intention of enabling integration of data analytics and big data streams into system dynamics execution processes. The capability to execute models in Python opens other doors too, in this context namely that it is possible to place the execution behind a Web API.

3.2.2.4 SDEVERYWHERE

http://sdeverywhere.org/

SDEverywhere is a Vensim to C transpiler that handles a broad range of System Dynamics models. It supports some advanced features of Vensim Modeling Language, including subscripts, subranges, and subscript mapping. The Vensim model is converted to human-readable C code.

Using SDEverywhere, you can deploy interactive System Dynamics models in mobile, desktop, and web apps for policymakers and the general public. Or you could perform model analysis using general-purpose languages, running the model as high-performance C code.
It is in some ways comparable to Pysd - albeit using only Vensim files as input instead of XMILE files. While both formats are widely used, only XMILE is an 'open standard'. Furthermore, SDEverywhere uses C as a target programming language that has a smaller number of developers and applications.

3.2.2.5 ISEE Exchange

https://exchange.iseesystems.com/

Exchange is the ISEE platform for sharing and running models built with Stella software. The platform allows the user to change values of some of the variables, run the model and compare results. The usability is defined by the modeller, with the limitations of the editor included in Stella Architect software. There are also storytelling tools to make the model more understandable for no SD specialists.

3.2.2.6 NetLogo

https://ccl.northwestern.edu/netlogo/

NetLogo is a programmable modelling environment for simulating natural and social phenomena. It focuses on agent based modelling, but it has a System Dynamics module.

NetLogo is a free, open source software specially thought for students, to understand the behaviour of complex systems over time.

3.3 Comparison and classification matrix

The matrix below provides a side-by-side comparison of how the various software packages that have been evaluated implements key features (i.e. features that are key to realize PoliRural objectives).

Tool name	Processing capacity	Graphic User Interface	Storytelling	XMILE support	Environment	Licensing	Limited free version
Tool	*Low - Intermediate - High*	*Entry level - Intermediate - Expert*	*Yes / No*	*Yes / No*	*Desktop - Web - Library - Service*	*Commercial / Free*	*Yes - no - n/a*
Vensim	High	Expert	Yes	Yes	Desktop	Commercial	Yes
Stella	High	Expert	Yes	Yes	Desktop	Commercial	Yes
Anylogic	High	Expert	No	No	Desktop	Commercial	Yes
PowerSim	High	Expert	Yes	No	Desktop	Commercial	Yes
Insight maker	Intermediate	Entry-level	No	Yes	Web	Free	n/a
Simantics	Intermediate	Expert	No	Yes / read only	Desktop	Free	n/a
PySD	High	N/A	No	Yes	Library	Free	n/a
SDEVERYWHERE	Intermediate	N/A	No	No	Service	Free	n/a
NetLogo	High	Expert	No	No	Desktop	Free	n/a
ISEE Exchange	Intermediate	Entry-level	Yes	Yes	Web	Free	n/a

Table 2 Comparison of SD software packages

Having conducted the comparison above, we find that Vensim and Stella offer similar value in context of PoliRural and therefore may be considered as equivalent when it comes to authoring the 'expert layer' of the models. Stella may have a slight edge in terms of native support for XMILE. Thus, either of these platforms could be used interchangeably for the demonstration purposes in the project - while in an operational context it may be entirely interchangeable with either Vensim or — in the domain of free software — Simantics.

Furthermore, for the purpose of making models executable outside of the expert systems, Pysd offers a unique value proposition in terms of translating XMILE models to Python executable code that can easily be integrated with Web Services and thus HTML5 and JavaScript applications. The uniqueness is regrettably not only on account of its relative excellence compared to alternative solutions but also because there are few alternatives. This is not a critical issue but nonetheless a matter that it is useful to be aware of going forward.

4 PoliRural system dynamics model authoring

In this chapter the structure of the models and their authoring is to be defined. The structure is based on D5.2 PoliRural Model (ed. 2), and it contains differentiates between two different types of models, a high-level abstract model, called the 'Polirural Base Qualitative Model', and the 'Local Customized Model'.

In the following points these two models are explained from the point of view of authoring, together with a brief explanation of the feedback process to enrich the Base Model and develop a community with contributions from both local rural areas and experts in any subject affecting rural development.

4.1 PoliRural Base Qualitative Model

This is the model designed by PoliRural, and so authored by the PoliRural team, which will be delivered at the end of the project.

At this stage the model contains four modules, and it combines quantitative data and qualitative values. The feeding of the model will come from the KPI - Drivers matrix, as explained in D5.2.

The final version will be based on a Stella model translated into an open source language and it will be embedded in the Innovation Hub. Once in the hub, and with an appropriate GUI, it will allow any rural area to introduce local quantitative and qualitative data and be run, without the need of Stella or any other modelling software.

The results will give a first approximate foresight for the rural area and will also allow testing policy actions and draw different scenarios. In this sense the predicted project product *Policy Options Inventory* will help to define template scenarios in the model.

4.2 Local customized model

The aim of the Base Qualitative Model is to give support to the foresight analysis. In this sense the model is embedded in the foresight exercise and can not be detached.

Nevertheless, the results provided by the model do not cover the local context with a high level of detail. This is why local authorities may want to develop the model further on and get more accurate results and interpretations of local reality and perspectives.

In this case a revision of the model takes place, beginning with the experts' layer and then bringing back the results to a public layer, as explained in the following points.

4.2.1 Experts Layer

Taking the Base Model as the reference, rural areas have the possibility to modify, refine or enlarge the model. These are the main tasks that can be afforded from the experts layer:

- Refine calibration, in order to have more accurate forecasts and scenarios.
- Modify variables or variable relations, to better fit local reality and challenges.
- Introduce new modules, to change exogenous variables by endogenous ones, reflecting local dynamics.

In this case the model will be manipulated by any modelling software, via a translation from XMILE language. To get the most of the model, a proprietary software is recommended. The authoring of the new model will be in the first instance the expert undertaking the work, and then, whenever the contract will allow it, the local authority. Normally local authority is the owner of the final work, but it is highly advised to specify the public ownership of the model in the contracts regulating the consultancy. This would permit subsequent exploitation and evolution of the models by any party, ultimately contributing to improvements that may benefit all.

4.2.2 Public Layer

The expert layer may create a deep understanding of the local dynamics and the impact of policies and actions. Proprietary software packages have a number of tools allowing a very sophisticated analysis of the models, including sensitivity analysis, variable optimization, Montecarlo testing, etcetera.

But there is a growing need to share decision making with broader communities. To allow a wise decision making, the model can be shared via a public platform (eg ISEE Exchange), or been translated again into XMILE language and been uploaded to the GIT repository with a connection to the PoliRural Innovation Hub.

The more accurate analysis is normally part of a consultancy work, and the knowledge extracted is very specific and locally rooted. But some parts of the model can be generalized as 'local rural dynamics' expressed in SD feedback loops or even in CLD (causal loop diagrams). PoliRural should provide a database gathering local rural dynamics, including description, application and also the SD expression.

4.3 Feedback Process and Workflow

There are two ways to refine the model in a feedback process: the first one is done by the community of local experts; the second one is done by specialists in the different subjects that affect rural development. Both ways are explained in sections 4.3.1 and 4.3.2 below.

The workflow scheme is draw in 4.3.3, including authoring and PoliRural scope.

4.3.1 Expert's Community

When working in the Local Customized Model, those dynamics found to be useful for other contexts, can be introduced in a file into the Git repository with tags specifying field of expertise (eg. tourism, agriculture, local food chain etc.) in which it is working. The file must include a CLD or a scheme of the main feedback loops intervening.

The repository can be consulted by the community of local experts, and then adapted to other rural areas changing variables, parameters or relations to better reflect local realities.

Ultimately, this feeding process may finish with the improvement of the 'PoliRural Base Qualitative Model', by adding new modules or changing the existing ones.

4.3.2 General Trends

General trends affecting rural development can also modify existing models. Reports or prospective analysis dealing with subjects such as the effect of CAP in agriculture employment, connectivity in rural areas, climate change local effects, etc, may be producing specific data segregated by area. The results of these reports (authored and published by public bodies or even the EU as such) can introduce changes in the parameters, relations or even in the structure of the model.

A general procedure should be put in place to identify relevant data for the models to be modified. The procedure must embrace modifications of both local model and Base Qualitative Model.

As an exemple, if the EU produced a general report of the effect of CAP on rural employment, and the data is segregated by NUTS 2 areas, the results can be introduced in the local models.

4.3.3 Workflow

The image below represents the workflow the system is intended to support.

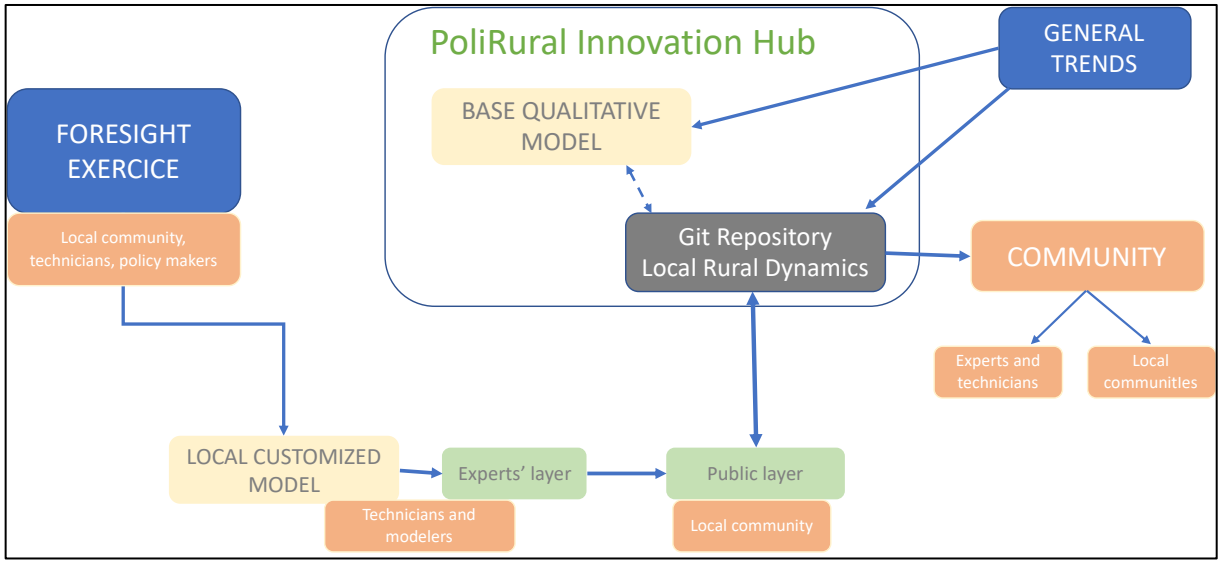


Figure 5 Expected Workflows

5 PoliRural System Dynamics Model Execution

In coherence with the model structure explained in the previous point, the execution will follow the paths defined below. The first consideration refers to the models to be executed, Base Qualitative model (5.1.1) and Local Customized models (5.1.2). Then the extended models of execution and the possibility of embedding them within external applications are explained (5.2).

5.1 Models to be executed

In this section, the two types of models that exist within the PoliRural system dynamics application ecosystem are described in terms of how they will be executed and by whom.

5.1.1 PoliRural Base Qualitative Model

The Base Qualitative Model will be designed and built in Stella environment, as defined in the project agreement. Nevertheless this model is not limited to execution within the Stella suite of software.

The model will be translated into XMILE and executed as part of an embedded application within the Innovation Hub, with an appropriate end-user interface for non-expert in system dynamics (i.e. policy maker, consultants, developers, etcetera). As stated above, the end user will allow some basic changes in the variables and relations of the model ('public layer'), but not in its structure ('expert layer').

Some of the ratios used in the model will be obtained from the data entered by the pilot areas, while some others will be defined graphically or numerically by the user.

The structure and modules of the model can evolve to more sophisticated versions, either of the model itself or the graphic user interface. These changes have to be made by PoliRural members during the project and whoever is determined once the project is finished.

5.1.2 Local Customized Models

Local models will be executed mostly by local consultants and experts. They will work with proprietary software, building on the Base Model in XMILE format as provided by the PoliRural project.

When a local model is finished, a version will be introduced in the Git Repository, including tags and a description of the main dynamics intervening (in the form of CLD or literal description), apart from an identification of the area, the circumstances involving the model, and the authoring (local authority and technical team).

5.2 Extended modes of model execution

In this section the two extended modes of model execution offered by PoliRural are described. Needless to elaborate, it suffices to mention that execution within expert modelling software is available, and that this mode of execution likely will be by system dynamics subject matter experts who are familiar with these environments.
However, it is a fundamental assumption of the PoliRural project that execution and manipulation of model parameters, without modifying the model itself, will improve the process of policy and decision making. For this to be possible, we are trying to tear down a number of obstacles to successful exploitation of system dynamics by non-experts, namely

- Learning threshold: by offering simple user interfaces that make the “expert concepts” of the model opaque to end-users
- Cost of use: by making models executable outside proprietary and costly subject matter expert software suites
- Change resistance: by allowing system dynamics to be embedded within third party applications - enabling system dynamics to become a module in pre-existing decision support systems rather than a new, stand-alone workflow that puts additional demands and labor on existing, over-stretched policy and decision makers.

To break down these three barriers is a matter of two technical steps as described in the sub-sections below.

5.2.1 Execution via Web API

First, it is necessary to enable the execution of the models outside the execution engines in i.e. Stella Architect. This will deal with the element of cost.

The work here involves identifying a suitable execution environment that is developed and released under a permissive Open Source license.

Second, it is necessary to build web-services on top of the execution environment and integrate these into a web API complete with access control (API-keys) and documentation (e.g. Swagger).

This API must then be hosted on the public Internet and be made accessible from any HTML5/Javascript application — or for that matter any third party application capable of issuing requests and interpreting responses over Http.

For more information on how PoliRural will realize this Web API, please refer to the recommended solution chapter below.

5.2.2 Embedding into third party end-user applications

Having enabled execution of system dynamics models outside of e.g. Stella, it is now possible to call the web services from an application.

PoliRural will provide its own demonstration application that can be used as a 'template' for building additional third-party applications — or completely independent third-party integrations can be made directly against the Web API documentation.

The purpose of these end-user applications are to expose the user-configurable input parameters of the models to end-users and allow them easy means to provide input data, then to execute the models and finally to visualize the results — or time series thereof by means of (geo)statistical visualization.

6 Recommended solution

This chapter provides the specification for the recommended solution resulting from the analysis conducted in chapters 2-5 above.

The aim of this section is to describe a generic solution that includes the minimum requirements for operation and that can be replicated within other professional domains. This is possible because the technical steps involved in translating an SD model from a professional platform into a Web API for integration is technology and content agnostic. It relies exclusively on standardized open interchange formats.

Things become different once we move towards the end-users. The third-party integrations need to include user interfaces that are customized to the needs of their intended "beneficiaries" and end-users, thus requiring further interaction with the stakeholders at that time. For a brief discussion on these concerns, please refer to section 7.3 below.

6.1 System requirements

This section provides an itemized set of system requirements that the system dynamics tool must satisfy with regard to overall, functional, technical and integration considerations.

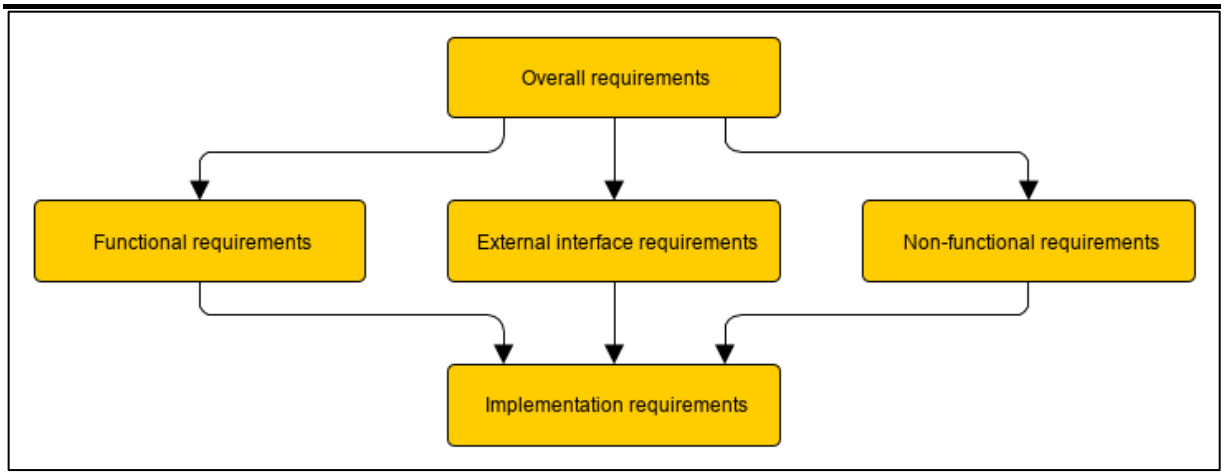


Figure 6 System requirements

6.1.1 Overall requirements

This section describes high-level requirements that precede functional, interface and non-functional requirements.

6.1.1.1 Networking Architecture Components

Authoring can happen flexibly in any networked or non-networked environment. Models are transported from the authoring platform to the system dynamics server as XMILE files over a regular public internet connection negotiated between the individual clients and their ISPs.

The system dynamics API server must be accessible over https/ssl on the public Internet and be configured to operate in a cross-domain environment to facilitate its inclusion into client-server applications relying on HTML5/JavaScript.

6.1.1.2 Users and User Groups

The system dynamics API should be subject to an API-key that can be used to monitor the use and load distribution of the system.

6.1.1.3 Operating Environment

Authoring can be done in any OS agnostic environment as long as models can be saved as or exported to XMILE.

The server will be operated on a high performance yet light-weight Linux and will be deployed containerized as Docker images.

6.1.1.4 Design and Implementation Constraints

The system should be accessible to non-technical end-users, this will be solved by making an API that enables integration with existing applications, familiar to the target end-users.

The system should be affordable and not add excessive licensing constraints to the cost sheet of prospective end-users.

6.1.1.5 User Documentation

The system should have comprehensive documentation in English language.

6.1.1.6 Assumptions and Dependencies

The proposed architecture assumes the continuation of XMILE as an interchange standard initiative within system dynamics modelling.

6.1.2 Functional Requirements

This section describes functional requirements, i.e. business functions that must be exposed by the proposed system in order to support the objectives of the Polirural project.

6.1.2.1 Authoring

Authoring of models ('experts' layer') will take place either (A) in expert system software like Stella Architect or (B) programmatically in interpreter languages like Python etc.

Prior to execution of models, end-users must be able to specify input parameters to the models ('public layer') through a graphical user interface that includes various textual, numerical and graphical input controls.

Authored models are licensed (by default public unless otherwise specified) and published to the PoliRural System Dynamics Git repository.

6.1.2.2 Execution

The system must permit execution of models through a RESTful web service endpoint.

The services must be executable in a Cross-origin resource sharing (CORS) context to facilitate easy integration with HTML5 applications.

6.1.2.3 Third party integration

The web services must be possible to integrate with third party applications and exchange data with these for visualization and further analytics purposes.

6.1.3 External Interface Requirements

6.1.3.1 User Interfaces

In the development of end-user interfaces, the principles of UX design and Web Accessibility shall be observed.

6.1.3.1.1 Configuration of public layer

- Text, number inputs.
- Graphical inputs (histograms, barcharts, line charts with brushes).

6.1.3.1.2 Model execution

- Streaming output of table record data.

6.1.3.1.3 Results visualization

- Charts (pie, line, bar, row etc).
- Maps (choropleth, bubble, line, point, pie).
- Tables (regular tables, pivot tables).
- Dashboards with coordinated views, i.e. when applying a filter to a chart dimension, all other charts will be updated to reflect the filter.

6.1.3.2 Software Interfaces

Data shall be interchangeable through the XMILE format and/or the proprietary yet widely used Vensim model format.

Integration towards other Polirural systems:

- Semantic extraction, analysis
- Geostatistical visualization

6.1.4 Non-functional Requirements

This section describes non-functional requirements for PoliRural system dynamics components.

6.1.4.1 Performance Requirements

All synchronous models must be capable of being executed in <= 5 seconds.

All asynchronous models must start yielding results within 5 seconds. The output time can be longer depending on the purpose or nature of the simulation taking place.

6.1.4.2 Security and Safety Requirements

The web API must be hosted over Https with an SSL certificate.

6.1.4.3 Software Quality Requirements

The web API must accept erroneous input without ‘breaking’.

All public methods in the API must be documented.

6.1.4.4 Internationalization requirements

The API, application and all associated components and documentation will be made available in standard English.

6.1.5 Implementation Recommendations

This section describes observations rather than requirements regarding the implementation of the PoliRural system dynamics components.

6.1.5.1 Observations specific to professional users

PoliRural recognizes that professional system dynamics users are familiar with one or more of the professional software suites. Thus, our architecture assumes that experts will continue to

work in these environments, using open standards as a bridge to execution environments targeting non-professional users.

6.1.5.2 Observations specific to non-professional users

System dynamics is, today, predominantly an expert discipline. Thus, introducing this capability to non-professional users necessitates overcoming some learning barriers. Policy and decision makers work in a very wide field. Much of their work is qualitative, relying on evidence gathered from multiple sources. A proliferation of different decision support tools aims at improving their evidence base - but the number of systems in itself constitute an additional barrier to effective use.

PoliRural therefore observes that it is necessary to be able to integrate system dynamics alongside other modules in existing applications instead of introducing new, end-to-end workflows that must be run parallel to - and in addition to - the existing workload of policy and decision makers.

6.1.5.3 Centralized versus distributed infrastructure

Due to the ongoing evolution of policies surrounding data storage and sensitivity, PoliRural recognizes that the system provided ought to work both as SaaS in a centralized infrastructure (for parties that do not have the technology required to run the system themselves) as well as in a distributed infrastructure where tech-savvy organizations can self-host their own instance of the provided APIs on their own in-house or private cloud infrastructure, thus removing any trust issue that might constitute a barrier to the uptake and adoption of the system.

6.1.5.4 Recommendations on software and hardware

On the server end, the APIs must be capable of running off of standard business level server hardware on Windows or Linux platforms and behind a Http server.

- Minimum system requirements
  - >= 2 core CPU at >= 2.4 GHz
  - >= 8 GB of RAM
  - >= 100 GB of storage
  - Gigabit networking
  - Http/https
  - SSL certificate

6.2 Solution architecture

The below diagram shows the proposed workflow of the systems dynamics tool from a data and actor perspective.

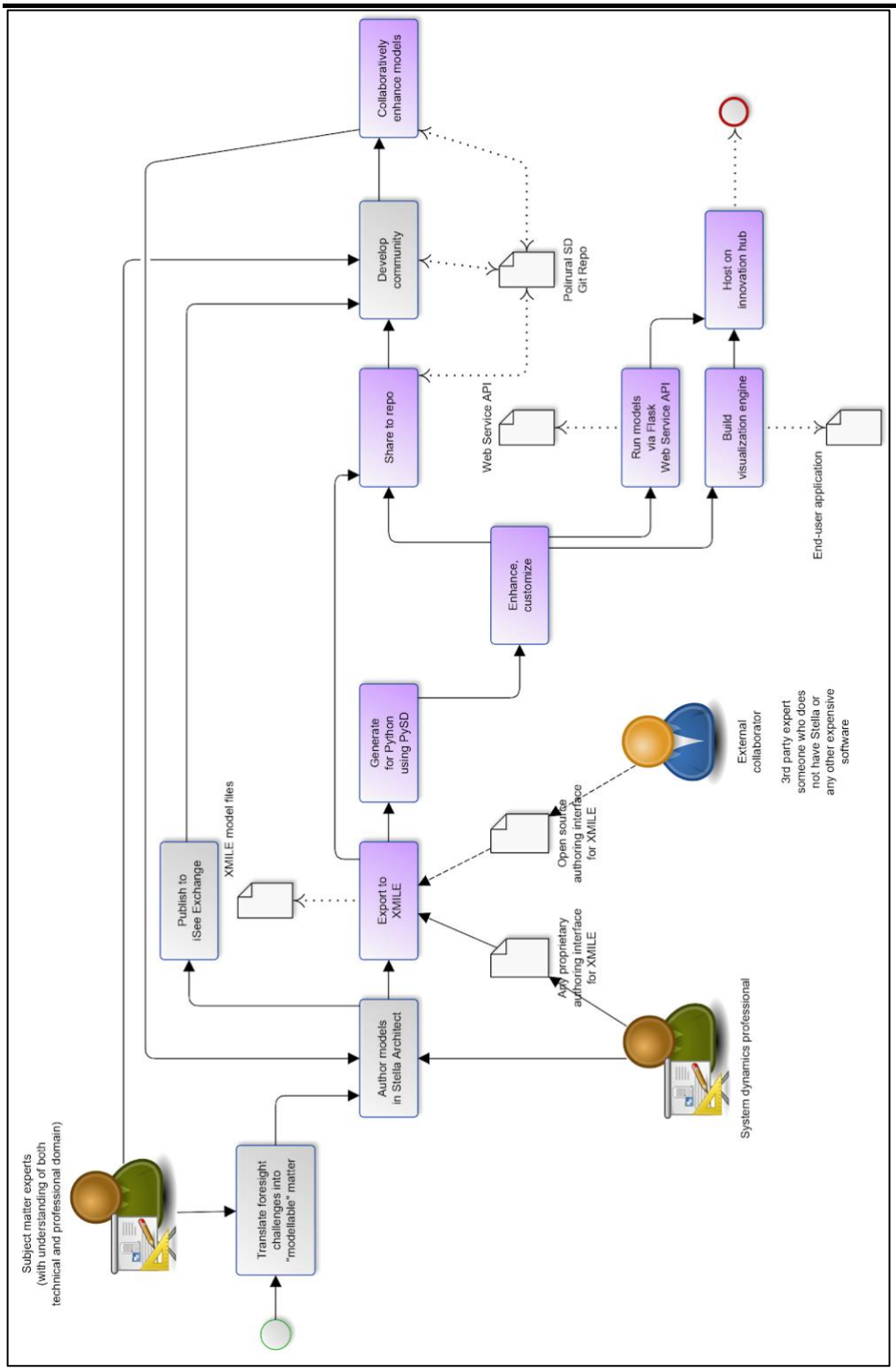


Figure 7 Solution architecture diagram

The diagram below shows a component perspective of the proposed solution architecture distributed on four application layers:

1. A data access layer
2. A business layer
3. An application layer

4. A presentation layer

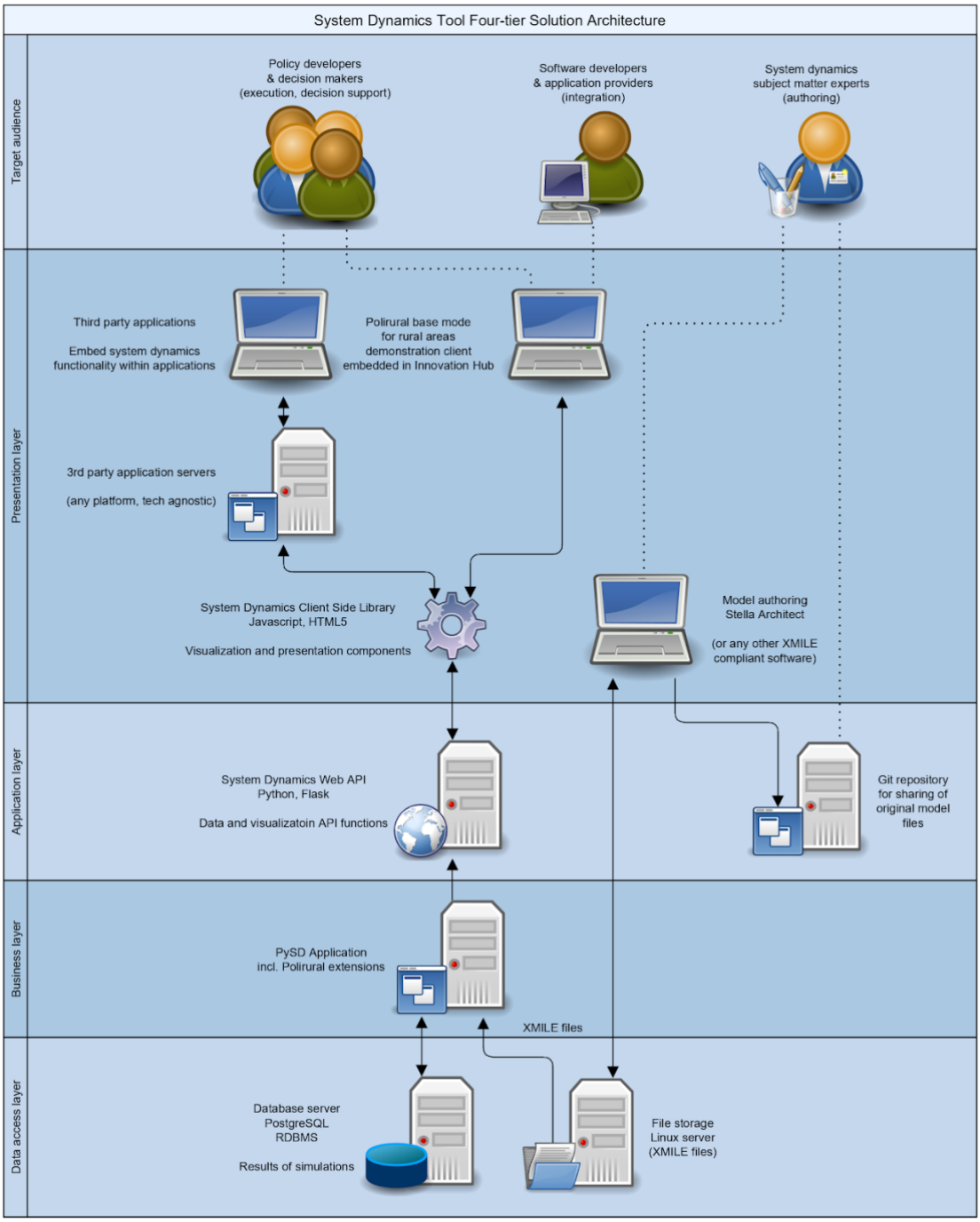


Figure 8 Application layers

6.3 Component description

In this section, each of the boxes shown in the proposed system architecture is described in greater detail.

The first step of the process is to identify the intersection points between system dynamics and foresight analysis as policy development tools. It is necessary to identify the specific policy challenges that can be modelled and that have wide reuse potential across regions - both within the project consortium and beyond.

Having identified this, it is necessary to develop the model. For this purpose, system dynamics models define a schematic notation that can be used to build and expand models visually through a diagramming technique as described in chapter 3 above. This is where the first software component of the software architecture enters the stage; a system dynamics modelling application that supports export to/storage of models in the form of the interchangeable XMILE format - subsidiarily models may be developed in the proprietary Vensim format.

To fulfill the requirements of this component, Polirural has selected **Stella Architect** for reasons outlined in chapter 3 above. This solves the editing environment.

Next, it is necessary to identify how to make the models executable in conjunction with other decision support software, outside the confines of the expert domain software and non permissive licensing policies of proprietary system dynamics software suites.

Instrumental to this end is the existence of the aforementioned **XMILE interchange format** that is described in greater detail in chapter 4 above. The ability to transport models between software packages is an essential step of making models executable through APIs.

Enter **PySD**, a Python based software package that is capable of reading XMILE (or Vensim) model files and translating them into executable Python code that in turn can be invoked programmatically.

By placing these methods behind a synchronous or, in the case of long-running models, asynchronous API built using Pythons web service framework **Flask**, the system dynamics models can be flexibly integrated into third party software packages where it may be combined with **geostatistical visualization tools** to provide policy developers and decision makers with powerful and effective tools in assessing the impact of policies during their development.

7 Conclusion, next steps, summary

We have studied and compared a number of common/popular SD tools and related software modules in order to identify a tool chain that will enable us to effectively...

1. Author
2. Exchange, refine
3. Publish
4. Execute and;
5. Embed into third-party apps

...SD models for use by non-professional, particularly policy and decision makers (and their agents acting on their behalf) in order to improve the evidence base for rural policy development and foresight.

7.1 Recommended components

For each of these steps, we have evaluated candidate technologies in terms of how well they fit into existing processes and workflows in order to choose a tool chain that provides the best return on investments in terms of a low learning threshold, low costs and minimized change management.

Authoring of SD models is still a specialized field that is the domain of subject matter experts, a community of relatively few members compared to other branches of science. These experts predominantly realise their models within one of the mainstream commercial simulation software packages with Stella and Vensim having the widest user communities for our application area.

Interoperability in SD is achieved through the XMILE format that to differing degrees is supported by several SD software packages. Thus, PoliRural will rely on this format both for exchange between subject matter experts in different software silos - as well as between professional authoring environments and embedded Python based execution environments that can power SD functionality in third party apps.

The critical step of translating XMILE SD models to executable code could be realized through either SDEverywhere or Pysd. The choice falling on Pysd is not exclusive and there are scenarios where SDEverywhere might be a suitable candidate, however the target language of Python offers a much wider development community and thus a greater chance of establishing a critical mass of developers to form part of the PoliRural SD interest community.

This deliverable describes the requirements for the PoliRural system dynamics components and architecture.

7.2 Next steps

The next step is to implement the proposed architecture through development of workflows, repositories and software applications capable of fulfilling the requirements specified in this document.

The first version of the system dynamics components is due in M21, i.e. February 2021. From now until then, the WP will mobilize the available development resources and conduct fortnightly development status meetings to plan the work, conduct development and evaluate/test intermediate results along the development period.

The organization of the development effort will be based on the Scrum methodology with 22Sistema acting as the ‘product owner’ and Avinet leading the development team. Based on this deliverable, a backlog of tasks will be extracted and assigned to four major Scrum iterations providing incremental functionality month-on-month from now until the deliverable in February.

7.3 Important considerations going forward

The proposed solution is meant to demonstrate the potential of making SDM available as a web service to be integrated into existing decision support applications used by planners and foresight experts. The process itself is generic but the implementations are specific.

Going forward, WP3 will consult with the pilot stakeholders while developing the demonstrator client to be include in the PoliRural Hub. This client must absorb common use cases that are shared by several pilots and that may be assumed to be representative of wider use cases with a significant number of stakeholders.

The demonstrator application along with models published to the web will become the key show cases for the use of SDM in planning and foresight processes and thus instrumental to gain an audience that is willing to listen and adopt the solutions beyond the project consortium.

The same can be said about the interest community PoliRural seeks to mobilize around the development of SDM models for rural planning and foresight processes. Involvement and information along and following the implementation of this solution will be critical and it is hoped that feedback from this community, positive or negative, may be used to tweak the solution during the lifespan of the project.

Several partners who are currently providing geospatial data visualization platforms widely used by government agencies and other public bodies have expressed interest in integrating SDM functionality within their systems. Thus, the development going forward must not only liaise with end-users but also with system integrators both within and without the project consortium. To this end, the implementation process will include interaction events.

8 References

- PoliRural Grant Agreement 818496 - April 2019.
- Deliverable 5.2 PoliRural Model (ed.2) - May 2020.