ALGEBRAIC EXPERIMENT LINE: AN APPROACH TO REPRESENT SCIENTIFIC EXPERIMENTS BASED ON WORKFLOWS

Anderson Souza Marinho

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2015

ALGEBRAIC EXPERIMENT LINE: AN APPROACH TO REPRESENT SCIENTIFIC EXPERIMENTS BASED ON WORKFLOWS

Anderson Souza Marinho

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof[a]. Marta Lima de Queirós Mattoso, D.Sc.

_____
Prof[a]. Vanessa Braganholo Murta, D.Sc.

_____
Prof. Mario Roberto Folhadela Benevides, D.Sc.

_____
Prof. Marco Antonio Casanova, Ph.D.

_____
Prof. Antônio Tadeu Azevedo Gomes, D.Sc.

_____
Prof. Leonardo Gresta Paulino Murta, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
DEZEMBRO DE 2015

Marinho, Anderson Souza

Algebraic Experiment Line: An Approach To Represent Scientific Experiments Based on Workflows / Anderson Souza Marinho – Rio de Janeiro: UFRJ/COPPE, 2015.

XIII, 94 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Vanessa Braganholo Murta

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 83-90.

1. Scientific Workflows. 2. Experiment Line. 3. Provenance. I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

To my Parents.

Acknowledgments

To God for the several blessings during this time. Certainly I could not finish this work without Him. In special, for the God's deliverance from a serious motorcycle accident that I suffered that let me off work for a long period.

To my parents Luiz Antônio and Lydia Maria, who always supported me. This work would not be a reality without them. This achievement also belongs to them.

To my advisors Marta Mattoso e Vanessa Braganholo, who accepted to conduct this work and believed in my potential. I am very happy to have had them as my advisors. They are professors that I have a profound admiration. Without doubt they were very important to the conclusion of this work. They always were present to discuss about problems or issues that arouse during the research. I learned a lot from them.

To all of my research group that contributed to this work. In special to Daniel Oliveira, Eduardo Ogasawara, and Vitor Silva that helped me presenting the several previous works and tools of the research group and for the tips and advices to improve my work. Also, to Kary Ocana that helped me in the evaluation of my approach by sharing her knowledge in bioinformatics and introducing to me a real experiment of phylogenomic analysis.

To my colleagues in Petrobras that could witness most of my concerns, frustrations, and achievements during the doctoral. I thank them for understanding this difficult period and for their support.

To the professors Leonardo Murta, Mario Benevides, Marco Casanova, and Antonio Tadeu, who accepted the invitation of joining the committee to evaluate my thesis. Their analysis having a different point of view about the work was very important to further improve this thesis.

To UFRJ, COPPE, and CAPES for funding my doctoral work and providing an excellent environment for study and research. I'll keep in my memory the 11 years that I spent in that place and the friends that I got. In special, to Rodrigo Santos and Eldanae Teixeira, beloved friends since my under-graduation.

And to all persons not above mentioned that contributed directly or indirectly to this work.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

LINHA DE EXPERIMENTO ALGÉBRICA: UMA ABORDAGEM PARA REPRESENTAR EXPERIMENTOS CIENTÍFICOS BASEADOS EM WORKFLOWS

Anderson Souza Marinho

Dezembro/2015

Orientadores: Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

Programa: Engenharia de Sistemas e Computação

A natureza exploratória de experimentos científicos computacionais envolve não somente variações de execução de um mesmo *workflow*, mas também modificações na definição do *workflow*. Sem um apoio computacional, a definição da hipótese científica e as diversas tentativas de execução do *workflow* não são registradas, ficam apenas na cabeça do cientista. Ogasawara et al. (2009a) definiram uma abordagem chamada Linha de Experimento, que representa um experimento como uma linha de produto, modelando suas diversas variações de implementação em um modelo único e integrado. Entretanto, esse processo de derivação da linha de experimento é *ad-hoc* e não há um registro de dados resultantes de cada execução, os correlacionando com os diferentes níveis de abstração. Esta tese propõe uma extensão da abordagem de linha de experimento, denominada Linha de Experimento Algébrica (LEA). LEA utiliza uma álgebra de workflow centrada em dados que facilita o mapeamento da linha de experimento até as execuções de workflows correspondentes. Com esta representação é possível consultar dados de proveniência que associem dados de execução dos workflows derivados da abordagem LEA com as suas correspondentes definições abstratas do experimento.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ALGEBRAIC EXPERIMENT LINE: AN APPROACH TO REPRESENT SCIENTIFIC EXPERIMENTS BASED ON WORKFLOWS

Anderson Souza Marinho

December/2015

Advisors: Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

Department: Systems and Computer Engineering

The exploratory nature of a computational scientific experiment involves not only executing variations of the same workflow but also changing the workflow definition. Without computational support, the scientific hypothesis definition and the several workflow trials are not registered, they are kept only in the mind of the scientist. Ogasawara et al (2009a) defined Experiment Line, which represents an experiment as a product line, modeling its several implementation variations within a unique and integrated model. However, this derivation process in the experiment line is ad-hoc and there is no tracking of data generated of each execution, correlating them to the different levels of abstractions. This thesis we propose an extension of the experiment line approach, called Algebraic Experiment Line (AEL). AEL uses a data-centric workflow algebra that easies the mapping of the experiment line to the corresponding workflow executions. With this representation it is possible to query provenance data that correlates workflow execution data from the AEL derived workflows with their corresponding abstract experiment definition.

General Index

Figure Index

Table Index

Chapter 1 Introduction

Scientific experimentation is one of the ways used by scientists to investigate phenomena with the aim of acquiring new knowledge or correcting and integrating previously established knowledge (WILSON JR, 1991). In the last decades, High Performance Computing (HPC) has evolved as an important resource to scientists, assisting in computational simulation and interpretation of experimental scientific data. As a result, the main components of an experiment (*e.g.*, the experiment environment, objects, and subjects) began to be simulated, giving rise to a new category of experiments, the computational experiments (SACKS et al., 1989).

Scientific workflows are used as an abstraction that structures the steps of a computational scientific experiment. They may be represented as a graph of activities, in which nodes correspond to data processing activities and edges represent the dataflow dependency between them (TAYLOR et al., 2007b). Scientific workflows enable an organized composition of programs as a sequence of activities aiming at a specific result. Scientific workflow management systems (SWfMS) are becoming an indispensable tool in several scientific domains. For example, in the medical imaging area, scientific workflows are deployed on grids for addressing the computing and storage needs arising from the manipulation of large fragmented medical data sets on wide area networks (MAHESHWARI et al., 2009). Climatologists use scientific workflows for visual exploration of climate data (SANTOS et al., 2009). Other examples can be found in domains such as physics (FOX et al., 2006), ecology (MICHENER et al., 2005), chemistry (SUDHOLT; ALTINTAS; BALDRIDGE, 2006), astronomy (BLYTHE et al., 2005), bioinformatics (GOECKS; NEKRUTENKO; TAYLOR, 2010; OCAÑA et al., 2011; OINN et al., 2004), oil & gas (DIAS et al., 2015).

SWfMS have been developed to support scientists in the process of composing, executing, and analyzing the results of the scientific workflow execution. Thanks to these systems, scientists do not need to do tedious tasks such as activities dispatching or data transferring from one program to another (STEVENS et al., 2003). These tasks are responsibilities of the SWfMS, which manage the workflow orchestration, including its parallel execution. Furthermore, SWfMS improve knowledge sharing, reuse, repeatability and data provenance capture of workflows (GODERIS et al., 2005).

The exploratory nature of a computational scientific experiment (SACKS et al., 1989) involves not only executing variations of the same workflow using several data combinations, but also changing workflow activities to explore different methodologies, approaches, or tools. To follow a scientific hypothesis, the scientist plans a computational experiment, which may lead to several workflow execution trials (*i.e.,* the different workflows executions that scientists experiment using different assumptions). Mattoso et al. (2010) define a computational scientific experiment as the set of all workflow executions that correspond to the planning and computational evaluation of a scientific hypothesis. From now on, we call it simply *experiment*. The life cycle of an experiment can be described as the phases that comprise the experiment from its conception, implementation, until its final results (MATTOSO et al., 2010). Figure 1 illustrates the experiment life cycle where the arrows indicate that iterations happen in the entire cycle, but also in each particular phase. Despite the several proposals of experiment life cycle views (DEELMAN et al., 2009; GÖRLACH et al., 2011), the innovation introduced by Mattoso et al. (2010) is related to the provenance database, at the center, which promotes an integration of all phases. Provenance data provide historical information about data manipulated in a workflow (FREIRE et al., 2008). This historical information expresses how data products were generated, showing their transformation processes from primary input and intermediary data (FREIRE et al., 2008).

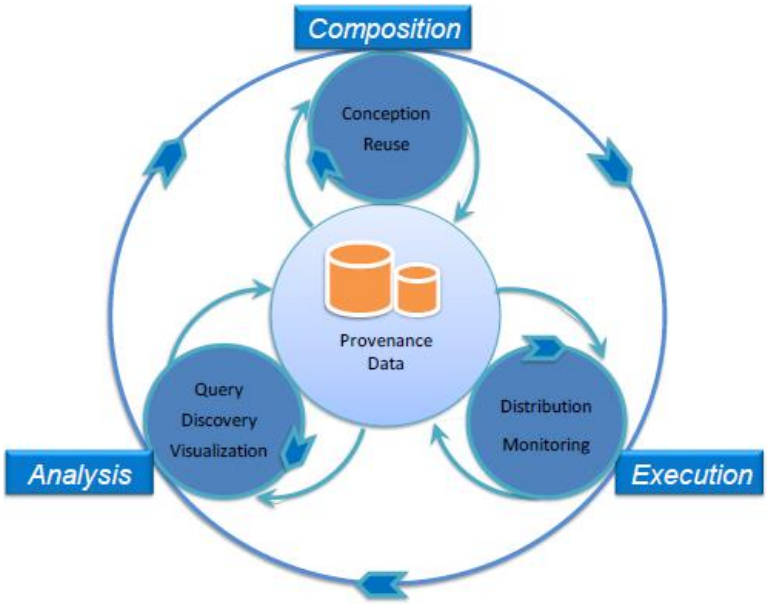

Figure 1. Provenance data supporting the whole scientific experiment life cycle (MATTOSO et al., 2010)

The interaction of different workflows produced in an experiment life cycle can happen into three main phases:

(i) **Composition:** this phase deals with any workflow configuration such as the activity scope definition, the selection of an adequate program or component to be used in a workflow activity, and the data flow between the workflow activities. The focus on this phase is in the development of workflows at different abstraction levels, using different representation models as textual scripts or graphical modeling (GIL, 2007).

(ii) **Execution:** the focus of this phase is on the workflow execution, including the distribution of its programs and data (COUVARES et al., 2007) in a computational environment (e.g., clouds, grids, and clusters) and monitoring of program executions (HULL et al., 2006).

(iii) **Analysis:** this phase focuses on the evaluation of the experimental results generated from workflow executions, which include activities such as data visualization, query and provenance analysis. Furthermore, the experiment results are documented in order to be shared with other scientists that want to re-execute the experiment (FREIRE et al., 2008).

Despite the highly effective SWfMS support in the workflow execution and analysis phases, they still lack a higher level support for the workflow composition phase. The scientific experiment life cycle is not managed as a whole, relating all the workflow trials. Without computational support, the scientific hypothesis definition and the several workflow execution trials are not registered and kept only in the mind of the scientists. There is no representation or supporting system to integrate these trials within the same analytical environment. As a consequence, the scientist may lose track of the trials during the experiment's plan and hypothesis evaluation.

Before executing a workflow, scientists have to compose their workflows in a specific workflow language. To do so, they have to decide which SWfMS to use. There are multiple options, such as VisTrails (CALLAHAN et al., 2006), SciCumulus (OLIVEIRA et al., 2010a), Kepler (ALTINTAS et al., 2004), Taverna (HULL et al., 2006), Swift (WILDE et al., 2011), Pegasus (DEELMAN et al., 2007), and Askalon (FAHRINGER et al., 2005). Each of these SWfMS has its own language and specificities such as parallel execution, visualization support, and semantics. However,

none of the existing SWfMS provides support for modeling several workflow trials as a unique experiment. They all represent a single workflow (SHOSHANI; ROTEM, 2009) at a time, *i.e.,* only one of the possibilities of a workflow composition from the experiment.

Since SWfMS are focused on execution, the composition phase support is very basic. Workflows are usually specified in two abstraction levels[1]: abstract and concrete (CAVALCANTI et al., 2005; MATTOSO et al., 2010). In the abstract workflow, activities are specified with the sole concern of modeling what needs to be done. It is a way to express the workflow using a "common language" that can be comprehensible by any scientist from the application domain area. The abstract workflow is used to guide the specification of the concrete workflow, which establishes the programs that are going to be used in each activity modeled in the abstract workflow. Furthermore, in practice, there is still an additional level, which can be called *execution level* that represents the derivation of the concrete workflow parameterized with different values and resource allocation to be executed in the SWfMS.

The importance of composing scientific experiments at abstraction levels higher than the ones provided by SWfMS has been reckoned before. There are many approaches in the literature for representing experiments aiming at improving semantics of the abstract workflow (ACHER et al., 2010; BELHAJJAME et al., 2014; GIL et al., 2007; LIN et al., 2009; LUDASCHER; ALTINTAS; GUPTA, 2003; PLANKENSTEINER; MONTAGNAT; PRODAN, 2011; QIN; FAHRINGER; PLLANA, 2007; SALAYANDIA et al., 2006; SANTOS et al., 2009; WILKINSON; VANDERVALK; MCCARTHY, 2011). However, they fail at providing a model or a supporting system to integrate the variations that an experiment can present within the same analytical environment. They do not formalize these variations in a way that allows for tracking and querying the results from workflows generated for the same experiment, classifying the origin of each derived workflow to its pre-defined variation.

---

[1] In the literature, some work (TAYLOR et al., 2007b) uses the same concrete workflow definition presented in this paper to define abstract workflows. Concrete workflows for these authors are workflows implemented in an execution environment and executed with specific parameters. For more detail, see Section 2.5.

This same limitation is found in semantic approaches that derivate configurations from a high level abstract representation. For example, based on a high level ontology definition (NOY, 2004; W3C, 2012) inference systems may derivate concrete level definitions that are consistent to the high level ontology. Other examples are the hierarchical planning techniques (CIARLINI et al., 2010; FURTADO et al., 2008; LIMA; FURTADO; FEIJÓ, 2015; NAU; GHALLAB; TRAVERSO, 2004). The high level plan representation may represent the experiment in a high level and through planning derivation techniques several correct concrete compositions are automatically generated. Another example is the software product line derivation techniques. All these semantic approaches are very powerful to generate correct compositions at the concrete level. They are very helpful in scenarios where arriving at the composition is the end of the process, like Business Process Modeling. However, in the scientific scenario there is one more step to decide if the composition is final. This next step requires a real execution of the workflow composition, and possibly variations on the parameters of the workflow. The analysis of the execution results may point to the derivation of a new composition. However, these semantic approaches do not present any mapping or representation to the "execution" behavior of the final composition. They are not prepared for provenance registering or supporting the analysis of the execution results mapped to the higher level structures. There is no support for the exploratory nature of workflow trials, which requires analyzes different workflow composition results and mapping back to the high level experiment definition.

The problem of representing scientific experiments in different levels of abstraction becomes particularly apparent when scientists attempt to analyze the results produced by different executions of scientific workflows as part of the same experiment. In this scenario, there are several missing links for connecting prospective and retrospective provenance data[2] (FREIRE et al., 2008), particularly in the occurrence of distribution of concurrent activity executions in HPC environments. This prevents scientists to perform queries related to the experiment as opposed to analyzing a single isolated workflow execution. For example, “which of the workflow trials produced the best result?” “did

[2]Provenance can be categorized into two main kinds (FREIRE et al., 2008): prospective and retrospective. The prospective provenance captures a workflow specification as a recipe for future data derivation. The retrospective provenance captures past workflow execution and data derivation information.

the workflow trial that used program P1 to implement the abstract activity A performed better than the one that used program P2 for this same activity?" Therefore, besides the concrete and abstract levels that are used to model a workflow, there is a need of a higher abstraction level that integrates the experiment information, in such a way that it can also be queried with the corresponding derivations and executions. This is a new level of experiment management enabling users to have a broader view of the experiment as whole. We call it **experiment level**.

The advantages in representing experiments at a higher level than an abstract workflow are described as follows. A higher abstract model (I) relates the information of different workflows created during the experiment life cycle, representing in a (II) unique and integrated model the existing workflows. Additionally, this promotes the experiment (III) reuse and (IV) reproducibility, since it formalizes the workflows common characteristics and their variations.

However, one of the major difficulties in establishing such representation is the lack of models that define relationship mappings traversing different levels of abstraction. Workflows represented in the existing levels of abstractions are not linked, and so cannot be easily related and queried. They are considered separate units, that is, the information that relates one abstract activity to the corresponding activity at another abstraction level is not registered. This is not trivial to infer. Furthermore, modifications in one level of abstraction are not automatically mapped to the others. Even when dealing with only one level of abstraction, the lack of support in managing multiple workflows associated with the same experiment hinders the experiment analyses, because it is complex to view and understand the several approaches used to execute the experiment. There is no identification that a particular activity is an alternative to a given activity – this is not explicit among different workflow trials. Scientists have to use textual annotations to register alternatives, but that is not uniform and it is hard to query. The replacement of a particular activity, in an existing workflow, is considered a different workflow or a new version by the current SWfMS and not an alternative in the context of trials from the same experiment. Even if two workflows are related by a provenance database, the concept of activity alternatives is not registered. Therefore, there is a semantic loss of valuable information from the characterization and conduction of the experiment, due to insufficient automatic support to different

abstraction representations and their corresponding executions, as discussed in the related work (Chapter 3).

Even the approaches that share our motivation of defining a model at the experiment level present some limitations (e.g., absence of abstraction level mapping support, focus only in the concrete level, lacking of models to represent the experiment characteristics, etc.). Due to these limitations, scientists have a narrow view of the experiment conception and management. Even when the experiment is documented using higher abstract level models it is not related to the concrete level. Finally, the resulting execution is disconnected from the concrete and abstract levels. This results in weak provenance analyses and massive use of tacit information that will be lost and unknown for other scientists that want to reproduce the experiment.

1.1 Experiment Line: using software product line to address experiments

Over the last years, the software engineering community has intensively worked on managing reusable artifacts in the software development process (POHL; BÖCKLE; LINDEN, 2005). The main idea behind managing these artifacts is to allow software developers to generate new software with minimal development effort, reducing financial costs, development time, and improving customer satisfaction. Following this idea, software product lines (SPL for short) (NORTHROP, 2002) were created for supporting this development task.

SPL can be defined as a set of software engineering methods, tools, and techniques for creating a collection of similar software systems from a shared set of software assets using common means of production (NORTHROP, 2002). In this work, we refer to these software assets as *components*. In SPL, a set of software that share common characteristics are grouped into a *family*, *i.e.*, they can be developed based on a set of assets. These assets include architecture, reusable software components, domain models, requirement statements, etc. The task of generating a software system based on a SPL is called *derivation*.

SPL are based on two fundamental orthogonal concepts: *variability* and *optionality*. When there is more than one alternative component that implements a specific characteristic in the SPL, we say there is a variability. In other words, variability allows

for software developers to change and configure components according to a given context. The second concept, optionality, allows a specific component of the SPL to be suppressed when a software system is derived. In order to manage this, components in a SPL can be marked as mandatory or optional. This way, when developers are generating new software based on a SPL, they basically have to determine which artifacts implement a specific component and which components are going to be suppressed in the final software system.

The exploratory nature of scientific experiments can be addressed using the experiment line approach defined by Ogasawara *et al.* (2009a). Inspired by a related problem in the software engineering field, the experiment line approach is an analogy for software product line. Experiment lines represent at a high abstraction level a family of scientific workflows that presents common characteristics (*i.e.*, related to a specific scientific domain) in a unique and integrated model. When experiment lines are modeled, they are used to derive abstract and concrete workflows.

1.2 Problem Definition

Considering the limitations of the existing solutions previously presented in the absence of abstraction level mapping support, focus only in the concrete level, lacking of models to represent the experiment characteristics, etc., the problem definition of this thesis is based on the limitations of the original proposal of experiment line defined by Ogasawara *et al.* (2009a). Such proposal is a step forward to solve the experiment representation problem, but it presents a lot of limitations that inhibit its full potential.

First of all, the workflow derivation process is not formalized. It is done in an *ad-hoc* way. As a consequence, there is no guarantee that workflows derived from the experiment line are adherent to the experiment conception. This problem gets worse due to the absence of a common representation model among the workflow levels. Such model, which clearly defines the data and activities involved in the different experiment abstractions, also provides mappings and relationships to each other. There is limited analytical power in the experiment conduction when the required information has to be manually discovered and related to be queried. For example, we cannot track the origin of a program in its concrete level to its conception in the abstract level or a data product to its specification.

Considering the experiment line representativeness, the proposal of Ogasawara *et al.* (2009a) lacks of semantics to make the dataflow explicit in terms of which operations each workflow activity performs and which data is involved. By adding these semantics, we enrich the experiment line model and improve the workflow derivation process by deriving abstract workflows that are ready to be instantiated in a SWfMS. Moreover, the concrete workflows can be better validated since they have to follow the data operations specifications defined in the experiment line model. Finally, the original experiment line proposal is only based on the concepts of variability and optionality. However, these concepts are not enough to represent other complex experiments characteristics such as conflicts and dependency relationships. The incorporation of these relationships is important to improve the expressiveness and analysis of the experiment line model.

1.3 Main Goal

The main goal of this thesis is to support the exploratory nature of the scientific experiment by providing a representation model for the experiment definition and a process to guarantee a correct workflow derivation and the workflow execution adherence. To address the experiment definition we propose the Algebraic Experiment Line approach (AEL), which builds on the experiment line of Ogasawara *et al.* (2009a). In this thesis, we incorporate new features to the experiment line to make it formal, applicable in practice, and verifiable.

To provide consistency among the derived workflows we introduce a uniform data model that adds semantics to the derived workflows. These semantics represent what and how the data is consumed and produced in the derived workflows. Due to that, even in the abstract level, the experiment line is capable of verifying compatibility between workflow activities by comparing their associated schema. AEL builds on a workflow algebra (OGASAWARA et al., 2011), referred here as the SciWfA algebra. This algebra defines a uniform data model for scientific workflows that describes in high level of detail how the data is consumed and produced in the workflow. Since SciWfA is the algebraic representation of all abstraction levels supported by AEL, this allows for querying traversal from workflow execution data to their corresponding abstract experiment definition. The workflows derived from the AEL can be used by a SWfMS compatible to SciWfA (such as SciCumulus cloud-based SWfMS (OLIVEIRA et al.,

2010a)) and benefit from algebraic transformations to do automatic optimizations in the workflow execution plan. These transformations would still be consistent with the abstract level.

1.4 Hypothesis

The general hypothesis of this work is that *an algebraic approach for the experiment line drives a correct derivation of abstract and concrete workflows that are adherent to the experiment specification. Moreover, it also guarantees that an algebraic workflow execution is adherent to the experiment line semantics by means of common algebraic operations and operands that define, at a high level of abstraction, how data is processed.*

The advantages of a correct derivation and common representation along the abstract, concrete and execution are the consistent analytical power of the components involved in the scientific experimentation. To be able to confirm this hypothesis, the experiment model should represent, in an integrated way, the several alternatives of an experiment that can be explored (*i.e.*, the different workflows that are created during the exploratory nature of conducting an experiment). The experiment management process should dictate how the experiments derive their workflows. From that, scientists can explore derived workflows ready to be executed by selecting specific experiment conceptions at a high level of abstraction.

As a consequence of this experiment model, scientists may benefit from a controlled derivation process, experiment data analysis and resources for the experiment reproducibility.

1.5 Proposed Approach

To guarantee that every workflow derived from the experiment line is valid we define the workflow derivation process and its corresponding system architecture. The proposed architecture describes the environment where the experiment lines are composed and how the workflow derivation process is implemented. The architecture also comprehends the provenance strategy that is used to collect and relate experiment information at different levels of abstraction in an integrated provenance model. To improve the formulation of the analytical queries, the components of the experiment are modeled according to the definitions of the W3C PROV recommendation (MOREAU;

MISSIER; BELHAJJAME, 2013). Additionally, this architecture is implemented in a prototype named ExpLine for supporting the creation of experiment lines and composition of workflows in different levels of abstraction.

Finally, a case study is defined and executed using a real experiment from the phylogenomic area. In this case study, ExpLine is used to compose an experiment line for the phylogenomic analysis experiment. Then, this experiment line is used to derivate concrete workflows to be executed in the SWfMS SciCumulus. To evaluate our hypothesis, we create some queries using the provenance model from Expline. Results show that queries combining data from the experiment line and the corresponding executions improve the experiment data analysis. It also shows a direct way of querying results from different trials involving activity variations, which are defined at the experiment line level of abstraction. Additionally, we have analyzed how the experiment management process was improved when using experiment lines.

In summary, the contributions of this work are as follows.

- A well-defined and formalized workflow derivation process that now works at three abstraction levels: the experiment level; the abstract workflow level; and the concrete workflow level.
- A provenance approach that tracks the information generated in the AEL, enabling the creation of provenance queries that correlates experiment line concepts defined at high abstraction level with derived workflows and their execution data.
- Incorporation of the SciWfA algebra into the experiment line approach, to make explicit the data description and the operation in the experiment abstraction level, thus improving consistency in the model.
- Specification and implementation of a prototype named ExpLine that supports the new functionalities present in AEL.
- A case study that evaluates the AEL approach on managing several trials of an experiment in the phylogenomic area.

1.6 Organization

Besides this chapter, this document is organized as follows. Chapter 2 presents a real astronomy data analysis experiment that will consistently be used in the rest of the thesis as our motivating example and to demonstrate our approach. Moreover, the astronomy experiment is used as example to discuss the challenges involved in the definition of different levels of abstraction to support the experiment management. Chapter 3 discusses related work. Chapter 4 presents the AEL approach, formalizing its main concepts, and specifies the architecture for supporting the creation and management of algebraic experiment lines. Chapter 5 presents the prototype named ExpLine that implements the AEL approach. Chapter 6 presents the evaluation study of the AEL implementation, with the phylogenomic analysis experiment. Finally, Chapter 7 concludes the thesis.

Chapter 2 Motivating Example

To illustrate a real problem of modeling several scientific workflows that represent different trials in the context of the same experiment, let us consider the Montage astronomy experiment (JACOB et al., 2009) that creates image mosaics to perform data analysis, which has been modeled in SciWfA by Silva, Oliveira and Mattoso (2014). We use Montage because it has been used as a benchmark in the analysis of scientific workflow techniques. We consider the Montage workflow (SILVA; OLIVEIRA; MATTOSO, 2014) as the basis of an experiment that uses a series of tools for assembling astronomical images into custom mosaics of the sky suitable for large scale processing. It is possible to build mosaics in the Flexible Image Transport System (FITS) format (GREISEN; CALABRETTA, 2002), according to common astronomy coordinate system, arbitrary image sizes and rotations, and all World Coordinate System (WCS) map projections.

In this Chapter, we introduce the MontageExp experiment, which is an adaptation of the Montage workflow (SILVA; OLIVEIRA; MATTOSO, 2014) to be expressed as a generic astronomy data analysis experiment. MontageExp can be implemented by using either Montage or another similar astronomy image mosaic composition service. Therefore, the workflow activities from MontageExp were redefined in a higher level of abstraction to ease the explanations.

Before that, we present a workflow formalism that encompasses both a data model, a dataflow model, and describes the SciWfA workflow algebra (OGASAWARA et al., 2011). The SciWfA algebra is illustrated in detail when instantiated in this astronomy workflow. The SciWfA algebra definitions presented in this chapter are also the fundamental basis to the experiment line formalization proposed in this thesis, which is presented in Chapter 4. The remaining of this Chapter is organized as follow. Section 2.1 introduces the SciWfA workflow algebra. Section 2.3 describes the MontageExp experiment. Section 2.4 presents the algebraic representation of MontageExp using SciWfA. Section 2.5 discusses the problems that arise when supporting different levels of abstractions using the MontageExp experiment as example.

2.1 SciWfA

The SciWfA algebra improves the semantics in the workflow specification by making explicit the workflow activities, data consumption and production and their associated high-level operations. This algebra adopts relations as the basic data operands for the algebraic operators corresponding to data consumption and production between workflow activities. Assuming this data model, Ogasawara *et al.* (2011) express a scientific workflow as follows.

2.1.1 Data Model

Workflow input and output data as well as the combination of data consumed and produced by activities correspond to relations in the database relational model (CODD, 1990, p. 2). In this context, during the execution, workflow activities consume and produce tuples that follow schema relations, which are defined using a set of primitive data (integer, real, alphanumeric, date, etc.) or references to files. Thus, each relation R is a set of tuples (*i.e.*, $\{r_1, ..., r_n\}$, considering n as the cardinality of R) that follows a schema R, with its respective attribute types.

Each relation R presents the following basic properties: (i) it is composed of one or more attributes; (ii) each attribute has a unique name in the relation, a specific type and associated constraints; (iii) it has a set of attributes that together form a key (that guarantees uniqueness). An example of schema is presented below:

$$R = (Attr1: Integer, Attr2: Char, Attr3: Boolean)$$

Definition 1. (SCIENTIFIC WORKFLOW) A scientific workflow w is a Directed Acyclic Graph (DAG) defined by the tuple $<Nodes, Edges>$. *Nodes* in w correspond to the set of workflow activities (*i.e.*, the set $A = \{a_1, .., a_n\}$, considering that A has n activities), whereas *Edges* correspond to the set of data dependency ($Dep(a_i, a_j)$) that exists between each pair (a_i, a_j) of activities in A, $i \neq j$. An activity a_i is a quadruple $<\{R_1,..., R_m\}, \mathcal{I}, P, op>$, which states that a_i consumes a set of relations $\{R_1,..., R_m\}$ with schemas respectively compatible with $\{R_1,..., R_m\}$ and produces an output relation with schema compatible with $\mathcal{I}$. Additionally, a_i is associated with an algebraic operator op. This means that a_i can only be invoked in association with that particular algebraic operator op. Finally, P is a program that is executed when a_i is invoked. In addition,

Input(a) defines the input relations consumed by activity *a*. Similarly, *Output(a)* defines the output relation produced by *a*. This way, given two activities a_i and a_j, the dependency between them is defined as $Dep(a_i, a_j) \leftrightarrow \exists R_k \in Input(a_j) \mid R_k = Output(a_i)$.

2.1.2 Algebraic Operators

The algebraic operators are associated to workflow activities according to how activities consume and produce tuples. Operators add semantics to data consumption and production along workflow activities. There are five operators in SciWfA algebra: Map, Filter, SplitMap, Reduce, and MRQuery (OGASAWARA et al., 2011). All operators are unary (*i.e.*, consume a single relation), with the exception of MRQuery, which is an n-ary operator (*i.e.*, consumes *n* relations). These algebraic operators are detailed as follows.

Definition 2. (MAP) *Map* is represented as $T \leftarrow Map(a, R)$. Map produces a single tuple *t* in the output relation *T* for each tuple *r* consumed by activity *a* from the input relation *R*. Considering *n* as the cardinality of the input relation *R*, *Map* executes activity *a n* times. *Map* is a one tuple consumption to one tuple production operation.

Definition 3. (FILTER) *Filter* is represented as $T \leftarrow Filter(a, R)$. It selects a tuple *r* from the input relation *R* to be copied to the output relation *T*, as long as *r* complies with a criterion defined in activity *a*. In this case, the output relation, which follows the schema *T*, is a subset of the input relation *R*, which follows the schema *R*. Considering that the cardinality of the input relation *R* is *n* tuples, this operator executes activity *a n* times. The operator produces *m* selected tuples in the output relation *T*, such that $m \leq n$. The selection criterion is internal to the program represented by activity *a*.

Definition 4. (SPLITMAP) *SplitMap* is represented as $T \leftarrow SplitMap(a, R)$. It produces a set of tuples in the output relation *T* for each tuple from the input relation *R* consumed by activity *a*. The splitting criterion is internal to the program represented by activity *a*. *SplitMap* is a one tuple consumption to *n* tuple production operation.

Definition 5. (REDUCE) *Reduce* is represented as $T \leftarrow Reduce(a, G_a, R)$. It produces a single tuple *t* in the output relation *T* from each subset of tuples in the input relation *R*. The tuples from *R* are grouped by a subset of *q* attributes from the relation *R*, which can be represented as $G_a=\{attr_1, ..., attr_q\}$. The set G_a is used to establish the criteria for horizontal partitioning over *R*. Summarizing, the Reduce operator executes the activity

a, consuming each partition at a time and producing an aggregated tuple *t* for each partition, similarly to the Map-Reduce model (DEAN; GHEMAWAT, 2010).

Definition 6. (MRQUERY) *MRQuery* is represented as $T \leftarrow MRQuery(a, \{R_1,..., R_o\})$. It invokes activity *a* to consume tuples from a set of relations $\{R_1,..., R_o\}$ in order to produce a single output relation *T*. It is similar to the join operator in the relational algebra, but using more than two relations. Notice that the join criterion is internal to the program represented by activity *a*.

2.2 Workflow Validation

Definition 7. (VALID SCIENTIFIC WORKFLOW) A minimal set of conditions for a scientific workflow *w* to be valid is:

- each activity must be associated to an algebraic operator;
- the activity signature (i.e., input and output relation cardinalities, the relation schema characteristics) must comply with its associated algebraic operator;
- the input and output relations of an activity must comply with their associated schemas;
- the activities and their dependency relationships must comply with the DAG constraints.

2.3 MontageExp

The MontageExp workflow is composed of five activities as shown in Figure 2.a. The first activity (List Images) transfers image files from an external astronomy repository and extracts every astronomy reference files from each image (which can be more than 20 types of domain data). Each image file corresponds to a mosaic creation. The second activity (Projection) computes the projection of these astronomy-positioning references into a specific plane. Here, we can find the first challenge. There are several coordinate systems that can be used to make the projection and each of them suits specific projection transformations. In our experiment, we consider the Cartesian, Spherical, and Celestial coordinate systems.

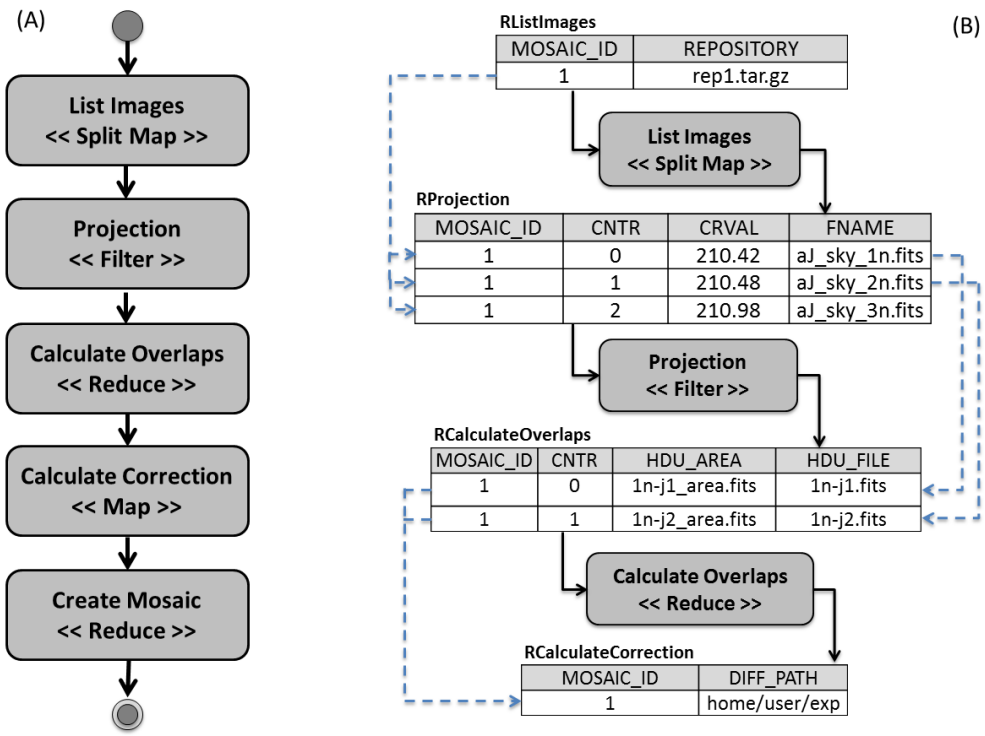

Figure 2. (A) Astronomy data analysis workflow adapted from Silva, Oliveira and Mattoso (2014) **and (B) the three first activities in detail with their relationships to the input and output relations**

The third activity (Calculate Overlaps) analyzes an image metadata table to determine which images overlap on the sky. In this astronomy experiment we selected two different algorithms to calculate overlaps. The first aims at generating better quality (with correction data) mosaics while the second focuses on performance (without correction data). The fourth activity (Calculate Correction) depends on the chosen algorithm for the previous activity. This fourth activity is only executed if the improved quality algorithm was chosen during workflow composition. This means that this activity can be suppressed if the higher quality data is not generated. Finally, the fifth activity (Create Mosaic) generates the mosaic joining all the images. As the previous activity (Calculate Correction), this activity requires that the image correction has previously been processed. Therefore, two algorithms can be used to generate, respectively, a corrected or uncorrected mosaic. Uncorrected mosaics do not have color corrections, but are frequently used because they are faster to obtain and allow scientists to have a short preview of what they are doing.

Summarizing, when analyzing the variabilities and restrictions of this example, we find six different ways to create image mosaics in our astronomy experiment. We have three coordinate systems to project the images and two options to create mosaics (with or without corrections) resulting in six distinct workflows to manage for a single

experiment. It is worth to mention that this experiment was chosen for didactic reasons. Complex experiments with several variabilities and optionalities can have hundreds of distinct workflows.

2.4 Algebraic representation of the astronomy experiment

Figure 2.b illustrates in detail the SciWfA algebraic specification for the first three workflow activities of the astronomy experiment. Besides the algebraic operations described as annotations in the abstract workflow activities from Figure 2.a, Figure 2.b explicitly shows the input and the output relations for these three workflow activities. The input and output relations are represented as tables and the content of these tables are tuples, which represent possibilities of workflow data executions. For instance, the workflow activity List Images is defined as a Split Map operation and its input and output relations are named, respectively, as RListImages and RProjection. On the other hand, RProjection also acts as the input relation for the workflow activity Projection since this is dependent from the workflow activity List Images. This dependency is represented by a solid line arrow connecting the relation to the workflow activities.

Also, in Figure 2.b, the dotted line arrows represent the relation tuples transformations along the workflow. These tuple transformations occur according to the algebra operation specified for each activity. For example, the workflow activity List Images, characterized as a Split Map operation, takes one tuple from the input relation RListImages (which contains a tar.gz file with the sky images) and generates three tuples (the tree sky images uncompressed from tar.gz file) in the output relation RProjection. The provenance information defined in this fine grain level of detail is captured by the SciWfA algebra, which allows for scientists to improve reasoning of the workflow execution analysis. From that, it is possible to visualize that one of the sky images (aJ_sky_3n.fits) was discarded in the Projection activity. A coarse grain provenance support, which only maps the dataflow at relation level, could let scientists to have a wrong idea about the workflow execution. Scientists could conclude that all of the sky images included in the tar.gz were used to create the mosaic.

2.5 Supporting Different Levels of Abstractions: Limitations and Open Issues

Abstraction levels provide a separation of the experiment conceptual information from its implementation, which, according to the environment where the experiment is executed, can show several differences. Let us clarify this problem by using the astronomy experiment. Figure 3.a shows the abstract workflow of the astronomy experiment according to an agnostic notation and its implementations in VisTrails notation (CALLAHAN et al., 2006) (Figure 3.b) to represent concrete executable workflows and some examples of workflow executions with specific parameters (Figure 3.c). Although this Figure presents several alternative concrete workflows using the VisTrails notation, the discussion presented here transcends a particular SWfMS.

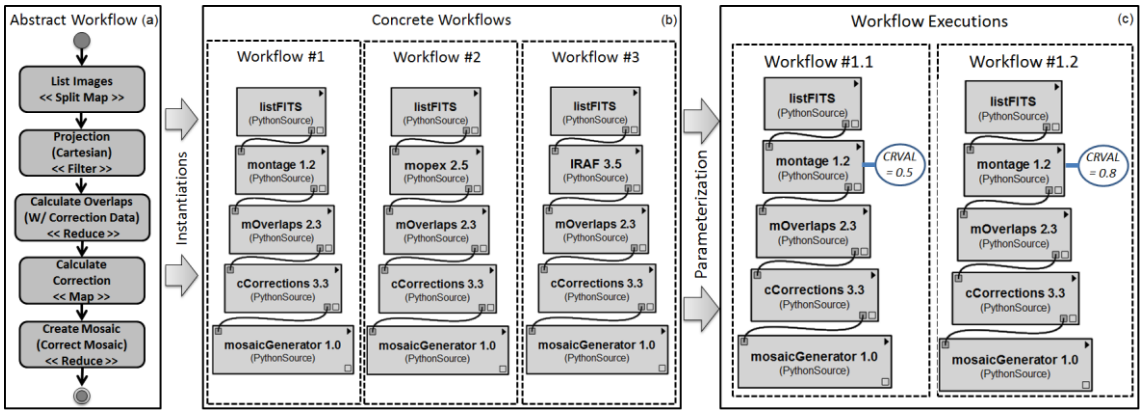

Figure 3. Several instantiations from one astronomy data analysis abstract workflow

Since the abstract level hides some implementation details, it is simpler and easier to understand than the concrete alternatives presented in Figure 3.b (Workflow#1 to Workflow#3). It is clearer at the abstract level that there is a "Projection" activity using the Cartesian coordinate system approach. We can choose several different programs to implement it. In this example, at the concrete level, it uses, alternatively, the programs: Montage (JACOB et al., 2009) (Workflow #1); Mopex (MAKOVOZ; KHAN, 2004) (Workflow #2); and IRAF (VALDES, 1998) (Workflow #3) for this same abstract activity. In addition, for the execution, scientists can choose among several different versions of the same program (*e.g.*, Montage1.31 and Montage2.0) to explore possibilities and evaluate different results considering just the abstract Montage activity.

Finally, at the execution level, the executions of the chosen Montage program in Workflow #1 were parameterized with two different values (CRVAL=0.5; CRVAL=0.8), generating two workflow execution instances (Workflow #1.1 and

Workflow #1.2). It is worth to mention that, in this example, we are dealing with only one experiment, represented by one abstract workflow trial in Figure 3.a. This abstract workflow represents only one possible experiment execution. According to the astronomy experiment, there are two other variations for the activity "Projection": spherical and celestial coordinate systems. It would be necessary to construct three distinct abstract workflows for the astronomy experiment only to represent all of these variations of the "Projection" activity. Nevertheless, these abstract workflows would not be correlated and we would fall into the same problem of the current approaches that develop several individual concrete workflows for each implementation variation.

The common exploration strategy adopted in traditional SWfMS approaches consists in generating several disconnected instances of concrete workflows with different program choices and parameterization for each concrete workflow alternative, also disconnected. This strategy presents several limitations regarding analysis, scalability and maintenance since there is no semantics involved in the workflow and activities representations. For example, it is complex to query mosaic results encompassing variations of Projection activity in the astronomy experiment. Only scientists involved with the experiment conception and implementation are able to do it, that is, find and manually join all the trials if the scientists can remember the names given to all of them. They should individually query the results of each workflow trial executed in a SWfMS, which is error prone, and make their conclusions based on their tacit knowledge. Other scientists that are not directly involved with the experiment may not realize, for instance, that workflows' activities implementing different coordinate systems for image projection (Cartesian, Spherical, and Celestial) have been evaluated in the experiment.

Figure 4 illustrates the solution space of MontageExp experiment grouped by important abstract concepts that scientists usually use in a typical provenance analysis. The space solution represents the implementation of MontageExp experiment, i.e., concrete workflows that were created during its lifetime. In traditional SWfMS solutions, the provenance analysis is done using information directly related to the concrete workflow. It is unfeasible to make provenance queries using filters/parameters at a higher level of abstraction that regards approaches used in the experiment or important experiment characteristics. For instance, it is unfeasible to analyze the mosaics that were generated in MontageExp using the Cartesian coordinate system for the projection (i.e., the

"Cartesian projection" set of Figure 4). Additionally, it is complex to gather all the mosaics that were generated without correction process to make a deep analysis (i.e., the "without correction" set of Figure 4).

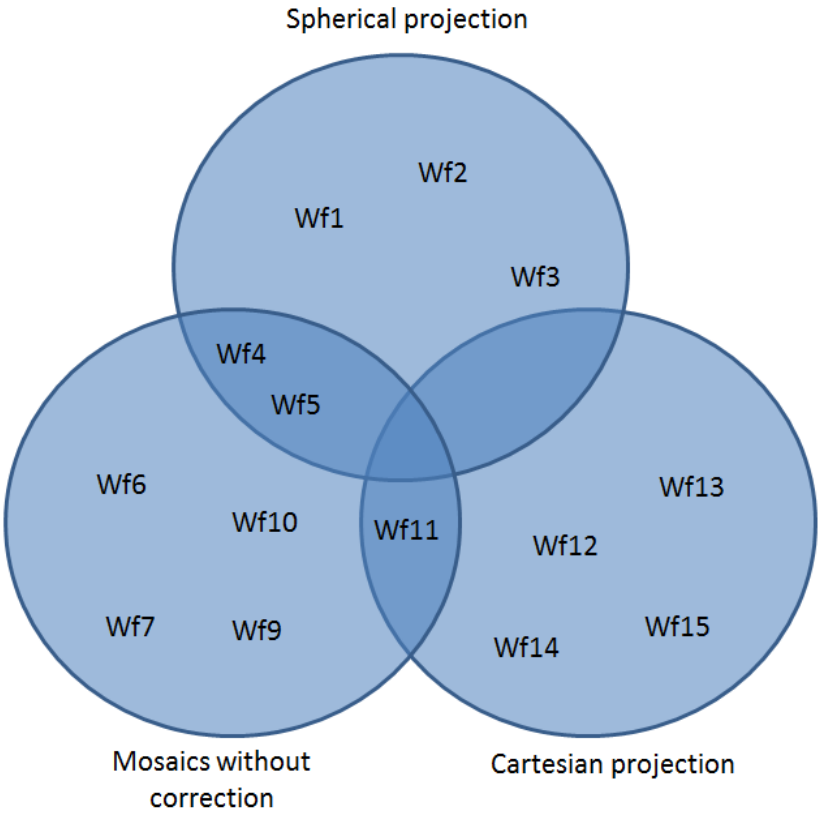

Figure 4. MontageExp's space solution grouped by some abstract concepts

On the other hand, in the semantic approaches, when the experiment trials are modeled in an integrated way and the derivation process metadata is registered, it would be possible to submit the following query: is there any workflow composition generated without correction process? However, it would not be possible to find out their related executions, and even more complex, to find which ones have those resulting values. Even if the scientist has access to all workflow execution data, it would be very difficult and error prone to filter out which results come from compositions generated without correction process.

In the case of using the original experiment line approach (OGASAWARA et al., 2009a) to represent the MontageExp, we could integrate all the information in a unique model. Similar programs present in the concrete workflows would be described in terms of variabilities and optionalities in the experiment line. However, the description about the approach used for each program would be lost and we would not be able to organize

the experiment using that information. Scientists would be able to use the experiment line as a guide to select which program to be chosen in a specific workflow activity and a concrete workflow would be derived. Nevertheless, the derivation process would not be registered using a provenance database and so there would be no tracking about the derived workflow correlating the features (in this case, programs) selected in the experiment line and their corresponding execution data. Therefore, the previous presented provenance queries are not supported by the original experiment line either. In Chapter 5, we explore other analytical provenance queries. For more details, see Section 3.2.

2.6 Final Remarks

The astronomy experiment presented in this chapter is just a simple scenario – there are many more complex scenarios with several variabilities and restrictions. Even the Montage experiment becomes much more complex in a larger real scale. This is also a reality in other domains such as Computational Fluid Dynamics (CFD) (ELIAS; COUTINHO, 2007; GUERRA et al., 2012; LINS et al., 2009), Uncertainty Quantification in Oil & Gas applications (DIAS et al., 2015) and other scenarios of large-scale experiments (*i.e.,* experiments that manipulate large amounts of data and demand HPC capabilities). Particularly, in the HPC scenario, scientists must analyze many possible implementation alternatives of the experiment just to improve performance. When scientists are not satisfied with a particular workflow composition, they replace a specific approach or program by another, modifying the workflow composition. Yet, there are scenarios where it is necessary to amend parts of the experiment, as in the case of an exchange of a particular technological infrastructure (*e.g.*, moving from clusters to clouds). In such cases, workflows may have to be completely recomposed. In Chapter 5, we present a case study with an experiment from bioinformatics, which may derive 30 distinct workflows. In this case, the scientist may easily get lost among which ones have and have not been executed and finding which ones are closer to validate their hypothesis.

Chapter 3 Related Work

Currently, widely used SWfMS such as VisTrails (CALLAHAN et al., 2006), Triana (TAYLOR et al., 2007a), Kepler (ALTINTAS et al., 2004), Taverna (HULL et al., 2006), Pegasus (DEELMAN et al., 2007), and Swift (WILDE et al., 2011) represent workflows only at the concrete level, providing graphical interfaces to help scientists to better visualize their workflows. VisTrails, in particular, has a mechanism to keep the change history of the workflows that can be used to control several versions of workflows. However, the workflows versioned by this mechanism are not classified in different categories, that is, it is hard to know the workflows that are ready to be reused by other scientists (*i.e.*, stable releases) from those that are under development, without quality guaranties (*i.e.*, under development versions) (OGASAWARA et al., 2009b). In VisTrails, a simple parameter modification in the workflow is considered as a new concrete workflow version, regardless of the fact that this new version does not present any difference at the implementation level from the previous one. Furthermore, the way the information is represented does not help scientists to have a global view of the experiment. For instance, it is unfeasible to find all alternative implementations of a specific activity because it is spread in different nodes of the version tree. Finally, VisTrails provides a functionality called parameter exploration that allows scientists to execute a specific workflow version by sweeping a set of parameters (*i.e.*, generating several workflow instances). The results of all these workflow execution instances can be visualized in an integrated way in a grid panel.

The previously mentioned SWfMS use terms and concepts specialized to implementation details. Workflows expressed in that way are hard to be analyzed even by other scientists from the same experiment domain. On the other hand, there are some initiatives in the literature for representing experiments at a higher level of abstraction. They are discussed in the next sections. However, we start this chapter (Section 3.1) by presenting some comparison criteria that we use to analyze related work.

3.1 Comparison Criteria

To guide the discussion about related work, we use a set of criteria. These criteria were defined from a literature review and from an analysis of the main contributions and limitations of the approaches described in this chapter. Furthermore, some criteria were

extracted from the main characteristics of the experiment knowledge management problem that this thesis intends to solve. These criteria are listed below.

- **Abstraction models:** Analyses what abstraction models are used to represent the experiment information. For example, if only abstract workflows are used to represent the experiment knowledge at a high abstraction level or if other ways of representation (*e.g.*, ontology (STAAB; STUDER, 2004), feature model (KANG; LEE; DONOHOE, 2002), etc.) are used to represent the several implementation variations of an experiment.
- **Abstraction level information mapping:** Indicates whether the information defined in different levels of abstraction in the approach are mapped to each other.
- **Variability:** Indicates whether the approach supports the modeling of variabilities in the scientific experiment definition, describing in which abstraction level this is supported (*i.e.*, concrete level, abstract level, and experiment level).
- **Optionality:** Indicates whether the approach supports the modeling of optional characteristics in the scientific experiment definition that can be suppressed or not during the workflow execution, describing in which abstraction level this is supported (*i.e.*, concrete level, abstract level, and experiment level).
- **Conflict and dependency relationship:** Indicates whether the approach supports the modeling of conflict and dependency relationship between characteristics (it does not include activity input and output data, since it is already defined in the workflow specification) in the scientific experiment definition to be used to guide the users during the workflow derivation process. (PS: These characteristics does not include input and output data, since this is already defined in the dataflow/workflow)

3.2 GExpLine

GExpLine (OLIVEIRA et al., 2010b) is the tool that implements the original approach of experiment line of Ogasawara et al. (2009a). This tool allows the creation of experiment lines via a graphical interface that can be further derived into concrete workflows. Therefore, GExpLine works at two abstraction levels: experiment line and concrete workflow. The experiment line is the model that represents in terms of

variability and optionality the program variations that a concrete workflow can use and their associated input and output data schema. Concrete workflows are the workflows derived from the experiment line instantiated with specific programs and following a particular SWfMS specification.

GExpLine is the pioneer prototype for modeling experiment lines, but it presents some limitations. The variations expressed in the experiment line model concerns only about concrete level information. In other words, it only describes the programs variations that each activity can have in the workflow. We cannot abstract the workflow activity description in terms of approaches or algorithms similar to an abstract workflow. Additionally, the experiment line model of GExpLine is limited only to variability and optionality. It does not support the representation of conflict and dependency relationship between the activities of the experiment line. Moreover, there is not a provenance approach for tracking the derived concrete workflows to the experiment line. Therefore, we cannot make provenance analysis that aggregates workflows based on a specific program used or a data product.

3.3 Kepler-Onto

Ludäscher *et al.* (2003) present an approach (which we name as Kepler-Onto) to relieve scientists from directly designing concrete workflows. The scientist has only the burden of constructing abstract workflows based on directed acyclic graphs using concepts and terms from underlying domain ontologies. These abstract workflows are automatically converted to web-service concrete workflows by their approach. This is done by using database mediation techniques that automatically map abstract workflow activities into concrete ones based on the domain ontology definitions. The authors claim that specific database mediation techniques, such as global-as-view style query rewriting, can be used to optimize the workflow execution. However, this has not been defined yet. Currently, it is possible to validate the workflow structure by verifying the connections between the workflow activities and their input and output data specifications.

The mapping proposed by the approach is powerful and independent of SWfMS. However, the abstract workflow model used by the approach does not represent variability or optionality. Consequently, it does not support dependency and conflict relationship either. It is not possible to check whether the workflow activities are optional or may be implemented by more than one approach. Therefore, there is not a

higher abstraction model to represent all the experiment information in a consistent and integrated way.

3.4 WDOs

Salayandia *et al.* (2006) define an approach to represent abstract information for scientific experiments. It was initially originated to solve a problem in the domain of geology by sharing experiment knowledge among scientists. In this approach, the experiment knowledge is represented at different abstraction levels, using one of three models: Workflow-Driven Ontologies (WDOs) (SALAYANDIA et al., 2006), Semantic Abstract Workflows (SAW) (SALAYANDIA; DA SILVA, 2010), and Proof Markup Language (PML) (DA SILVA; MCGUINNESS; FIKES, 2006).

WDOs are used to capture experiment vocabulary of a specific domain in order to be used to construct scientific workflows and encode provenance data. WDOs are encoded in OWL and its information is represented as extension of two classes: Data and Method.

The concepts defined in the ontology of WDO model are used to construct abstract workflows, also encoded in OWL, called Semantic Abstract Workflows (SAW). Therefore, the workflow activities and the data input and output are instances from the data and method classes defined in the WDO. Besides, the data classes used in the SAW can assume specific types (*e.g.*, string, integers, etc.). This guarantees a validation of the structure of workflow analyzing both workflow activities operations and data.

Finally, Proof Markup Language (PML) is a domain-independent language that defines concepts and relations that are useful to encode provenance data. The PML model is related to WDO and SAW models. The level of detail of provenance to be captured in PML is defined in workflows documented in SAWs and the PML concepts are aligned to the OWL concepts.

The work of Salayandia *et al.* (2006) shares our same motivation since it allows scientists to represent experiments in high levels of abstraction and document multiple execution alternatives for a workflow. However, their work only deals with information at high level of abstraction without considering their relationship with executable workflows. Consequently, there is no mapping of this concrete information to abstract information.

3.5 ASKALON

ASKALON (FAHRINGER et al., 2005) provides a graphic modeling tool, called Teuta, which is capable of representing grid workflow activities in UML 2.0 activity diagrams. The motivation of this work is to increase the collaboration among different grid-related projects using a standard well-diffused language. Workflows in ASKALON are expressed in AGWL (Abstract Grid Workflow Language), which is an XML-based language. Teuta uses the extension mechanism provided by the UML specification to specialize the UML diagram activity to represent the AGWL semantics.

The abstract workflows composed in Teuta are automatically converted to concrete workflows to be executed in the ASKALON grid environment. This is possible because the abstract workflows created in ASKALON make references to concepts of the execution grid environment, which are represented at high level of abstraction. For example, abstract workflow activities are described by activities types. An activity type is an abstract description of a group of activity instances deployed in the grid which have the same input and output data structures. From that, ASKALON can validate workflows by checking the activity types and input and output data specification.

The main advantage of this approach is the use of UML diagrams to represent the experiment knowledge instead of using a non-standard specification. Nevertheless, this tool does not support representing variability, optionality, dependency, and conflict properties in a workflow.

3.6 TSL

The task specification language (TSL) (LIN et al., 2009) is an initiative coupled to the VIEW SWfMS (LIN et al., 2008) to represent scientific workflows in two abstraction layers: interface and concrete. The interface layer consists of the workflow definition representation that describes the main functionalities of the experiment using terms familiar to the scientist without considering implementation details. The concrete layer represents the workflow definition considering the implementation details and so that includes underlying implementation functionalities. The motivation of this work for representing scientific workflows in a higher abstraction way is due to the fact that current SWfMS represent workflows with a lot of implementation details that hinder the scientist from understanding the experiment. Most of these implementation details are

related to shim (or adapter) activities that convert data from incompatible activities ports.

This approach evolves the scientific workflow composition process to a higher level of abstraction, enabling scientists to create their workflows without considering implementation details. However, this high abstraction model provided by TSK is not capable of representing, in an integrated way, the different trials of an experiment, i.e., the workflow specification does not support the concepts of variability, optionality, dependency, and conflict.

3.7 VISMASHUP

Santos *et al.* (2009) propose a framework called VISMASHUP that simplifies the creation of customizable visualization applications from previously developed workflows in VisTrails. The purpose of these customizable applications is to turn the workflow configuration process user friendly, even for those users with no knowledge in the domain, preventing them from interacting directly with complex workflows.

The VISMASHUP framework supports various stages in the application development process, including mining and exploration of visualization workflows sets, creation of simplified workflow view, and automatic generation of applications and interfaces. Moreover, VISMASHUP allows the generation of applications by aggregating multiple visualization workflows that were previously developed. The application creation process is defined as follows. The user selects configurable parts in the workflow by creating templates. These templates are used to generate applications. Finally, the user can define suggestions of values for the variable input in the templates for further ease of use.

VISMASHUP is a good approach for modeling variabilities in a workflow. However, it focuses only on the workflow variability at implementation level, *i.e.*, at the concrete workflow level. Some variabilities dealt by the approach are, for instance, parameters for executing an application tool, configuration of resolution and size of an image, configuration of Web service address ports, etc. In other words, it is not possible to model the workflow knowledge at different levels of abstraction without considering computational resources and showing the different variations of execution that an experiment can have. Finally, VISMASHUP does not support representing optionality, conflict, and dependencies properties in the workflow.

3.8 FMCOp

Acher et al. (2010) propose a product line approach to ease the construction of service-based scientific workflows in a computational grid. The motivation emerged from medical domain where authors noted that the services used in scientific workflows are highly configurable making the task tedious and error-prone to the users. Therefore, the approach organizes the services in a product line architecture where feature models (which they name as Feature Model Composition Operators - FMCOp) are used to organize important information in terms of service variability.

When the product line is defined, the user selects the services to build a new scientific workflow. As the services are being selected, the user has to choose the characteristics of each service in its respective model features (*e.g.*, the communication protocol type, data format, quality of service (QoS), etc.). According to the features selected, the system performs an analysis to find conflicts between characteristics of the services. For example, a service *A* with the feature "HTTP communication protocol" cannot be connected to a service *B* that does not have this feature.

The approach, like ours, is able to model variabilities of a scientific experiment. However, this model works only at a low level of abstraction, disregarding the experiment level.

3.9 Chiron

Chiron (OGASAWARA et al., 2013) is a data-centric scientific workflow engine implemented to parallelize scientific workflows that demands parameter sweep techniques in workflow activities. Chiron is based on an algebraic model that represents workflows in a high abstraction level releasing scientists from coding in low level parallelism mechanisms such as parameter sweep techniques on workflow activities. The parallelism is done automatically due to the way the workflow is represented, where workflow data are expressed as relations (similar to relational model) and workflow activities are ruled by algebraic operators. Moreover, this algebraic model also allows the execution engine of Chiron to optimize the workflow execution using different parallelism and distribution strategies. Other underlying mechanisms such as workflow validation are available thanks to the algebraic model characteristics. It is possible to validate the workflow composition structure based on the operation

definitions and the input and output data defined for the workflow activities and their connections. Finally, Chiron can parallelize the execution of scientific workflows dispatched from a local machine into a cloud environment, such as Amazon EC2 (AMAZON EC2, 2014) or GoGrid (GOGRID, 2012), when used in combination with the SciCumulus system (OLIVEIRA et al., 2010a).

Even though Chiron uses an abstract model to represent scientific workflows, it is not abstract enough to help scientists with low computer skills to create their own workflows. In addition to that, there is no automatic mechanism to help scientists to derivate their workflows. Scientists have to compose workflows using algebraic operations that are not intuitive and workflow alternatives and optionalities cannot be represented. Additionally, Chiron does not support representing dependency and conflict relationship in the workflow. Finally, the workflow has to be textually constructed using the XPDL language (GUELFI; MAMMAR, 2006), which is based on XML (BRAY et al., 2008).

3.10IWIR/SHIWA

Plankensteiner *et al.* (2011) define a common workflow language named IWIR (Interoperable Workflow Intermediate Representation) for use as an intermediate exchange representation for multiple workflow systems. This work has the main objective to solve the interoperability issue that users from different workflow systems face to share the workflow knowledge.

The first step took by the authors to create an intermediate workflow language was to separate the abstract from the concrete information in the workflow specification. IWIR language deals only with the abstract part of the workflow specification (*i.e.*, the workflow orchestration, the workflow activities relations, and dataflow). The concrete part (*i.e.*, the information of how the workflow activities are executed, specifying the computational resources, etc.) is handled by a mechanism that maps the abstract information in the IWIR to the workflow system. The second step was to establish that each workflow system needs to adjust its front-end to translate its source input language into the IWIR language. Therefore, a workflow specification generated in a workflow system can be easily exported to another IWIR compliant workflow system.

IWIR is a good initiative to solve the interoperability problem between workflow systems. The definition of a common language to be used as a communication protocol

between workflow systems is essential. However, the study conducted to define the IWIR language was restricted because it considered a limited number of workflow languages. It considered just workflow languages of workflow systems inserted in the EU FP7 SHIWA project (SHIWA, 2011) (ASKALON (FAHRINGER et al., 2005), Moteur (GLATARD et al., 2007), P-GRADE (KERTÉSZ; SIPOS; KACSUK, 2007), Triana (TAYLOR et al., 2007a), and Pegasus (GIL et al., 2007)). Consequently, characteristics of other workflow systems may not be comprised in the IWIR language. For instance, IWIR language does not support the representation of workflow change history available in some SWfMS workflow languages (*e.g.*, VisTrails). This can contribute negatively to its full adoption in the community.

Like our approach, IWIR is also focused on having a high abstract level to be mapped to a lower level. However, the IWIR language lacks the abstractions of variabilities, optionalities, conflict and dependency in a workflow. In the current version of the language, it is necessary to model several workflows individually to infer that each instance is a variation of "one" workflow.

3.11 Research object model

Research object model (BELHAJJAME et al., 2014) is a set of ontologies to represent the experiment information. It formalizes not just workflows and associated provenance but other important concepts related to the experiment as the hypothesis, conclusions, and so forth. One of the research object ontologies is called workflow motif (GARIJO et al., 2014), which is used to abstract the workflow representation. Motifs are annotations in the workflows to ease their comprehension when shared to other scientists. Motifs present powerful abstractions to explicit and formalize the workflow control-flow structure and the way data is consumed and produced in the workflow.

Research objects can be modeled and managed through a set of tools and libraries, which are called Research Object Digital Library (RODL). Additionally, the virtual research environment myExperiment (DE ROURE; GOBLE, 2007) was extended to support the creation of research objects in its process. myExperiment enables collaboration by sharing and publishing workflows (and experiments) that do not need to be specified in a particular workflow management system.

Research object is a good initiative to represent in a uniform way the entire experiment information including hypothesis and conclusion. However, the work has some

restrictions if we individually analyze each component of the research object model. For instance, the queries supported by motifs at abstract level in the workflow composition are separated from the workflow execution data. It is not possible to make provenance queries joining information from different abstraction levels. Finally, Motifs and all the other research object models do not allow representing workflow variabilities, optionalities, conflicts and dependencies.

3.12 SADI

SADI (WILKINSON; VANDERVALK; MCCARTHY, 2011) is a framework that defines Semantic Web Services design patterns to simplify the publication and consumption of bioinformatics web services. The main idea of SADI is to help scientists with recommendations and tools in the creation of web services with well-documented ontologies in order to automate the discovering and construction of complex workflows.

In SADI, ontology descriptions are used to document and add semantics in the services input and output data. Using a unique ontology description or mapping several ontologies from a same domain, it is possible to chain web services by inference to build or reproduce a specific data product. Moreover, ontologies are also used to map transformations of input to output data, describing at abstract level the "meaning" of the relationships.

The Semantic Health and Research Environment (SHARE) (VANDERVALK; MCCARTHY; WILKINSON, 2009) is one of the SADI tools specific to help scientists to reproduce experiments. It differs from the traditional analysis techniques where scientists query provenance data to understand how the data were produced. In SHARE, scientists request the reproduction of a specific data and, based on the ontology description, a workflow is composed to be executed on the fly.

Like our approach, the main idea of SADI is to help scientists with recommendations and tools to automate the discovering and construction of complex workflows. However, SADI is restricted to the bioinformatics domain and the technology of Web Services.

3.13Wings

Wings (GIL et al., 2007) uses semantic workflows to enrich the traditional workflow representations by adding metadata and constraints to each element in the workflow. These metadata and constraints are organized in ontologies to formalize their concepts and relationships. From that, Wings uses AI planning and semantic reasoners to process all these data, which are stored in catalogs, to guide scientists during the workflow composition (GIL et al., 2011). These catalogs are classified by domains and can be selected individually according to the experiment characteristics.

The workflow information is organized hierarchically in classes similar to the object-oriented paradigm. Workflow activities and data are created as extensions of preexistent elements and so they inherit the characteristics of their ancestors. These characteristics include constraints, and, especially for workflow activities, their signatures (i.e., the input and output interface). The facts and rules registered in these catalogs are the basis for Wings to validate the workflows and to suggest workflow candidates to scientists during the template composition phase. Also, additional rules (in RDF) can be defined by the scientist when necessary.

In a later work (GIL, 2013), Gil describes an approach for Wings to derive abstract workflows for different SWfMS. Wings is a powerful tool by exploring ontologies and being independent from a SWfMS language. This approach is the closest approach to our work. It can infer variabilities in the experiment definition by using ontologies, but it does not support the concept of optionality. Therefore, scientists cannot explicitly declare, in the abstract workflow specification, which activities can be included or not in the concrete workflow. Wings allows the definition of conflict relationships, but this is only used to validate the workflow structure. As a result, other relationships such as dependency and conflict cannot be effectively used in the workflow specification to guide the scientist in the workflow composition. For instance, there is not a mechanism that indicates that if one activity is inserted in the workflow, another activity should be included or removed, as we do in our approach AEL. Furthermore, Wings does not provide a way to integrate the abstract workflow with the provenance data generated from the execution of the derived workflow. The provenance analysis has to be done using separate tools, one for querying the abstract definition and another for the

corresponding execution of the derived workflow. As a result, the integration must be done in an *ad-hoc* way.

3.14 Related Work Comparison and Final Remarks

Table 1 classifies the approaches discussed in this chapter according to the criteria described in Section 3.1. We can see that even the approaches that share our motivation of defining a model at the experiment level present some limitations (e.g., absence of abstraction level mapping support, focus only in the concrete level, lacking of models to represent the experiment characteristics, etc.). Due to these limitations, scientists have a narrow view of the experiment conception and management. Even when the experiment is documented using higher abstract level models it is not related to the concrete level. This results in weak provenance analyses and massive use of tacit information that will be lost and unknown for other scientists that want to reproduce the experiment.

Table 1. Related Work Comparison

	Experiment Level	Abstraction Levels	Variability	Optionality	Conflict and Dependency Relationship	Abstract to Concrete Level Mapping
ASKALON	Yes	UML activity diagram; Abstract Grid Workflow Language;	No	No	No	Yes
Chiron	No	Abstract level; Concrete level	No	No	No	No
FMCOp	Yes	Software product line; Concrete workflows	Yes (Concrete level)	No	Only conflict	Yes
GExpLine	Yes	Experiment Line; Concrete Workflow	Yes	Yes	No	No
IWIR/SHIWA	Yes	Abstract level; Concrete level	No	No	No	No
Kepler	No	Concrete level	No	No	No	No
Kepler-Onto	No	Abstract level; Concrete level	No	No	No	Yes
Pegasus	No	Concrete level	No	No	No	No
Research Object Model	Yes	Ontology; Concrete	No	No	No	Yes

		workflow				
SADI	Yes	Ontology; Concrete workflow based on web services	Yes	No	Only conflict	Yes
Swift	No	Concrete level	No	No	No	No
Taverna	No	Concrete level	No	No	No	No
Triana	No	Concrete level	No	No	No	No
TSL	Yes	Interface layer; Concrete layer	No	No	No	No
VisMashup	No	visualization apps; Concrete level	Yes (Concrete level)	No	No	No
VisTrails	No	Concrete level	No	No	No	No
WDOs	Yes	Workflow-Driven Ontologies; Semantic Abstract Workflows; Proof Markup Language	Yes	Yes	Only conflict	No
Wings	Yes	Ontology; Concrete workflow	Yes	No	Only conflict	Yes

Representing workflows at the abstract level individually is not enough to represent the experiment, considering that during the experiment life cycle a lot of workflow variations are defined to run in different trials. These explorations must be represented to register the experimentation phases. Hence, it is necessary to integrate the information of all these workflows trials in a unique model of higher abstraction level with the aim of helping scientists in managing all the phases (composition, analysis, execution) of the experiment life cycle.

In Chapter 4 we discuss how the AEL approach proposed by this thesis mitigates these issues.

Chapter 4 Algebraic Experiment Line

Over the last years, the software engineering community has intensively worked on managing reusable artifacts in the software development process (POHL; BÖCKLE; LINDEN, 2005). The main idea behind managing these artifacts is to allow software developers to generate new software with minimal development effort while reducing financial costs, development time, and improving customer satisfaction. Following this idea, software product lines (SPL for short) (NORTHROP, 2002) were created for supporting this development task.

SPL can be defined as a set of software engineering methods, tools, and techniques for creating a collection of similar software systems from a shared set of software assets using common means of production (NORTHROP, 2002). In this work, we refer to these software assets as *components*. In SPL, a set of software that share common characteristics are grouped into a *family*, *i.e.*, they can be developed based on a set of assets. These assets include architecture, reusable software components, domain models, requirement statements, etc. The task of generating a software system based on a SPL is called *derivation*.

SPL are based on two fundamental concepts: *variability* and *optionality*. When there is more than one alternative component that implements a specific characteristic in the SPL, we say there is a variability. In other words, variability allows for software developers to change and configure components according to a given context. The second concept, optionality, allows a specific component of the SPL to be suppressed when a software system is derived. In order to manage this, components in a SPL can be marked as mandatory or optional. This way, when developers are generating new software based on a SPL, they basically have to determine which artifacts implement a specific component and which components are going to be suppressed in the final software system.

Using the concepts of the software product line approach in the context of scientific experiments, Ogasawara *et al.* (2009a) proposed the experiment line approach to model scientific workflows at the experiment level. Experiment line represents not only one, but also all possible ways for representing a scientific experiment (variabilities and optionalities intrinsic to a specific experiment), *i.e.,* the workflow derivation space. The

experiment line model represents all possible alternatives to model a workflow using an abstract and uniform notation. This notation can be considered as an abstract workflow since it is not bound to infrastructure issues.

In this work we reformulate the original proposal of experiment line from (OGASAWARA et al., 2009a) and present a new approach, called Algebraic Experiment Line. This approach enhances the workflow derivation process by means of using more semantics in the experiment line definition. We incorporate the SciWfA workflow algebra (OGASAWARA et al., 2011) in the experiment line approach. This algebra adds data consumption and production semantics to workflow activities in order to standardize the data model in all abstraction levels. This standardization guarantees the creation of queries that join workflow execution data with their corresponding abstract experiment definition. Additionally, derived workflows can use the algebraic transformations and workflow fragments to do automatic optimizations in the execution plan. However, since the SciWfA algebra (OGASAWARA et al., 2011) is not aware of the experiment line abstractions, we formally define algebraic operations and operands to obtain AEL, an enhanced experiment line approach built on top of the SciWfA algebra. This further improves the potential of runtime optimizations by the workflow engine.

For simplicity, we use the term experiment line instead of algebraic experiment line to present the main concepts of the approach along the text. Therefore, from now on, anytime the term experiment line is mentioned in the text please refer to the AEL approach.

The remaining of this Chapter is organized as follows. Section 4.1 presents an overview of the proposed approach. Section 4.2 formalizes the experiment line approach, defining its main concepts. Finally, Section 4.3 presents the proposed architecture for AEL that supports the composition of experiment lines and the derivation of workflows in different execution platforms. Also, this architecture comprises the provenance strategy for gathering and mapping the provenance data, which are distributed at different abstraction levels.

4.1 Overview

An experiment line is one of the ways of implementing the experiment abstraction level. The relation of an experiment line to the traditional abstract and concrete workflow

abstraction levels defined in the literature can be described as follows. The experiment line (*i.e.*, the higher abstract level) represents all *n* possible ways to conduct and represent a scientific experiment, according to the hypothesis of the scientist. Based on experiment lines, workflows can be derived for the experiment through the concepts of variability, optionality, derivation, and instantiation. Variability means that for a specific activity in an experiment line there is more than one abstract alternative (*e.g.*, there is more than one approach that could be applied to perform the task denoted by that activity). The concept of optionality, on the other hand, defines that a specific activity of the experiment line can be suppressed when a workflow is derived. In an experiment line, each activity is associated to an operator in the workflow algebra, and to input and output schemas. Although experiment lines are agnostic of implementation, connecting experiment line activities with algebraic operators exposes the semantics (represented by such operators) behind each activity at this abstraction level.

The experiment line model is similar to a workflow model, which defines the main flow of activities, with an inclusion of a set of annotations that define the multiple variations of the workflow activities. The workflow activities are associated to their corresponding algebraic operators[3] (see Section 2.1.2), leading to the workflow representation by means of algebraic expressions. Using the experiment line model, scientists are able to derive workflows that represent a specific cutting of an experiment line. In this cutting, scientists choose specific approaches from (possibly) several that are associated to the activities. These derived workflows represent the second abstraction level, which is called *abstract workflow*.

Finally, the concrete workflow (*i.e.*, the concrete level) takes these algebraic expressions and implements them in a specific SWfMS with a specific workflow language, *i.e.*, it instantiates the abstract workflow activities to computational resources. This process is called instantiation. Summarizing, we have two main processes for transforming an experiment line into a final concrete workflow that is going to be executed in a specific SWfMS: **derivation** and **instantiation**. The derivation process takes as input an experiment line, and produces an abstract workflow. In other words, in the derivation process, there is a cutting of the execution approach space of the

[3] Each algebraic operator is associated to the activity in the experiment line (and in the abstract workflow as well) according to how it consumes and produces tuples.

experiment line workflow, generating a valid and unique execution approach, *i.e.*, an abstract workflow (containing algebraic expressions). The instantiation process takes as input an abstract workflow and produces a concrete one. More specifically, abstract workflow activities are instantiated in terms of computational resources, generating a concrete workflow.

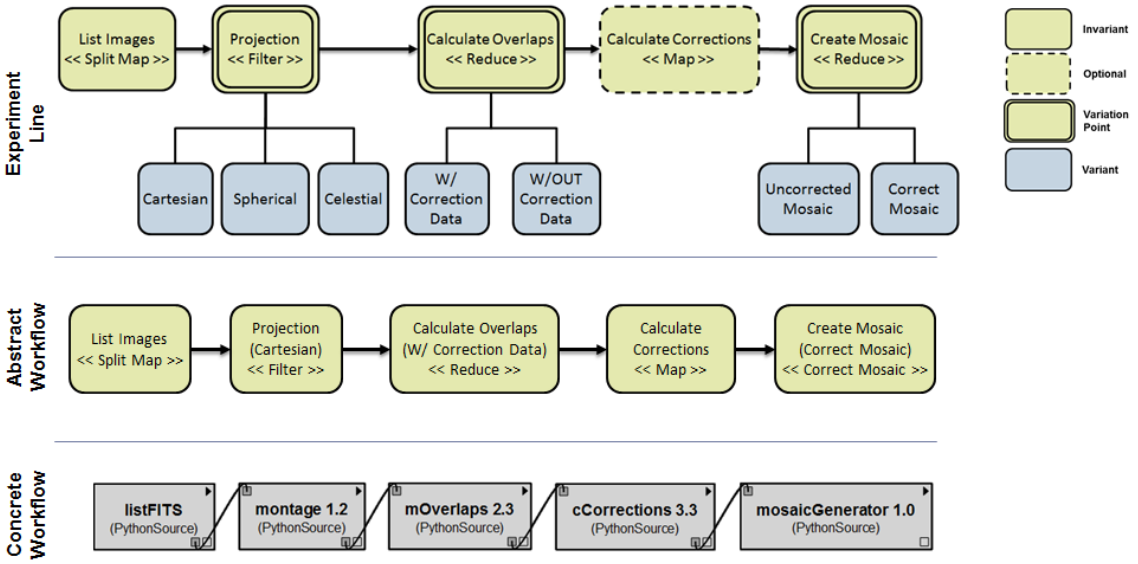

Figure 5. An example of experiment line in the context of an astronomy data analysis experiment

Figure 5 presents an example of an experiment line in the context of an astronomy data analysis experiment (for details, see Chapter 2), with abstract and concrete workflows derived from it. The notation used to represent the experiment line is the one we implemented in the prototype tool (for details, see Chapter 5). Each workflow activity has an annotation that describes its associated algebraic operator, as described in Section 2.1.2. In the experiment line, the activities List Images, Projection, Calculate Overlaps, and Create Mosaic are mandatory (*i.e.*, they must be present in any derived abstract workflow). Mandatory activities are represented by solid line rectangles. On the other hand, the activity Calculate Corrections is optional and can be present or suppressed in the derived workflow, depending on what kind of algorithm was selected in the workflow activity Calculate Overlaps, as described in Chapter 2. Optional activities are represented by dashed line rectangles. The experiment line variabilities are represented by variant activities and they are grouped by variation point activities. For instance, Cartesian, Spherical and Celestial are variants of the variation point activity Projection. Variation points are represented by double-line rectangles, while the variants are represented by blue rectangles. Activities that have no variants are called invariants (*e.g.*, List Images, Calculate Corrections.).

4.2 Experiment Line Formalism

With the intent of clarifying and validating the experiment line approach, this section presents a formalization of the approach. This formalization is organized in five sections described as follows. Section 4.2.1 defines the basic concepts related to the experiment line. Section 4.2.2 introduces the functions used in the workflow derivation process. Section 4.2.3 describes the workflow activity types present in the experiment line. Section 4.2.4 presents the composition rules. Finally, Section 4.2.5 discusses how to validate workflows derived from an experiment line.

4.2.1 Basic Concepts

Definition 8. (EXPERIMENT LINE) An experiment line *el* is represented by a triple <*AEL*, *DepEL*, *Rl*>, where AEL corresponds to the set of activities in an experiment line, denoted by $AEL = \{ael_1, ..., ael_n\}$; *DepEL* represents the set of edges of the experiment line, denoted by $DepEL = \{depel_1, ..., depel_m\}$, and *Rl* is the set of rules that constrains the possible workflow derivation, denoted by $Rl = \{rl_1, ..., rl_o\}$.

It is worth mentioning that each ael_i inherits the workflow activity definition from Definition 1 (Section 2.1.1) and is associated to one of the algebraic operators defined in Section 2.1.2.

Optionality and Variability are the basic concepts to represent variation in the experiment line. Optionality is a concept that defines if an experiment line activity can be suppressed or not when a workflow is derived. Variability is a concept that defines if an experiment line activity can be represented by more than one abstract alternative (e.g., more than one approach that could be applied to perform the task denoted by that activity). Their definitions are described as follows.

Definition 9. (OPTIONALITY) An experiment line activity *ael* can be classified as mandatory, when its presence is required in all derived workflows, or optional, when the workflow can be derived without its presence. The set of all mandatory activities is denoted by $MA \mid MA \subseteq AEL$, and the set of all optional activities is denoted by $OA \mid OA \subseteq AEL$. Thus, $AEL = MA \cup OA$ and $MA \cap OA = \emptyset$.

Definition 10. (VARIABILITY) An experiment line activity *ael* can be classified as a variation point iff there is more than one abstract activity associated with it. Otherwise

ael is classified as invariant (when it can be represented by a singular abstract activity). The set of all variation point activities of an experiment line *el* is denoted by $VPA \mid VPA \subset AEL$, and the set of all invariant activities is denoted by $IA \mid IA \subseteq AEL$. Additionally, each activity $vpa_i \mid vpa_i \in VPA$ has a set of approach variations (variants) $VA \mid VA \subset vpa_i$. Thus, $AEL = VPA \cup IA$ and $VPA \cap IA = \emptyset$.

Definition 11. (ABSTRACT WORKFLOW) An abstract workflow *aw* is a valid workflow (see Definition 7 from Section 2.2) represented by the tuple $<AA, ADep>$, where *AA* is the set of abstract workflow activities, denoted by $AA = \{aa_1, ..., aa_n\}$; and *ADep* corresponds to the set of edges of the abstract workflow, denoted by $ADep = \{adep_1, ..., adep_m\}$.

It is worth mentioning that $AA \subseteq AEL$, where *AEL* is the set of activities of an experiment line *el*. Each abstract workflow aw_i derived from an experiment line *el* is unique and their activities (*i.e.*, *AA*) are associated to specific operators in the workflow algebra. The function $Oper(aa_j)$ returns the operator associated to a specific abstract activity (*i.e.*, its semantics). A derivation of an experiment line *el* is performed by cutting its derivation space, forming a derived abstract workflow. This derivation is based on a derivation function (presented in Definition 12).

4.2.2 Derivation and Instantiation Functions

Definition 12. (DERIVATION FUNCTION) The derivation of an abstract workflow *aw* from an experiment line *el* is represented by a function called *DerivationFunction(el, dp)*, such that $DerivationFunction(el, dp): el \rightarrow aw$, where *el* is the experiment line, *aw* is the target abstract workflow, and *dp* is the derivation plan that describes which elements are selected in the experiment line.

Definition 13. (INSTANTIATION FUNCTION) The instantiation of a concrete workflow *cw* from an abstract workflow *aw* is represented by a function called *InstantiationFunction(aw, swfms)*, such that $InstantiationFunction(aw, swfms): aw \rightarrow cw$, where *aw* is the abstract workflow, *cw* is the target concrete workflow, and *swfms* is the target scientific workflow management system where the concrete workflow will be instantiated.

Definition 14. (VALID WORKFLOW DERIVATIONS) All valid derivations of an experiment line *el* (*i.e.*, the derived abstract workflows that respect the rules described in Definition 22 – presented next) are represented by the set $AW = \{aw_1, \ldots, aw_n\}$.

For simplicity, from now on when workflow activities are mentioned in this thesis with no abstraction level modifier please refer to activities of an experiment line. At the abstract and concrete levels, activities are mentioned with their respective adjectives, *i.e.*, abstract activities and concrete activities. Figure 5 illustrates an abstract workflow derived from the experiment line. In the abstract workflow, the Cartesian approach was selected for the variation point activity Projection and the optional activity Calculate Corrections was included.

Once the relationships between experiment line and abstract workflow were explained by introducing the concepts of variability and optionality, now we discuss with more detail the different activity types originated from these two concepts.

4.2.3 Workflow Activity Types

Definition 15. (MANDATORY ACTIVITY) Given the set $AW = \{aw_1, \ldots, aw_n\}$, which represents all the valid abstract workflow derivations from an experiment line *el*, and the set $MA = \{ma_1, \ldots, ma_m\}$, which represents all mandatory activities of an experiment line *el*, we have $\forall aw_i \in AW, MA \subset aw_i.AA$.

Definition 16. (OPTIONAL ACTIVITY) An optional workflow activity may or may not be present in any derived abstract and concrete workflows. A minimal condition for an activity to be optional is to have the schema preservation characteristic, i.e., the workflow activity input and output schemas (see Section 2.1.1) must be the same.

Definition 17. (INVARIANT ACTIVITY) An invariant activity *ia* is an activity that presents only one implementation solution.

Invariant activity can be considered a particular case of a variation point that has only one variant activity. Hence, when an abstract workflow is derived it can be automatically selected if it is also classified as a mandatory activity.

Definition 18. (VARIATION POINT ACTIVITY) A variation point activity *vpa* is an activity that is associated to a set of variant activities $VA \mid VA \subset vpa$. A *vpa* should be

represented by one and only one variant $va_i \mid va_i \in VA$ in a derived abstract workflow *aw*. In other words, considering *AA* as the set of activities of the abstract workflow *aw*, we have $\exists! \; va_i \in VA \mid va_i \subset AA$.

It is worth mentioning that *vpa* is a unique and exclusive element of the experiment line *el*. The abstract activities AA of a derived abstract workflow *aw* contain only the variant activity *va* selected for a specific variation point activity *vpa*. Therefore, the abstract workflows *AW* derived from AEL must comply with the following constraint: $\forall \, aw_i \in AW, \; \forall \, vpa_j \in VPA \mid vpa_j \not\subset aw_i$.

Definition 19. (VARIANT ACTIVITY) A variant activity in an experiment line *el* represents one of the different approaches of a variation point activity. A minimal set of conditions for a workflow activity to be a variant of a variation point activity is:

- the input relation set of the variation point must have the same schema of the input relation set of the variant activity;
- the output relation of the variation point must have the same schema of the output relation of the variant activity.

It is worth to mention that the dependencies *dep* of the variation point activity to the other activities defined in the experiment line *el* ($dep \subseteq el.DepEL$) is the same, no matter which variant activity *va* is selected, since the input and output schemas are always preserved.

Additionally, optionality and variability are orthogonal concepts. Therefore, experiment line activities are classified based on these two concepts. For example, if an experiment line activity is categorized as invariant (*i.e.*, variability characteristic) it can also be categorized as optional or mandatory using the variability classification. In the case of variant activities, they are not classified based on optionality characteristics. This is due to the fact that they do not belong to the experiment line workflow definition but exclusively to the variation point definition.

4.2.4 Composition Rules

With the help of variability and optionality concepts (represented by the different activity types), it is possible to model a large set of experiment lines. However, some characteristics such as conflict and dependency relationships between activities in the

experiment line cannot be expressed using only these activity types. Therefore, in the experiment line we have rules that allow scientists to model more sophisticated and customized relationships.

Definition 20. (RULE) An experiment line *el* has a set of composition rules $Rl = \{rl_1, ..., rl_n\}$. A rule rl_i is composed of a logical implication of the type $\alpha \rightarrow \beta$, where α is the premise and represents absence and presence conditions of activities in the derived abstract workflow *aw*, and the consequent β (or conclusion) represents insertions or removals of workflow activities in *aw* (*e.g.*, $oa_i \wedge va_j \rightarrow oa_k$).

A rule rl_i is valid when it restricts the experiment line selection (cutting) to a valid workflow composition, *i.e.*, it does not violate any other rule rl_j or any experiment line basic definition (*e.g.*, invariant, variant, variation point, mandatory, and optional rules). Additionally, the workflow activities used in an implication must be optional or variants in an experiment line *el* (*e.g.*, a rule rl_i is not allowed to remove a mandatory workflow activity in the abstract workflow).

Definition 21. (VALID RULE) A rule *rl* is considered valid iff $ValidRule(el, rl) \leftrightarrow (Optional(el, a_i) \vee Var(el, a_i)), \forall a_i \mid a_i \in RuleActivities(rl)$. The function *RuleActivities(rl)* returns the workflow activities used in the rule.

In the astronomy experiment line of Figure 5, in order to create corrected mosaics (i.e. select the variant activity Corrected Mosaic for the variation point activity Create Mosaic) it is necessary to calculate the image corrections (i.e., include the optional activity Calculate Corrections). The inclusion of Calculate Corrections also implies that during the image overlap calculus the correction data should be processed (i.e., select the variant activity W/ Correction Data for the variation point activity Calculate Overlaps). Therefore, we have two rules: Rl_1 (Corrected Mosaic → Calculate Corrections) and Rl_2 (Calculate Corrections → W/ Correction Data). The derived abstract workflow should meet these rules to be valid.

4.2.5 Validation of Derived Workflows

The SciWfA workflow algebra is used to verify how a specific activity can consume and produce data. As described in Section 2.1.1, the input and output relations of the workflow activities represented in the SciWfA workflow algebra follow their respective

schemas. This way, by using the workflow algebra we can support derivation by generating concrete workflows whose programs are compatible.

Besides verifying compatibility among programs, we have to validate derived workflows. The process of validating a derived workflow has to take into account all composition rules defined by scientists (or workflow modelers). For example, it must verify the inclusion of mandatory activities, the choice of a single variant for variation point activities, and so forth. Moreover, the additional composition rules (*i.e.*, the set *Rl*) associated to the experiment line should be analyzed to check if they are met in the derived abstract workflow.

Definition 22. (VALIDATION OF WORKFLOW DERIVATION) An abstract workflow *aw* derived from an experiment line *el* is valid iff:

- *aw* is a valid workflow, according to Definition 7 (Section 2.2);
- all the mandatory activities defined in *el* is present in *aw*, according to Definition 15;
- for each selected variation point activity *vpa*, a unique variant activity *va* is selected, according to Definition 18;
- *aw* respects all the dependencies defined in *el*;
- *aw* respects all the rules defined in *el*.

4.3 Architecture

The AEL approach was conceived to be independent of SWfMS as much as possible. Therefore, experiment lines are composed and abstract workflows are derived using a generic language. The dependency to SWfMS only happens in the instantiation process when concrete workflows are created for a specific workflow language. However, in order to address these characteristics, the architecture of AEL approach is extensible and modular, as shown in Figure 6. The core architecture is composed of three components: Interface Module, Derivation Module, and Instantiation Module. From these three components, the experiment line is modeled (Interface Module), abstract workflows are derived from the experiment line (Derivation Module), and concrete workflows are instantiated in specific SWfMS.

The Derivation module uses a knowledge base to store the experiment line general rules and concepts to be further processed during the workflow derivation. The knowledge base implements the Definitions presented in Section 4.2. The purpose of using the knowledge base by the Derivation Module is twofold. The first one is to validate the workflows derived from the experiment line. The second one is to guide the scientist on selecting the workflows activities that will be used in the derived workflow. The Derivation module can calculate the implications (i.e., dependencies or conflicts) on selecting or unselecting one element in the experiment line.

The Instantiation module is based on the concept of cartridges (GAMMA et al., 1994), which is extensible to support different instantiation implementations to cover any SWfMS. The instantiation module provides a generic instantiator, which defines the interface that the specific modules must follow. Therefore, when scientists want to instantiate the abstract workflow generated from the experiment line into a concrete workflow, they have to choose one cartridge in a set of available cartridges. Each cartridge generates concrete workflows for a specific representation language (*e.g.*, XML, SCUFL, SwiftScript) of a specific workflow engine (*e.g.*, SciCumulus, VisTrails and Taverna).

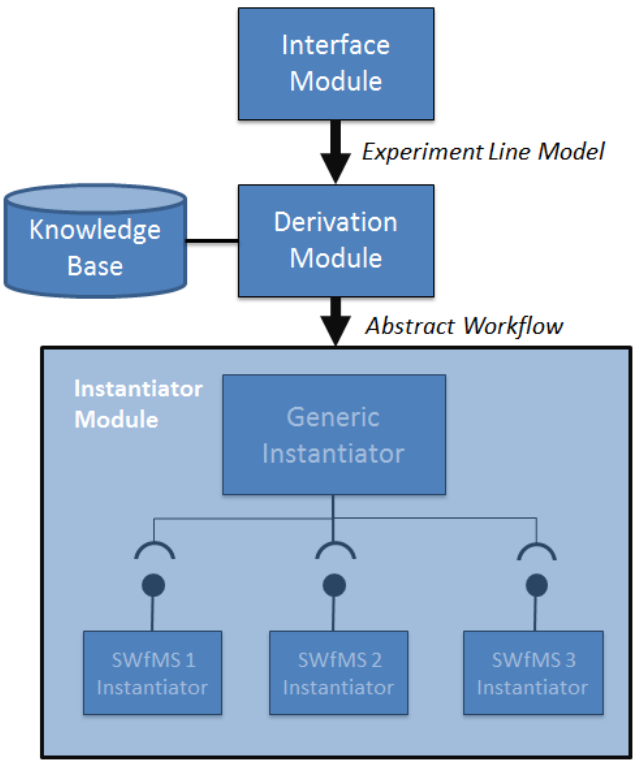

Figure 6. The core architecture of AEL approach

An important missing part on Figure 6 (that will be connected later on this chapter) regards provenance. In the AEL approach, there are three abstraction levels (experiment line, abstract workflow and concrete workflow) that correlate each other to represent the experiment information. Provenance is an important feature for mapping these abstraction levels in order to help scientists on tracking the abstract model transformations, from the experiment line until the concrete workflow. Additionally, another challenge is to collect the data from derived workflows that can be executed in different workflow systems. These problems can be dealt by the provenance management system we previously proposed named ProvManager (MARINHO et al., 2012). This system manages, in an integrated way, provenance data that are distributed in several workflow systems but are part of the same experiment context.

The remaining of this section is organized as follow. Section 4.3.1 presents the provenance management system ProvManager. Section 4.3.2 discusses how to use ProvManager in the AEL approach to benefit from its functionalities. Section 4.3.3 describes the provenance model proposed to the AEL approach.

4.3.1 ProvManager

ProvManager (MARINHO et al., 2012) was originated to deal with experiments that during their life cycles are executed by several workflow systems or, in a worst scenario, use a combined solution of different workflow systems where each one executes a segment of the experiment. ProvManager abstracts the technological complexity of different and heterogeneous workflow systems and integrates the experiment provenance data enabling scientists to visualize them in a unique and standard Web interface.

ProvManager consolidates all the provenance data from the experiment on its own repository. The provenance data is collected in two different moments: the prospective provenance is collected from the workflow specifications used to define the experiment; while the retrospective provenance is collected during the workflow execution via instrumentation (MARINHO et al., 2012) in the workflow specification.

The ProvManager operation can be divided into three main phases: (a) provenance mechanism configuration, (b) provenance gathering, and (c) provenance analysis. Figure 7 illustrates the operation phases.

At the configuration phase, when the workflows are composed in the SWfMS and ready to be executed, the scientist should first provide these workflow specifications to ProvManager (step 1 of Figure 7). These workflow specifications are automatically interpreted by ProvManager to extract the prospective provenance. Moreover, ProvManager also adapts these workflow specifications, including special activities that are responsible for gathering retrospective provenance during execution. This process of adding provenance components into the workflow is called instrumentation. These instrumented workflow specifications are then returned to the scientist (step 2 of Figure 7). They can now be put into execution in the SWfMS (step 3 of Figure 7). During the execution of workflows, retrospective provenance data are gathered by the before mentioned special activities instrumented in the workflows and published in the ProvManager repository via a Web Services API (Provenance Publishing API) (step 4 of Figure 7). At this moment, ProvManager binds the previously collected prospective provenance data with the just collected retrospective provenance data. Finally, at the provenance analysis phase, scientists may use ProvManager to query the experiment provenance data in an integrated fashion (step 5 of Figure 7).

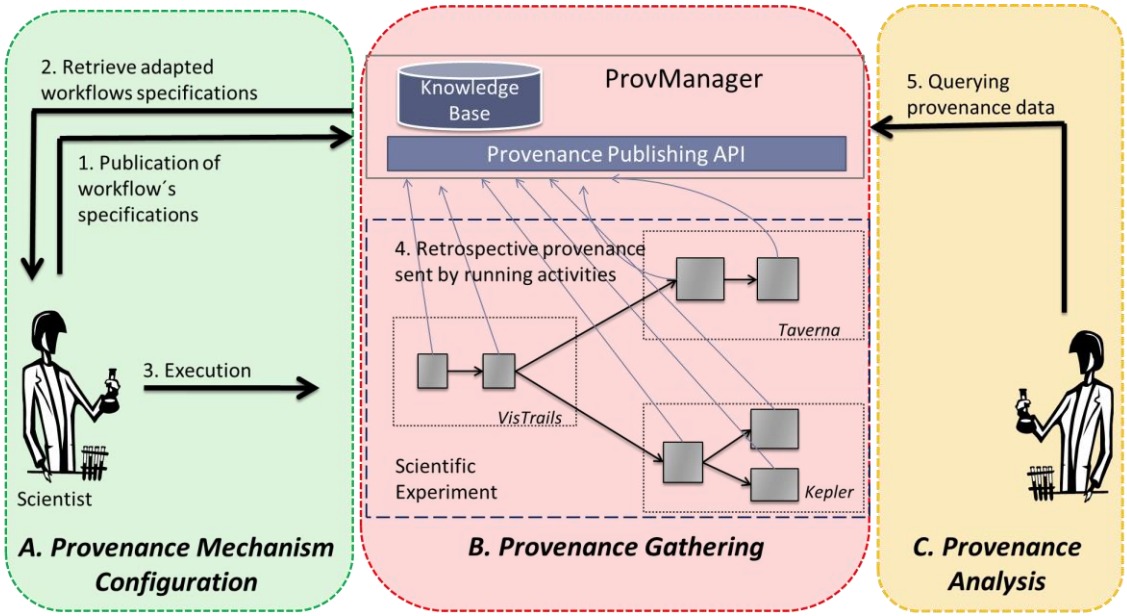

Figure 7. ProvManager in operation

4.3.2 Using ProvManager in the AEL Approach

ProvManager suits the characteristics of the AEL approach since it is not possible to know in advance which workflow system will be selected by the scientist to execute the derived workflows. Additionally, the scientist can interchange to different workflow

systems during the experiment life cycle. ProvManager enables scientists to freely do that without concerning about losing provenance data.

ProvManager is incorporated in the AEL approach to collect and manage prospective and retrospective provenance data from the experiment line using abstract/concrete information. Prospective provenance in AEL consists of information about the experiment line, abstract and concrete workflows, and workflow components such as activities, relations, and schemas. Retrospective provenance consists of information associated to the execution of the experiment (*i.e.*, derived workflow execution).

ProvManager can act in three important moments, as show in Figure 8. In the composition phase (a), when the experiment line is modeled, ProvManager collects prospective provenance information. The second moment (b) is in the workflow derivation and instantiation. At this moment, ProvManager configures the derived workflows via instrumentation to collect retrospective provenance information outside of the AEL domain when they will be executed in the workflow system. The third moment (c) is the publication of retrospective provenance information to ProvManager from the instrumented workflow activities that are executed in the workflow system.

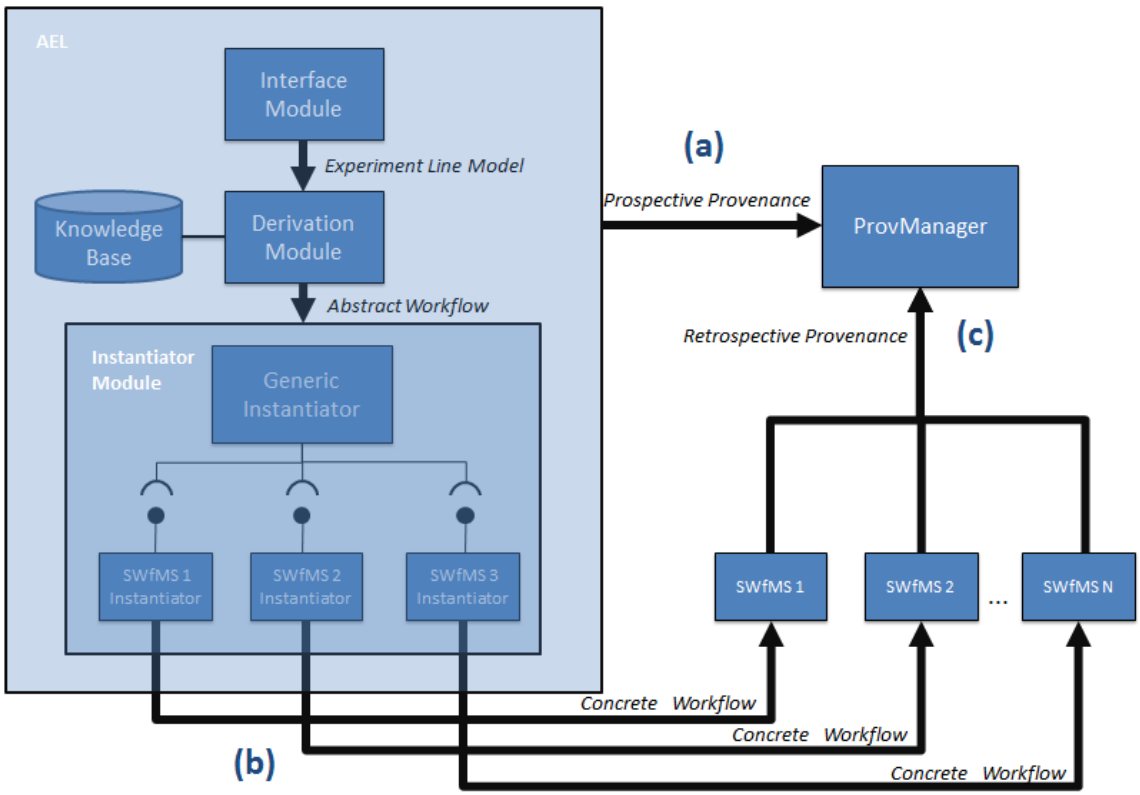

Figure 8. Integration of ProvManager to the AEL approach

4.3.3 Provenance Model

Figure 9 presents the provenance model proposed for the AEL approach, which is used in ProvManager to support the experiment line concepts. This provenance model is designed to represent provenance data from experiment lines that can derive concrete workflows for different SWfMS. The proposed provenance model is an extension of PROV-Wf (COSTA et al., 2013), which is based on the PROV W3C recommendation (MOREAU; MISSIER; BELHAJJAME, 2013), to represent the experiment line abstractions. PROV allows for representing entities, people, and processes involved in the generation of a piece of data so that further extensions can be defined.

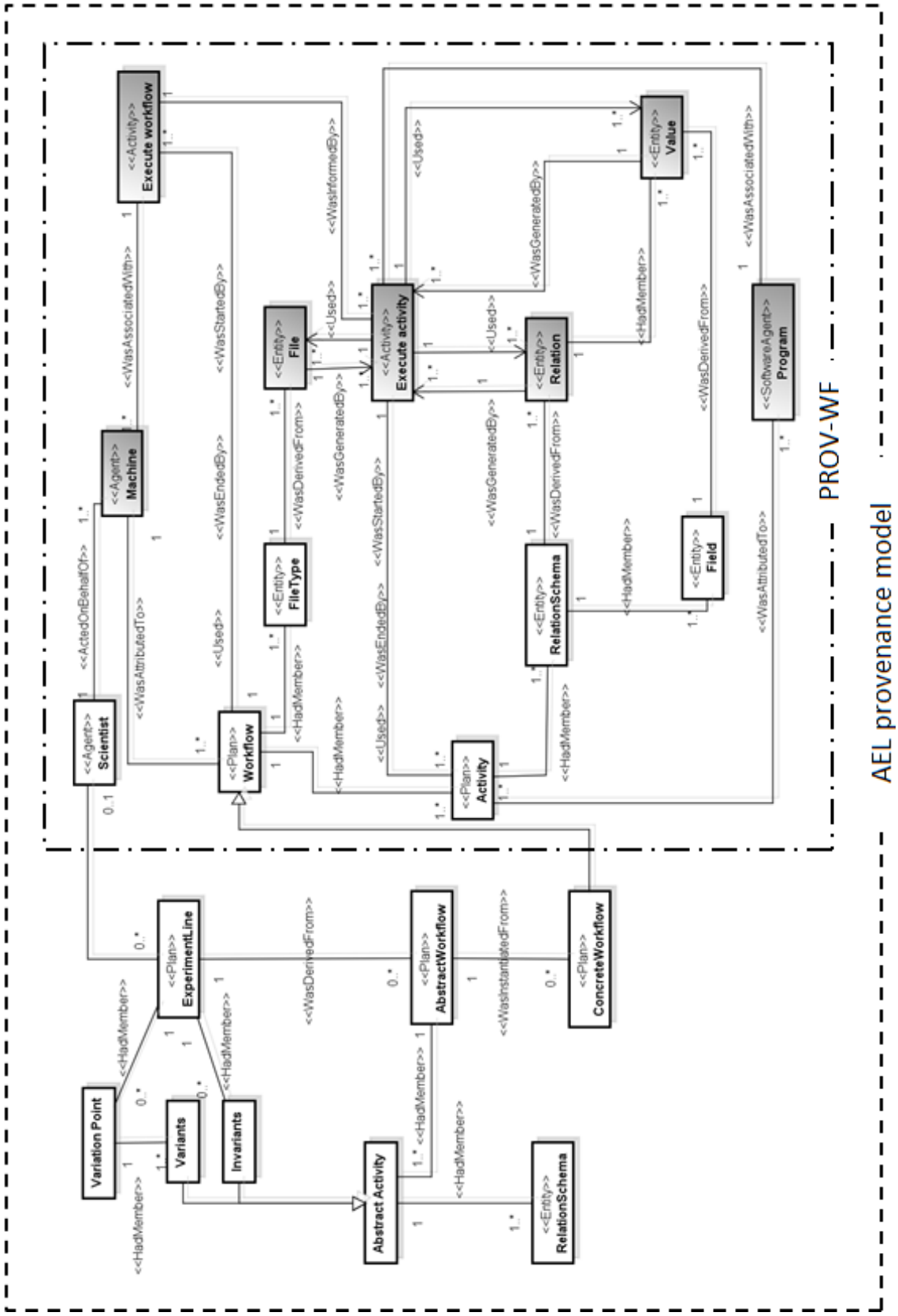

Figure 9. The provenance model proposed for the AEL approach

The provenance model is composed of two main parts: the structure of the experiment (white classes in the UML class diagram), and the execution of the experiment (gray

classes). Furthermore, the stereotypes in the UML class diagram are used to represent PROV components. The agent *Scientist (i.e., <<Agent>> Scientist)* represents a person that models and executes the experiment line. The entity ExperimentLine (*i.e.,* <<Plan>> *ExperimentLine*) is composed of a set of activities, where each activity is responsible for representing variabilities and optionalities in the experiment line level. Experiment lines can be derived (represented by the relation <<WasDerivedFrom>> between <<Plan>> ExperimentLine and <<Plan>> AbstractWorkflow) into abstract workflows. Consequently, concrete workflows can be instantiated from abstract workflows (represented by the relation <<WasInstantiatedFrom>> between <<Plan>> AbstractWorkflow and <<Plan>> ConcreteWorkflow). The entity workflow (*i.e.,* <<Plan>> *Workflow*) is composed by a set of activities (*i.e.,* <<Plan>> *Activity*). Each activity in a workflow is responsible for representing an approach (abstract level) or a program (concrete level). To express all data that is consumed and produced by execution instances (and to verify compatibilities and to add semantics), the entity *RelationSchema* is associated with a schema and can be defined with multiple fields. Each field (*i.e.,* << Entity>> *Field*) describes the meaning of each parameter associated to a program that is associated to an *Activity*. The other elements (gray classes) are associated to the execution of the experiment. Although they are very important information, detailing them transcends the scope of this work. For more details, please refer to Costa *et al* **(2013)**.

4.4 Final Remarks

Considering the issues and the difficulties discussed in Chapter 2 of managing a complex scientific experiment, which can be explored in different trials during its lifecycle, this Chapter presented the AEL approach. The AEL approach aims to solve these problems with the following characteristics/functionalities:

1) definition of a new abstraction level model to fully represent scientific experiments;
2) adaptation of the software product line approach to model scientific experiments;
3) supporting different levels of abstractions on scientific experiment management;
4) formalization of the main concepts of the experiment line approach (variability and optionality) and its production processes (derivation and instantiation);
5) definition of an architecture to support the modeling of experiment lines and automatic derivation of workflows;

6) definition of a provenance mechanism to track the information generated in the AEL approach:
    a. integration to ProvManager to gather provenance information from derived workflows executed in different SWfMS;
    b. definition of a provenance model to map the different abstraction models supported in the AEL approach.

Next, in Chapter 5, we introduce a prototype that implements the AEL approach.

Chapter 5 ExpLine

In this Chapter we present a tool for supporting the creation of experiment lines for composing workflows in different levels of abstraction. This tool, named ExpLine, is complementary to existing SWfMS, thus offering extra higher abstract representation layers to be coupled to the existing approaches. Its goal is to provide an environment for scientists to model their scientific experiments using an abstract level language that automates as much as possible the generation of executable workflows in pre-defined SWfMS such as VisTrails, Kepler or SciCumulus.

The Chapter is organized as follow. Section 5.1 discusses the technical details about the ExpLine specification, describing the main technologies and employed tools. Section 5.2 presents ExpLine in action illustrating its operation with some screenshots of the prototype.

5.1 Technical Specification

Figure 10 illustrates the UML deployment diagram that describes the main components of the ExpLine architecture. ExpLine is a Java standalone application so that scientists can run this prototype in any platform that has the Java Runtime Environment (JRE) installed.

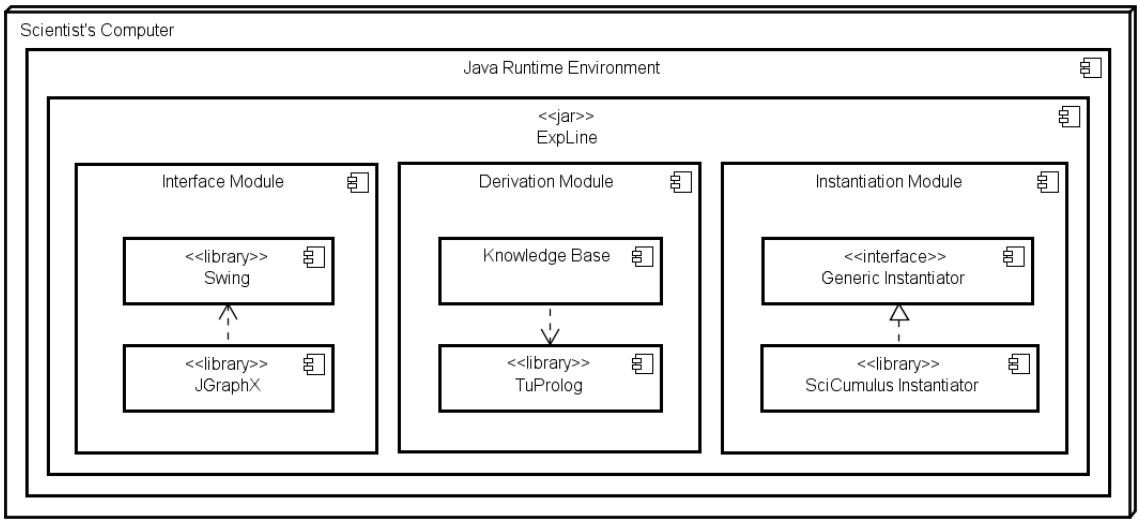

Figure 10. ExpLine architecture illustrated in a UML deployment diagram

ExpLine provides an intuitive interface where experiment lines can be graphically designed. This resource is implemented using the JGraphX library (ALDER, 2015).

JGraphX is a Java Swing diagraming tool that allows graph edition and visualization. JGraphX provides several graphical elements that can be used as nodes and vertices of a graph. We use some of these elements to build the experiment line graphic notation. However, some more specific elements were developed from scratch extending the JGraphX library. Moreover, a lot of graph edition events in the library were customized to support the experiment line rules.

In the Derivation module, the knowledge base used to store the experiment line rules is implemented in Prolog (BRATKO, 2001). Therefore, the definitions presented in Chapter 4 (Section 4.2) were translated into Prolog facts and rules. Additionally, scientists can create composition rules (Definition 20 and Definition 21 from Chapter 4) in the interface module (see Section 5.2.1), which are also translated into Prolog and stored in the knowledge base. When the knowledge base is completely populated it is used by the Derivation Module to calculate the implications on selecting or unselecting one activity during the workflow derivation process (see Section 5.2.2).

The Instantiation module, as specified in Chapter 4, is designed to be extensible to instantiate derived abstract workflows in any SWfMS. This module provides a generic instantiation interface (Generic Instantiator), which any SWFMS specific instantiator component must implement. In ExpLine, as a proof of concept, we implement a specific instantiator component for the SWfMS SciCumulus (OLIVEIRA et al., 2010a). SciCumulus implements the SciWfA workflow algebra in which the experiment line model of ExpLine is based on. This is the SWfMS chosen to be used with the AEL approach in the evaluation presented in Chapter 6.

SciCumulus Instantiator uses the Java API for XML processing to convert XML documents generated in the ExpLine to the XML schema supported by SciCumulus. It adopts the DOM Parser/Builder (LEITHEAD, 2015), which loads the whole ExpLine XML structure into memory and allows writing the SciCumulus XML document with XSLT transformations (CLARK, 2015).

5.2 Usage guide

Figure 11 illustrates the main screen of ExpLine prototype, which can be segmented in 4 main regions: edition canvas, library pane, outline view, and toolbar. Edition canvas is the place where experiment lines are modeled. Experiment lines can be modeled using drag and drop feature to ease the edition. Library pane contains workflow activity types

to represent variability and optionality in the experiment line. Outline view provides an overview of the model and the visible region in the Edition canvas. It also eases the navigation in the model. Toolbar contains basic edition commands that interact with the experiment line model (e.g., copy, paste, zoom in/out, undo, redo, etc.).

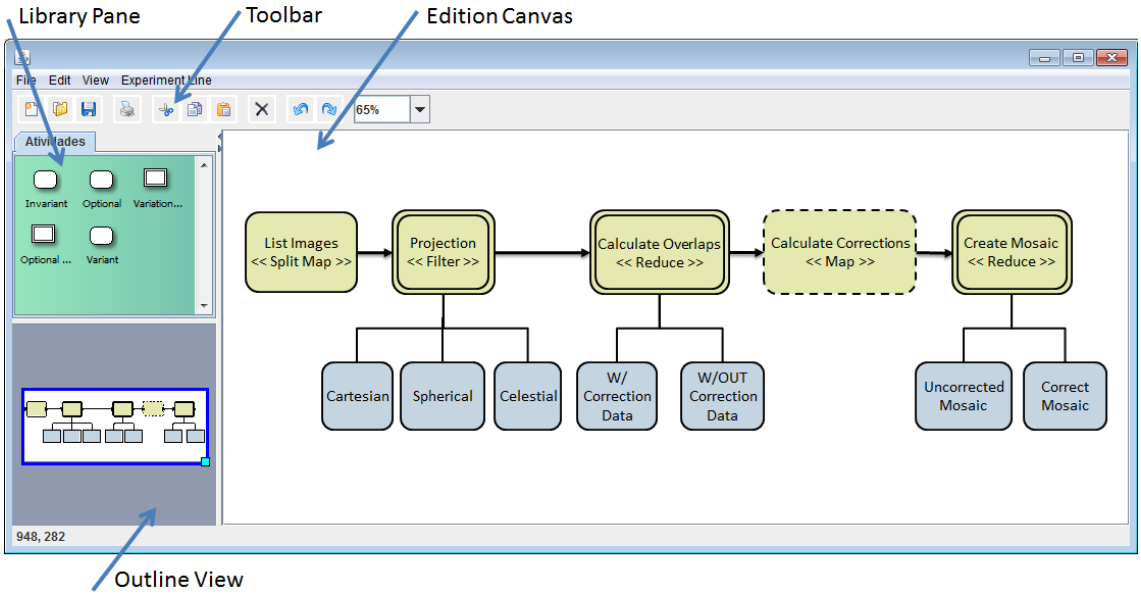

Figure 11. ExpLine main screen

There are two main processes of usage of ExpLine: (a) experiment line composition; and (b) workflow derivation. These two processes are described in details in Section 5.2.1 and Section 5.2.2. The astronomy experiment presented in Chapter 2 is used as an example to illustrate some activities of these processes.

5.2.1 Composing an Experiment Line

The experiment line design process can be described with the following activities:

- Compose workflow;
- Define algebraic operations for the workflow activities;
- Define input and output relation schema for the workflow activities;
- Define variabilities;
- Define optionalities;
- Define composition rules.

Figure 12 illustrates this process in a UML activity diagram. It is worth mentioning that the activity sequence defined for this process is not unique and can be modified. There are other activity sequence variations that scientists could follow.

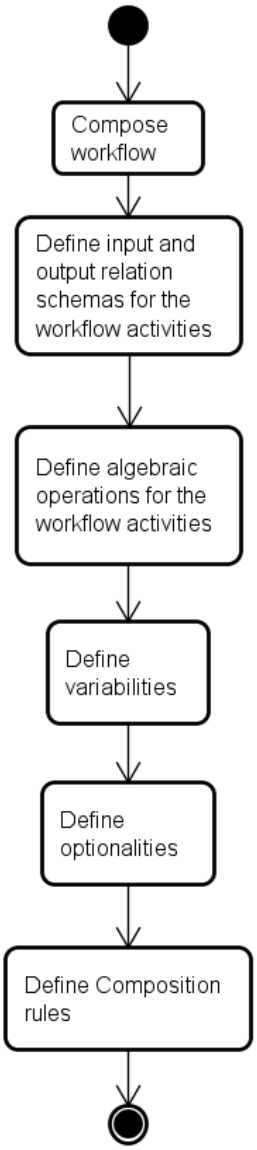

Figure 12. Experiment line design process

Figure 11 illustrates the astronomic data analysis experiment line modeled in ExpLine. Invariant activities are represented by solid single-line rectangles, whereas the optional ones are represented as dashed lines. Variation point activities are represented by double-line rectangles while their variants are represented by blue rectangles. They are associated with undirected edges. Therefore, the yellow part of the model represents the main workflow of the experiment line. The data flow between the workflow activities is represented by directed edges. Finally, the blue part represents the variations that can be selected in the workflow.

Figure 13 illustrates the properties screen for a specific workflow activity where scientists can define its associated algebraic operation and input and output relation schemas. In this example, the workflow activity “List Images” has the algebraic

operator parameter set to “Split Map”. We also see in Figure 13 that the input relation of “List Images” presents the following schema:

Relation 1 = (Mosaic_ID: Int, Repository: String)

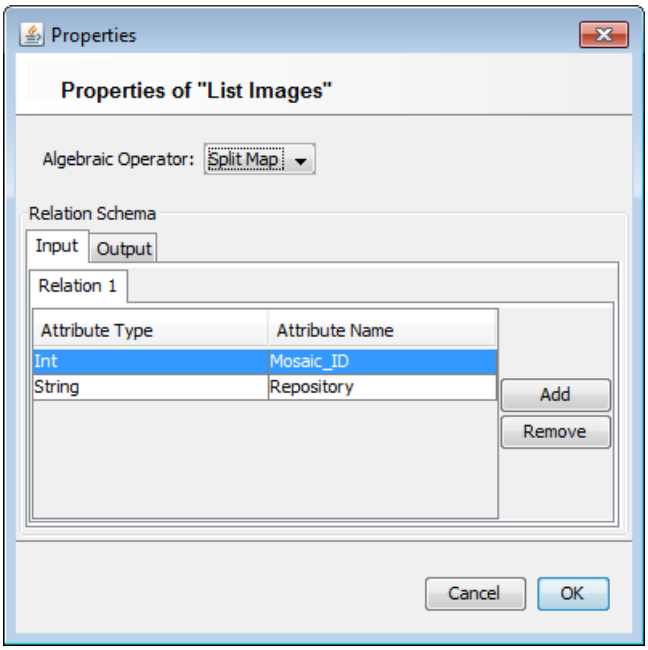

Figure 13. Properties screen of ExpLine

During the experiment line design process, scientists can create composition rules to complement the variabilities and optionalities rules that are intrinsically present in the experiment line. For example, in the astronomy experiment, there is a dependency relationship between the workflows activities “Create Mosaic”, “Calculate Corrections”, and “Calculate Overlaps”. When the variant “Corrected Mosaic” is selected for the workflow activity “Create Mosaic” the optional workflow activity “Calculate Correction” must be selected. Moreover, when “Calculate Correction” is selected the workflow activity “Calculate Overlaps” should generate correction data (i.e., the variant “W/ Correction Data” must be selected).

Figure 14 presents this rule created in ExpLine. “Selected activity” parameter is the condition that triggers the rule (i.e., the selection of “Corrected Mosaic”). The implications are a list of actions which represent selections or deselections of elements in the experiment line. These actions can be processed as a conjunction (logical AND operation) or as a disjunction (logical OR operation). In this rule, the implications are set to be processed as a conjunction since all the actions should be met. It is worth mentioning that this rule was simplified with “Corrected Mosaic” being directly

dependent from “W/ Correction Data” to avoid the creation of two rules, and thus simplifying the demonstration in this example.

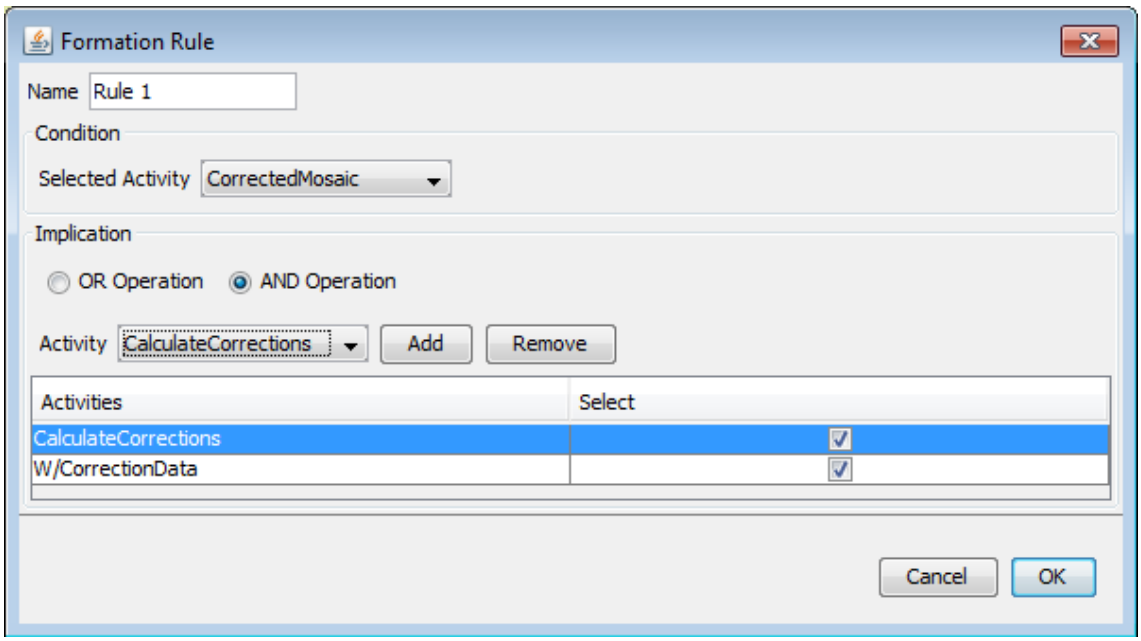

Figure 14. Composition rule edition screen of ExpLine

5.2.2 Workflow Derivation

The workflow derivation is an iterative and interactive process. ExpLine assists the scientist in some activities in order to automate the process as much as possible. The process comprehends from the experiment line activities selection until the instantiation of the derived workflow in a specific SWfMS. It can be described with the following activities:

- Select variant or optional workflow activity;
- Calculate implications;
- Confirm implications;
- Derive abstract workflow;
- Define programs for the abstract workflow activities;
- Select SWfMS;
- Define SWfMS parameters;
- Instantiate abstract workflow to SWfMS.

Figure 15 illustrates this process in a UML activity diagram. The instantiation of the derived abstract workflow can be optional. Therefore, the process has two endpoints to represent the two distinct paths. Additionally, the process is shown in swimlanes to

discriminate the activities performed by ExpLine from the activities performed by the scientist.

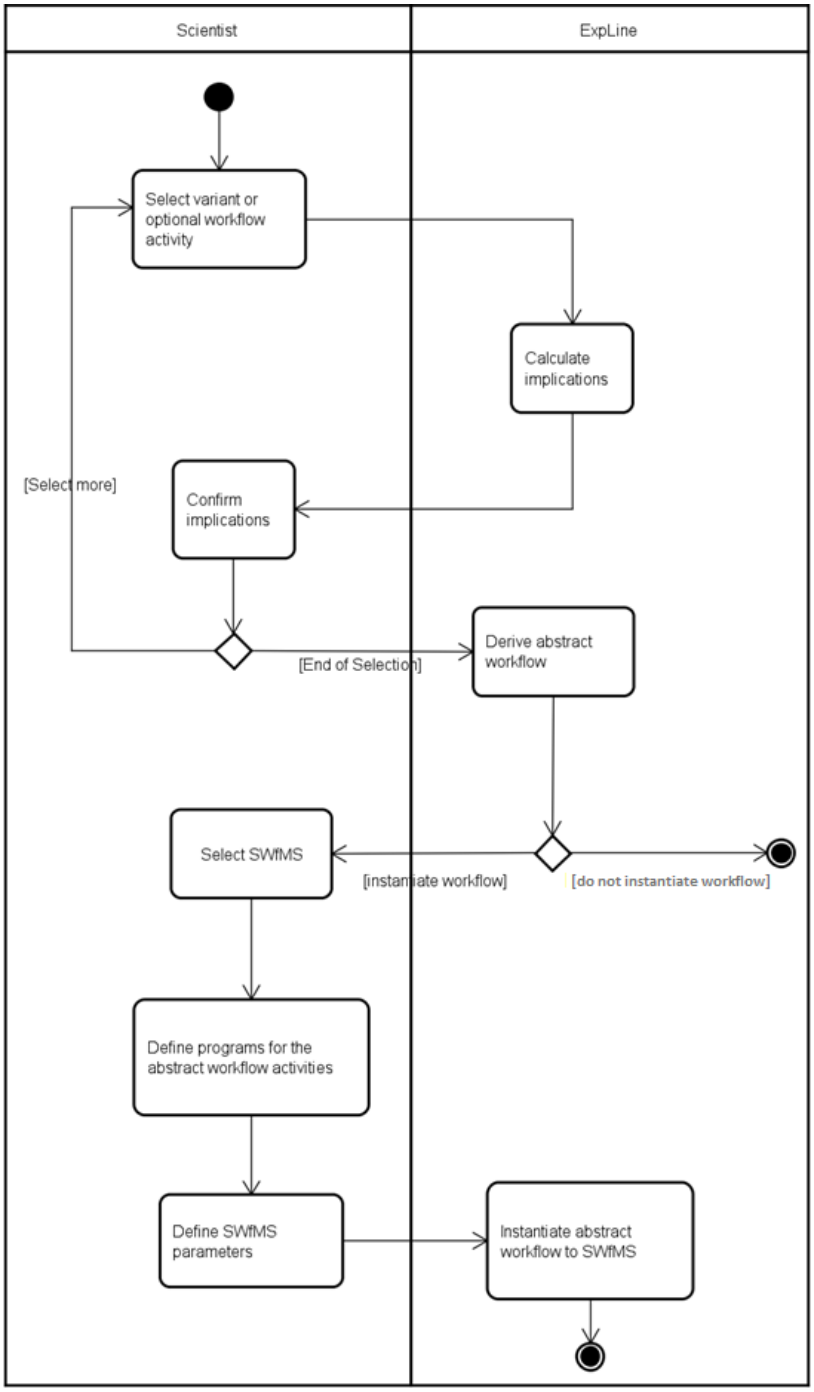

Figure 15. Workflow derivation process

Once the experiment line is modeled, scientists can go in the "Experiment line" menu and click in "Derive Workflow". This action causes ExpLine to change the Experiment Line Edition view to the Workflow Derivation view, as shown in Figure 17. In this view, scientists have to choose the workflow activities of the experiment line that they want to use to derive an abstract workflow. The workflow activities not selected are

graphically represented in semi-transparent colors, while the selected ones are represented in opaque colors.

For each of the selected workflow activities, ExpLine processes the possible implications that this selection may have. The implications are a list of required actions that have to be done in the experiment line (i.e., selecting or deselecting other workflow activities) in order to make the selection valid. However, depending on the complexity of the experiment line model, the implications can be numerous. We can have different implication options to make the selection valid. The knowledge base of ExpLine implemented in Prolog processes all the valid implications and present them to the scientist to be confirmed. For more detail, Appendix I presents the Prolog database dump that contains all the predicates generated by ExpLine for the astronomy experiment line.

For example, the astronomy experiment can have another composition rule which defines that the use of the variant "W/ Correction Data" of the workflow activity "Calculate Overlaps" requires that the workflow activity "Projection" uses only the "Cartesian" or the "Celestial" coordinate system. Figure 16 illustrates the Prolog rules derived from the astronomy experiment line including the aforementioned hypothetical rule (i.e., "W/ Correction Data" -> NOT Spherical).

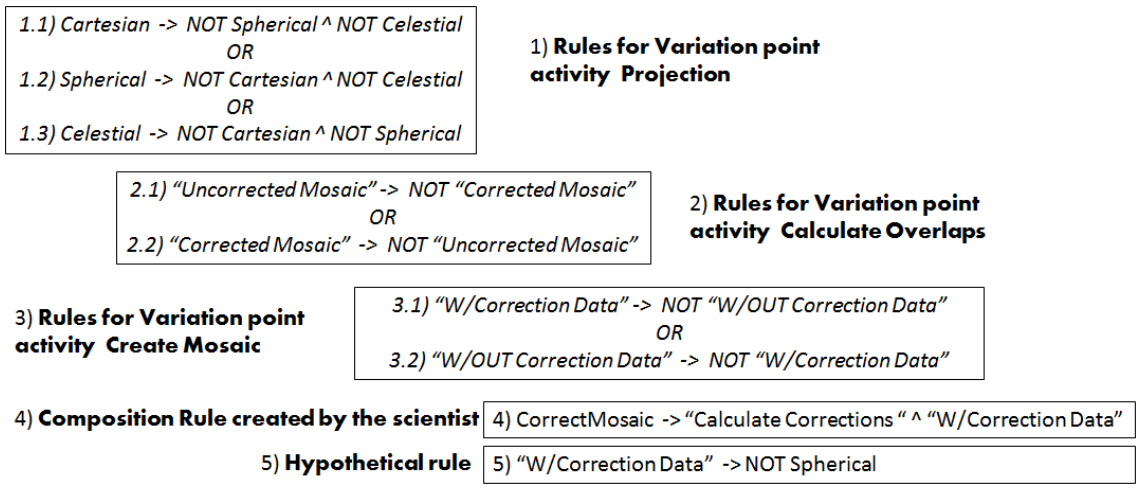

Figure 16. Prolog rules derived the astronomy experiment line

Considering the current selection configuration for the experiment line illustrated in Figure 17, we have two implication options when the variant activity "W/ Correction Data" is selected.

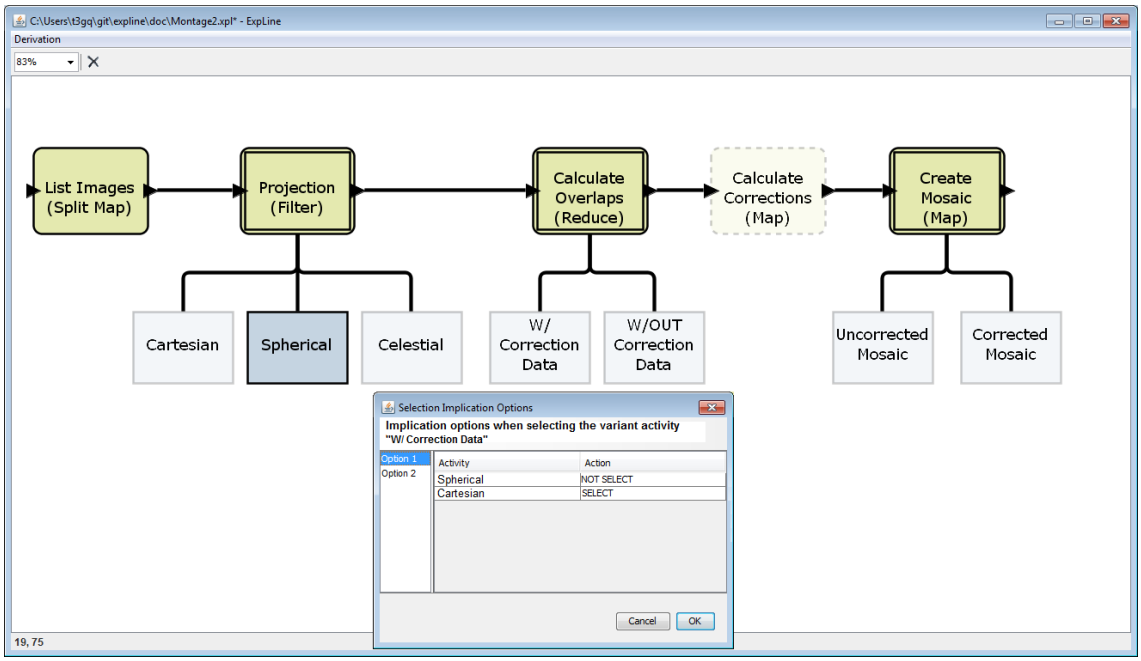

Figure 17. Derivation implication screen

Figure 18 illustrates the inference tree processed by the knowledge base of ExpLine to determine the implication options. Each tree path that begins from the root until a leaf node of the tree is an implication option. Finally, as the scientist chooses one of the implication options, ExpLine automatically updates the experiment line model selecting and deselecting the workflow activities.

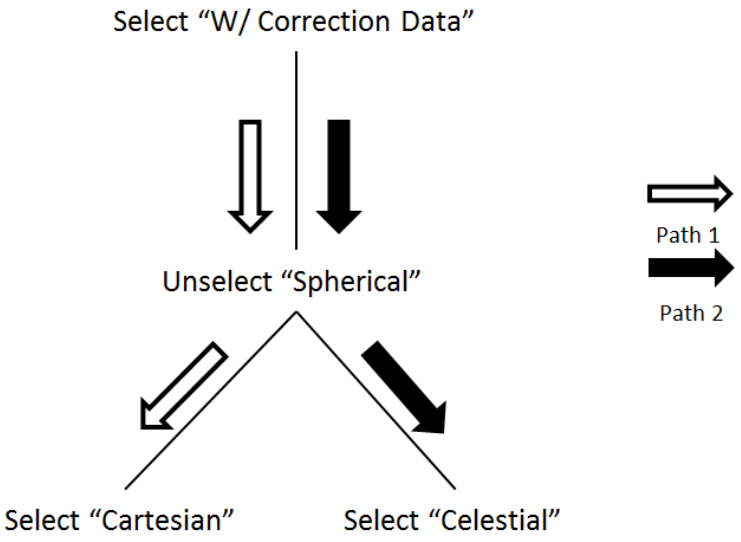

Figure 18 - Inference tree

When the selection is complete, the next step is to derive the abstract workflow. However, before that, ExpLine analyzes the selection configuration to verify if it satisfies the experiment line model specification. In other words, ExpLine verifies if the

variability, optionality and composition rules defined in the experiment line are met. In positive case, ExpLine derives the abstract workflow exporting to a XML file using the workflow notation defined for the AEL approach. In negative case, ExpLine lists to the scientist the rules not met yet in the current selection configuration. Figure 19 illustrates a snippet of the XML document of the abstract workflow derived from the astronomy experiment. In this snippet we can see the specification for the workflow activity "List Images" and its input and output relations. Appendix II illustrates the XML schema (FALLSIDE; WALMSLEY, 2004) of the abstract workflow used by ExpLine. The schema is presented via a graphical hierarchical notation available in the XML editor solution XML Spy (ALTOVA, 2014) to ease the visualization.

```
<?xml version="1.0" encoding="UTF-8" ?>
<AbstractWorkflow>
  <Activity id="2" algebraicOperator="SplitMap" name="List Images">
    <Ports>
      <InputPort id="4">
        <RelationSchema>
          <RelationSchemaAttribute name="Mosaic_ID" type="Int" />
          <RelationSchemaAttribute name="Repository" type="String" />
        </RelationSchema>
      </InputPort>
      <OutputPort id="3">
        <RelationSchema>
          <RelationSchemaAttribute name="Mosaic_ID" type="Int" />
          <RelationSchemaAttribute name="CNTR" type="Int" />
          <RelationSchemaAttribute name="CRVAL" type="Float" />
          <RelationSchemaAttribute name="FNAME" type="String" />
        </RelationSchema>
      </OutputPort>
    </Ports>
  </Activity>
  ...
</AbstractWorkflow>
```

Figure 19. Snippet of the XML document of the abstract workflow derived from the astronomy experiment

After the workflow derivation, scientists have the option of instantiating the abstract workflow selecting one of the SWfMS supported by ExpLine. In case of selecting SciCumulus system, scientists have to fill a form with some information about the SciCumulus execution environment and the programs that will be used to represent the abstract activities. This form is divided in several categories and organized in tabs, as shown in Figure 20. Figure 20 illustrates, in special, the specific tab to list the programs and parameters that will be used to instantiate the workflow.

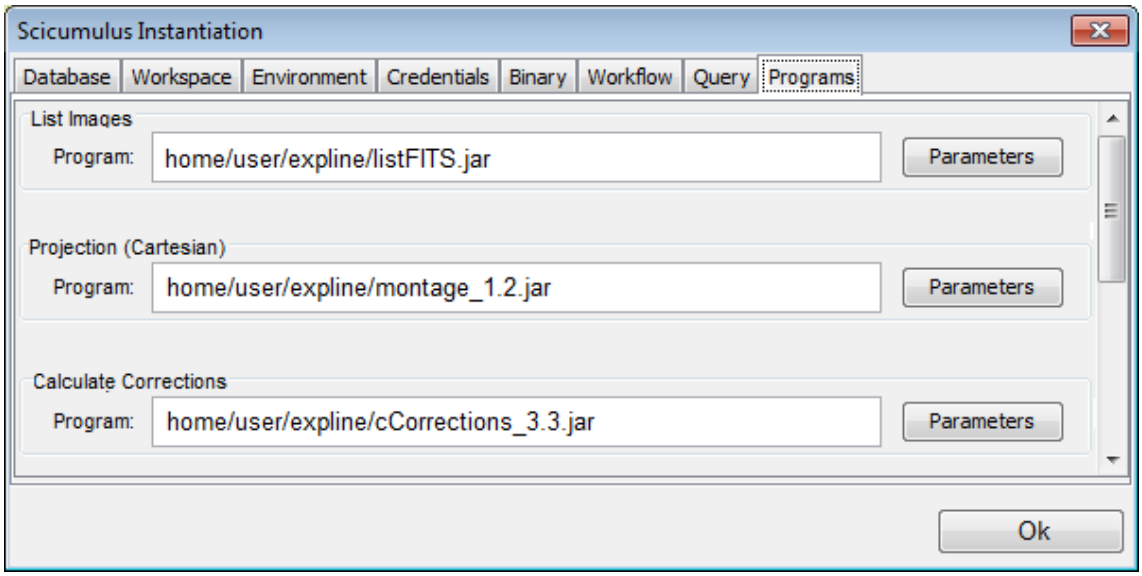

Figure 20. SciCumulus instantiation screen

5.3 Final Remarks

ExpLine provides an environment where scientists can compose experiment lines, derive and instantiate workflows to be executed in SciCumulus system. However, the current version of ExpLine prototype does not cover the provenance mechanism specification described in the AEL approach, which proposes integration with ProvManager to collect and store provenance data. Apart from this limitation, ExpLine contributes in showing some evidences that the AEL approach can be implemented and providing some insights on how the approach can be enhanced.

The implementation of this prototype was the first step for the AEL approach evaluation. The next step is to use ExpLine in a real scenario and analyze if this tool can improve the management, in special, of complex experiments that during their life cycle can have several workflow specifications but are not related to each other. In Chapter 6, we present a case study with a real experiment of the bioinformatics area to evaluate that.

Chapter 6 Case Study and Evaluation

This Chapter presents a case study of an experiment in the phylogenomic area applying the AEL approach. The main idea is to validate the hypothesis of this thesis, which can be summarized as:

An algebraic approach for the experiment line drives a correct derivation of abstract and concrete workflows that are adherent to the experiment specification. Moreover, it also guarantees that an algebraic workflow execution is adherent to the experiment line semantics by means of common algebraic operations and operands that define, at a high level of abstraction, how data is processed.

To analyze that, we used the prototype ExpLine combined with SWfMS SciCumulus (OLIVEIRA et al., 2010a) and some minor manual activities to simulate the whole process of composing experiment lines, deriving workflows, and analyzing provenance.

However, we begin by describing a typical implementation of this phylogenomic experiment and the difficulties and issues that scientists have to deal using state of practice SWfMS. We show how each problem detected in this implementation can be addressed by the AEL approach. In the meantime, we cite other approaches in the literature that could be used in this case study showing their limitations compared to the AEL approach.

The Chapter is organized as follows. Section 6.1 presents the phylogenomic experiment and discusses a typical implementation. Section 6.2 presents the implementation of the phylogenomic experiment in the AEL approach. Section 6.3 briefly analyses other approaches that could be applied in the context of phylogenomic experiment describing their limitations.

6.1 Phylogenomic Analysis Experiment

Phylogenomics is a bioinformatics domain that uses large genomic data sets to address difficult phylogenomic[4] problems. Alternatively, phylogenomics may be described as the use of phylogeny and comparative analysis to infer biological events of genome

[4] Phylogenomics is the study of evolutionary relationships among groups of organisms – complete genes (*e.g.*, species).

evolution (LEIGH et al., 2011). Phylogenomic experiments produce phylogenomic trees that depict existing evolutionary relationships and are used for inferring phylogeny. The Phylogenomic experiment presented by Oliveira *et al.* (2013) has one example of a workflow that processes a large collection of multi-fasta files (*e.g.,* DNA, RNA or protein sequences) to obtain phylogenomic trees.

Figure 21 presents the workflow for the phylogenomic analysis experiment. The workflow is composed of seven activities. The first four activities are specifically for the phylogenetic analysis: MSA (Multiple Sequence Alignment) construction, evolutionary model election, phylogenetic tree construction, and MSA conversion. The last three activities represent the phylogenomic analysis itself: MSA concatenation, evolutionary model selection, and phylogenomic tree construction. In the end of the workflow, there is still a manual activity for electing the best evolutionary tree, which is done by the scientist.

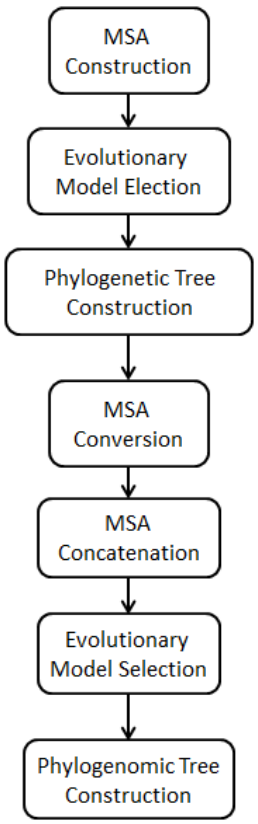

Figure 21. Phylogenomic Workflow

The activity "MSA Construction" uses a specific MSA program to produce individual multiple sequence alignments. There are several available programs that implement this activity (*e.g.,* ClustalW, Kalign, MAFFT, Muscle or ProbCons). Each one of these programs follows a specific bioinformatics algorithm (*i.e.,* approach) (*e.g.,* maximum

expected accuracy, Viterbi alignment or Wu-Manber string-matching (WU; MANBER, 1992)), which clearly provides different performance and results that directly impacts on the choice of them. In addition, scientists do not know *a priori* which MSA algorithm provides the best results, so they have to try different programs in different workflow executions (trials). Each MSA program receives a multi-fasta file as input and produces a MSA as output.

Each MSA is then tested by activity "Evolutionary Model Election" to find out the best evolutionary model (BLOSUM62, CPREV, JTT, WAG, and RtREV) using ModelGenerator. Activity "Phylogenetic Tree Construction" generates phylogenetic trees using a maximum likelihood implementation that varies its execution according to the selected evolutionary model. Activity "MSA Conversion" converts each MSA to the PHYLIP format (FELSENSTEIN, 1989). Although useful, this conversion is optional. This way, scientists have to decide if the PHYLIP conversion is really needed. Next, all individual MSA (converted or not) are concatenated to obtain a "super alignment" in activity "MSA Concatenation". In activity "Evolutionary Model Selection", a specific evolutionary model is selected and is used in activity "Phylogenomic tree construction" to construct phylogenomic trees consuming each "super alignment". Finally, scientists have to choose one evolutionary model between five possibilities. All phylogenomic trees have to be analyzed to elect the one that best reflects the organisms' evolutionary relationship hypothesis.

In phylogenomic experiments it is common to explore several available alternatives of algorithms to investigate which one produces the best results. This way, scientists have to explore all alignment algorithms implemented in different forms. In addition, in several cases, scientists explore different programs for the same algorithm to evaluate which one presents better data quality × performance ratio as presented by Ocaña *et al*. (2011). If we consider that scientists have five different MSA algorithms, five different evolutionary models, and the optionality in activity MSA format conversion, there are 50 different ways of obtaining phylogenomic trees. In other words, we have 50 different workflows that can be modeled for a single phylogenomic analysis experiment (*i.e.,* 5 MSA × 5 evolutionary model × 2 options for MSA conversion activity).

6.1.1 Typical Implementation

Phylogenomic experiments are commonly implemented using SWfMS that deals with the different workflows of the experiment independently, *i.e.*, they are not related (**Problem 1**). Examples of these SWfMS are VisTrails, Triana, Kepler, Taverna, Pegasus, and Swift. It is worth mentioning that the workflows are managed only in a concrete level. There is not a conceptual view of the experiment (i.*e.*, a higher abstraction model) (**Problem 2**) that helps scientists to identify the origin of the concrete workflows. Actually, this information is kept only in the mind of the scientists. Therefore, there is a risk that non-adherent workflows to the experiment specification are generated (**Problem 3**). In other words, workflows may have missing, incorrect or non-expected characteristics of the experiment specification.

As the workflows are treated independently, scientists create several projects in the SWfMS to manage them. However, these workflows share a lot of common characteristics and scientists model them over and over again for each project. This is not efficient and can be error-prone. There is not a formalized strategy of reuse to help scientists on the workflow composition (**Problem 4**). For instance, in the phylogenomic experiment several workflow projects are created for each MSA approach available for the workflow activity MSA Construction. The specification for the remaining of the workflow activities is identical for all of these workflow projects.

In this scenario, the workflows projects files are typically kept in directories using a convention in the file name to distinguish the different versions. The simple task of searching a specific workflow for the experiment is very complex (**Problem 5**). Figure 22 illustrates an example of how the workflows are organized in the file system. Each folder is discriminated by the MSA program and the evolutionary model that the workflow uses.

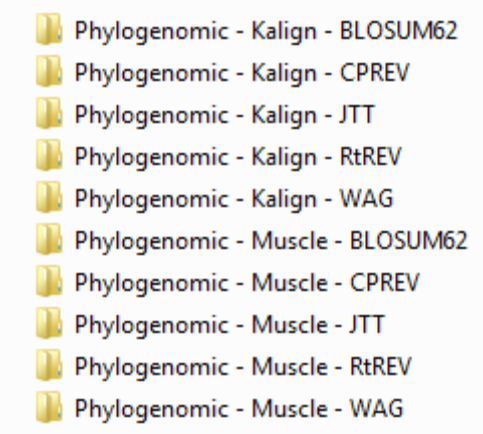

Figure 22. Workflow variations from the phylogenomic experiment organized in folders

To execute the experiment, scientists have to manually load each workflow project in the SWfMS. Sometimes scientists have to execute a lot of workflows, varying a specific approach in a workflow activity. Nevertheless, to gather all the workflows that meet this particular characteristic is an unfeasible task due the way the workflows projects are stored. There is not an automatic process to generate/select workflows according to experiments characteristics defined by the scientist (**Problem 6**).

Another problem faced by scientists is about provenance. The independency of the experiment workflows executed in different projects in the SWfMS leaves little room for analysis. Since the provenance information is not integrated, scientists can make queries only in the context of a particular workflow composition. It is not possible to correlate information of different workflows. Even though it would be possible, the problem of not having a conceptual view of the whole experiment makes the reasoning complex. We cannot correlate the data generated in the execution level to those defined in the experiment conception. There are several missing links for connecting prospective and retrospective provenance data (**Problem 7**).

A common analysis is to find out which MSA approach produces phylogenetic trees in less meantime. In a typical implementation, this is done manually by the scientist. It is necessary to group each workflow execution of a specific approach and calculate the meantime. However, the workflow execution data is distributed in different workflow projects. For instance, the MSA approach Viterbi is distributed in at least five workflow projects. Each workflow represents a combination of the Viterbi approach to a specific Evolutionary Model (BLOSUM62, CPREV, JTT, WAG, and RtREV).

The problems identified in the phylogenomic experiment using a trivial implementation can be mitigated using the AEL approach. This approach leverages the experiment management at a new level providing resources for modeling experiments and their variabilities seamlessly. Furthermore, workflows can be derived and executed using a guided process, which collects and maps the provenance information at different levels of abstraction. This guarantees that sophisticated provenance queries can be done correlating workflow execution data with their corresponding abstract information.

6.2 Modeling and Deriving a Phylogenomic Analysis Experiment Line

Based on the workflows activities characteristics of the phylogenomic experiment described in Section 6.1, the first step is to elicitate all variabilities and optionalities that will be part of the phylogenomic experiment line. For example, the MSA Construction activity is classified as a variation point, which is associated to three variant activities: Wu-Manber, Viterbi, and MaxExpAcc. Each one of these activities represents a specific approach that can be chosen to derive a distinct workflow. In fact, scientists do explore each one of these variations.

The second step to be performed is to model the experiment line. It can be created in a top-down way from the highest abstract level to the concrete level. The experiment line is modeled just once and scientists derive workflows to be executed or make annotations any time after that. It prevents scientists from composing a lot of similar workflows, which is very complex to manage when we are dealing with several workflow variations (mitigates **Problem 4**). Furthermore, the experiment line model provides to the scientist an integrated view of the experiment at a higher level of abstraction (mitigates **Problem 2**). In a unique model it is possible to visualize the main workflow for the phylogenomic experiment and their variations represented to each variation point activity as illustrated in Figure 23.

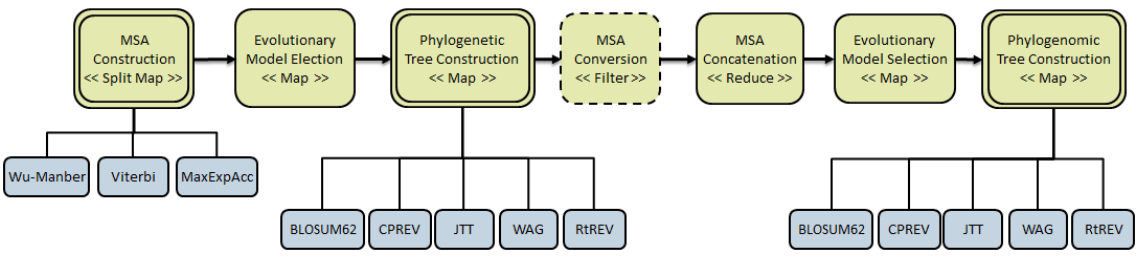

Figure 23. Phylogenomic experiment line

From the experiment line model, scientists conduct experiments by deriving and instantiating, respectively, abstract and concrete workflows associated to that experiment. ExpLine supports both instantiation and derivation processes automating a large part of them (mitigates **Problem 6**). In the derivation process, ExpLine provides a friendly graphical interface, illustrated in Figure 24, which let scientists to select the workflow that they want to derivate (mitigates **Problem 5**). This is done by selecting and deselecting variabilities and optionalities represented in the experiment line model. For each selection event, ExpLine processes the experiment line rules to identify implications that have to be done in order to turn the workflow selection valid. For instance, there could be a rule to determine that the evolutionary model JTT should be used whenever the Wu Manber approach is selected for the workflow activity MSA Construction. The implications can be multiple, depending on the experiment line characteristics. Therefore, the implications alternatives are presented to the scientist in order to select the one that fits better. The scientist can only derive the abstract workflow when the requirements are met in the experiment line specification (i.e., all the mandatory activities selected, only one variant activity selected for each variation point activity, and the composition rules respected) (see Definition 22 from Section 4.2.5). This results in the derivation of only correct workflows that are adherent to the experiment specification (mitigates **Problem 3**).

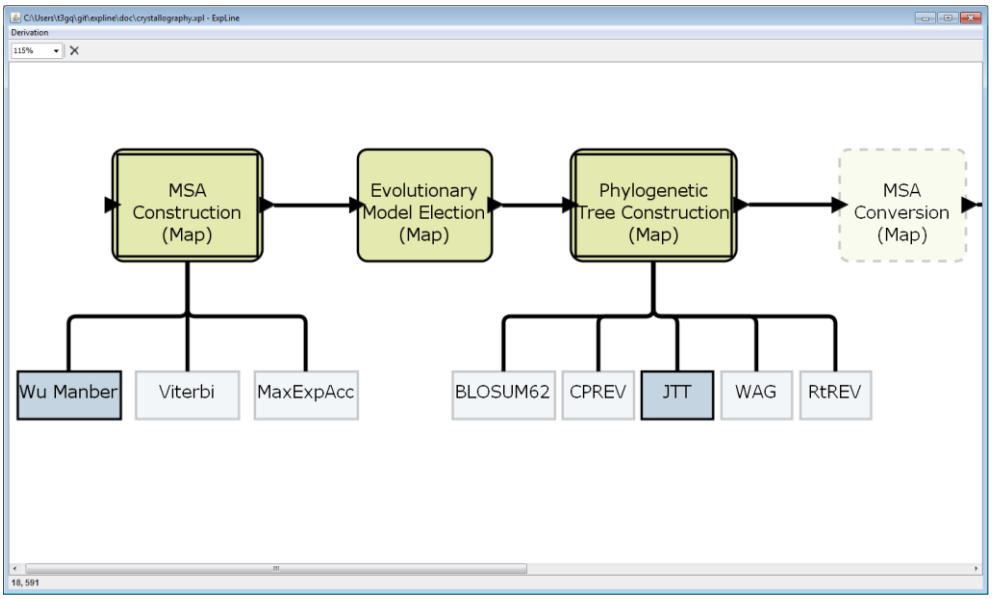

Figure 24. ExpLine derivation screen

In the instantiation process, ExpLine takes the derived abstract workflow written in a generic XML format and converts it to a specific SWfMS format. In this case study we

instantiated the workflow to the SciCumulus system. However, the process is not completely automatic. The scientist must pass to ExpLine some information about the instantiation. This information contains details about the programs to be used for each abstract workflow activity, their parameters, and so forth.

In this case study we explore two different compositions for the experiment. In the first one, scientists choose the Wu-Manber approach as variant activity to the MSA Construction variation point. In the second one, scientists choose the maximum expected accuracy approach (MaxExpAcc) as variant activity. These choices will lead to two different abstract workflows.

Figure 25 illustrates a preview of these two abstract workflows instantiated in SciCumulus (i.e., two concrete workflows), which were derived from the phylogenomic analysis experiment line (modeled in ExpLine). We only present the first activities of the concrete workflows for the sake of simplicity. One of them uses Kalign for the first activity and the other uses ProbCons. These programs are associated with Wu-Manber and MaxExpAcc algorithms, respectively. These two variations of the same experiment produce results with different quality and execution performance. For example, if we take the phylogenomic experiment conducted by Oliveira *et al*. (2012), we can observe that the concrete workflow variation that uses the Kaling program is more efficient in execution time than the ProbCons variation, as shown in Figure 26.a.

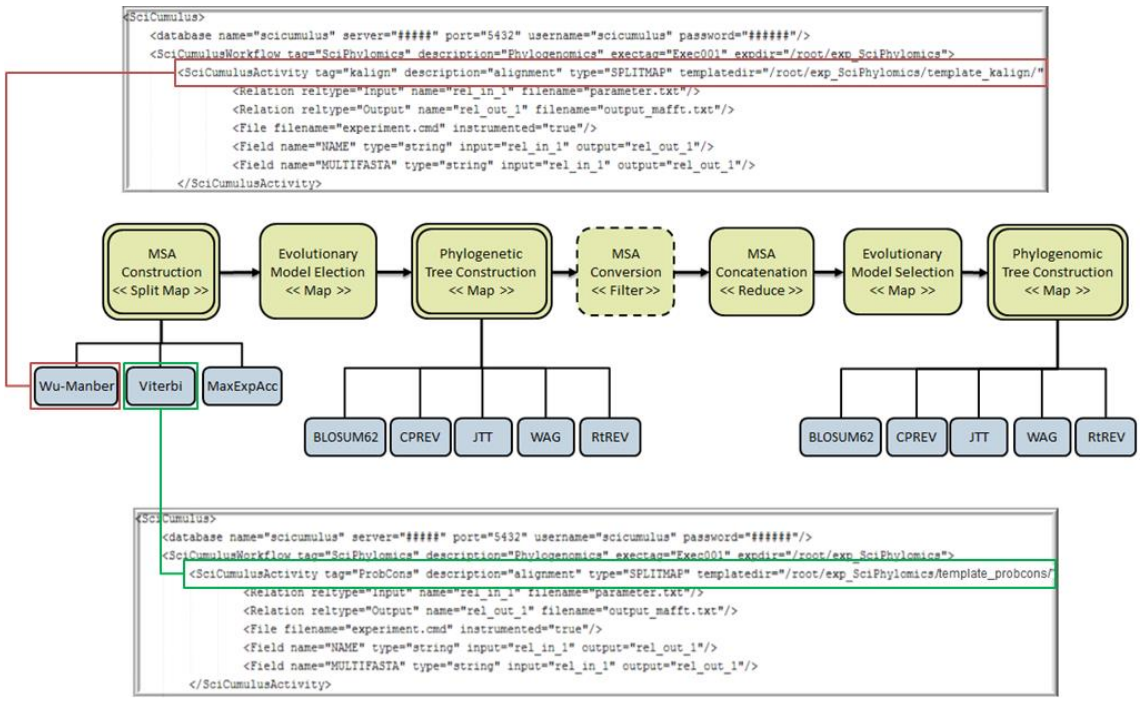

Figure 25. Excerpt of derivation of phylogenomic experiment line into concrete workflows

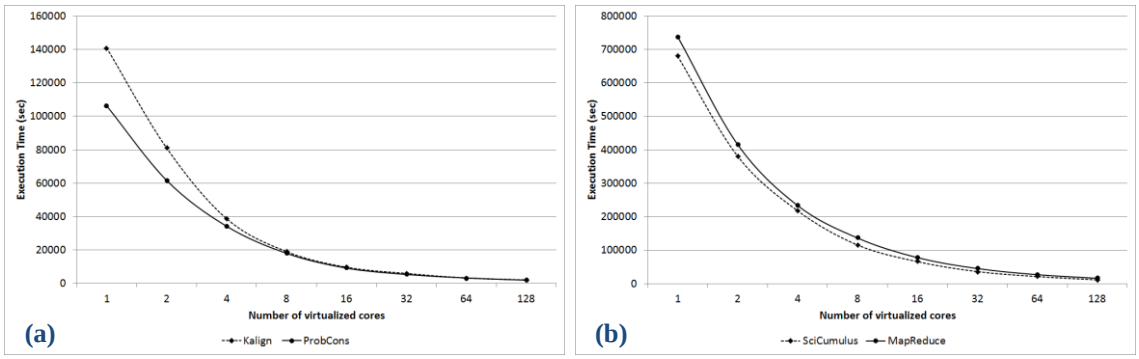

Figure 26. Execution time using (a) Kalign and Probcons and (b) using our derived algebraic workflow in SciCumulus and a Hadoop implementation as presented by Oliveira *et al.* (2012)

A typical analysis in phylogenomic experiments is about the quality of the results. This can be done by comparing signalized set of phylogenomic trees using tree distance, reinforcing phylogenetic inferences. The idea is to analyze which results presents bootstrap replication values greater than the other trees. Using this test we concluded that these two methods provide trees with different quality. In this experiment, Kaling produced phylogenomic trees with more quality than ProbCons. Since the scientific workflow that contains Kalign produced results with more quality and executed faster than the workflow that contains ProbCons program, it is natural that scientists will opt to use Kalign in future larger executions. In this case, the experiment line helped scientists to explore a space of options to conduct their experiments.

Once the phylogenomic analysis experiment was modeled using experiment line, the experiment line activities and derived abstract workflows are directly associated to an algebraic operator and the dependencies are represented in the algebraic expression. The application of the SciWfA algebra systematically in all abstraction levels contributes for the process of querying workflow execution data correlating with their corresponding abstract experiment definition (mitigates **Problem 7**).

Each workflow derived from the experiment line is registered in the ExpLine provenance database. From that, scientists can track which features were selected in the experiment line for a particular derived workflow. Additionally, the provenance data generated in the SWfMS related to the derived workflow executions are collected via workflow instrumentation by ProvManager. ProvManager gathers this data and publishes in the ExpLine provenance database. For more details, see Section 4.3.1 (Chapter 4).

The integration of ExpLine and ProvManager is not implemented yet. Nevertheless, in order to evaluate this study case a database based on the ExpLine provenance model presented in Figure 9 (Chapter 4) was created. The provenance data generated from SciCumulus system were collected and inserted manually in this database.

Therefore, using this simulated database, scientists can perform queries in the provenance model of ExpLine to analyze the results of the experiment as a whole. Querying provenance corresponding to all workflow executions within the same experiment line would require a significant effort when using only the SWfMS provenance support. Since the main abstraction in the existing SWfMS is the workflow itself, it is not simple to gather and combine provenance results from distinct workflow executions. On the other hand, when using ExpLine, these queries are directly formulated (mitigates **Problem 1**). For example (Example I), scientists may want to analyze which MSA approach led to the best execution time. This can be queried using a SQL statement as illustrated in Figure 27, which would result in a table as depicted in Figure 27 as well. Another example (Example II) of query is to discover which MSA approach is more used in the derived abstract workflows. Figure 28 illustrates the SQL statement for this query and its respective table result.

```
SELECT abstActv.Name, avg(eWf.endTime-eWf.startTime) as meantime
FROM AbstractActivity abstActv, AbstractWorkflow abstWf, ExpLine el, Derivation der, VariationPoint vp, Variant var,
ExecutionWorkflow eWf, ExecutionActivity eActv, ConcreteWorkflow concWf, ConcreteActivity concActv, Instantiation inst
WHERE el.Name = 'Phylogenomic' AND vp.Name = 'MSA Construction'
    AND vp.ExpLineID = el.ID AND vp.ID = var.VariationPointID AND var.AbstractActivityID = abstActv.ID
    AND der.ExpLineID = el.ID AND der.AbstractWorkflowID = abstWf.ID AND der.AbstractActivityID = abstActv.ID
    AND inst.AbstractWorkflowID = abstWf.ID AND inst.ConcreteWorkflowID = concWf.ID
    AND inst.AbstractActivityID = abstActv.ID AND inst.ConcreteActivityID = concActv.ID
    AND eWf.ConcreteWorkflowID = concWf.ID AND eActv.ConcreteActivityID = concActv.ID
    AND eActv.ExecutionWorkflowID = eWf.ID
GROUP BY abstActv.Name;
```

Name	meantime
Wu-Manber	30000
Viterbi	33000
MaxExpAcc	35000

Figure 27. Example (I) of SQL statement and respective result table for Phylogenomic experiment

```
SELECT abstActv.Name, count(abstActv.Name) as count
FROM AbstractActivity abstActv, AbstractWorkflow abstWf, ExpLine el, Derivation der, VariationPoint vp, Variant var
WHERE el.Name = 'Phylogenomic' AND vp.Name = 'MSA Construction'
      AND der.ExpLineID = el.ID AND der.AbstractWorkflowID = abstWf.ID AND abstActv.ID = der.AbstractActivityID
      AND vp.ExpLineID = el.ID AND vp.ID = var.VariationPointID AND var.AbstractActivityID = abstActv.ID
GROUP BY abstActv.Name;
```

Name	count
Viterbi	34
Wu-Manber	20
MaxExpAcc	10

Figure 28. Example (II) of SQL statement and respective result table for Phylogenomic experiment

6.3 Limitation of other approaches

There are others approaches in the literature that could be used to mitigate the problems present in the phylogenomic experiment. In Chapter 3, a comparison made between the approaches pointed Wings as one of the most advanced approaches. Even though Wings provides great features and functionalities, it has several limitations compared to AEL approach. These limitations would be noticed may Wings be applied in the phylogenomic experiment.

For instance, the first major problem is the ontology model provided by Wings. It is not as representative as the experiment line model. The concept of optionality present in the experiment line cannot be expressed in Wings. In the phylogenomic experiment it is essential in the workflow specification to define that the workflow activity MSA Conversion is optional. It means that depending on the kind of workflow derived from the experiment, this activity can be included or suppressed. Something as simple as selecting a variability in the workflow may determine the optionality of MSA Converstion. For instance, selecting the Wu-Manber approach for the workflow activity MSA Construction can require the inclusion of the workflow activity MSA Conversion since the MSA generated is in other format than the PHYLIP. This relationship of implication is not supported in Wings either.

Another major problem of Wings is related to provenance. Wings does not present an approach for integrating its own prospective provenance data to the retrospective provenance data generated independently in the targeted SWfMS where the derived workflows are executed. Therefore, the provenance queries (Figure 27 and Figure 28) elaborated in this study case are not supported by Wings.

6.4 Final remarks

Is this case study, we got some evidences that the AEL is capable of reaching a new level of experiment management. The problems faced by scientists in the phylogenomic experiment using a generic scientific workflow management approach are satisfactorily addressed by the AEL approach.

The AEL approach contributes to a better definition, exploration, and analysis of the experiment execution option space. Since the information is related in an integrated representation, it is possible to connect information generated by the concrete workflow with the abstract workflow representation, so scientists can take advantage of their expertise to better analyze experiments results. Furthermore, these relationships allow scientists to analyze results of different concrete workflows derived from the same experiment. In other words, we provide the capability of analyzing provenance data related to a higher level of abstraction. Finally, comparing to other related approaches, AEL demonstrated to be the best approach to manage the phylogenomic experiment presented in this study case.

It is worth mentioning that the results obtained from this evaluation cannot be generalized due to some constraints described as follows. Even though the experiment used in this evaluation is real and original from bioinformatics domain area, the prototype was not tested by users (i.e., scientists) from this area. All the activities executed in the prototype were only performed by the prototype author. Therefore, we could not measure the difficulty level or the learning curve of using the prototype by unexperienced users. Another question is the number of experiment samples to evaluate our hypothesis. We have used only the phylogenomic experiment, which can also be an experiment very compatible to the problem. Finally, the provenance database and the integration between ExpLine and ProvManager were manually implemented. This could bias the evaluation result due to errors in this manual activity.

Chapter 7 Conclusions

The exploratory nature of the science and the increase of performance and availability of computational resources contributed to the growing of complex computational experiments. Managing the trials of these complex experiments with their respective data generated from several workflows execution is becoming an unfeasible task without an appropriate computation support. It is continuously necessary to propose new approaches to consolidate all these data and new abstraction models to abstract the experiment information in order to ease the work of the scientist.

Therefore, this thesis presented a practical approach to deriving workflows using experiment lines based on an algebraic model. The AEL approach is a way to ease the creation and control of the several workflows that are developed during the experiment life cycle in a unique and integrated representation model. From this model, the implementation alternatives of an experiment are expressed in a high abstract language without considering technical issues. When experiment lines are defined, abstract workflows can be derived and instantiated into concrete workflows to be further executed in specific SWfMS, representing a particular, addressable, experiment alternative implementation.

In a broader view, out of the scope of this thesis, ExpLine is an important component to a bigger project of scientific experiment management carried out in our research group. In this project, we can cite at least four main components: Chiron (OGASAWARA et al., 2013), Achilles (DIAS et al., 2011), SciCumulus (OLIVEIRA et al., 2010a), and ExpLine. Figure 29 summarizes the roles played by each of these components and how they correlate with each other during the experiment life cycle. Chiron is an engine of scientific workflows that provides advanced parallelism execution mechanisms. Chiron is the reference solution that implements SciWfA workflow algebra in which ExpLine is based on. The execution engine of Chiron can be deployed in a computational grid or cluster. Chiron can be also executed in cloud computing environments when it is coupled with the middleware SciCumulus. SciCumulus uses an adaptive execution approach where the elasticity of the cloud virtual machines can be dynamically configured during the workflow parallel execution.

The ExpLine plays the role of an interface of all system, helping scientists during the experiment composition, execution and analysis. In the composition phase, ExpLine enables scientists to model their experiment lines and derive workflows to be executed in Chiron. During the experiment execution, Achilles (DIAS et al., 2011) is a mechanism for enabling scientists to configure the workflow at runtime. These mechanisms are a result of workflow steering (CHIN et al., 2003) (MATTOSO et al., 2015). Normally, workflow steering is done "offline" at the concrete level, *i.e.*, scientists manipulate concrete workflows and adjust some application parameters after the execution. Achilles is unique in providing user steering and workflow adjustment at runtime (DIAS et al., 2015). Using ExpLine in integration with Achilles we can improve this configuration by also providing it at a higher level of abstraction. From this integration, we can evolve the workflow steering from dynamic data exploration to workflow composition configuration allowing scientists, for instance, to select the variabilities of a workflow activity or include or not an optional workflow activity at runtime. One of the main advantages of using ExpLine in this dynamic workflow adjustment is to maintain the workflow specification valid and to guide scientists suggesting correct workflow variations.

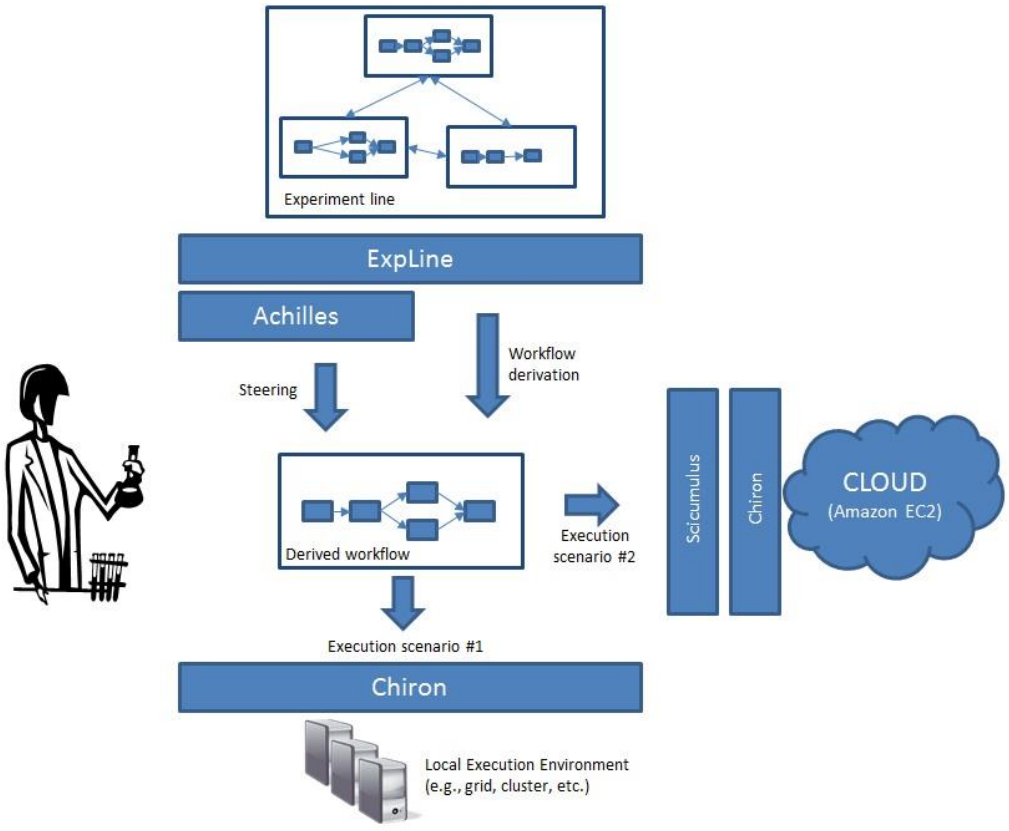

Figure 29. ExpLine working combined with Chiron, SciCumulus, Achilles

7.1 Contributions

This thesis presents the results of a research work that aimed to propose a novel experiment line approach. Among the main contributions, we can list:

- Definition of a new abstraction level model to represent scientific experiments;
- A well-defined and formalized workflow derivation process that now works at three abstraction levels: the experiment level; the abstract workflow level; and the concrete workflow level;
- A provenance approach that tracks the information generated in the AEL enabling the creation of provenance queries that correlates experiment line concepts defined at high abstraction level with derived workflows and their execution data;
    - Integration to ProvManager to gather provenance information from derived workflows executed in different SWfMS;
    - Definition of a provenance model to mapping the different abstraction models supported in the AEL approach;
- Incorporation of the SciWfA algebra into the experiment line approach to explicit the data description and the operation in the experiment abstraction level, thus improving consistency in the model;
- Definition of an architecture to support the modelling of experiment lines and automatic derivation of workflows;
- Implementation of a prototype named ExpLine that supports the new functionalities present in AEL approach;
- A case study that evaluates the AEL approach on managing several trials of an experiment in the phylogenomic area.

7.2 Limitations

During the research, there were some events that hindered the approach development. Furthermore, other limitations were identified from a critical analysis done in both approach and prototype developed. A list of the main limitations is presented as follows:

- As stated in the prototype chapter, the current version of ExpLine does support the provenance mechanism specified in the AEL approach. The integration of

ProvManager and ExpLine is not implemented, and so the provenance data generated in the SWfMS by the derived workflows execution is not gathered and neither mapped to the abstract information of experiment line;

- Currently, the abstract workflows derived from ExpLine can only be automatically instantiated to concrete workflows for the SciCumulus system. It is possible to manually create a concrete workflow for other SWfMS. However, ExpLine loses the information tracking and, as a manual task, it can be error-prone;
- AEL approach does not present a strategy for creating a knowledge base of concrete programs that can be used to instantiate abstract workflow activities from an experiment line. Instead, scientists have to list manually the programs that will be used during the workflow instantiation;
- The composition rules edition in ExpLine is limited to create implications that use only conjunction and disjunction operations. It is not possible to define other operators such as an exclusive disjunction or quantifiers to manipulate set of elements of the experiment line (e.g., if one of the optional activities be selected then select variant "Var1" from the variation point "VP1");
- In the current version of ExpLine, due to graphical limitations, workflows activities that are associated to the algebraic operator MRQuery have only two input relations representation. However, as described in Definition 6, MRQuery algebraic operator can have *n* input relations generating one output relation.

7.3 Future Work

This work is a first step towards the development of an environment to manage scientific experiments at a higher abstraction level using experiment lines. At this phase, we were concerned on formalizing the approach and developing a first functional release of a prototype to compose experiment lines and derive scientific workflows. In the prototype we listed the employed technologies and established the process of usage. Therefore, this work opens a promising perspective for future work. A detailed list of the main future work is described as follows:

- Implement a mechanism to monitor the execution of workflows derived from ExpLine. Furthermore, this mechanism have to be integrated to Achilles for enabling scientists to configure the workflow at runtime;

- Improve the execution optimization capabilities from SciWfA workflow algebra[5] by including experiment line information (such as, variabilities, constraints and dependency, etc.) on the calculus of workflow execution plans. A new workflow execution plan could consider changing an approach of a workflow activity or adding/removing an optional workflow activity. All these modifications could be done at runtime, validating if all the experiment line rules are met. In the negative case, other modifications could be calculated and added to the new workflow execution plan;
- Implementation of the provenance mechanism specified in AEL approach that integrates with ProvManager (Chapter 4);
- Creation of a mechanism that uses the provenance model of ExpLine to guide scientists in the workflow composition. Scientists could use the execution data collected of all the workflows derived from the experiment line to analyze which derived workflow better fits to a particular scenario. For instance, considering an execution scenario of the phylogenomic experiment (Chapter 2), a scientist wants to derive a workflow to be executed in a cloud or a grid environment. The main interest in this trial is to compose a workflow that results in a minimal cost when executed in the environment. A minimal cost could consider the number of execution nodes and the total amount of execution time. Based on the execution history, the scientist concludes that a workflow which uses the Wu-Manber approach for the MSA activity and BLOSUM62 as the evolutionary model is statically the best option. This selection process could be incorporated in the workflow engine optimizer to become automatic during the workflow execution. The workflow engine optimizer could adapt the workflow execution plan at

[5] Derived concrete workflows instantiated in a SWfMS that implements the SciWfA workflow algebra (e.g., Chiron/SciCumulus) in which the AEL is based on can be optimized by doing algebraic transformations. The algebraic model provides the data consumption and production semantics to the abstract workflow activities and a uniform representation in the experiment line. This optimization allows a reduction of the amount of data to be processed by managing the workflow activities execution order and the definition of parallelism points during the workflow execution.

runtime by replacing the scientists' predefined preferences or suggesting changes to be confirmed by the scientists;

- Study the possibility of abstracting even more the experiment line model using feature models (KANG; LEE; DONOHOE, 2002). At this abstraction level, experiment line will be represented purely by experiment concepts (features) that have no direct association with workflows (e.g., workflow activities, input and output data, dataflow, and so forth). This approach could improve even more the experiment understanding, but the tracking of this abstraction level to the existing ones will be more complex;
- Extend the workflow instantiation to other workflow systems implementing other SWfMS specific instantiator modules (e.g., Kepler, Pegasus, VisTrails, etc.);
- Improve the automation of the instantiation process in ExpLine. Currently, when scientists derive a workflow instantiated to a concrete workflow (in that case, to SciCumulus system), ExpLine only returns the SciCumulus specification file (see Section 5.2.2). The rest of task of loading this specification in SciCumulus system and executing it is the scientist's responsibility. This could be improved by delegating this manual task to ExpLine and implementing an integration with SciCumulus;
- Evaluate the approach in other real scenarios to obtain more solid evidences that AEL approach contribute to solve the problem studied in this thesis.

References

ACHER, M. et al. Managing Variability in Workflow with Feature Model Composition Operators. In: BAUDRY, B.; WOHLSTADTER, E. (Eds.). . **Software Composition**. Lecture Notes in Computer Science. [s.l.] Springer Berlin / Heidelberg, 2010. v. 6144p. 17–33.

ALDER, G. **JGraphX: Java Graph Drawing Component**. Disponível em: <http://www.jgraph.com>. Acesso em: 9 mar. 2014.

ALTINTAS, I. et al. **Kepler: An Extensible System for Design and Execution of Scientific Workflows.**, 2004.

ALTOVA. **Altova XML Spy**. Disponível em: <http://www.altova.com/xmlspy.html>. Acesso em: 15 nov. 2015.

AMAZON EC2. **Amazon Elastic Compute Cloud**. Disponível em: <http://aws.amazon.com/ec2/>. Acesso em: 1 jul. 2014.

BELHAJJAME, K. et al. **D2.2v2 Design, implementation and deployment of workflow lifecycle management components - Phase II**. [s.l.] http://www.wf4ever-project.org/, 2014. Disponível em: <http://repo.wf4ever-project.org/Content/49/D2.2v2.pdf>.

BLYTHE, J. et al. **Task scheduling strategies for workflow-based applications in grids**Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on. **Anais**... In: CLUSTER COMPUTING AND THE GRID, 2005. CCGRID 2005. IEEE INTERNATIONAL SYMPOSIUM ON. 2005Disponível em: <10.1109/CCGRID.2005.1558639>. Acesso em: 13 ago. 2010

BRATKO, I. **Prolog programming for artificial intelligence**. Harlow, England; New York: Addison Wesley, 2001.

BRAY, T. et al. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. Disponível em: <http://www.w3.org/TR/xml/>. Acesso em: 9 mar. 2014.

CALLAHAN, S. P. et al. **VisTrails: Visualization Meets Data Management**Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. **Anais**...New York, NY, USA: ACM, 2006Disponível em: <http://doi.acm.org/10.1145/1142473.1142574>. Acesso em: 16 mar. 2014

CAVALCANTI, M. C. et al. Managing structural genomic workflows using web services. **Data & Knowledge Engineering**, v. 53, n. 1, p. 45–74, 2005.

CHIN, J. et al. Steering in computational science: Mesoscale modelling and simulation. **Contemporary Physics**, v. 44, n. 5, p. 417–434, 2003.

CIARLINI, A. E. M. et al. Event relations in plan-based plot composition. **Comput. Entertain.**, v. 7, n. 4, p. 1–37, 2010.

CLARK, J. **XSL Transformations (XSLT)**. Disponível em: <http://www.w3.org/TR/xslt.html>. Acesso em: 9 mar. 2015.

CODD, E. F. **The relational model for database management: version 2**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

COSTA, F. et al. **Capturing and Querying Workflow Runtime Provenance with PROV: A Practical Approach**Proceedings of the Joint EDBT/ICDT 2013 Workshops. **Anais**...: EDBT '13.New York, NY, USA: ACM Press, 2013Disponível em: <http://doi.acm.org/10.1145/2457317.2457365>. Acesso em: 2 dez. 2013

COUVARES, P. et al. Workflow Management in Condor. In: **Workflows for e-Science**. [s.l.] Springer, 2007. p. 357–375.

DA SILVA, P. P.; MCGUINNESS, D. L.; FIKES, R. A proof markup language for semantic web services. **Inf. Syst.**, v. 31, n. 4, p. 381–395, jun. 2006.

DEAN, J.; GHEMAWAT, S. MapReduce: a flexible data processing tool. **Communications of the ACM**, v. 53, n. 1, p. 72–77, 2010.

DEELMAN, E. et al. Pegasus: Mapping Large-Scale Workflows to Distributed Resources. In: TAYLOR, I. J. et al. (Eds.). . **Workflows for e-Science**. [s.l.] Springer, 2007. p. 376–394.

DEELMAN, E. et al. Workflows and e-Science: An overview of workflow system features and capabilities. **Future Generation Computer Systems**, v. 25, n. 5, p. 528–540, 2009.

DE ROURE, D.; GOBLE, C. myExperiment – A Web 2.0 Virtual Research Environment. **In International Workshop on Virtual Research Environments and Collaborative Work Environments.**, 2007.

DIAS, J. et al. **Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow**6th Workshop on Workflows in Support of Large-Scale Science. **Anais**...: WORKS '11.Seattle, WA, USA: ACM, 2011

DIAS, J. et al. Data-centric iteration in dynamic workflows. **Future Generation Computer Systems**, v. 46, p. 114–126, maio 2015.

ELIAS, R. N.; COUTINHO, A. L. G. A. Stabilized edge-based finite element simulation of free-surface flows. **International Journal for Numerical Methods in Fluids**, v. 54, n. 6-8, p. 965–993, 2007.

FAHRINGER, T. et al. **ASKALON: a Grid application development and computing environment**6th IEEE/ACM International Workshop on Grid Computing. **Anais**...Seattle, Washington, USA: IEEE, 13 nov. 2005

FALLSIDE, D. C.; WALMSLEY, P. **XML Schema Part 0: Primer Second Edition**. Disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em: 9 mar. 2014.

FELSENSTEIN, J. PHYLIP - Phylogeny Inference Package (Version 3.2). **Cladistics**, v. 5, p. 164–166, 1989.

FOX, P. et al. Semantically-Enabled Large-Scale Science Data Repositories. In: CRUZ, I. et al. (Eds.). . **The Semantic Web - ISWC 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. v. 4273p. 792–805.

FREIRE, J. et al. Provenance for Computational Tasks: A Survey. **Computing in Science Engineering**, v. 10, n. 3, p. 11–21, 2008.

FURTADO, A. L. et al. Analysis and Reuse of Plots Using Similarity and Analogy. In: LI, Q. et al. (Eds.). . **Conceptual Modeling - ER 2008: 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 355–368.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. [s.l.] Addison-Wesley Professional, 1994.

GARIJO, D. et al. Common motifs in scientific workflows: An empirical analysis. **Future Generation Computer Systems**, v. 36, p. 338–351, jul. 2014.

GIL, Y. Workflow Composition: Semantic Representations for Flexible Automation. In: **Workflows for e-Science**. [s.l.] Springer, 2007. p. 244–257.

GIL, Y. et al. **Wings for Pegasus: Creating Large-Scale Scientific Applications Using Semantic Representations of Computational Workflows**The National Conference On Artificial Intelligence. **Anais**...Vancouver, BC, Canada: Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007

GIL, Y. et al. A semantic framework for automatic generation of computational workflows using distributed data and component catalogues. **Journal of Experimental & Theoretical Artificial Intelligence**, v. 23, n. 4, p. 389–467, 2011.

GIL, Y. **Mapping Semantic Workflows to Alternative Workflow Execution Engines**IEEE, set. 2013Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6693545>. Acesso em: 18 maio. 2014

GLATARD, T. et al. Workflow-Level Parametric Study Support by MOTEUR and the P-GRADE Portal. In: **Workflows for e-Science**. [s.l.] Springer, 2007. p. 279–299.

GODERIS, A. et al. **Seven Bottlenecks to Workflow Reuse and Repurposing**The Semantic Web – ISWC 2005. **Anais**...Galway, Ireland: 2005Disponível em: <http://dx.doi.org/10.1007/11574620_25>. Acesso em: 15 jul. 2008

GOECKS, J.; NEKRUTENKO, A.; TAYLOR, J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. **Genome Biology**, v. 11, n. 8, p. 86, 2010.

GOGRID. **Cloud Hosting, Cloud Servers, Hybrid Hosting, Cloud Infrastructure from GoGrid**. [s.l: s.n.]. Disponível em: <http://www.gogrid.com/>. Acesso em: 8 set. 2012.

GÖRLACH, K. et al. Conventional Workflow Technology for Scientific Simulation. In: YANG, X.; WANG, L.; JIE, W. (Eds.). . **Guide to e-Science**. Computer Communications and Networks. [s.l.] Springer London, 2011. p. 323–352.

GREISEN, E. W.; CALABRETTA, M. R. Representations of world coordinates in FITS. **Astronomy and Astrophysics**, v. 395, n. 3, p. 1061–1075, dez. 2002.

GUELFI, N.; MAMMAR, A. **A formal framework to generate XPDL specifications from UML activity diagrams**Proceedings of the 2006 ACM SAC. **Anais**...Dijon, France: ACM, 2006Disponível em: <http://portal.acm.org/citation.cfm?id=1141277.1141566>. Acesso em: 2 jan. 2009

GUERRA, G. et al. Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using Workflow Management Engine. **International Journal for Uncertainty Quantification**, v. 2, n. 1, p. 53–71, 2012.

HULL, D. et al. Taverna: a tool for building and running workflows of services. **Nucleic Acids Research**, v. 34, n. 2, p. 729–732, 2006.

JACOB, J. C. et al. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. **International Journal of Computational Science and Engineering (IJCSE)**, v. 4, n. 2, p. 73–87, 2009.

KANG, K. C.; LEE, J.; DONOHOE, P. Feature-oriented product line engineering. **IEEE Software**, v. 19, n. 4, p. 58–65, 2002.

KERTÉSZ, A.; SIPOS, G.; KACSUK, P. Brokering Multi-grid Workflows in the P-GRADE Portal. In: LEHNER, W. et al. (Eds.). . **Euro-Par 2006: Parallel Processing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. v. 4375p. 138–149.

LEIGH, J. W. et al. Evaluating Phylogenetic Congruence in the Post-Genomic Era. **Genome Biology and Evolution**, v. 3, n. 0, p. 571–587, 28 jun. 2011.

LEITHEAD, T. **DOM Parsing and Serialization**. Disponível em: <http://www.w3.org/TR/DOM-Parsing/>. Acesso em: 9 mar. 2015.

LIMA, E. S.; FURTADO, A. L.; FEIJÓ, B. Storytelling Variants: The Case of Little Red Riding Hood. In: CHORIANOPOULOS, K. et al. (Eds.). . **Entertainment Computing - ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29 - Ocotober 2, 2015, Proceedings**. Cham: Springer International Publishing, 2015. p. 286–300.

LIN, C. et al. **Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System**Services. **Anais**...IEEE Computer Society, 2008Disponível em: <http://portal.acm.org/citation.cfm?id=1447882>. Acesso em: 5 mar. 2010

LIN, C. et al. **A Task Abstraction and Mapping Approach to the Shimming Problem in Scientific Workflows**Proc. Services 2009. **Anais**...Bangalore, India: IEEE Computer Society, 2009Disponível em: <http://portal.acm.org/citation.cfm?id=1638036>. Acesso em: 6 dez. 2009

LINS, E. F. et al. Edge-based finite element implementation of the residual-based variational multiscale method. **International Journal for Numerical Methods in Fluids**, v. 61, n. 1, p. 1–22, 2009.

LUDASCHER, B.; ALTINTAS, I.; GUPTA, A. **Compiling abstract scientific workflows into web service workflows**Scientific and Statistical Database Management. **Anais**...Cambridge, MA: IEEE Computer Society, 2003Disponível em: <http://portal.acm.org/citation.cfm?id=1116873>. Acesso em: 5 ago. 2009

MAHESHWARI, K. et al. **Medical image processing workflow support on the EGEE grid with taverna**22nd IEEE International Symposium on Computer-Based Medical Systems, 2009. CBMS 2009. **Anais**... In: 22ND IEEE INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS, 2009. CBMS 2009. 2009

MAKOVOZ, D.; KHAN, I. **Mosaicking with MOPEX**. In: PROCEEDINGS OF ADASS XIV. Pasadena: 2004

MARINHO, A. et al. ProvManager: a provenance management system for scientific workflows. **Concurrency and Computation: Practice and Experience**, v. 24, n. 13, p. 1513–1530, 2012.

MATTOSO, M. et al. Towards Supporting the Life Cycle of Large-scale Scientific Experiments. **International Journal of Business Process Integration and Management**, v. 5, n. 1, p. 79–92, 2010.

MATTOSO, M. et al. Dynamic steering of HPC scientific workflows: A survey. **Future Generation Computer Systems**, v. 46, p. 100–113, maio 2015.

MICHENER, W. et al. Data Integration and Workflow Solutions for Ecology. In: LUDÄSCHER, B.; RASCHID, L. (Eds.). . **Data Integration in the Life Sciences**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. v. 3615p. 321–324.

MOREAU, L.; MISSIER, P.; BELHAJJAME, B. **The PROV Data Model and Abstract Syntax Notation**. [s.l: s.n.].

NAU, D.; GHALLAB, M.; TRAVERSO, P. **Automated Planning: Theory & Practice**. [s.l.] Morgan Kaufmann Publishers Inc., 2004.

NORTHROP, L. M. SEI's software product line tenets. **IEEE Software**, v. 19, n. 4, p. 32–40, 2002.

NOY, N. F. Semantic integration: a survey of ontology-based approaches. **SIGMOD Rec.**, v. 33, n. 4, p. 65–70, 2004.

OCAÑA, K. A. C. S. et al. **Optimizing Phylogenetic Analysis Using SciHmm Cloud-based Scientific Workflow**Proceedings of the 7th IEEE International Conference on e-Science (e-Science). **Anais**... In: IEEE E-SCIENCE 2011. Stockholm, Sweden: IEEE, 7 dez. 2011

OGASAWARA, E. et al. **Experiment Line: Software Reuse in Scientific Workflows** (M. Winslett, Ed.)Scientific and Statistical Database Management. **Anais**... In: SSDBM

2009. New Orleans, Louisiana, USA: Springer Berlin Heidelberg, 2009aDisponível em: <http://www.springerlink.com/content/l2142986j44gl160/>. Acesso em: 2 nov. 2011

OGASAWARA, E. et al. **Comparison and versioning of scientific workflows**Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models. **Anais**...Washington, DC, USA: IEEE Computer Society, 2009b

OGASAWARA, E. et al. An Algebraic Approach for Data-Centric Scientific Workflows. **Proceedings of the 37th International Conference on Very Large Data Bases (PVLDB)**, v. 4, n. 12, p. 1328–1339, 2011.

OGASAWARA, E. et al. Chiron: A Parallel Engine for Algebraic Scientific Workflows. **Concurrency and Computation**, v. 25, n. 16, p. 2327–2341, 2013.

OINN, T. et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. **Bioinformatics**, v. 20, n. 17, p. 3045–3054, 2004.

OLIVEIRA, D. et al. **SciCumulus: a lightweight cloud middleware to explore many task computing paradigm in scientific workflows**3rd International Conference on Cloud Computing. **Anais**... In: INTERNATIONAL CONFERENCE ON CLOUD COMPUTING. Washington, DC, USA: IEEE, 2010a

OLIVEIRA, D. et al. **GExpLine: A Tool for Supporting Experiment Composition**Provenance and Annotation of Data and Processes. **Anais**...: Lecture Notes in Computer Science.Springer Berlin / Heidelberg, 2010bDisponível em: <http://dx.doi.org/10.1007/978-3-642-17819-1_28>

OLIVEIRA, D. et al. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. **Future Generation Computer Systems**, v. 29, n. 7, p. 1816–1825, set. 2013.

OLIVEIRA, D. DE et al. A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds. **Journal of Grid Computing**, v. 10, n. 3, p. 521–552, 2012.

PLANKENSTEINER, K.; MONTAGNAT, J.; PRODAN, R. **IWIR: a language enabling portability across grid workflow systems**Proceedings of the 6th workshop on Workflows in support of large-scale science. **Anais**...: WORKS '11.New York, NY, USA: ACM, 2011Disponível em: <http://doi.acm.org/10.1145/2110497.2110509>. Acesso em: 11 mar. 2012

POHL, K.; BÖCKLE, G.; LINDEN, F. J. VAN DER. **Software Product Line Engineering: Foundations, Principles and Techniques**. 2005. ed. [s.l.] Springer, 2005.

QIN, J.; FAHRINGER, T.; PLLANA, S. UML based Grid Workflow Modeling under ASKALON. In: **Distributed and Parallel Systems**. [s.l: s.n.]. p. 191–200.

SACKS, J. et al. Design and Analysis of Computer Experiments. **Statistical Science**, v. 4, n. 4, p. 409–423, nov. 1989.

SALAYANDIA, L. et al. **Workflow-Driven Ontologies: An Earth Sciences Case Study**Proceedings of the Second IEEE International Conference on e-Science and Grid Computing. **Anais**...IEEE Computer Society, 2006Disponível em: <http://portal.acm.org/citation.cfm?id=1192571>. Acesso em: 30 maio. 2010

SALAYANDIA, L.; DA SILVA, P. On the Use of Semantic Abstract Workflows Rooted on Provenance Concepts. **Provenance and Annotation of Data and Processes**, p. 216–220, 2010.

SANTOS, E. et al. VisMashup: streamlining the creation of custom visualization applications. **IEEE transactions on visualization and computer graphics**, v. 15, n. 6, p. 1539–1546, dez. 2009.

SHIWA. **SHIWA: SHaring Interoperable Workflows for large-scale scientific simulation on Available DCIs**. [s.l.] http://www.shiwa-workflow.eu, 2011.

SHOSHANI, A.; ROTEM, D. **Scientific Data Management: Challenges, Technology, and Deployment**. 1. ed. [s.l.] Chapman and Hall/CRC, 2009.

SILVA, V.; OLIVEIRA, D.; MATTOSO, M. **Exploratory analysis of raw data files through dataflows**Proceedings of the Workshop on Parallel and Distributed Computing for Big Data Applications (WPBA 2014). **Anais**...Paris, France: 2014

STAAB, S.; STUDER, R. **Handbook on Ontologies**. 1. ed. [s.l.] Springer, 2004.

STEVENS, R. et al. **Performing in silico experiments on the Grid: a users perspective**Proceedings of the UK e-Science programme All Hands Conference. **Anais**...Nottingham, UK: 2003

SUDHOLT, W.; ALTINTAS, I.; BALDRIDGE, K. Scientific Workflow Infrastructure for Computational Chemistry on the Grid. In: ALEXANDROV, V. N. et al. (Eds.). . **Computational Science – ICCS 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. v. 3993p. 69–76.

TAYLOR, I. et al. The Triana Workflow Environment: Architecture and Applications. In: **Workflows for e-Science**. [s.l.] Springer, 2007a. p. 320–339.

TAYLOR, I. J. et al. **Workflows for e-Science: Scientific Workflows for Grids**. 1. ed. [s.l.] Springer, 2007b.

VALDES, F. G. The IRAF Mosaic Data Reduction Package. **Astronomical Data Analysis Software and Systems VII**, v. 145, 1998.

VANDERVALK, B. P.; MCCARTHY, E. L.; WILKINSON, M. D. SHARE: A semantic web query engine for bioinformatics. In: **The Semantic Web**. [s.l.] Springer, 2009. p. 367–369.

W3C. **OWL Web Ontology Language**. [s.l: s.n.].

WILDE, M. et al. Swift: A language for distributed parallel scripting. **Parallel Computing**, n. 37(9), p. 633–652, 2011.

WILKINSON, M. D.; VANDERVALK, B.; MCCARTHY, L. The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation. **Journal of Biomedical Semantics**, v. 2, n. 1, p. 8, 24 out. 2011.

WILSON JR, E. B. **An Introduction to Scientific Research**. Rev Sub ed. [s.l.] Dover Publications, 1991.

WU, S.; MANBER, U. Fast text searching: allowing errors. **Communications of the ACM**, v. 35, n. 10, p. 83–91, 1 out. 1992.

APPENDIX I – PROLOG DATABASE FOR PHYLOGENOMIC EXPERIMENT

```
selectElement([[E, true, Rule], Path]) :-
       not isSelected(E,Path),
       not isDesselected(E,Path),
       assertz(currentSelection(E,Path, true)),
       flagAsIncomplete(Path),
       rule([[E, true, Rule]|Path]), !.

selectElement([[E, true, Rule], Path]) :-
       isDesselected(E,Path),
       flagAsIncomplete(Path).

selectElement([[E, false, Rule], Path]) :-
       not isDesselected(E,Path),
       not isSelected(E,Path),
       assertz(currentDesselection(E,Path, true)),
       flagAsIncomplete(Path),
       rule([[E, false, Rule]|Path]), !.

selectElement([[E, false, Rule], Path]) :-
       isSelected(E,Path),
       flagAsIncomplete(Path).

flagAsIncomplete([[E, true, _]| Path]) :-
       retract(currentSelection(E, Path, true)),
       assertz(currentSelection(E, Path, false)), !.

flagAsIncomplete([[E, true, _]| Path]):-
       currentSelection(E, Path, false), !.

flagAsIncomplete([[E, false, _]| Path]) :-
       retract(currentDesselection(E, Path, true)),
       assertz(currentDesselection(E, Path, false)), !.

flagAsIncomplete([[E, false, _]| Path]) :-
       currentDesselection(E, Path, false), !.

flagAsIncomplete([]).

isSelected(E, [[E, true, _]|_]) :- !.

isSelected(E,[Head|Path]) :-
       isSelected(E,Path).

isDesselected(E, [[E, false, _]|_]) :- !.

isDesselected(E,[Head|Path]) :-
       isDesselected(E,Path).

ruleAlreadyUsed(Rule, [[_, _, Rule]|_]) :- !.

ruleAlreadyUsed(Rule,[Head|Path]) :-
```

```
      ruleAlreadyUsed(Rule,Path).

processResults(_) :-
      currentSelection(A, B1, true),
      append([[A, true, '']], B1, C),
      assertz(option(C)).

processResults(_) :-
       currentDesselection(A, B1, true),
       append([[A, false, '']], B1, C),
       assertz(option(C)).

removeRedundants(_) :-
      option(Path),
      option(Path2),
      not (Path = Path2),
      length(Path, Length),
      length(Path2, Length),
      isEqual(Path, Path2),
      retract(option(Path)).

isEqual([], _).

isEqual([[A, B, _]|List], List2) :-
      member([A, B, _], List2),
      isEqual(List, List2).



rule(Path) :- (not ruleAlreadyUsed('R1', Path), isSelected('A12',Path)) ->
(selectElement([['A14', false, 'R1'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R2', Path), isSelected('A14',Path)) ->
(selectElement([['A12', false, 'R2'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R3', Path), isSelected('A28',Path)) ->
(selectElement([['A29', false, 'R3'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R4', Path), isSelected('A29',Path)) ->
(selectElement([['A28', false, 'R4'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R5', Path), isSelected('A35',Path)) ->
(selectElement([['A36', false, 'R5'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R6', Path), isSelected('A35',Path)) ->
(selectElement([['A37', false, 'R6'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R7', Path), isSelected('A36',Path)) ->
(selectElement([['A35', false, 'R7'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R8', Path), isSelected('A36',Path)) ->
(selectElement([['A37', false, 'R8'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R9', Path), isSelected('A37',Path)) ->
(selectElement([['A35', false, 'R9'], Path])).

rule(Path) :- (not ruleAlreadyUsed('R10', Path), isSelected('A37',Path)) ->
(selectElement([['A36', false, 'R10'], Path])).
```

```
rule(Path) :- (not ruleAlreadyUsed('ruleA', Path), isSelected('A29',Path)) ->
(selectElement([['A12', true, 'ruleA'], Path]), selectElement([['A20', true,
'ruleA'], Path])).

rule(Path).
```

APPENDIX II – ABSTRACT WORKFLOW SCHEMA OF EXPLINE