[DEMO] OpenAlea: Scientific Workflows Combining Data Analysis and Simulation*

Christophe Pradal
UMR AGAP, CIRAD and Inria
Montpellier, France
christophe.pradal@cirad.fr

Christian Fournier
INRA
Montpellier, France
christian.fournier@inra.fr

Patrick Valduriez
Inria and LIRMM, Montpellier, France
Patrick.Valduriez@inria.fr

Sarah Cohen-Boulakia
Inria, Montpellier, France
LRI CNRS 8623, U.Paris Sud
cohen@lri.fr

ABSTRACT

Analyzing biological data (e.g., annotating genomes, assembling NGS data...) may involve very complex and interlinked steps where several tools are combined together. Scientific workflow systems have reached a level of maturity that makes them able to support the design and execution of such in-silico experiments, and thus making them increasingly popular in the bioinformatics community. However, in some emerging application domains such as system biology, developmental biology or ecology, the need for data analysis is combined with the need to model complex multi-scale biological systems, possibly involving multiple simulation steps. This requires the scientific workflow to deal with retro-action to understand and predict the relationships between structure and function of these complex systems. OpenAlea (openalea.gforge.inria.fr) is the only scientific workflow system able to uniformly address the problem, which made it successful in the scientific community. The main originality is to introduce *higher-order dataflows* as a means to uniformly combine classical data analysis with modeling and simulation. In this demo paper, we provide for the first time the description of the system involving an original combination of features. We illustrate the demonstration on a high throughput workflow in phenotyping, phenomics, and environmental control designed to study the interplay between plant architecture and climatic change.

1. INTRODUCTION

Classical bioinformatics analysis (e.g. annotating genomes, building phylogenetic trees, assembling NGS data) involves the management and processing of huge data sets together with the chaining of numerous complex and interlinked tools. Scientific workflow systems aim at facilitating and rationalizing the design and management of such tasks.They clearly separate the workflow specification from its execution and offer useful capabilities on both aspects, such as a user interface to design workflows by composing tools [10], a scheduler to optimize the processing of huge amounts of data [5], a provenance module [3] to keep track of the data used and generated during an execution and ensure the reproducibility of the experiments [13].

However, the complexity of biological analysis increases in emergent interdisciplinary domains addressing the study of complex multi-scale systems that require numerical simulations. In system biology for instance, analyzing the emergent behavior of a large number of interactions within a biological system requires simulating the interplay between the topological and geometrical development of the structure and its biological functioning. This involves coupling models from different disciplines, integrating experimental data from various sources at different scales (gene, cell, tissue, organism and population), and analyze the reconstructed system with numerical experiments.

While scientific workflow systems have mainly been designed to support data analysis and visualization [10, 7, 8, 1], only a few systems have attempted to support iteration or simulation [1, 10]. Most of the systems use either control flow edges or define loops in the workflow specification with specific routing nodes (e.g. switch [6]). Kepler [1] uses black box actors with different models of computation to provide iteration processes. These solutions can lead designing overly complex workflows that are difficult to understand, reuse, and maintain [1]. Expressing control flow (iteration) in scientific workflows is actually a difficult problem due to the absence of state variable and side-effect. To address this problem, we have proposed the concept of λ-dataflow, that makes use of higher-order constructs in the context of dataflows theory and thus allows to represent control flow using algebraic operators [5] (e.g., conditionals, map/reduce...). λ-dataflow allows to model retro-action. OpenAlea [12] is a workflow system based on λ-dataflow and is able to uniformly deal with classical data analysis, visualization, modeling and simulation tasks.

In this paper, we show how the notion of λ-dataflow allows OpenAlea to uniformly deal with workflows involving data analysis and simulation steps. Our demonstration introduces the capabilities of OpenAlea on a workflow involving high-throughput phenotyping, phenomics, and environmental control, to study the interplay between plant architecture and climatic change. OpenAlea is a Python open source project (openalea.gforge.inria.fr) that provides support to a large community of users and developers.

2. USE CASE

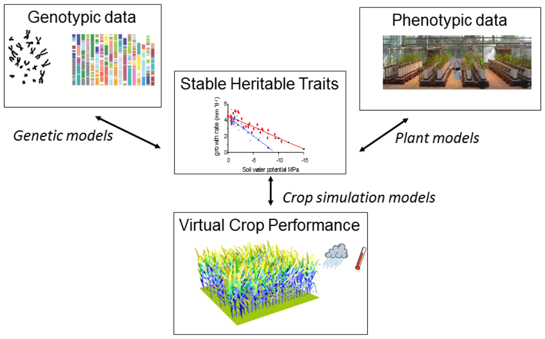


Figure 1: Use case

We consider a use case in the context of crop plant breeding [11], where high throughput data analysis needs to deal with simulation models at different scales (genes, organism and population). The objective of a breeding program is to produce plants that perform better than others (higher or more stable yields) in a given environment. This is challenging as environmental conditions vary a lot among cropping areas, and are subjected to rapid change in a global warming context. Model-assisted breeding aims at tackling this issue. It combines ecophysiological plant models, which reduce phenotypic plasticity to a set of environment-independent plant parameters (stable traits), genetic models that link genetic profile (allele set) to stable traits, and simulation models that run virtual experiments to predict crop performance in a large set of environmental conditions. Model-assisted breeding recently gained interest thanks to the development of automated phenotyping platforms that allow for the measurement of plant traits for a large number of accessions in controlled conditions. For example, the M3P-PhenoArch facility[1] allows to characterize daily plant growth and transpiration for 1,600 individuals at a time, together with precise control of water availability to the plants. It respectively generates 52 GB of data per day, 2.75 TB per essay (for a typical 50 days experiment) and 11 TB per year.

Managing such experiments is particularly challenging due to the volume of data involved, and the multi-disciplinary nature of the tasks, as it requires biological data production, data analysis, mathematical and biological modeling, and computer simulation. From a scientific workflow perspective, the main issue remains to combine (model-assisted) data analysis and model simulation with retro-action.

3. OPENALEA

This section introduces the OpenAlea system, with its programming and execution models.

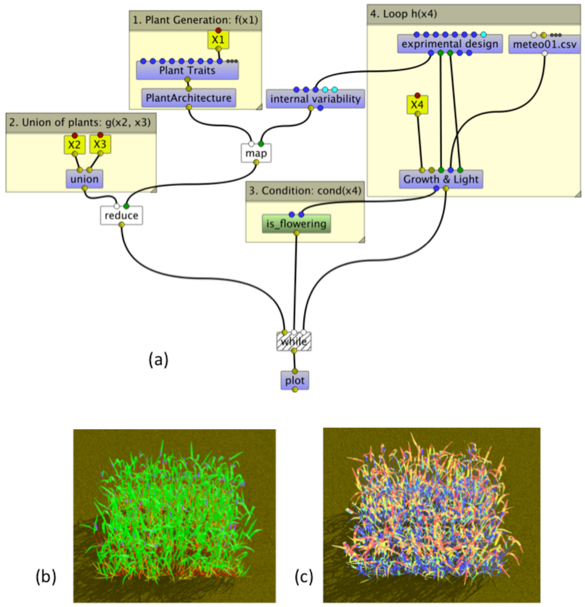


Figure 2: (a) OpenAlea workflow for simulating Maize and Wheat crop performance based on phenotypic and environment data, and two image outputs (b and c). Colors represent the organ's type in (b) and the amount of intercepted light in (c).

Actors and workflows. An actor in OpenAlea is an elementary brick (a.k.a. component or activity) that has a name, a function object (a functor, a program, a web service or a composite actor), and explicitly defined input and output ports. A semantic type [1] is associated to each port (with a corresponding color). A workflow is represented as a directed multi-graph where nodes are actors, and directed edges are data links between output and input ports (see Fig. 2(a)). A workflow can become a (composite) actor in another workflow to allow composition.

Dataflow variable. One of the major originality of OpenAlea lies in the way iteration is handled by introducing a specific kind of actor, called *dataflow variable* X. It allows to specify that, at a given port, an actor receives an unbound variable rather than a value. Connecting an X to an actor transforms a workflow into a lambda function, and allows to express higher-order programming providing control flow behavior using a set of algebraic operators. The three iteration types can be expressed as [6, 4]: (1) counting loops without dependencies (*map* operator), (2) counting loops with dependencies (*reduce* and *for* operators) and (3) conditional loops (*while* operator). In Fig. 2(a), the dataflow variables and the algebraic operators are represented using yellow and white nodes, respectively.

Execution. Dataflow execution in OpenAlea is orchestrated in a *model-driven* manner (rather than input-driven): the

[1] http://www6.montpellier.inra.fr/lepse/M3P

execution of a given workflow is launched in response to requests for data of one of its actors. Such an actor can satisfy the request when the upstream subworkflow has been executed, that is, when all the relevant actors connected to its input ports have been executed. When such an actor has received its data on its input ports, it executes and places data on its output ports. OpenAlea is able to deal with extremely large datasets to perform big data analysis in parallel environments. Additionally, it allows actors to be *lazy* and *blocked*. When an actor is blocked, the execution is not propagated to the upstream subworkflow and when the actor is lazy, the execution is performed only if the actor's inputs have not changed compared to its previous execution. This type of orchestration performs only the operations needed to produce the required result, executing the subset of the graph relevant to the output [2].

Algebraic operators and λ-dataflow evaluation. An algebraic operator is an actor that iterates over first-order function calls, and thus takes one or more functions as inputs. Ports that require a function have an associated semantic type *Function* (colored in white). For instance, the first input port of the *map* and *reduce* operators requires a function as input (see Fig. 2.(a)). λ-dataflow evaluation differs from the classical evaluation when the workflow contains at least one *dataflow variable* X. In this case, the execution is decomposed into two stages. First, for each port of type *Function*, a subworkflow is computed if the upstream subworkflow contains at least one dataflow variable. This subworkflow is defined by all the actors needed to produce the data on this port, *i.e.* the upstream subworkflow and the connected output port. This subworkflow is dynamically transformed into a function (*i.e.* an actor) of one or several variables corresponding to its dataflow variables. Second, the evaluation of this function by algebraic operators consists in replacing the variables by real data and evaluating the subworkflow using the model-driven algorithm described above.

Reproducibility. OpenAlea allows to make experiments reproducible by providing two capabilities. First, it is able to capture both prospective and retrospective provenance (following the PROV-DM model[2]), that is, it is equipped of a provenance module that keeps track of the complete description of the workflows as well as the full history of the data produced and consumed during each execution. Second, and very originally, OpenAlea's architecture is based on IPython and makes use of IPython Notebooks [14] to generate executable papers (see Fig. 3.(b)).

4. DEMONSTRATION

This section describes the main points of our demonstration based on the workflow depicted in Fig.2. We show how users can create or interact with highly expressive workflows (able to perform analysis, modeling and simulation tasks), both using the visual programming environment (Fig. 3(a)) and IPython notebooks (Fig. 3(b)) of OpenAlea.

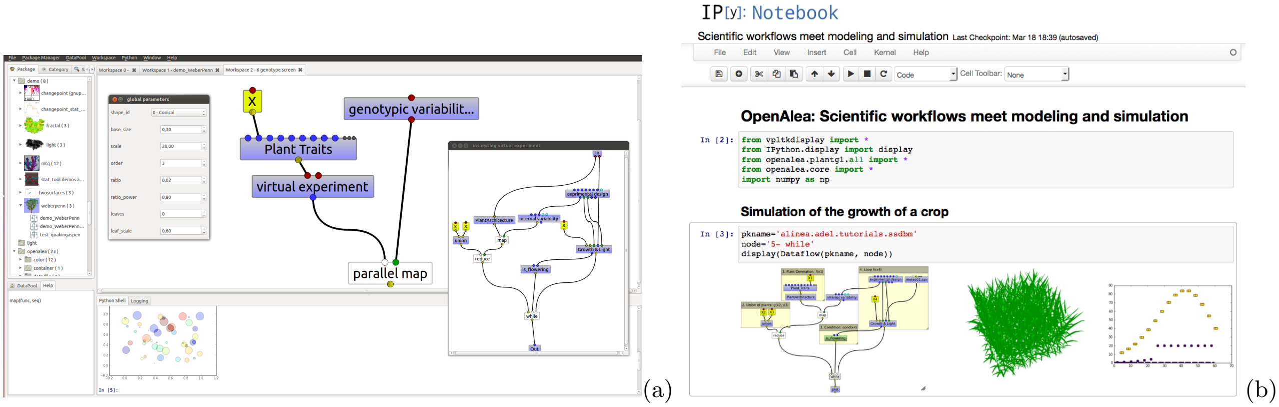
Reusing or designing a workflow. OpenAlea offers a visual programming environment where users are provided with a set of predefined workflows and libraries of tools to be combined to form new workflows (see the left part of Fig. 3(a), "Package Panel"). Users can create new wrapped tools by implementing them in Python. Each tool and workflow is associated with some documentation and saved. Ports of actors are typed and widgets can be associated with data types to allow users interacting with the data (see the widgets depicted in Fig. 3(a)).

(Re)Running a workflow. To execute a workflow, users have to click on their output of interest (as OpenAlea is *model-driven*). For instance in Fig. 2, by clicking on the *plot* actor, users trigger the execution of all the actors of the workflow, from top to bottom. If users click again on any actor of the same workflow, they can visualize or access to intermediate results. OpenAlea determines whether or not any calculation has to be redone (default Lazy mode). If no input or parameter change has occurred, data is not recomputed. Otherwise, the subworkflow impacted by the change is executed again. In Fig. 2(a), the *is_flowering* actor is a non-lazy or eager actor, colored in green. It is always recomputed, even if its input data has not changed.

Using algebraic operators for simulation. Algebraic operators are higher-order actors that take function as argument. In our demonstration, we use three different operators: `map`, `reduce` and `while`. Other types of algebraic operators in OpenAlea follow the same principle while users can define their own operators. The `map` operator is a higher-order function $map :: (\alpha \rightarrow \beta) \rightarrow [\alpha] \rightarrow [\beta]$. Its argument are a function $f :: \alpha \rightarrow \beta$ (first port) and a set of elements of type α (second input port). The `map` operator applies f to each element of the set and returns the set of resulting elements of type β. Similarly, the `reduce` operator takes a function g of two variables and a sequence of elements $[x_i]$ and returns one element. `while` is an iteration operator that takes three inputs: an initial element t_0, a boolean function *cond* and function h. It initializes a variable t with t_0 and iteratively applies the function h on t while $cond(t)$ is true. In the workflow in Fig. 2, the `map` actor takes a λ-subworkflow f and a sequence of parameters $\mathcal{S}$. The λ-subworkflow f, composed of two actors (*Plant Traits* and *PlantArchitecture*), takes a parameter set that corresponds to one plant trait (*e.g.*, leaf growth dynamic) and generates an object that represents a fully parameterised individual plant model to be simulated. The sequence $\mathcal{S}$ is produced by the actor *internal variability*, and represents the intra-genotype (inter-individual) variability of the trait. During the execution, the `map` actor produces a sequence of individual plant models. The `reduce` operator concatenates this sequence of plants into one graph corresponding to the crop canopy. Finally, the `while` operator simulates the development of the crop by iterating a growth function, that takes into account environmental data (meteo01.csv), the state and the specific parameters of each plant and the light intercepted by each 3D organs. The later is computed from the 3D geometry of the canopy. The simulation stops at the flowering stage.

Last, this workflow is reused as a composite workflow (see

[2] http://www.w3.org/TR/prov-dm/



Scientific workflows meet modeling and simulation Last Checkpoint: Mar 18 18:39 (autosaved)

OpenAlea: Scientific workflows meet modeling and simulation

```
In [2]: from vpltkdisplay import *
        from IPython.display import display
        from openalea.plantgl.all import *
        from openalea.core import *
        import numpy as np
```

Simulation of the growth of a crop

```
In [3]: pkname='alinea.adel.tutorials.ssdbm'
        node='5- while'
        display(Dataflow(pkname, node))
```

Figure 3: OpenAlea (a) visual programming environment and (b) IPython notebook

Fig. 3.(a)) and run on a large set of genotypes to select the most efficient plant variety in a given environment.

From workflow to executable paper. Execution of OpenAlea workflows can be embedded into IPython notebooks (Fig. 3(b)), able to produce *executable papers*, where users can share, visualize and interact with input and output produced by each step of an in-silico experiment in a web-based application.

5. CONCLUSION

Faced with the need of coupling data analysis with modeling and simulation, OpenAlea provides a unique solution able to extend the dataflow model of computation by introducing higher-order language constructs in a visual programming environment. Introducing first-class functions allows to design highly expressive workflows in a fully uniform way. First-class functions are increasingly popular and have also been introduced in several imperative languages like PHP, VisualBasic, C# or C++. While OpenAlea is in used since 2007 (160,000 downloads, 1,200 unique visitors a month, 20 active developers) leading to several biological findings (e.g., [9]), this paper is the first to provide an overview of its major capabilities and introduce the λ-dataflow concept. This demonstration deals with the study of plant response to climatic change illustrating the research challenges in areas of high and increasing interest including *big data analysis* and *reproducible science*.

Acknowledgements

Work done in the context of the Computational Biology Institute (http://www.ibc-montpellier.fr)

6. REFERENCES

[1] S. Bowers, B. Ludascher, A. H. Ngu, and T. Critchlow. Enabling scientific workflow reuse through structured composition of dataflow and control-flow. In *ICDE Workshops*, pages 70–70. IEEE, 2006.

[2] V. Curcin and M. Ghanem. Scientific workflow systems-can one size fit all? In *Proc. of Biomedical Engineering Conference*, pages 1–9, 2008.

[3] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.

[4] J. Dias, G. Guerra, F. Rochinha, A. L. Coutinho, P. Valduriez, and M. Mattoso. Data-centric iteration in dynamic workflows. *Future Generation Computer Systems*, 2014.

[5] J. Dias, E. Ogasawara, D. De Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Algebraic dataflows for big data analysis. In *Proc. of IEEE Big Data*, pages 150–155, 2013.

[6] E. Elmroth, F. Hernández, and J. Tordsson. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. *Future Generation Computer Systems*, 26(2):245–256, 2010.

[7] J. Freire, C. Silva, S. Callahan, E. Santos, C. Scheidegger, and H. Vo. Managing rapidly-evolving scientific workflows. *Proc. of IPAW*, pages 10–18, 2006.

[8] J. Goecks, A. Nekrutenko, and J. Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010.

[9] M. Lucas, K. Kenobi, D. Von Wangenheim, U. Voβ, K. Swarup, I. De Smet, D. Van Damme, T. Lawrence, B. Péret, E. Moscardi, et al. Lateral root morphogenesis is dependent on the mechanical properties of the overlaying tissues. *Proc. of the Nat. Academy of Sciences*, 110(13):5229–5234, 2013.

[10] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble. Taverna, reloaded. In M. Gertz, T. Hey, and B. Ludaescher, editors, *Proc. of SSDBM*, Heidelberg, Germany, 2010.

[11] B. Parent and F. Tardieu. Can current crop models be used in the phenotyping era for predicting the genetic variability of yield of plants subjected to drought or high temperature? *J. of Experimental Botany*, 65(11):6179–6189, 2014.

[12] C. Pradal, S. Dufour-Kowalski, F. Boudon, C. Fournier, and C. Godin. Openalea: a visual programming and component-based software platform for plant modelling. *Functional plant biology*, 35(10):751–760, 2008.

[13] G. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig.

Ten simple rules for reproducible computational research. *PLoS comp. biology*, 9(10):e1003285, 2013.

[14] H. Shen. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151–152, 2014.