D2.6
Design of the PoC platform

Project Title	dealii-X: an Exascale Framework for Digital Twins of the Human Body
Project Number	101172493
Funding Program	European High-Performance Computing Joint Undertaking
Project start date	1 October 2024
Duration	27 months

dealii-X has received funding from the European High-Performance Computing Joint Undertaking Programme under grant agreement Nº 101172493

Deliverable title	D2.6: Design of the PoC platform
Deliverable number	D2.6
Deliverable version	1.0
Date of delivery	31/03/2025
Actual date of delivery	28/03/2025
Nature of deliverable	Report
Dissemination level	Public
Work Package	WP2
Partner responsible	eXact lab

Abstract	dealii-X D2.6 outlines the design of the Proof of Concept (PoC) platform for the dealii-X project. This deliverable focuses on the strategies for creating intuitive interfaces for a broad range of users in the field of personalized medicine. It envisions both a no-code interface for high-level control and a low-code interface for deep customization, allowing biomedical engineers and surgeons with varying technical expertise to interact with the system. The platform aims to enable users to define custom no-code nodes based on low-code logic. This deliverable will detail the node-based low-code interface using deal.II as an example and sketch the no-code approach to simulation management using the defined nodes. Finally, it will outline the development plan and future outlook for the PoC.
Keywords	PoC platform; no-code interface; low-code interface; frontend; backend; metascheduler

Document Control Information

Version	Date	Author	Changes Made
0.1	03/03/2025	G.P. Brandino, M. Franzon, F. De Giorgi	Initial draft
0.2	26/03/2025	L. Heltai	Abstract, keywords, and editing
1.0	27/03/2025	G.P. Brandino	Final version

Approval Details

Approved by: M. Kronbichler
Approval Date: 28/03/2025

Distribution List

- Project Coordinators (PCs)
- Work Package Leaders (WPLs)
- Steering Committee (SC)
- European Commission (EC)

Disclaimer: This project has received funding from the European Union. The views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European High-Performance Computing Joint Undertaking (the ”granting authority”). Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction

One of the long-term goals of the dealii-X project is to develop key components for diagnostic and training tools in the fields of personalized medicine. In order for such tools to be used by a large number of professionals, some intuitive yet complete interfaces should be created. Moreover, different levels of interaction should be envisioned: a biomedical engineer should be able to access the mechanical properties of the organs or tissues, while a surgeon could be interested in intuitive ways to drag an MRI scan into the system and have an interactive simulation without having to deal with the technical details of the simulation.

Currently, the typical workflow for a user of the dealii-X libraries to run a simulation starting from medical data involves the following steps:

- Convert a medical input data (e.g. MRI, X-ray scan, etc.) into a mesh using an appropriate software
- Write a C++ program that models the physical process (the digital twin)
- Run the simulation on an HPC cluster
- Fetch the resulting data to a local machine
- Write/Use post-processing code to prepare the results for visualization
- Use an application to visualize the data

The flow described above is rather complex, and generally not suitable for a non-expert user. The goal of the PoC platform is to allow a user to perform all these steps in a single environment, while retaing the possibility of deep customization.

In view of these diverse use cases, we created a list of requirements that the dealii-X PoC platform should sport.

- A collaborative space, in which professionals with different needs can contribute
- A modular design, to allow components to be easily added
- An interaction that allows deep customization without requiring specific coding skills

- A transparent usage of computational resources, ranging from local machines to HPC clusters

Those requirements naturally bring into play the programming approach that goes under the name of ”Low-code/No-code”.

1.1 No-code/Low-code approach

In modern IT, the demand for rapid application development has driven the rise of the no-code/low-code approaches. These tools empower developers, both experts and domain specialists, to build complex software with minimal manual coding, reducing development time and lowering barriers to entry. Although such platforms are widely used in business and enterprise applications, their impact on scientific software development is particularly significant.

Scientific research often requires custom software solutions for data analysis, simulation, and visualization. Traditionally, developing such tools has been time-consuming, requiring specialized programming expertise. However, low-code and block-based platforms offer a transformative approach, enabling researchers and engineers to build and modify scientific applications more efficiently. These platforms facilitate collaboration between scientists and software developers, enhance reproducibility, and accelerate innovation in fields such as computational physics, bioinformatics, and environmental modeling. At the heart of many no-code/low-code platforms is a graph-based architecture, which is particularly intuitive for scientific applications. In this model, software is constructed as a directed graph where:

- Nodes represent operators or functional units. Each node performs a specific task, such as reading data, applying a mathematical transformation, running a machine learning algorithm, or rendering a visualization. Nodes typically have defined inputs (data they receive) and outputs (results they produce).
- Edges represent the flow of data between nodes, dictating how the output of one operator feeds into the input of another. This creates a pipeline or workflow that mirrors the logical steps of a scientific process.

This graph-based approach is inherently modular and flexible and this enhances reproducibility, as the workflow can be shared, inspected, or modified by others with

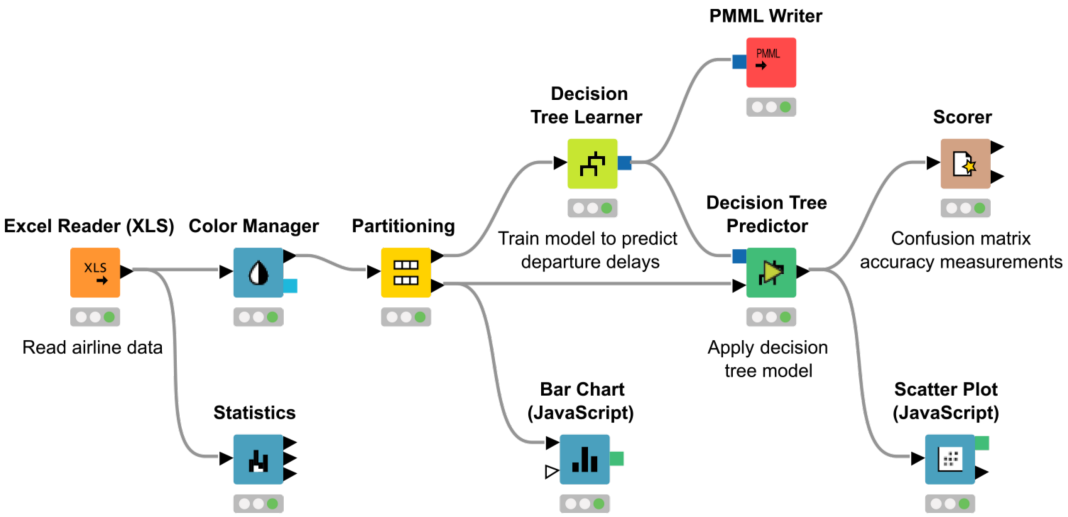

Figure 1: An examples of a data analysis workflow created using the KNIME IDE

ease. In Fig. 1 we report an example of a data analytics workflow created using the proprietary KNIME IDE. The workflow should be read from left to right, with the beginning being the reading of an XLS file. The data are then passed to both a node that does, in this example, color extraction, and to other tasks like partitioning and model training, but there are also branches for statistics and different kinds of plotting. Each of the nodes can perform complex operations, and custom nodes can be added to the system to extend its capabilities.

The approach depicted so far goes under the name of no-code programming since there is no coding logic involved, only relations between entities. Although this approach works for workflow definition, there could be more complex problem settings where some parts need to be manipulated by actually writing code. For these cases, it is still possible to rely on a visual approach that goes under the name of low-code. Fig. 2 shows an example of a piece of code that generates the Fibonacci sequence, written in CodeWire. While retaining the logic and control structure of a programming language, it abstracts the details of the specific programming language.

The approach we want pursue in the PoC platform for dealii-X should sport both strategies, allowing high-level control by means of the no-code interface, while retaining the possibility of deep customization through the low-code interface.

This will be achieved by allowing users to define their custom no-code nodes in

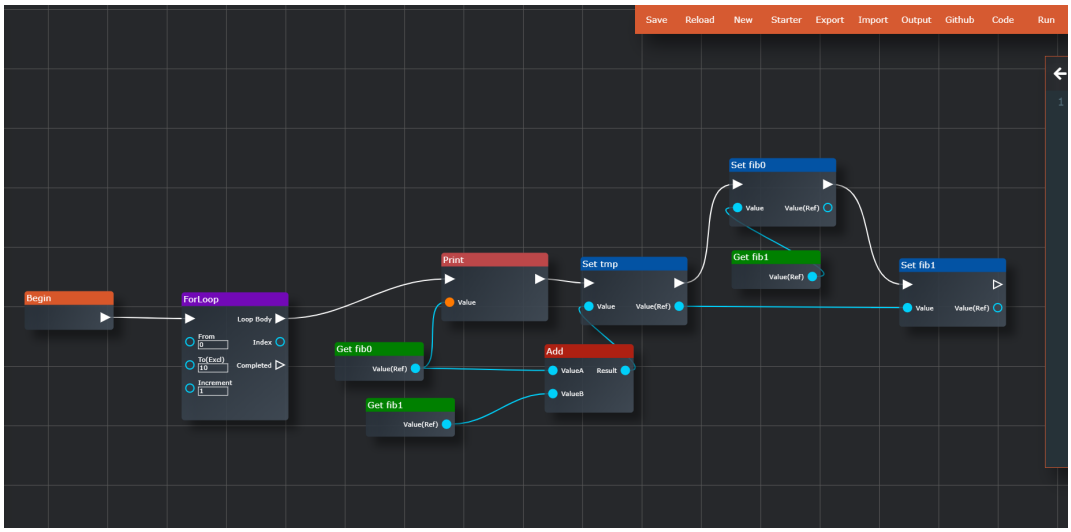

Figure 2: An examples of a Fibonacci sequence in CodeWire

term of low-code logic.

The rest of the deliverable is organized as follows: in section 2 we sketch the low-code approach with a practical example using the deal.II library. In section 3 we sketch instead the no-code approach to the simulation, using the nodes defined in section 2. In section 3 we outline the development plan of PoC and the relative improvements we envision.

2 Low-code approach to dealii programming

In the present section we go on the details of the design of a node-based low-code interface for an Object Oriented Library, using deal.II as an operative example.

A minimal set of entities and control loop that allow the creation of an object-oriented, Turing-complete language is the following:

- Variables and instances of objects
- Functions and member functions
- Loops
- Conditionals

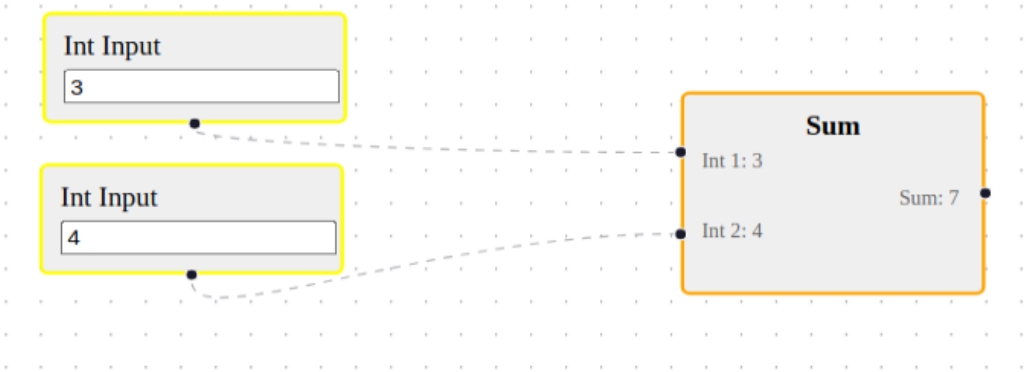

Figure 3: Node-based mockup of a sum operation

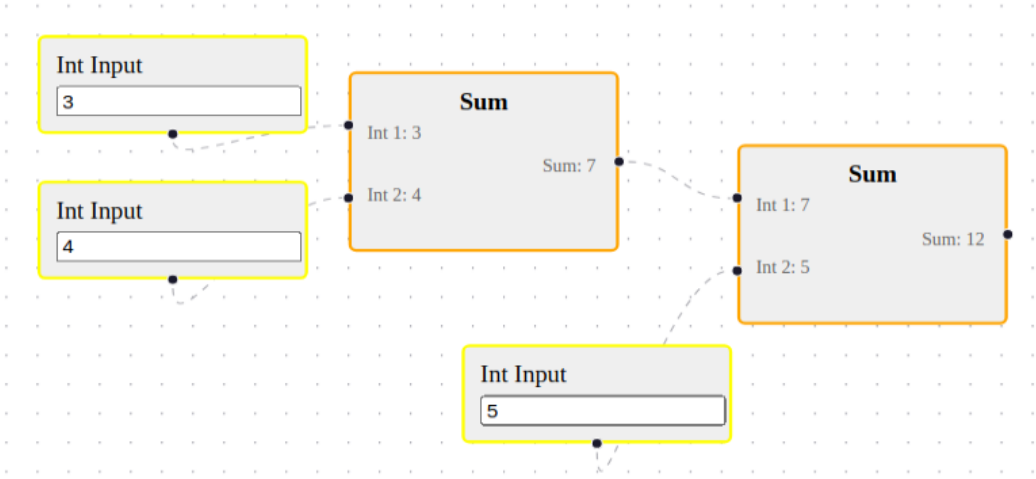

Figure 4: Node-based mockup of a two subsequent sum operations

Each of these entities can be represented as a node in a graph, with inputs and outputs. The graph is then traversed in a way that is consistent with the logic of the programming language.

For a example, a basic type like int can be represented, as shown in Fig. 3, as a node with a single output pin. The sum is then represented as a node with two input pins and one output pin.

To specify order of the operation, special connectors one the node are used, and special edges are display, to distinguish instruction flow from arguments. For example, subsequent sums would look like (fig.2).

Object work in the same way, but member functions nodes can be attached to the object instance node, and the output of the function can then be passed to other nodes.

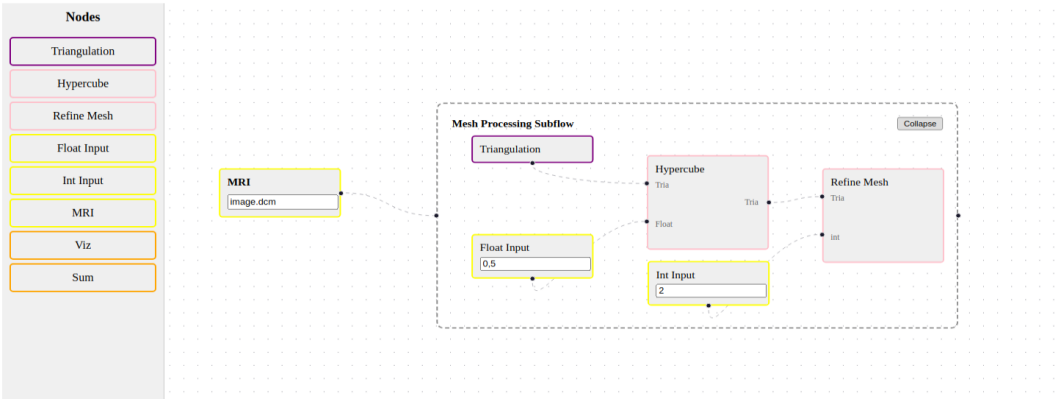

Figure 5: Node-based mockup of a mesh refinement operation

Fig. 5 shows an example of a node-based representation of a mesh refinement operation in deal.II. The mesh is read from a file, and then refined, calling a generic function and member functions of the mesh object.

The power of node-based programming is twofold: on one side, it allows to create complex operations without knowing the details of a specific programming language. On the other side, it allows to encapsulate a collection of nodes into a single node, that can be reused in other parts of the program. These hierarchical features unleash the possibility of creating high level entities that can be used by non-expert users. The latter is the key feature that links the low-code and no-code approach, allowing the creation of a single environment that can be used by both expert and non-expert users.

Moreover, the node-based approach naturally leads to the definition of a computational graph that can be used to optimize the execution of the program: by analyzing the graph, it is possible to determine which parts of it can be executed concurrently, which allows a potential second layer of parallelization on top of the one that is already present in the libraries. Operatively, we are currently investigating the use of the taskflow library to implement graph analysis and node scheduling.

3 No-code approach to simulation management

As noted in the previous section, given the hierarchical nature of node-based programming, it is possible to create high-level entities that completely hide the technical details, while still allowing the creation of complex dataflow. For example, the

Figure 6: Mockup of the high level no-code interaction

simulation block defined in fig.6 can be encapsulated in a single, opaque node, that only requires e.g. an MRI input and a place to display the results.

This high level view of the computational workflow will also allow the quick selection of the computational resource to be used, abstracting the details of different HPC or cloud systems. This system, which will be called ”metascheduler”, will be developed in the context of the dealii-X project, and will be used to manage the computational resources of the PoC platform.

The entire platform, from the low-code interface to the metascheduler, will be served by means of a webapp, that will allow the user to interact with the system from any device, and will be developed using the latest web technologies.

4 PoC development plan and outlook

The main development activities for the PoC platform will be carried out in task 2.6.2 and 2.6.3. The development activities will start in M7, with main frontend and backend of the low-code/no-code PoC platform being developed concurrently. The low-code system will be agnostic of the underling simulation library used, in order to easily integrate the progess of the various dealii-X libraries and applications for specific organs. The activities on the metascheduler will start in M7 as well. Both the low-code/no-code interface and the metascheduler will be developed in an iterative way, with the first version of the PoC platform being deployed in M17. Between M18 and month M22 the PoC platform will be tested by the partners, and the feedback will be used to improve the platform in the following months. The final tranche of the development will be between M23 and M27, in which the final version of the PoC platform will be deployed.

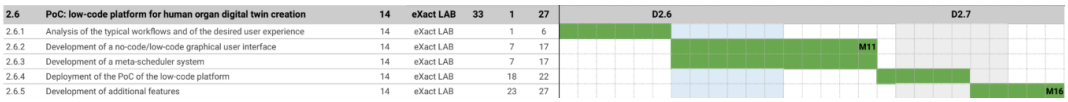

Figure 7: Timeline of the activities of task 2.6, Creation of the PoC platform

The Gantt chart of the activities is shown in fig.7. The frontend of the PoC platform will be in charge of Dualistic, while the backend and the metascheduler will be developed by eXact lab. The activities will be carried out in close collaboration with the other partners of the dealii-X project, UniPI in particular. The PoC platform is foreseeen to be a valuable exploitable result of the dealii-X project, and will be used to showcase the capabilities of the libraries developed in the project.