Reproducibility Analysis of Scientific Workflows

Anna Bánáti[3], Péter Kacsuk[1,2], Miklós Kozlovszky[1,3]

[1] MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary
[2] University of Westminster, 115 New Cavendish Street, London W1W 6UW
[3] Óbuda University, John von Neumann Faculty of Informatics, Bécsi út 96/b, H-1034 Budapest, Hungary
peter.kacsuk@sztaki.mta.hu, {banati.anna,kozlovszky.miklos}@nik.uni-obuda.hu)

Abstract: Scientific workflows are efficient tools for specifying and automating compute and data intensive in-silico experiments. An important challenge related to their usage is their reproducibility. In order to make it reproducible, many factors have to be investigated which can influence and even prevent this process: the missing descriptions and samples; the missing provenance data about the environmental parameters and the data dependencies; the dependencies of executions which are based on special hardware, changing or volatile third party services or random generated values. Some of these factors (called dependencies) can be eliminated by careful design or by huge resource usage but most of them cannot be bypassed. Our investigation deals with the critical dependencies of execution. In this paper we set up a mathematical model to evaluate the results of the workflow in addition we provide a mechanism to make the workflow reproducible based on provenance data and statistical tools.

Keywords: scientific workflows; reproducibility; analytical model; provenance; evaluation; gUSE

1 Introduction

In large computational challenges scientific workflows have emerged as a widely accepted solution for performing in-silico experiments. In general, these in-silico experiments consist of series of particularly data and compute intensive jobs and in most cases their executions require parallel and distributed infrastructure (supercomputers, grids, clusters, clouds). The complexity of workflows and the continuously changing nature of the environment make it hard or even prevent to reproduce or share the results in the scientist's community. The different users for different purposes may be interested in reproducing the scientific workflow

(SWf). The scientists have to prove its results, other scientists would like to reuse the results and reviewers intend to verify the correctness of the results [13]. A reproducible workflows can be shared in repositories and can become useful building blocks that can be reused, combined or modified for developing new experiments. The workflows have to be reproducible in order to be shared or reused. Unfortunately experiences have showed that many workflows failed on occasion of a later re-execution. Zhao et al. [23] [11] investigated the main purposes of the so-called *workflow decay*, which means that year by year the ability and success of the re-execution of any workflow significantly reduces. They found four main causes which have prevented the re-execution: 1. the missing environmental parameters, 2. missing third party resources; 3. missing descriptions about the workflows; 4. the missing samples of the experiments or the inputs and outputs of the workflows.

By incorporating these results into our previous paper [2] we have deeply investigated the requirements of the reproducibility and we have given a taxonomy of the different dependencies of the execution which can interfere with a later re-execution. To sum up our conclusions, in order to reproduce an in-silico experiment the scientist community and the system developers have to face three important challenges:

1) More and more meta-data has to be collected and stored pertaining to the infrastructure, the environment, the data dependencies and the partial results of an execution in order to make us capable of reconstructing the execution in a later time even on a different infrastructure. The collected data – called provenance data – help to store the actual parameters of the environments, the partial and final data results and system variables. Concerning the provenance, the challenge is what, where and how to store the captured information.

2) Descriptions and samples have to be stored together with the workflows which are provided by the user (scientist).

3) Some services or input data can change or become unavailable during the years. For example, third party services, special local services or continuously changing databases. Scientific workflows which are established on them can become instable and non-reproducible. In addition there are computations based on random generated values (for example, in case of image processing) thus, their executions are not deterministic so these computations cannot be repeated to provide the same result in a later time. These factors – we *call dependencies of the execution* - can especially influence the reproducibility of the scientific workflows, consequently, we have performed a deeper analysis.

The first issue can be solved by capturing detailed provenance information. The second one is the responsibility of the user (scientist), however, the scientific workflow management systems (SWfMS) can and should support the scientist to

provide detailed descriptions and samples. We have dealt with this issue in one of our previous paper [1].

In this paper I deal with the third issue. Based on a provenance database we introduce a so-called descriptor-space referred to the jobs of the workflow which contains all the parameters required to reproduce the jobs. The elements of the descriptor-space we call descriptors. Every descriptor has a name, a value and a so-called decay-parameter which refer to the fluctuation of the descriptor value. A workflow can be reproducible if all the descriptor values are known and storable. However, there are descriptors which cannot be stored (for example too big input), can become unavailable in later time (for example volatile third party resources), can vary in time (for example input originated from continuously changing database). Additionally, the descriptors can be either unknown if they are based on random generated values or other operation-related system-calls. In this case, the full reproducibility is very challenging task.

By our research, we intend to make the scientific workflow reproducible by extending the scientific workflow management system (SWfMS) with an analyzer tool. With the help of the expressions of the descriptors and the decay-parameters we can perform a pre-analysis before the execution. During this phase, we can examine the jobs of a given workflow and determine whether they are reproducible or not. If not, we determine the tools and the methods which can help to reproduce the job. According to the decay-parameter, the jobs can be grouped into four groups and executed in different ways. After the execution, based on provenance data a post-analysis can be performed by the application of statistical tools. An evaluation can be computed to replace the non-reproducible parts of the workflow.

In order to achieve our goal, on one hand we have analyzed [2] the criteria of the reproducibility on the other hand we have collected and have categorized all the necessary information which are required to reproduce the scientific workflows [1]. Finally, in this paper we set up a mathematical model to formalize the problem and determine certain statistical methods to predict, evaluate or simulate the results of the jobs and the re-executed workflows. We defined the descriptor space, the decay parameter of the descriptors and the reproducible job and workflow. Based on these definitions, we set up a mathematical model of the reproducibility analysis to formalize the problem and to give our solution.

The ultimate goal of our research is to make the workflows either reproducible by eliminating the dependencies or simulating the non-reproducible jobs of the scientific workflows.

Our paper is organized as follows. In the next subsection (1.1) we introduce the WS-PGARDE/gUSE system, in which we would like to test our results. Chapter 2 gives a brief summary about the related works. Chapter 3 represents our model and the components of the reproducibility analysis. In Chapter 4 we introduce the process and the phases of the analysis. Finally, we sum up our results in 5 and in

Chapter 6 we conclude our research with a brief provisioning of possible future research directions.

1.1. WS-PGRADE/gUSE

gUSE (grid and cloud user support environment) is a well-known and permanently improving open source science gateway framework developed by the Laboratory of Parallel and Distributed Systems (LPDS) that enables users the convenient and easy access to grid and cloud infrastructures. It has been developed to support a large variety of user communities. It provides a generic purpose, workflow-oriented graphical user interface to create and run workflows on various Distributed Computing Infrastructures (DCIs) including clusters, grids, desktop grids and clouds. [20]

The WS-PGRADE Portal [21] [10] is a web based front end of the gUSE infrastructure. The structure of WS-PGRADE workflows are represented by directed acyclic graphs.

The nodes of the graph, namely the jobs are the smallest units of a workflow. They represent a single algorithm, a stand-alone program or a web-service call to be executed. Ports represent input and output connectors of the given job node. Directed edges of the graph represent data dependency (and corresponding file transfer) among the workflow nodes. This abstract workflow can be used in the second step to generate various concrete workflows by configuring detailed properties (first of all the executable, the input/output files where needed and the target DCI) of the nodes representing the atomic execution units of the workflow.

A job may be executed if there is a proper data (or dataset in case of a collector port) at each of its input ports and there is no prohibiting programmed condition excluding the execution of the job. The execution of a workflow instance is data driven forced by the graph structure: A node will be activated (the associated job submitted or the associated service called) when the required input data elements (usually file, or set of files) become available at each input port of the node.

2 State of the Art

The researchers dealing with the reproducibility of scientific workflows have to approach this issue from two different aspects. First, the requirements of the reproducibility have to be investigated, analyzed and collected. Secondly, techniques and tools have to be developed and implemented to help the scientist in creating reproducible workflows.

2.1. Requirements

Researchers of this field agree on the importance of the careful design [8],[15], [16], [17], [22] which on one hand means the increased robustness of the scientific code, such as modular design and detailed description about the workflow, about the input/output data examples and consequent annotations [7]. On the other hand, the careful design includes the careful usage of volatile third party or special local services.

Groth et al. [10] based on several use cases analyzed the characteristics of applications used by workflows and listed seven requirements in order to enable the reproducibility of results and the determination of provenance. In addition, they showed that a combination of VM technology for partial workflow re-run along with provenance can be useful in certain cases to promote reproducibility.

Davison [7] investigated which provenance data have to be captured in order to reproduce the workflow. He listed six vital areas such as hardware platform, operating system identity and version, input and output data etc.

Zhao et al. [23] in their paper investigated the cause of the so called workflow decay. They examined 92 Taverna workflows submitted in the period between 2007 and 2012 and found four major causes: 1) Missing volatile third party resources 2) Missing example data 3) Missing execution environment (requirement of special local services) and 4) Insufficient descriptions about workflows. Hettne et al. [11] in their papers listed ten best practices to prevent the workflow decay.

2.2. Techniques and Tools

There are existing available tools, VisTrail, ReproZip or PROB [5], [9], [14] which allow the researcher and the scientist to create reproducible workflows. With the help of VisTrail [9], [12] reproducible paper can be created, which includes not only the description of scientific experiment, but all the links for input data, applications and visualized output. These links always harmonize with the actually applied input data, filter or other parameters. ReproZip [5] is another tool, which stitches together the detailed provenance information and the environmental parameters into a self-contained reproducible package.

The Research Object (RO) approach [3], [6] is a new direction in this research field. RO defines an extendable model, which aggregates a number of resources in a core or unit. Namely a workflow template; workflow runs obtained by enacting the workflow template; other artifacts which can be of different kinds; annotations describing the aforementioned elements and their relationships. Accordingly to the RO, the authors in [4] also investigate the requirements of the reproducibility and the required information necessary to achieve it. They created ontologies, which help to uniform these data. These ontologies can help our work and give us a basis

to perform our reproducibility analysis and make the workflows reproducible despite their dependencies.

Piccolo et al [18] collected the tools and techniques and proposed six strategies which can help the scientist to create reproducible scientific workflows.

Santana-Perez et al [19] proposed an alternative approach to reproduce scientific workflows which focused on the equipment of a computational experiment. They have developed an infrastructure-aware approach for computational execution environment conservation and reproducibility based on documenting the components of the infrastructure.

To sum up the results mentioned above, we can conclude that the general approach is that the scientist has to create reproducible workflows with careful design, appropriate tools and strategies. But none of them intended to solve the problem related to the dependencies rather they suggested to bypass them. Moreover, they did not deal with the following question: How an existing workflow can be made reproducible?

2.3. Reproducibility Support in WS-PGRADE/gUSE System

In the WS-PGRADE/gUSE system with the help of the "RESCUE" feature the user has the possibility to re-execute a job which does not own all the necessary inputs but the provenance data is available from the previous executions. (Fig. 1)

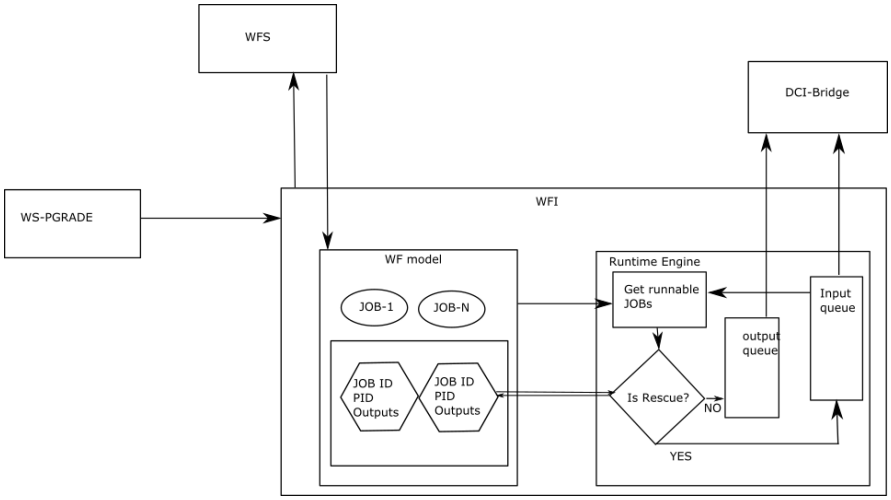

Figure 1

Operation of the *Rescue* feature in the WS-PGRADE/gUSE system

When submitting a job which has the identifier originated from the previous execution, the workflow instance (WFI) queries the description file of the workflow. This XML file includes the jobs belonging to the workflow. Their input

and output ports, their relations and the identifiers of the job instances executed previously with their outputs. After processing the XML file, a workflow model is created in the memory representing the given workflow during its execution. At this point the Runtime Engine (RE) takes over the control to determine the "ready to run" jobs then it examines whether these jobs have already stored outputs originated from previous executions. Concerning the answer the RE puts the job in the input or in the output queue.

3 Reproducibility Analysis

In this section, we introduce our mathematical model of reproducibility analysis. Next we give a method to handle the influence factors of the reproducibility of a job and to make the non-reproducible job reproducible under certain conditions or by a given probability. Finally we deal with jobs applying random generated values in an independent subsection.

3.1. The Model

In order to formalize the problem let us introduce the following notations and definitions:

- The scientific workflow (SWf) can be represented by a directed acyclic graph, where the vertices denote the jobs and the edges denote the dataflow between jobs.

  $\boldsymbol{V} = \{\boldsymbol{J_1}, \dots, \boldsymbol{J_N}\}$, where $\boldsymbol{N} \in$ N; the number of the job of a given workflow

  $$\boldsymbol{E} = \{(\boldsymbol{J_i}, \boldsymbol{J_j}) \in \boldsymbol{V} \times \boldsymbol{V} | \boldsymbol{i} \in [\boldsymbol{1}, \boldsymbol{2}, \dots \boldsymbol{N} - \boldsymbol{1}]; \boldsymbol{j} \in [\boldsymbol{2}, \boldsymbol{3}, \dots, \boldsymbol{N}] \; \boldsymbol{and} \; \boldsymbol{i} \neq \boldsymbol{j}\}$$

- The job J_i is *exit job* (exit node) in the graph, if $\nexists \boldsymbol{J_j} \in \boldsymbol{V}$: $(\boldsymbol{J_i}, \boldsymbol{J_j}) \in \boldsymbol{E}$; Notation: J_{exit}
- The job J_{i} is *entry job* (entry node) in the graph, if $\nexists \boldsymbol{J_j} \in \boldsymbol{V}$: $(\boldsymbol{J_j}, \boldsymbol{J_i}) \in \boldsymbol{E}$; Notation: J_{entry}
- The job, which is neither exit nor entry job, is an *inside job.*
- The *forward sub-workflow of a job* J_i is the part of the workflow, which contains all the successor jobs (nodes) and the edges between them.
- From our point of view the SWf is a function: $\mathbf{SWF}(t_0, J_1, J_2, \dots, J_N) = \mathbf{Y}$, where t_0 is a given time of the submission of the workflow and Y is the result of the workflow.

- Assuming that the workflow was successfully executed at least once and provenance database is available a descriptor-space $\boldsymbol{D_{J_i}}$ can be created to store all the necessary parameter needed to re-execute the job.
- The job J_i (i = 1, 2, …, N) has K_i descriptors: V_1, V_2, …, V_{Ki} , which are necessary to reproduce the workflows. The values of descriptors are: $\boldsymbol{D_{J_i}} = \{\boldsymbol{v_{i1}}, \boldsymbol{v_{i2}}, \dots, \boldsymbol{v_{iK_i}}\}$
- With the help of the descriptors every job can be written as a function: $\boldsymbol{J_i(t_0, v_{i1}, v_{i2}, \dots, v_{iK_i}) = Y_i}$
- For every descriptor we have defined a so called *decay-parameter* which indicates how the descriptor's value changes in time. There are four cases:
    1. The availability and the value of the descriptor is not changing in time. In this case the decay parameter is 0.
    2. The availability of the descriptor is changing in time. There are two cases: the probability distribution function of the descriptor's availability is known or not. In the first case, the decay parameter can be determined by the given distribution function and in the second one, the descriptor value is infinite.
    3. The value of the descriptor is changing in time. Similarly to the second item, the change of the value can be known or unknown. According to the actual case the value of decay parameter is a function describing the change or it is infinite.
    4. The value of the descriptor is not constant, but both its availability and change is unknown. For example a random generated value, which is used during the execution but it is not known. In this case the value of the decay-parameter is infinite.

In formal:

$$\boldsymbol{decay(v_i)} = \begin{cases} \mathbf{0}, & \text{if the value of the descriptor is not changing in time} \\ \infty, & \textbf{if the value of the descriptor is unknown} \\ \boldsymbol{F_i(t)}, & \textbf{distribution function} \text{ of the availability of the given value} \\ \boldsymbol{Vary_i(t,\, v_i)}, & \textbf{if the value of the descriptor is changing in time} \end{cases}$$

With help of these expressions we can define the reproducibility as the following way:

Definition (D1): The J_i job is *reproducible*, if the descriptor space $D_{J_i} = \{v_{i1}, v_{i2}, \dots, v_{iK_i}\}$ of job – which contains all the inputs and environmental parameters - is known and can be stored, in other words all the decay parameters are zero.

Notation: J_i^{repro}; $JOB_i^{repro}(v_{i1}, v_{i2}, \dots, v_{iK_i}) = Y_i$

Corollary: A reproducible job is invariable in time (time-independent):

$\boldsymbol{JOB_i^{repro}(t_0, v_{i1}, v_{i2}, \dots, v_{iK_i}) = JOB_i^{repro}(t_0 + \Delta t, v_{i1}, v_{i2}, \dots, v_{iK_i}) = Y_i}$ for every Δt.

Definition (D2): The *scientific workflow* is *reproducible*, if its exit jobs and the $SubWF_{J_{exit}}^{back}$ of the exit jobs is reproducible.

It can be easily proven, that if and only if every job of a SWf is reproducible, then the SWf is also reproducible.

We introduce other properties, namely the substitutional and the approximative reproducibility referring to that case, in which the decay parameter of one of its descriptors changes in time and this variation is known. There is two option: the first one is that the variation of result can be described with a function determined by the variation function of the descriptor; the second one is that the variation of result can be estimated. In formal:

Definition (D2): The J_i job is *reproducible by substitution*, if the descriptor space $\{v_{i1}, v_{i2}, \dots, v_{iK_i}\}$ of job and $\exists k \in [1, 2, \dots, K_i]: Vary_{ik}(\Delta t, v_{ik})$ is known, and based on vary function a $Vary_i^*(\Delta t, Y_i)$ can be unambiguously determined.

If

$$JOB_i(t_0, v_{i1}, v_{i2}, \dots, v_{iK_i}) = Y_i$$

Then

$$JOB_i(t_0 + \Delta t, v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots, v_{iK_i}) = Vary_i^*(\Delta t, Y_i)$$

Notation: $JOB_i^{vary}(v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots\ v_{iK_i})$

Definition (D3): The J_i job is *approximately reproducible*, if J_i is reproducible under condition that $\exists k \in [1, 2, \dots, K_i]: Vary_{ik}(\Delta t, v_{ik})$ is precisely known, and in accordance this function a $Vary_i^*(\Delta t, Y_i)$ can be estimated with an acceptable accuracy:

$$JOB_i(t_0 + \Delta t, v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots, v_{iK_i}) \approx Vary_i^{appro}(\Delta t, Y_i) = \widetilde{Y}_i$$

Notation: $\boldsymbol{JOB_i^{appro}(v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots\ v_{iK_i})}$

3.2. Pre-Analysis

The first step of the process of the reproducibility analysis is to create the descriptor space of all the jobs belonging to the given scientific workflow. The descriptors and their decay parameters can originate from three different sources: from the users, from the provenance database and it can be automatically generated by the SWfMS. [1]

Analyzing the decay parameters of the descriptors we have separated those, which can influence the reproducibility of the workflow in other words which have non-zero decay parameters. Four groups have been created:

1. With the help of additional resources or tools this dependency of execution can be eliminated. For example, in case of random generated values we are going to implement an operating system level tool, which captures the return value of the random generator, and stores it in the provenance database (see subsection 3.4)
2. With the help of approximation tools the value of the descriptor can be evaluated or even replaced. (see subsection 3.3)
3. A time interval can be given during which the descriptor is available by a given probability *p*.
4. There is no method to make the workflow reproducible.

3.3. Evaluation

In this subsection we investigate the case when one decay parameter of the job's descriptors is changing in time.

In case of the presented methods we assume two essential conditions:

1. The availability of the whole descriptor's space of the job in a given SWf, which means all the necessary information to reproduce the job.
2. The availability of a provenance database which contains the provenance information about the previous executions of a given SWf. For example, descriptor values, partial and final results of the jobs etc.

Based on provenance database a sample set can be defined which contains provenance data originated from s (where s is a natural number) previous executions:

$$S = \begin{Bmatrix} J_i\left(t_0, v_{i1}^0, v_{i2}^0, \dots, v_{iK_i}^0\right) = Y_i^0 \\ J_i\left(t_0, v_{i1}^1, v_{i2}^1, \dots, v_{iK_i}^1\right) = Y_i^1 \\ \dots \\ J_i\left(t_0, v_{i1}^{s-1}, v_{i2}^{s-1}, \dots, v_{iK_i}^{s-1}\right) = Y_i^{s-1} \end{Bmatrix} \quad (1)$$

If a v_{ij} ($i = 1 \ldots N$; $j = 1 \ldots K_i$) descriptor's value is not changing in time its decay parameter is 0, thus, in the sample set the given elements are equals:

$v_{ij}^{0} = v_{ij}^{1} = \ldots = v_{ij}^{s-1}$

Assuming that only one descriptor value is changing in time the sample set related to the job J_i can be written in a simpler form:

$$S = \{(t_0, v_{ij}^0, Y_i^0), (t_0, v_{ij}^1, Y_i^1), \ldots, (t_0, v_{ij}^{s-1}, Y_i^{s-1})\} \quad (2)$$

Based on the sample set $S = i$ the correlation can be investigated between the variables v_{ij}^{k} an Y_i^k (k =1, 2, … , s-1) with the? help of the following expression:

$$corr(v, Y) = \frac{\sum_{k=0}^{s-1}(v_{ij}^k - \widetilde{v_{ij}})(Y_i^k - \widetilde{Y_i})}{\sqrt{\sum_{k=0}^{s-1}(v_{ij}^k - \widetilde{v_{ij}})^2 \sum_{k=0}^{s-1}(Y_i^k - \widetilde{Y_i})^2}}, \quad (3)$$

where $\widetilde{Y_i}$ and $\widetilde{v_{ij}}$ are the empirical (sample) mean of the adequate variables.

In addition, based on provenance data we can determine the coverage of a given descriptor, which contains every job influenced by this descriptor. We can compute the correlation matrix of the $v_{i,j}$ descriptor and the results of all the successors of the job J_i.

$$\mathbf{R}_{(p+1)\times(p+1)} = \begin{pmatrix} cor(v_{i,j}, v_{i,j}) & cor(v_{i,j}, Y_1) & \ldots & cor(v_{i,j}, Y_p) \\ cor(Y_1, v_{i,j}) & cor(Y_1, Y_1) & & cor(Y_1, Y_p) \\ \\ cor(Y_p, v_{i,j}) & cor(Y_p, Y_1) & & cor(Y_p, Y_p) \end{pmatrix} \quad (4)$$

where $Y_1, Y_2 \ldots, Y_p$ is the results of the successors of job $J_{i.}$. The R matrix is symmetric and the values in the diagonal are 1.

The coverage of the given descriptor can be determined based on the first row of the correlation matrix. The non-zero values, which are close to 1 can show which J_i, i = 1, 2, …, p belong to the coverage zone.

Concerning to the value of the expression (3) we can differentiate two cases:

1. The result is close to 1, which means that the two variables are bounded? up with each other thus the result Y_i. can be evaluated by applying some approximation. For example, the linear regression consequently, the result Y_i. can be written as a linear combination of the changing descriptor.

   $$Y_i = \beta_0 + \beta_1 v_{ij} \quad (5)$$

   where the $\beta_{0 \text{ and}} \beta_1$ are the linear coefficients.

   In this way, $Y_i^t = \beta_0 + \beta_1 vary_{ij}(t, v_{ij})$, where t is arbitrary.

   If the result of (3) is closer to 0.5 then to 1, nonlinear regression or other curve fitting method can be used.

Storing the approximation and the final results in the repository makes it possible that during the re-execution of a workflow, the non-reproducible job can be replaced by these approximated or simulated results.

we call The scientific workflows associated to this group reproducible by substitution or approximately reproducible scientific workflows.

2. The result of the correlation coefficient is close to 0, which means that the descriptor v_{ij} does not influence the result Y_i. In this case, the analysis has to be continued and the correlations between the results of the successor jobs have to be investigated.

3.4. Random Based Dependency

Many jobs use applications and computations which are based on random generated values (RGV). For example, the image processing applications, the different simulators and workflows which simulate some physical or chemical phenomena or even cryptographic algorithms. In this case, during the execution a system call is performed which returns a random generated value but this result is stored only in the memory. Consequently, provenance information does not get into the provenance database. We have designed a tool which operates at the operating system level and it captures the return values of the system call. Next, it stores the given value in the provenance database or on a predefined location. With the help of this tool the random RGVs can be stored together with the workflow in a repository. In/on? occasion of a later re-execution, the SWfMS uses the originally stored value instead of the newly generated random value.

4 The Process of Reproducibility-Analysis

Based on the decay-parameter (DP) the pre-analyzer performs a classification of the jobs of the given SWf. Depending on the classification, the job can be executed in three ways:

1. Standard execution, if all the decay parameters are zero.
2. Replacing the execution with evaluation, if there are changing descriptor values or the availabilities are defined by a probability distribution function (PDF).
3. Execution with random value capture (RVC) tool, if the execution of the job is based on random generated value.

In all cases updating the Provenance Database (PDB) is performed occasionally by extra provenance information (for example a random value).

Based on the PDB the post-analyzer creates a sample set. The evaluator module computes the evaluated output of the given job. Figure 2 shows the flow-chart about the process and Figure 3 presents the block-diagram.

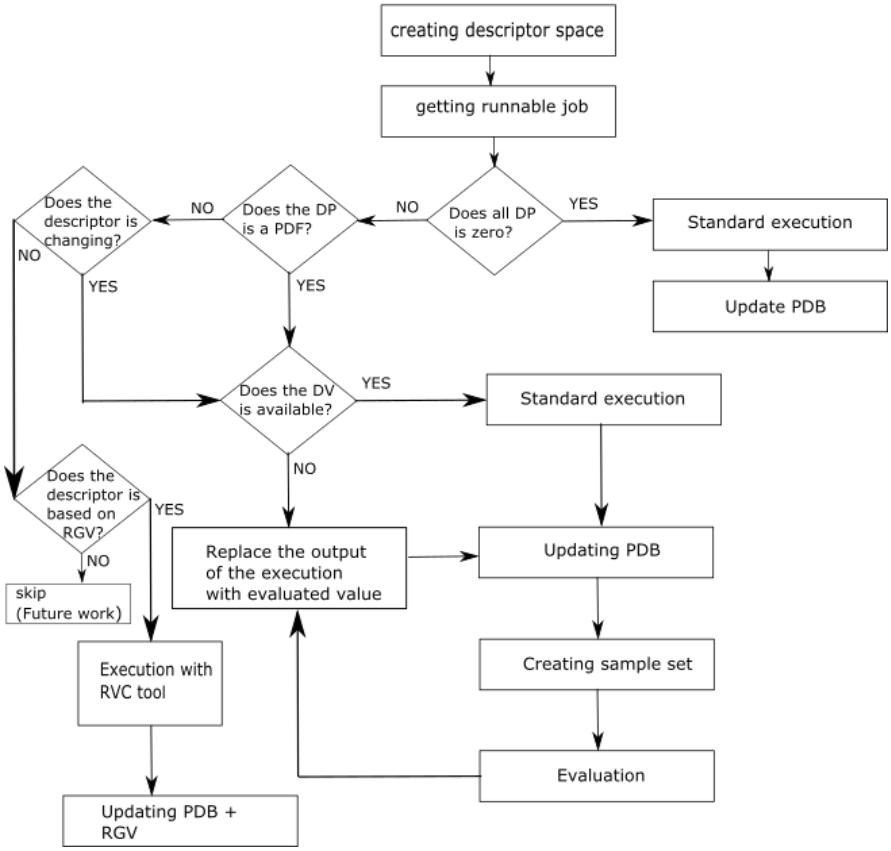

Figure 2

The flow-chart of the process of the reproducibility analysis

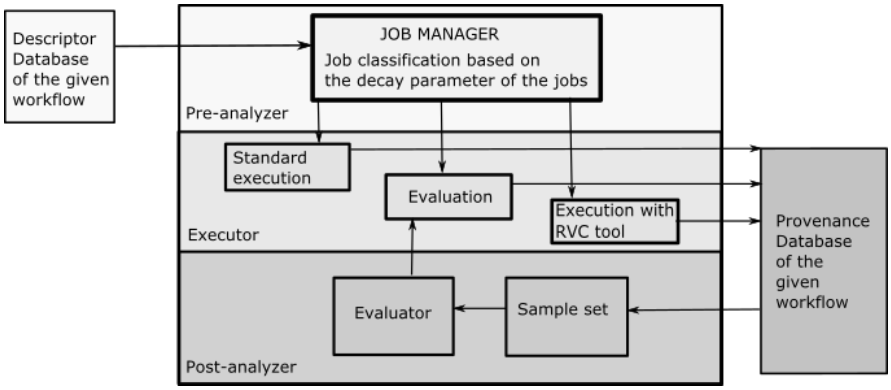

Figure 3

The block-diagram of the system of the reproducibility analysis

5 Results and Implementation

Since this investigation is based on the descriptor-space and the descriptor-space is based on the provenance database, the first step toward the implementation of an evaluating tool is the implementation of a provenance framework. The implementation of a Provenance manager (PROV-man) framework is already finished. It provides functionalities to create and manipulate provenance data in a consistent manner and ensures its permanent storage. It also provides a set of interfaces to serialize and export provenance data into various data format, serving interoperability [24], [25], [26]. Three main components constitutes the PROV-man framework:

- A set of methods to build and manipulate provenance data, while preserving full compliance with the PROV specifications
- A set of interfaces for provenance data sharing and interoperation. These interfaces covers serialization to formats of the PROV family of documents (e.g. XML, RDF, DC, etc.) and other specifically required format (e.g. Graphviz, PDF, JPG, etc.)
- A relational database that serves as a main repository for storing provenance data, reflecting the PROV-man data model

Additionally, the “rescue” feature and the tool which captures the RGVs and stores them in the PDB or in a predefined location are implemented in the WS-PGRADE/gUSE system. Furthermore, we intend to extend it with an evaluating tool in case the job cannot run because of the missing or unavailable inputs. This solution makes us able to apply not only the previously stored results but an evaluated or a simulated output of a previously executed job.

Conclusions

We analyzed the requirements of the reproducibility and the critical, continuously changing or non-deterministic descriptors of the scientific workflows to make them reproducible. To formalize the problem, we set up a mathematical model and gave definitions of the reproducible jobs and workflows. Based on the model, we worked out a reproducibility analysis process which involves three phases. The first is a pre-analysis based on the descriptor's space to determine the reproducible parts of the workflow and to classify the jobs according to their decay-parameter. The jobs in the different classes are executed in different ways. In the post-analysis phase, assuming that provenance data is available about the previous executions a sample set is created and the determination of the evaluating algorithm is performed. This information is stored together with the workflow in the repository or in the provenance database. On occasion of a re-execution of the workflow and in case of a non-reproducible job, instead of the standard execution we evaluate the outputs based on the stored sample set and on the evaluating parameters.

The presented framework is theoretical in the sense that the time is limitless. If the probability distribution function of the availability referring to a descriptor is given or can be estimated based on provenance, the limit of the function as time approaches infinity is 1, and the time – during the descriptor is available – can be determined, theoretically. If an estimating method can be determined for a changing descriptor, it is also "time-limitless" and the method can be applied at any time, maybe if the appropriate resources is out of time. However, the experience actually shows, that the technological development can be prevent the re-execution and can shorter the theoretical time given by our analysis.

In addition, we particularly have dealt with the job executions based on random generated values and we have developed a mechanism which is able to capture the return values of the system calls and to store it for a later re-execution.

In our future work, we would like to develop other tools to be able to handle more special dependencies of the workflow execution. Also we intend to explore other procedures to find a more general solution for the evaluating problems when many descriptors' values change, simultaneously. Furthermore, we plan to implement our methods and tools within the gUSE framework.

Acknowledgement

This work was supported by EU project SCI-BUS (SCIentific gateway Based User Support). The SCI-BUS project aims to ease the life of the e-Scientists by creating a new science gateway customization methodology based on the generic-purpose gUSE/WS-PGRADE portal family.

References

[1] Bánáti A., Kacsuk P., Kozlovszky M.: Minimal Sufficient Information about the Scientific Workows to Create Reproducible Experiment. In: IEEE 19th International Conference on Intelligent Engineering Systems (INES), Slovakia, 2015, pp. 189-194

[2] Bánáti A., Kacsuk P. Kozlovszky, M.: "Four Level Provenance Support to Achieve Portable Reproducibility of Scientific Workflows" In Information and Communication Technology, Electronics and Microelectronics (MIPRO) 2015 38th International Convention on (pp. 241-244) IEEE

[3] Bechhofer S., De Roure D., Gamble M., Goble C., Buchan I.: „Research Objects: Towards Exchange and Reuse of Digital Knowledge" In: he Future of the Web for Collaborative Science, 2010

[4] Belhajjame K., Zhao J., Garijo D., Gamble M., Hettne K., Palma R., Goble C.: „Using a Suite of Ontologies for Preserving Workow-Centric Research Objects" In: Web Semantics: Science, Services and Agents on the World Wide Web, 2015

[5] Chirigati F., S., Shasha D., Freire J.: „ReproZip: Using Provenance to Support Computational Reproducibility" Presented as part of the 5th USENIX Workshop on the Theory and Practice of Provenance, 2013

[6] Corcho O., Garijo Verdejo D., Belhajjame K., Zhao J., Missier P., Newman D., Goble C.: „Workflow-Centric Research Objects: First Class Citizens in Scholarly Discourse" In: Proceedings of Sepublica 2012, pp. 1-12, 2012

[7] Davison, A. „Automated Capture of Experiment Context for Easier Reproducibility in Computational Research", Computing in Science & Engineering, Vol 14/ 4, pp. 48-56, July 2012

[8] De Roure D., Belhajjame K., Missier P., Gmez-Prez J. M., Palma R., Ruiz J. E., Hettne K., Roos M., Klyne G., Hekkelman M. L.: „Towards the Preservation of Scientific Workows". In Proceedings of 8th International Conference on Preservation of Digital Objects (iPRES 2011) 2011

[9] Freire J., Koop D., Chirigati F. S, and Silva C. T.,: "Reproducibility Using VisTrails", Implementing Reproducible Research 33, 2014, OnlineAvailable: http://citeseerx.ist.psu.edu/viewdoc/download doi:10.1.1.369.9566

[10] Groth, P; Deelman E., Juve G., Mehta G., and Berriman B., „Pipeline-Centric Provenance Model", in Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, 2009, p. 4

[11] Hettne, K. Wolstencroft, K. Belhajjame, Goble C. A., Mina E., Dharuri H., D. De Roure, Verdes-Montenegro L., Garrido J., and M. Roos, „Best Practices for Workflow Design: How to Prevent Workflow Decay", in SWAT4LS, 2012

[12] Koop D., Freire J., and Silva C. T., „Enabling Reproducible Science with VisTrails", arXiv preprint arXiv:1309. 1784, 2013

[13] Koop, D., Santos, E., Mates, P., Vo, H. T., Bonnet, P., Bauer, B., ... and Freire, J. (2011) "A provenance-based Infrastructure to Support the Life Cycle of Executable Papers". Procedia Computer Science, 4, 648-657

[14] Korolev, A. Joshi, V. Korolev, M. A. Grasso, A. Joshi, M. A. Grasso, D. Dalvi, S. Das, V. Korolev, Y. Yesha, and others, „PROB: A Tool for Tracking Provenance and Reproducibility of Big Data Experiments.", Reproduce'14. HPCA 2014, Vol. 11, pp. 264-286, 2014

[15] Mesirov J. P., „Accessible Reproducible Research", Science, Vol. 327/ 5964, pp. 415-416, January 2010

[16] Missier P., Woodman S., Hiden H., and P. Watson, „Provenance and Data Differencing for Workflow Reproducibility Analysis", Concurrency and Computation: Practice and Experience, 2013

[17] Peng, „Reproducible Research in Computational Science", Science, vol. 334/ 6060, pp. 1226-1227, 2011

[18] Piccolo S. R., Lee A. B., Frampton M. B.: „Tools and Techniques for Computational Reproducibility". In: bioRxiv, Vol. 022707, 2015

[19] Santana-Perez I., Prez-Hernndez M. S.: „Towards Reproducibility in Scientific Workows: An Infrastructure-based Approach". In: Scientific Programming, Vol. 2015, p. 11, 2015

[20] SZTAKI LPDS: User's Guide. Http://guse.hu/about/home

[21] SZTAKI LPDS: User's Guide Http://sourceforge.net/projects/guse/_les/3.7.4/Documentation

[22] Woodman, Hiden H., Watson P., and Missier P., „Achieving Reproducibility by Combining Provenance with Service and Workflow Versioning", in Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science, 2011, pp. 127-136

[23] Zhao J., Gomez-Perez J. M., Belhajjame K., Klyne G., Garcia-Cuesta E., Garrido A., Hettne K., Roos M., De Roure D., and Goble C., „Why Workflows Break—Understanding and Combating Decay in Taverna Workflows", in E-Science (e-Science), 2012 IEEE 8th International Conference on, 2012, pp. 1-9

[24] http://nl.sharp-sys.com/PROV-man.html

[25] Kiss, Tamás, et al. "Ws-pgrade/guse in European Projects." Science Gateways for Distributed Computing Infrastructures. Springer International Publishing, 2014, 235-254

[26] Benabdelkader, A., Antoine AHC van Kampen, and Silvia D. Olabarriaga. PROV-Man: A PROV-Compliant Toolkit for Provenance Management. No. e1347. PeerJ PrePrints, 2015