Óbuda University

PhD Thesis

# Reproducibility analysis of the scientific workflows

Anna Bánáti

Supervisors:
Péter Kacsuk, Phd, Prof.
Miklós Kozlovszky, Phd

Doctoral School of Applied Informatics

Budapest, 2016.

# Statement

I, Anna Bánáti, hereby declare that I have written this PhD thesis myself, and have only used sources that have been explicitly cited herein. Every part that has been borrowed from external sources (either verbatim, or reworded but with essentially the same content) is unambiguously denoted as such, with a reference to the original source.

# Abstract

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 successive steps of an experiment are chained to a so called scientific workflow, which can be represented by a directed acyclic graph (DAG). The execution of these scientific workflows – depending on the field of science – can take a very long time, weeks or even months. The complexity of workflows and the continuously changing nature of the environment can hide the details of the execution, the partial results and the intermediate computations, and even the results of the execution of the same workflow can be different.

In order to repeat or reproduce a scientific workflow the scientist and also the SWfMS developers have to face several challenges. On one hand many workflows are based on special hardware or software with the appropriate settings, or third party resources which create dependencies of the execution. These dependencies have to be handled or even eliminated with tools developed for this purpose. On the other hand, the ancestry of the results may raise problems when someone wants to reuse the whole or a part of the workflow. To conserve this information rich provenance data have to be collected during the execution.

In this dissertation I deal with the requirements and the analysis of the reproducibility. I set out methods based on provenance data to handle or eliminate the unavailable or changing descriptors in order to be able reproduce an – in other way – non-reproducible scientific workflow. In this way I intend to support the scientist's community in designing and creating reproducible scientific workflows.

In the first two thesis groups I introduced the mathematical model of the reproducibility analysis, I investigated and proved the behavior of the changing descriptors referred to the jobs which can influence the reproducibility. In addition I presented methods to determine the coverage of the descriptors, the reproducible part of the workflow and the probability of the reproducibility. In the third thesis group I introduced two metrics of the reproducibility and I present algorithms to evaluate these metrics in polynomial time. Finally I classify scientific workflows from a reproducibility perspective.

# Kivonat

Napjainkban, a tudományos világban folytatott tudományos kísérletek egyre növekvő, hatalmas adathalmazokra épülnek, melyek feldolgozása és a rajtuk végzett számítások a hagyományos laboratórium adta lehetőségeket messzemenően meghaladják. Ennek következtében a tudós közösségek körében egyre népszerűbbé és nélkülözhetetlenebbé válnak az un. „in-silico" (számítógépeken végrehajtott) kísérletek, melyek futtatása párhuzamos és elosztott infrastruktúrákat igényelnek, mint a számítási rácsok (grid), fürtök (cluster) vagy egyre inkább a felhők (cloud). A kísérletek egyes lépéseinek láncba fűzésével un. tudományos munkafolyamatok jönnek létre, melyek futtatása - tudományterülettől függően - hetekig vagy akár hónapok is tarthat. A fentebb említett infrastruktúrák különbözőségéből és a folyamatosan változó természetükből fakadóan azonban a futás részletei, vagy akár a közbülső számítások és részeredmények is rejtve maradhatnak, sőt, két különböző végrehajtás eredményei eltérhetnek egymástól. A tudományos munkafolyamatok reprodukálhatóságának biztosítása érdekében, a tudós társadalomnak, - mint felhasználóknak - és a munkafolyamatokat futtató, kezelő rendszerek (Scientific Workflow Management system) fejlesztőinek két nagy kihívással kell szembenézniük: Egyrészt a munkafolyamatok végrehajtása gyakran speciális hardver/szoftver elemeken vagy harmadik féltől származó erőforrásokon alapszik, amelyek rendelkezésre állása megkérdőjelezheti egy újra futtatás sikerességét. Ennek megoldására olyan eszközöket és módszereket kell fejleszteni, melyek kezelik vagy esetleg megkerülik ezeket a függőségeket. Másrészről az eredmények eredetének nyomon követhetőségét biztosítani kell. Ennek érdekében, a munkafolyamat-kezelő rendszerek un. provenance adatokat gyűjtenek az adatfüggőségekről, a részeredményekről, a környezeti változókról valamint a rendszer beállításairól és paramétereiről.

Jelen kutatásban a tudományos munkafolyamatok reprodukálhatóságának feltételeivel és elemzésével foglalkoztam provenance adatok felhasználásával, továbbá a tudós társadalom támogatása céljából megoldási lehetőségeket kerestem az egyébként nem reprodukálható tudományos munkafolyamatok reprodukálhatóvá tételével kapcsolatban. A módszerek az elérhetetlenné váló és változó deszkriptorok kiküszöbölését és kompenzálását kezelik.

Az első két téziscsoportban a bevezetett matematikai modell építőelemeit definiáltam és vizsgáltam, nevezetesen a számítási feladatok reprodukálhatóságát meghatározó deszkriptorok romlási mutatóját, változásainak természetét, kapcsolatát és az eredményre vonatkozó hatását. Továbbá eljárást dolgoztam ki a deszkriptorok hatósugarának és a tudományos munkafolyamatok reprodukálható részének meghatározására, valamint a reprodukálhatóság valószínűségének kiszámítására. A harmadik téziscsoportban a reprodukálhatóság mértékeit definiálom és polinomiális lépésszámú algoritmust mutatok be a mértékek becslésére. Végezetül a tudományos munkafolyamatokat osztályoztam reprodukálhatósági szempontból.

# Content

# List of figures

# List of tables

# List of abbreviations

| Abbreviation | Meaning |
|---|---|
| ARC | Average Reproducibility Cost |
| DAG | Directed Acyclic Graph |
| HPC | High Performance Computing Infrastructures |
| NRP | Non-reproducibility probability |
| OPM | Open Provenance Model |
| PDB | Provenance Database |
| RGV | Random Generated Values |
| RBF | Radial Basis Function |
| RO | Research Object |
| SWf | Scientific Workflow |
| SWfMS | Scientific Workflow Management System |
| VM | Virtual Machine |
| W3C | World Wide Web Consortium |
| WFLC | Workflow Life Cycle |

ACKNOWLEDGEMENT

Firstly, I would like to thank my supervisors, Miklós Kozlovszky and Péter Kacsuk for being tolerant, and always helpful as a colleague and supervisor. Their guidance in general and technical suggestions in particular have really guided me during this study. Their calm and friendly demeanour allowed me to discuss and share ideas with him at any time.

Furthermore, I am thankful to Dr János Levendovszky, my first supervisor at Budapest University of Technology and Economics who showed me the beauty of the research and introduced me to network reliability.

Moreover, I would like to thank my friend and colleague, Eszter Kail who accompanied me along this struggling way.

Without the support of parents and family, this long period of research would not have been possible. The support I received from my mother cannot be described in words. My husband whose patience and care allow me to complete this work and our children who had to miss me so long.

# 1 INTRODUCTION

## 1.1 Scientific experiments – *In vivo, In vitro, In situ, In silico*

During the last decade, scientific workflows have emerged as a widely-accepted solution for performing *in silico* experiments for large computational challenges. The traditional scientific experiments are conducted on living organisms, called *in vivo* (Latin: "within the living"), in the nature, called *in situ* (Latin: locally, on site) or in laboratories, called *in vitro* (Latin: in glass) experiments. During *in vivo* experiments, the effects of various biological entities are tested in their original environment on whole living organisms, usually animals or humans. In situ observation is performed on site, typically in the habitat of the animal being studied and generally it is the environment that is modified in order to increase/improve the life conditions of a certain animal. The *in vitro* term refers to a controlled environment such as test tubes, flasks, petri dishes, etc. where the studied component is tested in an isolated way from their original, living surroundings. These experiments have fewer variables and simpler conditions than *in vivo* experiments and they can avoid the continuously changing impact and interactions of real life. This way/Thus they could allow a more fine-grained analysis of the studied phenomena. At the same time, correlating their results to real-world scenarios was not always straightforward, thus, generally *in vitro* results have to be verified in the original environment.

In contrast to the traditional methods, the *in silico* (Latin: in silicon, referring to semiconductor computer chips) experiments are performed on computer or via computer simulation, modelling the original components, variables and the studied effects. Thanks to the particularly fast growing of computer science technology these experiments become more and more complex, more data and compute intensive which requires parallel and distributed infrastructure (supercomputers, grids, clusters, clouds) to enact them. Generally, these in-silico experiments consist of a huge amount of activities (call jobs) – their number can reach hundreds or even thousands - which invoke particularly data and compute intensive programs. Tying the jobs to a single, multi thread chain provides a scientific workflow to model the *in-silico* experiments which can be executed by the Scientific Workflow Management Systems.

## 1.2 Reproducibility

To be able to proof or verify a scientific claim, the repeatability or the reproducibility of any type of experiments is a crucial requirement in the scientist's community. The different users for different purposes may be interested in reproducing of the scientific workflow. 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 (Koop & al, 2011). A reproducible workflow can be shared in repositories and it can become useful building blocks that can be reused, combined or modified for developing new experiments.

In the traditional method, the scientists make notes about the steps of the experiments, the partial results and the environment to make the experiments reproducible. Additionally, during the history of the scientific research, different standards, metrics, measurements and conventions had been developed to allow to provide the exact descriptions, the repeatability and the possibility of reusing each other's results. After all, certain types of the scientific experiments are unable to be repeatable because of the continuously changing environment such as the living organisms or nature in which many factors can be interacts and, in this way influence the results. Similarly, in case of the *in-silico* experiments, the same way has to be walked and has to develop tools to make them reproducible. On one hand, like the scientist make notes about the traditional experiments, provenance information has to be collected about the environment of the execution and the partial result of the scientific workflow. On the other hand, the ontologies of these type of experiments also has to be developed to allow the knowledge sharing and the reusability on the so called scientific workflow repositories. However, many researcher work in these fields the reproducibility of the scientific workflows is still a big challenge because of:

- The complexity and the ever-changing nature of the parallel and distributed infrastructure: Computations on a parallel and distributed computer system arise particularly acute difficulties for reproducibility since, in typical parallel usage, the number of processors may vary from run to run. Even if the same number of processors is used, computations may be split differently between them or combined in a different order. Since computer arithmetic is not commutative, associative, or distributive, achieving the same results twice can be a matter of luck. Similar challenges arise when porting a code from one hardware or software platform to another (Stodden & al., 2013)
- The labyrinthine dependencies of the different applications and services: A scientific workflow inherently can interconnect hundred or even thousand jobs which can be based on different tools and applications which has to work together and deliver data to each other. In addition, each job can depend on external inputs complicating the connections and dependencies.

- The complexity of the scientific workflows managing a huge amount of data.

## 1.3 Motivation

Zhao et al. (Zhao & al, 2012) and Hettne (Hettne & al, 2012) 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. In their investigation, they examined 92 Taverna workflows from myExperiment repository in 2007-2012 and re-execute them. This workflow selection had a large coverage of domain according to 18 different scientific (such as life sciences, astronomy, or cheminformatics) and non-scientific domains (such as testing of Grid services). The analysis showed that nearly 80% of the tested workflows failed to be either executed or produce the same results. The causes of workflow decay can be classified into four categories:

1. Volatile third-party Resources
2. Missing example data
3. Missing execution environment
4. Insufficient descriptions about workflows

By incorporating these results, we have deeply investigated the requirements of the reproducibility and I intended to find methods which make the scientific workflows reproducible.

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 have to be collected and stored about 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 in a different infrastructure. The collected data – called provenance data – help to store the actual parameters of the environments, the partial and final data product and system variables.
2. Descriptions and samples should 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, certain computations may base on random generated values (for example, in case of image processing) thus, its execution are not deterministic so these computations cannot be repeated to provide the same result in a later time. These factors – call dependencies of

the execution - can especially influence the reproducibility of the scientific workflows, consequently, they have been eliminated or handled.

In this dissertation, I deal with the third item.

The goal of computational reproducibility is to provide a solid foundation to computational science, much like a rigorous proof is the foundation of mathematics. Such a foundation permits the transfer of knowledge that can be understood, implemented, evaluated, and used by others. (Stodden & al., 2013)

However, nowadays more and more workflow repositories (myExperiment; CrowdLabs etc.) can help the knowledge sharing and the reusability, the reproducibility cannot be guaranteed by the systems. The ultimate goal of my research is to support the scientist by giving information about the reproducibility of the workflows found in the repositories. Investigating and analyzing the change of the components (call descriptors) required to the re-execution I reveal their nature and I can identify the crucial descriptor which can prevent the reproducibility. In certain cases, based on the behavior of the crucial component an evaluation can be performed for the case of unavailability which can replace the missing component with a simulated one making the workflow reproducible. With help of this reproducibility analysis also the probability of reproducibility can be calculated or the reproducible part of the workflow can be determined. To make the workflow reproducible, extra computations, resources or time are required which impose an extra cost for the execution. This cost can be measured and it can qualify the workflow from the reproducibility perspective. Additionally, the analysis presented in this dissertation can support the scientist not only to find the most suitable and reliable workflow on the repository but also can help to design a reproducible scientific workflow. The process, from the first execution of a workflow to achieving a complete and reproducible workflow is very long and the jobs get over a lot of change.

## 1.4 Research methodology

As a starting point of my research I thoroughly investigated the related work in the theme of reproducibility and the provenance which is the most significant requirements of the reproducibility. According to the reviewed literature I gave a taxonomy about dependencies of the scientific workflows and about the most necessary datasets required to reproduce a scientific workflow.

Based on this investigation I formalized the problem and set out the mathematical model of the reproducibility analysis. First, I introduced the necessary terms and definitions according to the reproducible job and workflow which serve as a building blocks to determine and prove the statements and the methods. With help of the mathematical statistics tool, I analyzed the nature

of the descriptors based on a sample set originating from the previous executions of the workflow to find statistical approximation tools to describe the relation between the descriptors and the results. Additionally, I introduced two metrics of the reproducibility based on the probability theory, the Average Reproducibility Cost (ARC) and the Non-reproducibility Probability (NRP) and defined a calculation method to calculate them in polynomial time. The universal approximation capabilities of neural networks have been well documented by several papers (Hornik & al., 1989), (Hornik & al., 1990), (Hornik, 1991) and I applied the Radial Basis Function (RBF) networks to evaluate the ARC in case if the exact calculation is not possible. To evaluate the NRP the Chernoff's inequality (Bucklew & Sadowsky, 1993) was applied based on Large Deviation Theory which concerns the asymptotic behavior of remote tails of sequences of probability distributions.

To perform the statistical calculations and prove the assumptions and the results, I used the MatLab and Excel applications.

### 1.5 Thesis structure

This dissertation is organized as follows: In the next section (2) the background of the scientific workflows is presented, their representation, life cycles and the most relevant Scientific Workflow Management Systems are described with special emphasis of their provenance and reproducibility support. Also in this section the WS-PGARDE/gUSE system is introduced since the implementation of this investigation is planned into it. In section 3 I deal with the requirements of the reproducibility and seven datasets are defined to establish the basis of this investigation, namely the descriptor-space which contains all the necessary information to reproduce a scientific workflow. Section 4 represents our mathematical model of the reproducibility analysis with the necessary definitions and terms. I introduce two ultimate characteristics of the descriptors, the theoretical and the empirical decay-parameter which help to analyze the behavior of the descriptors and the relation with the job results. In section 5 I deal with the effect of the changing descriptor, how many jobs are infected by the effect and the evaluability of the deviation of the result. Section 6 contains the probability investigation of the workflows and a method is presented to calculate the theoretical and the empirical probability of reproducibility. In section 7 I introduce the metrics of the reproducibility, ARC and NRP and two algorithms are determined to evaluate the metrics in polynomial time. In section 8 the classification of the scientific workflows is presented according to the reproducibility. Finally, I the results are concluded, the theses are described and I reveals some research direction along which this PHD research can be developed.

# 2 STATE OF THE ART

In this section the background of the scientific workflows, their natures, representation and lifecycle are presented, in addition a literature survey is given about the most relevant scientific workflow management systems (SWfMS) and their support of the reproducibility to highlight the focus and the background of this research.

## 2.1 Scientific Workflows

Applying scientific workflow to perform in-silico experiment is a more and more prevalent solution among the scientist's communities. Scientific workflow is concerned with the automation of scientific processes in which jobs are structured based on their control and data dependencies. In many research field, such as high-energy physics, gravitational-wave physics, geophysics, astronomy, seismology, meteorology and bioinformatics, these in-silico experiments consist of series of particularly data and compute intensive jobs. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications and scientific instruments need to be orchestrated while managing workflow operations within super/hypercomputers, grids, clusters or clouds (Gil & al, 2006) (Barker & Hemet, 2007).

### 2.1.1 Scientific Workflow Life Cycle

The various phases and steps associated with planning, executing, and analyzing scientific workflows comprise the scientific workflow life cycle (WFLC) (Deelman & Gil, 2006), (Gil & al, 2007) (Deelman & al, 2009). The following phases are largely supported by existing workflow systems using a wide variety of approaches and techniques. (Ludäscher & al, Scientific process automation and workflow management; Scientific Data Management: Challenges, Existing Technology, and Deployment, 2009)

*Hypothesis Generation* (Modification): Development of a scientific workflow usually starts with hypothesis generation. Scientists working on a problem, gather information, data and requirements about the related issues to make assumptions about a scientific process. From these data they build a specification which can be modified later during the whole lifecycle, or after the result analysis.

*Experiment / Workflow Design*: During the experiment an actual workflow is assembled based on this specification. This phase is the workflow development or design phase, which differs from

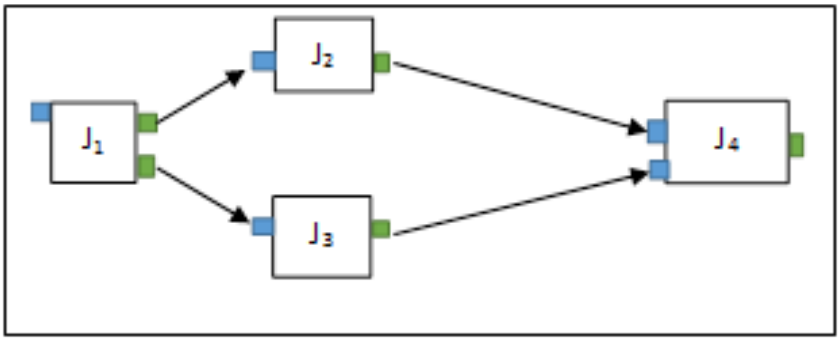
general programming in many ways. It is usually the composition and configuration of a special-purpose workflow from pre-existing, more general-purpose components, sub-workflows, and services. During workflow composition, the workflow developer either creates a new workflow by modifying an existing one or composes a new workflow from scratch using components and sub workflows obtained from a repository. In contrast to the business workflow world, where standards have been developed over the years (e.g., WS-BPEL 2.0 (Jordan, 2007)), scientific workflow systems tend to use a language set of internal languages and exchange formats (e.g., SCUFL (Taverna, 2009), GPEL (Wang, 2005), and MOML (Brooks, 2008)). Reasons for this diversity include the wide range of computation models used in scientific workflows and the initial focus of development efforts on scientist oriented functionality rather than standardization.
*Instantiation:* Once the workflow description is constructed, scientific workflow systems often provide various functions prior to execution. These functions may include workflow validation, resource allocation, scheduling, optimization, parameter binding and configuration. Workflow mapping is sometimes used to refer to optimization and scheduling decisions made during this phase.
*Execution*: After the workflow instantiation, the workflow can be executed. During execution, a workflow system may record provenance information (data and process history) as well as provide real-time monitoring and failover functions. Depending on the system, provenance information generally involves the recording of the steps that were invoked during workflow execution, the data consumed and produced by each step, a set of data dependencies stating which data was used to derive other data, the parameter settings used for each step, and so on. If workflow migration or adaptation (i.e.: change the workflow model or the running instance) is enabled or supported during execution (e.g., due to the changing environment), the evolution of such a dynamic workflow may be recorded as well to support subsequent event handling.
*Result Analysis*: After workflow execution, scientists often need to inspect and interpret workflow results. This involves evaluation of the results, examination of workflow execution traces, workflow debugging and performance analysis.
Data and workflow products can be published and shared. As workflows and data products are committed to a shared repository, new iterations of the workflow life cycle can begin.

### 2.1.2 Scientific workflow representation

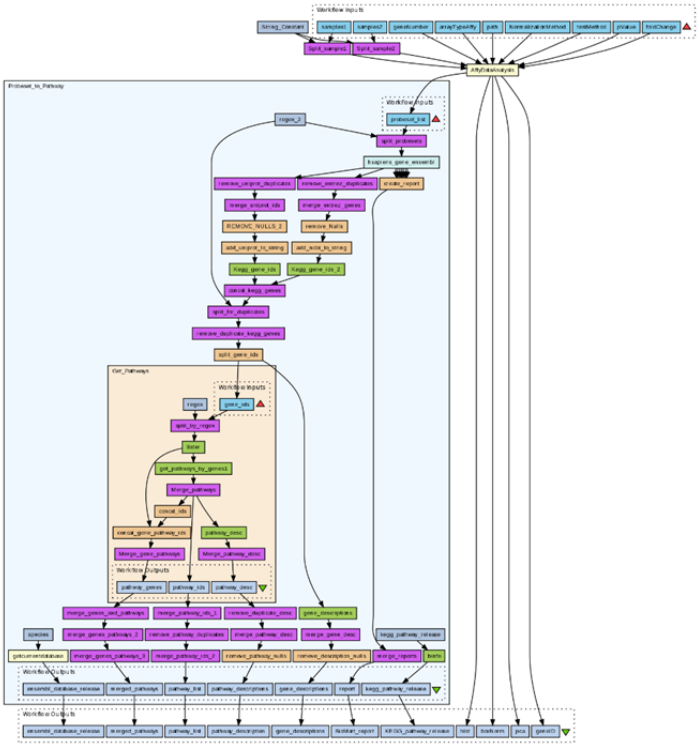
At the most abstract level, essentially all workflows are a series of functional units, whether they are components, jobs or services, and the dependencies between them which define the order in which the units must be executed. The most common representation is the directed graph, either acyclic (DAG) or the less used cyclic (DCG), which allow loops (Deelman & al, 2009). This latter

one represents the recursive scientific workflow. In this dissertation, I deal with the scientific workflow represented by DAG.

The nodes represent the jobs (denoted by $J_i$), which includes the experimental computations based on the input data accessed through their input ports. In addition, these jobs can product output data, which can be forwarded through their output ports to the input port of the next job. The edges of a DAG represent the dataflow between the jobs (Figure 1.). Figure 2 shows a more complex workflow downloaded from the myExperiment to demonstrate a typical scientific workflow.




*1. Figure: A simple scientific workflow example with four jobs ($J_1$, $J_2$, $J_3$, $J_4$) in gUSE*



http://www.myexperiment.org/workflows/10.html

*2. Figure:* A scientific workflow example from www.myexperiment.org

In this research, the scientific workflows represented by a directed acyclic graph denoted by G(V, E), where V denotes the set of jobs and E denotes the dataflow between jobs.
$V = \{J_1, \dots, J_N\}$, where $N \in \mathbb{N}$; the number of the job of a given workflow
$E = \{(J_i, J_j) \in V \times V | i \in [1, 2, \dots N-1]; j \in [2, 3, \dots, N] \; and \; i \neq j\}$

## 2.2 Scientific Workflows Management system

Scientific workflow systems are used to develop complex scientific applications by connecting different algorithms to each other. Such organization of huge computational and data intensive algorithms aim to provide user friendly, end-to-end solution for scientists (Talia, 2013). The following requirements should be met by the Scientific Workflow Management System (SWfMS):

- provide an easy-to-use environment for individual application scientists themselves to create their own workflows
- provide interactive tools for the scientists enabling them to execute their workflows and view their results in real-time
- simplify the process of sharing and reusing workflows among the scientist community
- enable scientists to track the provenance of the workflow execution results and the workflow creation steps.

Yu et al (Yu & Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, 2005) , (Yu & Buyya, 2005) gave a detailed taxonomy about the SWfMS for in which they characterized and classified approaches of scientific workflow systems in the context of Grid computing. It consists of four elements of a SWfMS: (a) workflow design, (b) workflow scheduling, (c) fault tolerance and (d) data movement. From the point of view of the workflow design the systems can be categorized by workflow structure (DAG and non-DAG), workflow specification (abstract, concrete) and workflow composition (user-directed, automatic). The workflow scheduling can be classified from the perspective of architecture (centralized, hierarchical and decentralized), decision making (local, global), planning scheme (static, dynamic) and strategies (performance driven, market-driven and trust-driven). The fault tolerance can be performed at task level and workflow level and the data movement can be automatic and user-directed.
In the next I introduce the most relevant SWfMS with special emphasis on their provenance and reproducibility support. The WS-PGRADE/gUSE is presented in more detailed manner since the

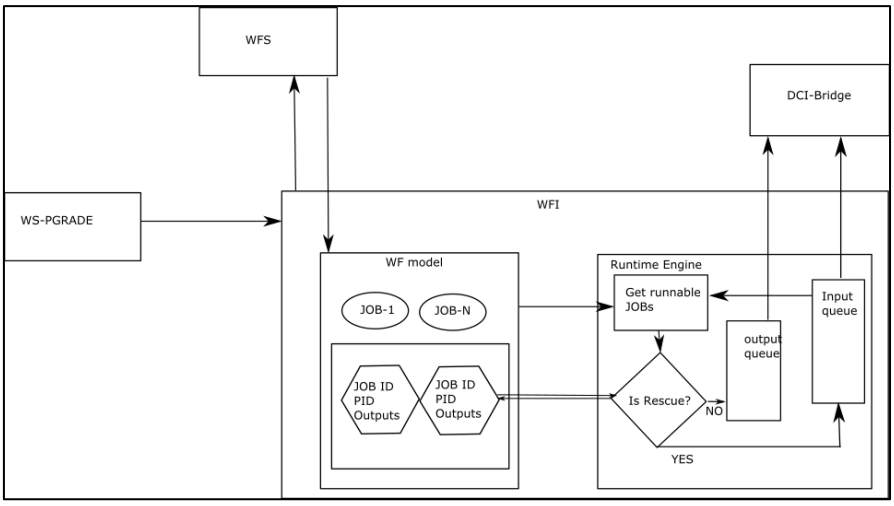
methods and the processes of the reproducibility analysis written in this dissertation will be implemented in it.

gUSE (Balaskó & al., 2013) (grid and cloud user support environment) is a well-known and permanently improving open source science gateway (SG) framework developed by 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. [ (gUSE)] The WS-PGRADE Portal [ (PGRADE)] is a web based front end of the gUSE infrastructure. The structure of WS-PGRADE workflows are represented by DAG. The nodes of the graph, namely 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.

In the WS-PGRADE/gUSE system with 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.

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. (Fig. 3)




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

Taverna (Oinn & al., 2006) (Oinn & al, 2004) is an open-source Java-based workflow management system developed at the University of Manchester. Taverna supports on one hand the life sciences community (biology, chemistry, and medicine) to design and execute scientific workflows on the other hand the in-silico experiments. It can invoke any web service by simply providing the URL of its WSDL document which is very important in allowing users of Taverna to reuse code that is available on the internet. Therefore, the system is open to third-part legacy code by providing interoperability with web services. In addition, Taverna use the myExperiment platform for sharing workflows; (Goble & al., 2010).

A disadvantage of integrating third-party Web Services is the variable reliability of those services. If services are frequently unavailable, or if there are changes to service interfaces, workflows will not function correctly on occasion of re-execution (Wolstencroft, 2013).

The Taverna Provenance suite records service invocations, intermediate and final workflow results and exports provenance in the Open Provenance Model format [ (OPM)] and the W3C PROV [ (PROV)] model.

Galaxy (Goecks, 2010), (Afgan, 2016) Galaxy is a web-based genomic workbench that enables users to perform computational analyses of genomic data. The public Galaxy service makes analysis tools, genomic data, tutorial demonstrations, persistent workspaces, and publication services available to any scientist that has access to the Internet. Galaxy automatically generates metadata for each analysis step. Galaxy's metadata includes every piece of information necessary to track provenance and ensure repeatability of that step: input

datasets, tools used, parameter values, and output datasets. Galaxy groups a series of analysis steps into a history, and users can create, copy, and version histories.

Triana (Taylor, 2004) (Taylor J. , 2005) is a Java-based scientific workflow system, developed at the Cardiff University, which combines a visual interface with data analysis tools. It can connect heterogeneous tools (e.g., web services, Java units, and JXTA services) in one workflow. Triana comes with a wide variety of built-in tools for signal-analysis, image manipulation, desktop publishing, and so forth and has the ability for users to easily integrate their own tools.

Pegasus (Deelman, 2005) is developed at the University of Southern California, it includes a set of technologies to execute scientific workflows in a number of different environments (desktops, clusters, Grids, Clouds). Pegasus has been used in several scientific areas including bioinformatics, astronomy, earthquake science, gravitational wave physics, and ocean science. It consists of three main components: the mapper, which builds an executable workflow based on an abstract workflow; the Execution engine, which executes in appropriate order the jobs; and the job manager, which is in charge of managing single workflow jobs. Wings (Gil, 2011), (Kim, 2008) providing automatic workflow validation and provenance frame work. It uses semantic representations to reason about application-level constraints, generating not only a valid workflow but also detailed application-level metadata and provenance information for new workflow data products. Pegasus maps and restructures the workflow to make its execution efficient, creating provenance information that relates the final executed workflow to the original workflow specification.

Kepler (Altintas, 2004) is a Java-based open source software framework providing a graphical user interface and a run-time engine that can execute workflows either from within the graphical interface or from a command line. It is developed and maintained by a team consisting of several key institutions at the University of California and has been used to design and execute various workflows in biology, ecology, geology, chemistry, and astrophysics. The provenance framework of Kepler (Bowers, 2008), (Altintas I. , 2006) keep track of all aspects of provenance (workflow evolution, data and process provenance). To enable provenance collection, it provides a Provenance Recorder (PR) component In order to capture run-time information event listener interfaces are implemented and when something interesting happens, the event listeners registered and take the appropriate action.

## 2.3 Provenance

Provenance data that carries information about the source, origin and processes that are involved in producing data play important role in reproducibility and knowledge sharing in the scientist community. Concerning provenance data lot of issues arise: during which workflow lifecycle phase data have to be captured, what kind of data and in what kind of structure need to be captured, captured data how can be stored, queried and analyzed effectively or who, why and when will use the captured information. The runtime provenance can be utilized in many area, for example fault tolerance, SWfMS optimization and workflow control.

There are two distinct forms of provenance (Clifford & al., 2008) (Davidson & Freire, 2008) (Freire & al., 2014),: prospective and retrospective. Prospective provenance captures the specification of a computational task (i.e., a workflow)—it corresponds to the steps that need to be followed (or a recipe) to generate a data product or class of data products.

Retrospective provenance captures the steps that were executed as well as information about the execution environment used to derive a specific data product— a detailed log of the execution of a computational task. (J.Freire & al., 2011), (Freire & al., 2012)

Despite the efforts on building a standard Open Provenance Model [ (OPM)], provenance is tightly coupled to SWfMS. Thus scientific workflow provenance concepts, representation and mechanisms are very heterogeneous, difficult to integrate and dependent on the SWfMS (Davidson & Freire, 2008). To help comparing, integrating and analyzing scientific workflow provenance, Cruz in (Cruz & al., 2009) presents a taxonomy about provenance characteristics.

PROV-man is an easily deployable implementation of the W3C standardized PROV. The PROV gives recommendations on the data model and defines various aspects that are necessary to share provenance data between heterogeneous systems. The PROV-man framework consists of an optimized data model based on a relational database system (DBMS) and an API that can be adjusted to several systems (Benabdelkader, 2014), (PROV) (Benabdelkader, 2011) (D-PROV)

Costa et al. in their paper (Costa & al., 2013) investigated the usefulness of runtime generated provenance data. They found that provenance data can be useful for failure handling, adaptive scheduling and workflow monitoring. Based on PROV recommendation they created their own data modelling structure.

The Karma provenance framework (Simmhan & al., 2006) provides generic solution for collecting provenance for heterogeneous workflow environments.

As an antecedent of this research four different levels of provenance data were defined because during the execution of a workflow four components can change that would affect the reproducibility: the infrastructure, the environment, the data and the workflow model. [7-B]

1. The first is a system level provenance, which stores the type of infrastructure, the variables of the system and the timing parameters. At this level happens the storing the details of the mapping process and as a result, we can answer the question of what, where, when and how long has been executed. This information supports the portability of the workflow which is a crucial requirement of reproducibility.
2. The environmental provenance stores the actual execution details which includes the operating system properties (identity, version, updates, etc.), the system calls the used libraries and the code interpreter properties. The execution of a workflow may rely on a particular local execution environment, for example, a local R server or a specific version of workflow execution software, which also has to be captured as provenance data or virtual machine snapshot.
3. The third category is data provenance. In the literature, the provenance often refers to data provenance, which deals with the lineage of a data product or with origin of a result. With this data provenance, we can track the way of the results and dependency between the partial results. This information can support the visualization, the deep and complete troubleshooting of the experimental model, the proving of the experiment but first of all the reproducibility. In addition, in one of our previous paper [B-10] we investigated the possibility and the need of user steering. We found that some parameters, filter criteria and input data set need to be modified during execution, which rely on data provenance.
4. The last provenance level tracks the modifications of the workflow model. The scientist during the workflow lifecycle often performs minor changes, which can be undocumented and later it is difficult to identify or restore. This phenomenon is usually referred as workflow evolution. Provenance data collected at this level can support the workflow versioning.

This structured provenance information of a workflow can support reproducibility at different levels if it meets the requirements of independency. In addition, extra provenance information can be stored in that cases, in which however the workflow contains some dependencies but these dependencies can be eliminated with usage of extra resources.

### 2.4 Reproducibility

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.

Researchers of this field agree on the importance of the careful design (Roure & al, 2011), (Mesirov, 2010), (Missier & al., 2013), (Peng & al., 2011), (Woodman & al.) which on one hand, it 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 (Davison, 2012). On the other hand, the careful design includes the careful usage of volatile third party or special local services.

Groth et al. (Groth & al., 2009) 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 (Davison, 2012) 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. (Zhao & al, 2012) 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. (Hettne & al, 2012) in their papers listed ten best practices to prevent the workflow decay.

### 2.4.1 Techniques and tools

There are available tools existing, VisTrail, ReproZip or PROB (Chirigati, D, & Freire, 2013), (Freire & al., 2014), (Korolev & al., 2014) which allow the researcher and the scientist to create reproducible workflows. With the help of VisTrail (Freire & al., 2014), (Koop & al, 2013) 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 (Chirigati, D, & Freire, 2013) 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 (Bechhofer & al, 2010), (Belhajjame & al., 2012) 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

(Belhajjame, 2015) 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 (Piccolo & Frampton, 2015) collected the tools and techniques and proposed six strategies which can help the scientist to create reproducible scientific workflows.

Santana-Perez et al (Santana-Perez & Perez-Hernandez, 2015) 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.

Gesing at al. in (Gesing & al., 2014) describe the approach targeting various workflow systems and building a single user interface for editing and monitoring workflows under consideration of aspects such as optimization and provenance of data. Their goal is to ease the use of workflows for scientists and other researchers. They designed a new user interface and its supporting infrastructure which makes it possible to discover existing workflows, modifying them as necessary, and to execute them in a flexible, scalable manner on diverse underlying workflow engines.

Bioconductor (Gentleman, 2004) and similar platforms, such as BioPerl (Stajich, 2002) and Biopython (Chapman & Chang, 2000) represent an approach to reproducibility that uses libraries and scripts built on top of a fully featured programming language. Because Bioconductor is built directly on top of a fully featured programming language, it provides flexibility. In the same time this advantage can be exploited by only users which has programming experience. Bioconductor lacks automatic provenance tracking or a simple sharing model.

# 3 REQUIREMENTS OF THE REPRODUCIBILITY

The implementation of the reproducible and reusable scientific workflows is not an easy task and many obstacles have to be removed toward the goal. Three main components play important role in the process:

- The SWfMS should support the scientist with automatic provenance data collection about the environment of execution and about the data production process. I determined the four levels of the provenance (subsection 2.3), and the different utilizations of the captured data in the different levels. Capturing provenance data during the running time of the workflow is crucial to create reproducible workflows.
- The scientists should carefully design the workflow (for example with special attention for modularity and robustness of the code (Davison, 2012) and give a description about the operation of experiment, the input and output data, even they should show samples. (Zhao & al, 2012) (Hettne & al, 2012).
- The dependencies of the workflow execution should be eliminated. A workflow execution may depend on volatile third party resources and services; special hardware or software elements which are available only in a few and special infrastructure; deadlines, which cannot be accomplished on every infrastructure or it can be based on non-deterministic computation which apply for example random generated values.

## 3.1 Dependencies

The execution of a workflow may require many resources, such as third party or local services, database services or even special hardware infrastructure. These resources are not constantly available, they can change their location, their access condition or the provided services from time to time. These conditions, which we refer to as dependencies, significantly complicate the chances of reproducibility and repeatability. We have classified the dependencies into three categories: infrastructural dependency, data dependency and job execution dependency as shown in table 1. [7-B]

| infrastructural | data | job execution |
|---|---|---|
| • spec. hardware demand | • changing<br>• TP demand<br>• local spec demand | • deterministic<br>• dependency between jobs<br>• Third Party demand<br>• Local spec demand |

*1. Table* Categories of workflow execution dependencies

By infrastructural dependency I mean special hardware requirements, which are available solely on the local system or not evidently provided by other systems, such a special processing unit (GPU, GPGPU).

In the group of data dependency, we listed the cases which does not guarantee the accessibility of the input dataset in another time interval. The causes can be that the data is provided by a third party or special local services. Occasionally the problem origins from the continuously changing and updated database that stores the input data. These changes are impossible to restore from provenance data.

The job execution can also depend on a third party or local services, but the main problem arises when the job execution is not deterministic. The operation of GPU or GPGPU are based on random processes consequently the results of re-executions may differ. Moreover, if the dependency factor is too high between the jobs, the reproducibility is harder to guarantee.

These conditions are all necessary to perform reproducibility of workflow execution. In section 5 we give a mathematical formula to determine the rate of reproducibility of a given workflow. With help of this measurement the scientist can see how much part of the workflow can be reproducible with 100 percent at a later period of time. Knowing this information, the scientist can decide to apply for example an extra provenance policy with extra resource requirement, which stores the whole third party data or apply virtual machine towards the reproducibility.

## 3.2 Datasets

To support and facilitate the work of the scientist by the SWfMS to create a well-documented and reproducible scientific workflow. The basic idea of our work is given by MIAME which describes the Minimum Information About a Microarray Experiment that is needed to enable the interpretation of the results of the experiment unambiguously and potentially to reproduce the experiment (MIAME) (Brazma & al, 2011). We collected and categorized the minimal sufficient information into seven different datasets, which target different problems to solve. Accordingly,

one of the types of data serves the documentation of experiment and helps to share it in a scientific workflow repository. Other type of data describes the data dependency and the process of data product and it is necessary for the proving and verification of the workflow. There is data which is needed to the repeatability or reproducibility of workflows in different infrastructure and environment. Finally, we collected information to help identifying the critical points of the execution which reduce the possibility of reproducibility or even arrest it [6-B].

The datasets are created in the different phases of the scientific workflow lifecycle (Ludäscher & al, 2009) and originate from three different sources. The scientist can give information when to design the abstract model, when to get the results or after the results are published. Other information can be gained from provenance database and there is information which can be generated automatically by the system.

With the help of our proposal we wish to solve the following problems:

- how to create a detailed description about scientific experiment;
- which minimal information is necessary to be collected from the scientists about their experiments to achieve a reproducible workflow;
- which minimal information is necessary from provenance to reproduce the experiments;
- which data and information can be generated automatically by the SWfMS in order to implement a reproducible scientific workflow;
- which jobs at which point do not meet the requirements of independencies.

If the goal is to repeat or reproduce the workflow execution on a different infrastructure, we have to store the descriptors and parameters of the infrastructure, the middleware and the operating systems in details too.

I defined seven types of datasets which contain the necessary and sufficient information about the experiment. An overview table summarizes the seven datasets and shows some examples about the stored data. (Table 1.) Data collected into different datasets target different problems to solve.

One part of the collected information of these datasets originates from the user, who creates the workflow. In the design phase the user establishes the abstract workflow model, defines the jobs, determines the input/output ports and specifies the input data and so on. Simultaneously, in order to achieve the reproducibility of workflow the user has to create the appropriate documentation about the experiment in a specific way, form and order. Such information is for example some personal data (name, date, etc.), the description of experiment (title, topic, goal, etc.), the samples about the necessary input, partial and output data, special hardware, application or service requirements and so on.

There are provenance data too in the datasets which have to be captured by the SWfMS in running time. For example, the version number and the variation of a given workflow, the number of submissions, the used data or parameter set during the previous executions, the makespan of execution or the number and types of failures occurred in running time. Information like these can be also crucial when the results of experiment have to be reproduced in a later time or in a different environment.

The third type of information is generated automatically by the system after the workflow is submitted, in the instantiation phase of the workflow lifecycle. This information can be obtained from the users too, but simpler, faster and even more precise and trusty if it is automated (for example workflow and job IDs, number of ports etc.). There exists such information too, which is created manually by the user at the beginning, but since the datasets and the database continuously grow and more and more data are collected, the system could "learn" certain information and fill in automatically the appropriate entries of datasets.

| | Scientist fills in in the design phase or before submit the workflow | filled in by Provenance in the execution phase | Scientist fill out after the execution |
|---|---|---|---|
| general description of experiment | title, topic, author(s), date, institute, laboratory, comment | number of ex-submission, number of failure, duration of execution, statistical data based on previous execution | publication details, experiences, comment |
| detailed environmental description of execution | infrastructure, OS, middleware, volume of resources, number of VM | start/end time of execution, statistical data based on the actual execution, resource usage (CPU, RAM, DISK, stb), | |
| detailed description of workflow | abstract wf (DAG), wf version, used parameter set, requirements (resources, libraries, applications with version number), place of input/output data files or storage), types of input/output data, | | |

| | | | |
|---|---|---|---|
| | constraints, deadlines, dependencies | | |
| research field specific information | | | |
| description of task-1<br>…<br>description of task-N | number of input/output ports, input/output data, types of input/output data, volume of input/output data, example input/output data, place of input/output data, necessary application, version number of app., dependencies, constrain | | |

*2. Table* Summary table about the datasets

General Description of Workflow (GDW).

This dataset contains general information about the scientific experiment such as title; author's name and its profile; the date; the institute's name and address, where the experiment is conducted and so on. In addition, general description of the experiment and data samples is also very important to be documented and stored. Most of the information originated from the users and it is necessary to create well-documented workflows, which will be reusable and understandable even after years. Certain entries are created in the design phase and others after the execution or later (for example publication details). However there exist information which is generated automatically by the SWfMS, such as Experiment ID, which is a unique identifier (expID) referred to the given workflow.

Detailed Description of Workflow (DDW)

The specification of the workflow is stored in the DDW. The experiment is modelled with an acyclic directed graph (DAG) (figure 1.) which is the most important part of this documentation in a graphical manner too. In addition, detailed information can be found in this dataset about the workflow (version number, parent workflows, required parameter set), the input/output data (number, type, amount, location, access method) the optional constraints or deadlines or other requirements. Automatically generated information is for example the number of input/output ports, the number of jobs, the number of entry/exit tasks

Detailed Description of Infrastructure (DDI)

If the goal is to repeat or reproduce the workflow execution on a different infrastructure, we have to store the descriptors and parameters of the infrastructure, the middleware and the operating systems in details too.

Detailed Description of Environment (DDE).

If the goal is to repeat or reproduce the workflow execution in a later time, we have to store the detailed environmental parameters. In this dataset, the following data can be found: the environmental variables and parameters; the circumstances of the execution; the state descriptors of the used resources; the time stamps; the required libraries, applications, data and services (with their exhaustive descriptions such as location, access method, version number etc.). This information can be captured during execution and can be stored as provenance data in a provenance database. The fields of this dataset filled in from this database.

## 3.3 Datasets for jobs

Every job has two datasets, the Detailed Description of Job (DDJ) and the Detailed Description of Environment of Job (DDEJ). Data in DDJ was collected based on two aspects: the first one helps understand the operation of a given job. The second one helps to follow the computational process and the partial or final results. DDEJ stores information about the environmental parameters of the execution, which serves the reproducibility. The number of DDJs (and also DDEJ) is equal to the number of jobs in the whole workflow.

Detailed Description of Job (DDJ)

The jobs in the abstract workflow model are organized into levels. The predecessors of any job are in lower level, the successors of a job are in upper level. This precedency appears in the naming convention of the job ID, which is referred to the exp ID and the sequence number of a level and the sequence number of a job in the given level. The entry job has not any input port or predecessor job, the exit job has not any output port or successor job. Also in this case, certain entries originate from the user (general description, job's name, sample input/output data, location and access method of input/output data, special hardware/application/service requirements etc.) and others are generated automatically by the system (job ID, predecessor and successor jobs, number of input/output ports, resource requirements).

Detailed Description of Environment of Job (DDEJ)

Provenance data can be used to fill in the most fields, such as type and number of failures; failure rate; start/end time of execution, waiting time, used resources, statistical data about previous executions and so on. The rest of necessary information can be generated automatically by the SWfMS such as type of code, compiler, resource requirements, virtual machine requirements and its state descriptors and so on.

## 3.4 Dependency dataset

In the instantiation phase of the workflow lifecycle, the SWfMS can examine the dependencies of the submitted workflow. With help of the given results together with the information gained from the user the system can create a so called Dependency Dataset, which will store all the jobs which depend on any external circumstances and may not be reproducible.

## 3.5 Conclusion

In this section, we investigated the necessary and sufficient information about scientific workflows to make them reproducible. We gave a proposal how to create the documentation of the scientific experiment to achieve this goal. The documentation consists of different datasets (related to the whole workflow and to the particular jobs) which are filled in from tree different sources: the scientist, the system and the provenance database. These datasets contain among others detailed information about the operation of the experiment; description and samples about input, partial and output data; and environmental descriptors. In addition, we specified another dataset about jobs, which depend on external conditions and can prevent the reproducibility or reusability of workflow. These datasets are necessary to create the so called descriptor-space introduced in the next section.

# 4 THE REPRODUCIBILITY ANALYSIS

In this section based on the datasets mentioned in the previous section I introduce the term of descriptor-space providing the basis of the reproducibility analysis. With help of the descriptor-space I give the definitions of reproducible job and workflows. In addition, I also introduce the term of decay-parameter to determine the behavior of the changing descriptors. Analyzing these changes, methods can be given to handle or eliminate the dependencies generated by them.

## 4.1 The different levels of the re-execution

The re-execution of a scientific workflow may have different purposes and goals and the different cases can require different conditions to perform this progress. Sometimes the exact repetition of the workflow is adequate for system developers to analyze the system and to develop a new one while another time the reproducing is necessary for the scientists to judge their scientific claims. Additionally, during the way which the scientists can take from designing a scientific workflow to verifying it, they pass the different phases of the re-execution from the repetition to the reproduction. Conversely, I separated the four goals of the re-execution: repetition, variation, repetition in different environment (portability) and reproduction (Table 3).

| **Level** | **Meaning** |
|---|---|
| repeatability | The workflow can be successfully re-executed using the original artifacts, data and conditions. |
| variability | The workflow can be successfully re-executed using the original artifacts, data and conditions., but with some measured modification of a parameter |
| portability | The workflow can be successfully re-executed using the original data and conditions but different artifacts. |
| reproducibility | The workflow can be successfully re-executed, independently from the scientist |

*3. Table* The different levels of the re-execution

To re-execute a single job of the scientific workflow, all the parameters must be stored such as inputs, code variables, program settings, environmental parameters etc. which unambiguously determine the job execution. This have to be done for every job of the workflow. The parameters needed to re-execution I call *descriptors* and they can be originated directly from the users or they can be collected from provenance information and system logs. At the first execution of a

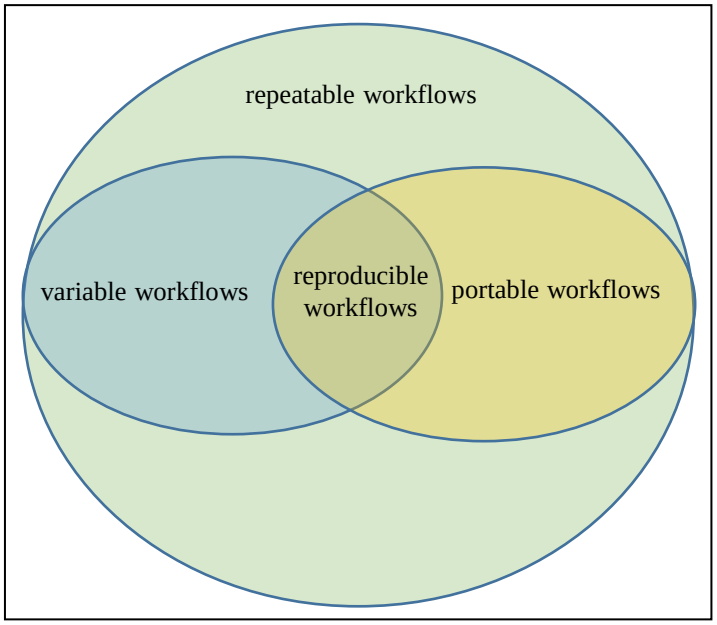
scientific workflows the value of the descriptors can be stored. Depending on which information are provided by the descriptors, they can be categorized into three groups: user specific, environmental and operation-related descriptors.

The user specific descriptors depend on the user such as inputs, variables or parameters of the job, the user can directly determine them or they can be captured by the provenance framework of the SWfMS.

The environmental descriptors refer to the parameters and variables of the enacting infrastructure such as the operating system with the appropriate version, the type of the CPU, the starting time of the job's running, the used libraries etc. Generally, they can be originated from the log and/or the provenance database.

The operation related descriptors relate to the operation of the system or reflect to the actual state of the system. In the most cases the value of these descriptors continuously change in time making the job nondeterministic. One of the examples is the random generated values (RGV). If a job based on RGV, this value kept unknown, it is not available nor in the provenance information nor in the logs and the result (output) of the job will never be the same. Since every generator is a pseudo random generator, knowing operation and the algorithm of the generator, the "random" result can be reproduced and the job can be made deterministic. Another way is that the RGV is captured and stored by an extra tool (script) developed for this purpose. Operation related descriptor can be also a return value of a system calls, which based on the actual time, the actual free amount of memory or other actual state of the system. In this cases the only possible solution is an extra tool developed for this purpose which can store these values.

The following figure can illustrate the relation of the different level (Figure 4.)




*4. Figure* The connection of the different levels of re-execution

### 4.1.1 Repeatability

Repeatability concerns the exact repetition of a scientific workflow, using the same experimental apparatus, the same inputs and settings of the jobs under the same conditions. It is a first step on the way toward the reproducibility and verifying the scientific claims. The arising failures during achieving the exact repeatability can expose hidden assumption about the experiment or the environment. Additionally, in certain research field the repetitions may not be 100% exact, due to the statistical variation and the measurement errors. Thus, the repetition is a useful process to calculate confidence intervals for the result of the scientific workflows. (Feitelson, 2015) According to repeatability, it can be assumed that the most descriptors does not change in time. The only decay factor may be found among the operation related descriptors are the random generated values, time based values or other system calls that depend on the actual state of the system. The user specific and the environmental descriptors are the same at every execution.

### 4.1.2 Variability

At the level of the variability the goal is to re-run the scientific workflow on the same infrastructure under the same condition with some intentional and measured modification of the jobs. The variation is the second step on the way toward the reproducibility. Variation can extend the understanding of the scientific experiment or the system being studied. (Feitelson, 2015) Performing several variations can provide a distribution of results, and give the possibility to

investigate whether the original result is in the middle of this distribution or in its tail. In this case, besides operation related descriptors user specific descriptors may also change.

### 4.1.3 Portability

The portability of a scientific workflow means the ability to run exactly the same workflow in a different environment or infrastructure under the same conditions. This is the third step on the way toward the reproducibility, and it is also one of the requirements of the reproducibility. Failures arising during achieving the portability can show the infrastructure dependent component of the execution and can provide important information about the robustness of the original scientific workflow. Additionally, it can depend on having a full and detailed descriptions of the original experiment which is also crucial to achieve the reproducibility and the reusability. According to the descriptors the environmental and the operation related descriptors can change while the user specific descriptors are the same.

### 4.1.4 Reproducibility

The term reproducibility means the ability for anyone who has access to the description of the original experiment and its results to reproduce those results independently, even under the different environment, with the goal to verify or reuse the original experimenter's claims. Consequently, a reproducible scientific workflow has the ability of repeatability, variability and portability too. It is the basis of sharing and reusing them in scientific workflow repositories. All the three type of the descriptor can change in time.

## 4.2 Non-determinisctic jobs

Typically, the operation-related descriptors such as random generated values, time-based values, etc. make the jobs non-deterministic preventing the reproducibility. This non-deterministic factor can be eliminated by operating system level tools developed for this purpose which can capture and store the return value of the system-calls. In this way, every job can be made deterministic thus hereafter in this dissertation I deal with deterministic jobs only.

## 4.3 The descriptor-space

Based on the datasets mentioned in the section 3, a so-called descriptor-space can be assigned to every job of a scientific workflows. In the datasets, the parameters - related to the descriptions of the SWf (sample data, descriptions, author's name etc.) – can be omitted and hereafter, I assume that a detailed and sufficient description is provided by the user about the SWf. which is enough to reproduce the workflow from that point of view. Based on the remain parameters a so-called descriptor-space can be defined. The theoretical descriptor-space contains all the descriptors which are necessary to re-execute the job. The descriptor-space assigned to the job $J_i$ can be denoted as follows:

$$D_{J_i} = \{d_{i1}, d_{i2}, \dots, d_{iK_i}\} \tag{4.3.1}$$

where $d_{ij}$ denotes the *j*-th descriptor of the job $J_i$

During an execution, the descriptors get a concrete value according to a given time $t_0$:

$$d_{ij}(t_0) = v_{ij}^{t_0} = v_{ij}^{(0)} \tag{4.3.2}$$

In this way, the concrete instantiation of a descriptor-space can be written as follows:

$$D_{ij}^{t_0} = \{v_{i1}^{t_0}, v_{i2}^{t_0}, \dots, v_{i3}^{t_0}\} \tag{4.3.3}$$

With help of the descriptor-space the deterministic scientific workflows and its jobs can be interpreted as a multivariate function:

$$SWF(t_0, J_1, J_2, \dots, J_N) = R \tag{4.3.4}$$

where *R* is the result (output) of the scientific workflow and *N* is the number of the jobs and

$$JOB_i\left(t_0, v_{i1}, v_{i2}, \dots, v_{iK_i}\right) = JOB_i\left(t_0, D_{J_i}\right) = {R_i}^{t_0}, \tag{4.3.5}$$

where $i = 1, \dots, N$ and $K_i$ is the number of the descriptors of the job $J_i$ and since the $t_0$ is indicated as the variable of the function, for the sake of simpler notation the $t_0$ upper index is omitted on $v_{ij}^{t_0}$.

In case of the nondeterministic jobs, stochastic function can be used, therefore the R result can be evaluated with a given probability.

$$JOB_i\left(t_0, v_{i1}, v_{i2}, \dots, v_{iK_i}\right) = \hat{R}_i \tag{4.3.6}$$

## 4.4 Definitions of reproducible job and workflow

Based on the descriptor-space the definition of a reproducible job can be determined as a time-invariant function, therefore

*Definition (D.4.3.1)*: A *job is reproducible if it meets the following requirement:*

$$JOB_i{}^{repro}\left(d_{i1}(t_0), v_{i2}(t_0), \dots, d_{iK_i}(t_0)\right) =$$

$$JOB_i{}^{repro}\left(d_{i1}(t_0 + \Delta t), d_{i2}(t_0 + \Delta t), \dots, d_{iK_i}(t_0 + \Delta t)\right) = R_i \tag{4.4.1}$$

for every Δt.

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

Since the scientific workflows consist of many jobs, the definition of the reproducible job has to be extended for the reproducible scientific workflows. In order to give the definition, some other term and their indications - which is used in the literature in different way – has to be laid down.

*Definition (D.4.3.2):* The job $J_i$ is *exit job* in the scientific workflow, if $\nexists J_j \in V$: $(J_i, J_j) \in E$, in other words if it has not successor job.

Notation: $J_{exit}$

*Definition (D.4.3.3):* The job $J_i$ is *entry job* in the scientific workflow, if $\nexists J_j \in V$: $(J_j, J_i) \in E$, in other words if it has not predecessor jobs.

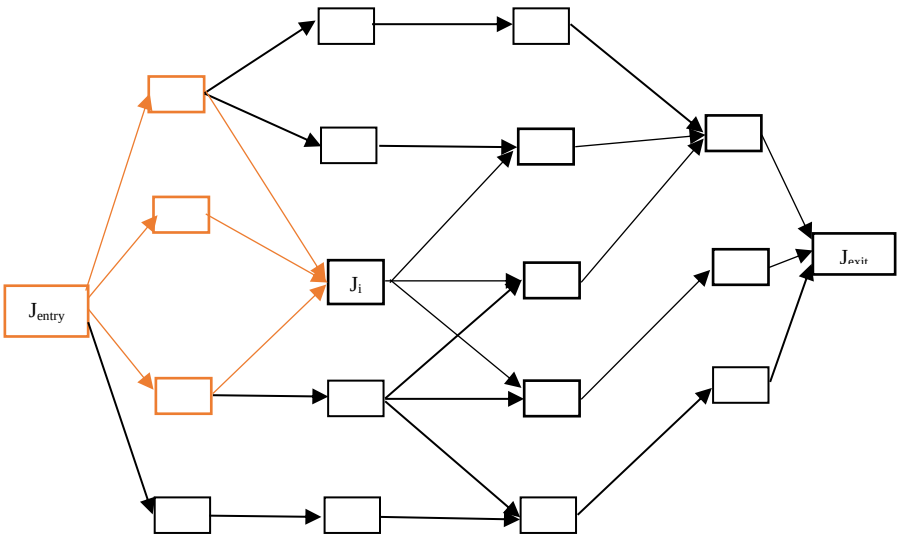
Notation: $J_{entry}$

*Definition (D.4.3.4):* The job, which is neither exit nor entry job, it is *inside job*.

In my research, I assume that every scientific workflows has at least one entry and one exit job.

*Definition (D.4.3.5):* The *backward subworkflow of a job* $J_i$ is a subgraph of the workflow graph where the exit job is $J_i$ and the entry job is the entry job of the original workflow graph. (Figure 5)

Notation: $SubWF_{J_i}^{back} = G_{sub}(V_{sub}; E_{sub})$ $where$ $\{V_{sub}\} \subseteq \{V\}$, $\{E_{sub}\} \subseteq \{E\}$




*5. Figure* The backward subworkflow of a job $J_i$

With the help of these terms the definition of the reproducible job can be extended for scientific workflow in the following way:

*Definition (D.4.3.6):* The *SWF* is *reproducible*, if the exit job and the $SubWF_{J_{exit}}^{back}$ of the exit job is reproducible.
Notation: $SWF^{repro}$;

Based on the two definitions (D.4.3.1) and (D.4.3.6) the following statement can be formulated and proved:
*Statement (S.3.3.1):* If and only if every job of a scientific workflow is reproducible, then the scientific workflow is reproducible.
Let the $SWF(J_1, J_2, \dots, J_N) = Y$
Proof:

a. If $\forall J_i^{repro}; i = 1, 2, \dots, N$ than the $SWF^{repro}$

   SWF is reproducible, if its exit job and sub-workflow of the exit job is reproducible.
   Let us assume that the swf has $k < N$ exit jobs: Since every job is reproducible, especially the exit jobs are also reproducible so this condition is fulfilled.
   Let us consider the *k* sub-workflows of the exit jobs (which may not be disjunctive). This *k* sub-workflow is reproducible, if on one hand its exit jobs are reproducible, on the other hand the sub-workflows of these exit jobs are also reproducible. Let us assume, that the *k* sub-workflows have $l < N\text{-}k$ exit jobs. Since every job is reproducible, especially these *l* exit jobs are also reproducible and so on. Since the size of the sub-workflows and the number of the exit jobs continuously decrease, this algorithm can be continued until there are not exit jobs and the sub-workflow of the last exit job consist of only the entry job, which is also reproducible.
   QED
   Lemma (L1): If we separate the exit jobs from its sub-workflows and in the sub-workflows we also separate the exit jobs from its sub-workflows and this procedure is repeated until there are no more exit job and the sub-workflow of the last exit jobs is the entry job, than every job in the workflow become exit job at least once.
   Proof : Since every swf has at least one exit job, we have to investigate only the inside jobs.
   Let us investigate an arbitrary inside job $J_i$, of sub-workflow $G_s$.
   Since $J_i$ is an inside job, $\exists J_j \in V: (J_i, J_j) \in E$. There are two options

   i. $J_j$ is an exit job. In this case in the sub-workflow $G_s^1$ we can separate the exit job $J_j$ from its sub-workflow $G_s^2$. In $G_s^2$ the job $J_i$ is necessarily become an exit job, since $G_s^2$ contains all the paths between the entry job and the predecessor job of $J_j$ which actually is $J_i$, consequently the job $J_i$ in $G_s^2$ has not successor job, it is an exit job. QED

ii. $J_j$ is an inside job. If $J_j$ is an inside job, $\exists J_k \in V: (J_j, J_k) \in E$, which is an exit job or an inside job. If $J_k$ is an exit job, after two separation step – first $J_j$ then also $J_i$ – become exit job. During the series of separation steps every inside job - found along the path from the actual inside and to the exit job - eventually becomes an exit job. QED

b. If the $SWF^{repro}$, than $\forall J_i^{repro}; i = 1, 2, \dots, N$

Since SWF is reproducible, its exit job is also reproducible. We separate the exit job from its sub-workflow, and the sub-workflow is also reproducible. Based on the lemma L1, during the separation procedure, every job become exit job at least one time which is reproducible, consequently every job is reproducible. QED

*Corollary (C1):* In the case of reproducible scientific workflow every job can be reproduced independently.

Proof: Based on T1, in a reproducible scientific workflow every job is reproducible. A job is reproducible based on the definition, if the descriptor-space is known and every decay-parameter is 0. If the descriptor space is known and stored, the execution of the job does not depend on neither time nor any external parameters, consequently it can be reproduced anytime and anywhere. This is true in the case of any job. QED

### 4.5 The sample set

During the process of the workflow lifecycle and the way in which the workflow can be formed to be reproducible many executions and re-executions are performed. The increasing number of the re-execution gives the possibility to collect and store the descriptor values originated from different executions generating a continuously growing dataset, called sample set. In the design phase, certain jobs are modified many times while the others remain unchanged. These latter ones, already during this phase can provide useful experience about the descriptor values. Although the sample set of the other type of jobs show slower growth, it can be still augmented while reach the level of reproducibility. Additionally, because of the users' demand for re-using each other's workflows, subworkflows or even individual jobs can be found in the repositories, with continuously increasing sample set. The sample set of the job originating from the different executions can be stored together with the job in the repository to support the reproducibility analysis when a user intends to reuse it.

The sample set used in this dissertation can be written in the following way:

$$S_{J_i} = \left\{ \begin{array}{c} d_{i1}(t_0), d_{i2}(t_0), d_{i3}(t_0), \dots, d_{iK_i}(t_0), {R_i}^{t_0} \\ d_{i1}(t_1), d_{i2}(t_1), d_{i3}(t_1), \dots, d_{iK_i}(t_1), {R_i}^{t_1} \\ d_{i1}(t_2), d_{i2}(t_2), d_{i3}(t_2), \dots, d_{iK_i}(t_2), {R_i}^{t_2} \\ \dots \\ d_{i1}(t_{s-1}), d_{i2}(t_{s-1}), d_{i3}(t_{s-1}), \dots, d_{iK_i}(t_{s-1}), {R_i}^{t_{s-1}} \end{array} \right\} \tag{4.5.1}$$

where *t* indicates the time when the scientific workflow was executed.

In the most section, I investigated the jobs in general, independently from the scientific workflow. Thus, for the sake of simplicity the index *i* referred to the job $J_i$ can be omitted, additionally the time $t_s$, according to the descriptor value originated from execution *s*-th, will be indicated in the upper index.

Conversely, the simpler form of the sample set is the following:

$$S = \left\{ \begin{array}{c} v_1^{(0)}, v_2^{(0)}, v_3^{(0)}, \dots, v_K^{(0)}, R^{(0)} \\ v_1^{(1)}, v_2^{(1)}, v_3^{(1)}, \dots, v_K^{(1)}, R^{(1)} \\ v_1^{(2)}, v_2^{(2)}, v_3^{(2)}, \dots, v_K^{(2)}, R^{(2)} \\ \dots \\ v_1^{(s-1)}, v_2^{(s-1)}, \dots, v_K^{(s-1)}, R^{(s-1)} \end{array} \right\} \tag{4.5.2}$$

where $v_i^{(j)}$ is the *i*-th descriptor value originated from the *j*-th execution.

## 4.6 The theoretical decay parameter

The descriptors in the descriptor-space was categorized depend on which information is provided. Additionally, they have another underlying attribute referring to their decay, namely how they change and how they can influence the re-execution of the job or the scientific workflow. The different descriptors can affect or even prevent the re-execution in different way. To describe the behavior of a descriptor I introduce a so called theoretical decay-parameter which creates four classes among the descriptors. The decay-parameter can be the following:

a. The decay-parameter of a descriptor can be zero. There are constant descriptor's values which do not change under any circumstances; the time does not influence their values and their availabilities. For example, a job may have constant inputs or parameters. If a job has two input port getting the values 2 and 3 and the result of the job is the summation of the inputs, the two descriptors of the job are *input1* and *input2*; the descriptor's values are 2 and 3 which cannot be influenced by the time on no conditions.

b. Some descriptors depend on external services or resources which can become unavailable during the years. The decay-parameter of these descriptors are a probability distribution function (generally exponential distribution function). This distribution may be given, evaluable or unknown. For example, third party services which can be unavailable at any time or can leave off to provide their services after the years.

c. Certain descriptors are continuously changing in time. For example, the statistics gained from continuously growing databases which are fed with more and more data from sensors or from other resources (in the field of astronomy, bioinformatics etc.). In this cases the decay-parameter of the descriptor is a function (*vary(v)*) which describes the change of the value. This function also can be unknown, known or even evaluable.

Formally:

$$decay(v_i) = \begin{cases} 0, & \text{if the value of the descriptor is not changing in time} \\ F_i(t), & \text{if the availabilitydistribution function of of the given value} \\ Vary_i(t,\ v_i), & \text{if the value of the descriptor is changing in time} \end{cases} \quad (4.6.1)$$

*Note:* There are descriptors with originally unknown descriptor value, if the descriptors are operation-related and extra tool is required to be able to capture and store their values. With help of this tool the decay-parameter can be identified.

*Statement (S.4.6.1)*: If every decay-parameter is zero in a job than the job is reproducible.

Proof: Let $J\big(d_1(t_0), d_2(t_0), \dots, d_K(t_0)\big) = R$ be a job. If every decay parameter is zero, the descriptors are constant thus they do not change in time, consequently $d_j(t_0) = d_j(t_0 + \Delta t); j = 1, 2, \dots, K$ for every $\Delta t$. In this way, the definition of the reproducible job *(D.4.3.1)* is fulfil.

$JOB_i^{repro}(d_1(t_0 + \Delta t), d_2(t_0 + \Delta t), \dots, d_K(t_0 + \Delta t)) =$

$JOB_i^{repro}\big(d_1(t_0), d_2(t_0), \dots, d_K(t_0)\big) = R$

QED

## 4.7 The distance metric

To be able to investigate the variation of a descriptor and the impact of the descriptors on the result, the deviation of the result or the descriptors must be measurable. Since every descriptor has a name and a value, in this case the assumption can meet the requirements in a simple way. In contrast, the outcome of a job can move on a wide range of the possibilities. They can be for example numerical data, vectors, matrices, diagrams, images, text files, audio files or video files etc. Additionally, a job can have more output, too. To find a measurable deviation between two different results belonging to the same job in can be simply performed in certain cases. It can be even automatically performed by the system as well, but in other cases, the scientist has to determine the underlying difference between two results from the perspective of the scientific experiment. For example, the size or the resolution of an image can be irrelevant but the rate of the three main colors can be the same at every execution. In this case, the difference between the rate of colors can be measured. Sometimes there are more important factors and two or three different type of deviation must be investigated and defined. Harking back to the previous example, assuming that the images can show a circle or a triangle, and the difference can be importance from only this point of view. In cases like this the distance can be 1 if the two form is different and 0 if they are the same.

Conversely, in the most cases, a measurable deviation can be defined over the field of the possible values of the results, which can be determined automatically by the system or with help of the scientist. Hereafter I deal with the scientific workflows which meet the requirements that a distance metric can be defined for the descriptors and the result of the jobs.

In formal:

$\Upsilon_{d_i}$: the set of the possible values of descriptor $d_i$

$\Delta_{d_i}: \Upsilon_{d_i} \times \Upsilon_{d_i} \longmapsto \mathbb{R}$,

Notation: $v_i, v_j \in \Upsilon_{d_i}: \Delta_{d_i}(v_i, v_j) = \|v_j - v_i\|$ (4.7.1)

$\mathcal{R}_i$: the set of the possible values of the results of job $J_i$

$\Delta_R: \mathcal{R}_i \times \mathcal{R}_i \longmapsto \mathbb{R}$,

Notation: $R_i, R_j \in \mathcal{R}_i: \Delta_{d_i}(R_i, R_j) = \|R_j - R_i\|$ (4.7.2)

## 4.8 The empirical decay-parameter concerned to time-dependent descriptors

During the increasing number of the executions, more and more precise knowledge can be collected based on the sample set about the behavior of the descriptors and most of all about the

changing descriptors. The nature of a descriptor can be very diverse, sometimes deterministic while in certain cases nondeterministic. For example, it can follow an unidirectional, continuously change which can be linear, exponential, logarithmic etc. or even irregular. Nevertheless, it can fluctuate about a determinable value and the fluctuation can be periodically or randomly as well. Additionally, a value of a descriptor can be fixed but at certain executions the descriptor may have the outliers. To be able to identify the nature of the descriptor and to measure the change of the descriptor I define the empirical decay-parameter in time-dependent cases and in time-independent cases too.

$$decay_{emp}^{time}(\Delta t, v_i, s) =$$

$$\begin{cases} 0, & \text{if } \sum_{j=1}^{s} \left\| {v_i}^{(j)} - v_i^{(j-1)} \right\| = 0 \\ \dfrac{\sum_{j=2}^{s} \frac{\left\| {v_i}^{(j)} - v_i^{(1)} \right\|}{t_j - t_1}}{\sum_{j=2}^{s} \frac{\left\| {v_i}^{(j)} - v_i^{(j-1)} \right\|}{t_j - t_{j-1}}}, & \text{if } \sum_{j=1}^{s} \left\| {v_i}^{(j)} - v_i^{(j-1)} \right\| \neq 0 \end{cases} \quad (4.8.1)$$

The numerator of the fraction determines the variation of the descriptors depending on time correlated to the first value of the descriptor. The denominator investigates the variation of the descriptors correlated to the previous value. Consequently, the rate between the two variations gives the empirical decay. In other words, this expression investigates the measure of the change of the descriptor values at the different executions while it observes also whether the values continuously diverge from the first value or they may fluctuate around a certain value (not inevitably around the first value).

The empirical decay also can be interpreted for the result of the job as well, if a distance metric of the result can be determined.

$$decay_{emp}^{time}(\Delta t, R, s) = \begin{cases} 0, & \text{if } \sum_{j=1}^{s-1} \left\| R^{(j)} - R^{(j-1)} \right\| = 0 \\ \dfrac{\sum_{j=1}^{s-1} \frac{\left\| R^{(j)} - R^{(1)} \right\|}{t_j - t_1}}{\sum_{j=1}^{s-1} \frac{\left\| R^{(j)} - v^{(j-1)} \right\|}{t_j - t_{j-1}}}, & \text{if } \sum_{j=1}^{s-1} \left\| R^{(j)} - R^{(j-1)} \right\| \neq 0 \end{cases} \quad (4.8.2)$$

## 4.9 The empirical decay-parameter concerned to time-independent descriptors

There are descriptors which do not depend on time and the time may become an embarrassing factor during the observation of their behaviors. For example, if a descriptor value shows a constant pattern with some outliers it does not depend the time but the time-dependent decay cannot show this phenomenon. To extend or complete the investigation of the descriptor value a time-independent decay also has to be introduced in the following way:

$$decay_{emp}(\Delta v_i, s) = \begin{cases} 0, & \text{if } \sum_{j=1}^{s-1} \left\| v_i{}^{(j)} - v_i^{(j-1)} \right\| = 0 \\ \frac{\sum_{j=1}^{s-1} \left\| v_i{}^{(j)} - v_i^{(1)} \right\|}{\sum_{j=1}^{s-1} \left\| v_i{}^{(j)} - v_i^{(j-1)} \right\|}, & \text{if } \sum_{j=1}^{s-1} \left\| v_i{}^{(j)} - v_i^{(j-1)} \right\| \neq 0 \end{cases} \quad (4.8.1)$$

The meaning of this expression is very similar to the time-dependent one the only different is that it overlooks the elapsed time between two values.

Note: If the sampling is equidistant the time-independent form of the empirical decay can be applied.

## 4.10 Investigation of the behavior of the descriptors

First, the definition of a reproducible job has to be investigated in case of the empirical decay-parameter. It is clear, that if a job is reproducible then the $decay_{emp}(\Delta t, \mathrm{R}, s) = 0$, conversely, the statement is not so evident. Concerning to the empirical decay, the size of the sample set ($s$) is an important information, it is an important characteristic of the empirical decay. If there is no information about the theoretical nature of the descriptor, all the knowledge about it can be based on only the samples and any prediction of the descriptor value cannot be guaranteed. The more samples can ensure the more probable prediction. Consequently, every statement of the empirical decay has to refer to the experience gained from the s executions.

- The empirical definition can be given as: If $\sum_{i=1}^{K_i} decay_{emp}(\Delta t, v_i, s) = 0$ and $decay_{emp}(\Delta t, R_i, s) = 0$ than $J_i$ is repeatable based on $s$ execution. This means that the descriptor values have not change during the $s$ executions thus it can be concluded that the job was successfully re-executed $s$ times without change of descriptors. In this case the reproducibility cannot be guaranteed, only the repeatability.
- If $\sum_{i=1}^{K_i} decay_{emp}(\Delta t, v_i, s) = 0$ and $decay_{emp}(\Delta t, R_i, s) \neq 0$ than the $D_i$ descriptor-space is not complete. Since the deterministic behavior of the jobs has been assumed, in this case there exist at least one unknown descriptor which influences the result of the job.
- If $\sum_{i=1}^{K_i} decay_{emp}(\Delta t, v_i, s) \neq 0$ and $decay_{emp}(\Delta t, R_i, s) = 0$ than $J_i$ is variable/portable/reproducible based on $s$ execution and over the $\{v_{i1}, v_{i2}, \dots, v_{iL}\}$ descriptor-set. The level of the re-execution is determined by the type of the descriptor. If the changing descriptors are user-defined descriptor, the job is variable. If the changing descriptors are environmental descriptors the job is portable and if both, the job is reproducible referring to the changing descriptors.

- If $\sum_{i=1}^{K_i} decay_{emp}(\Delta t, v_i, s) \neq 0$ and $decay_{emp}(\Delta t, R_i, s) \neq 0$ than $cor\left(\delta_{j,j-1}(v_i), \delta_{j,j-1}(R)\right)$ must be investigated. There is two cases:
    a. the connection between the descriptor and the result can be determined or even predicted,
    b. there is no correlation between the variables.

### 4.10.1 Simulations

The empirical decay had been introduced to give information about the nature of the descriptors. Thus, the possible values and the behavior of the "decay-function" have to be analyzed to be able to predict the change of the descriptors. To identify the nature of the change simulations were performed based on the sample sets containing 20, 50 and 100 elements. In time-dependent case the time intervals were generated randomly based on a non-determined "time-unit" which can be hours, days, weeks or even months. The measure of te "time-unit" does not influence the values of the empirical decay-parameter. The simulations showed that typically 20-30 – it depends on the nature of the change – samples, in other word executions are necessary to be able to correctly evaluate the change and 50 samples are enough, to clearly show the results. The figures, in the next subsections are created based on 50 samples. Some of the results can be proved by mathematical tools which are described below. I investigated the following different, typical sorts of changes:

- continuously increasing deviation from the starting value in irregular (random) and regular cases (linear, exponential, radical and logarithmic)
- fluctuating deviation in random and periodic (sinus) cases
- the descriptor values typically do not change but a few outliers can be found

### 4.10.2 Linearity

In the linear case both the time-dependent and time-independent decay-parameter can unambiguously determine the change of the descriptor by a well-defined expression or a concrete value.

Statement (S.4.10.1.1): If the descriptor is time independent and the change is linear than $decay_{emp}(\Delta v_i, s) = 1 + \frac{s-2}{2}$

Proof: Let $\delta_{m,n}$ indicates the distance between two instantiations of the descriptor value *i*-th: $\delta_{m,n}(v_i) = \left\| v_i^{(n)} - v_i^{(m)} \right\|$. Since the change is linear $\forall m,n \in [1,s]: \delta_{m-1,m}(v_i) = \delta_{n-1,n}(v_i)$ let be denoted by $\delta$.

$$decay_{emp}(\Delta v_i, s) = \frac{\sum_{j=1}^{s-1} \left\| v_i^{(j)} - v_i^{(1)} \right\|}{\sum_{j=1}^{s-1} \left\| v_i^{(j)} - v_i^{(j-1)} \right\|} = \frac{\sum_{j=1}^{s-1} \delta_{1,j}(v_i)}{\sum_{j=1}^{s-1} \delta_{j-1,j}(v_i)}$$

$$= \frac{\delta + 2\delta + 3\delta + \cdots + (s-1)\delta}{(s-1)\delta} = 1 + \frac{1 + 2 + 3 + \cdots + (s-2)}{(s-1)}$$

$$= 1 + \frac{(1 + (s-2))\frac{s-2}{2}}{(s-1)} = 1 + \frac{(s-1)(s-2)}{2(s-1)} = 1 + \frac{s-2}{2}$$

Statement (S.4.10.1.2): If and only if the descriptor is time-dependent and the change is linear than

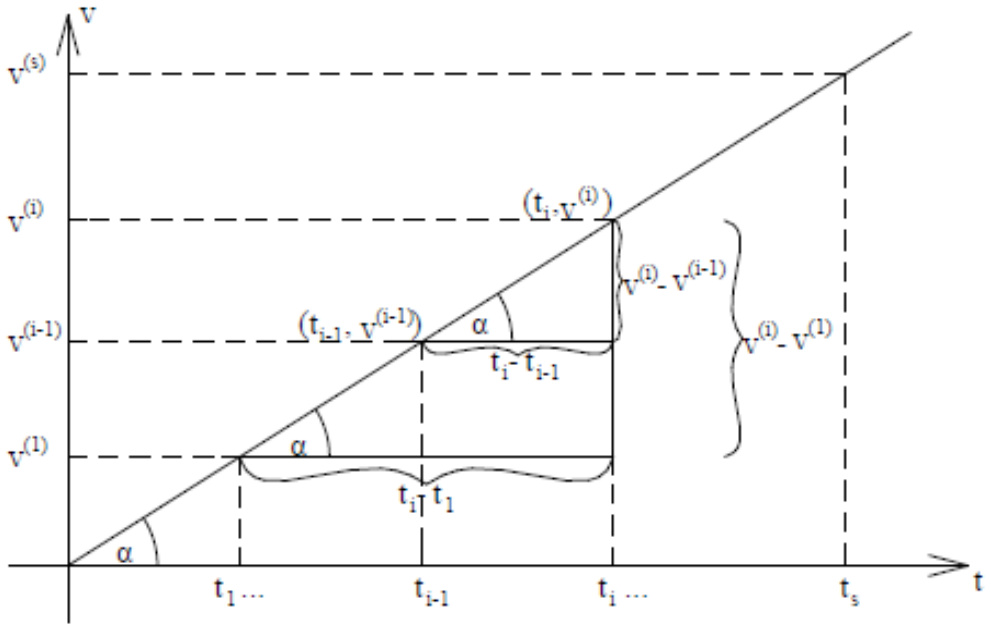
$decay_{emp}^{time}(\Delta v_i, s) = 1$ for every $s \in \mathbb{N}$.

Proof:

**a.** Let the change be linear.

Actually, both the numerator and the denominator of the expression (4.8.2) is a slope ($\tan \alpha$) of the line in a given time interval. In the case of the numerator the "big" triangle (($t_1$,$v_1$); ($t_i$,$v_1$); ($t_i$,$v_1$)) has to be investigated and in the case of the denominator the expression refers to the "little" triangle (figure 6). Assuming the linearity, all the slopes are equal in all the interval consequently

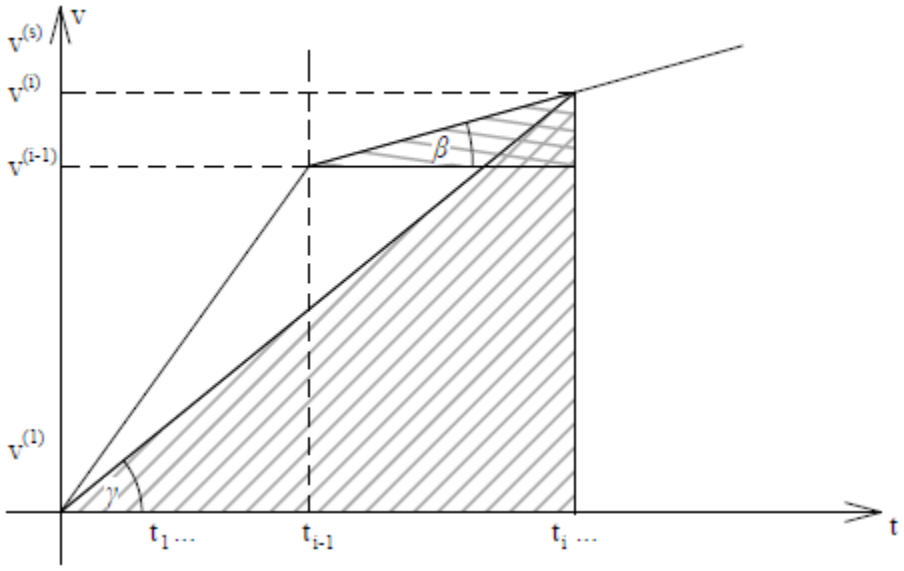
$$decay_{emp}^{time}(\Delta v_i, s) = \frac{\sum_{j=1}^{s-1} \frac{\left\| v_i^{(j)} - v_i^{(1)} \right\|}{t_j - t_1}}{\sum_{j=1}^{s-1} \frac{\left\| v_i^{(j)} - v_i^{(j-1)} \right\|}{t_j - t_{j-1}}} = \frac{(s-1) \cdot \tan \alpha}{(s-1) \cdot \tan \alpha} = 1$$




6. *Figure* The illustration of the numerator and the denominator in the time-dependent empirical decay

**b.** Let $decay_{emp}^{time}(\Delta v_i, s) = 1$ for every $s \in \mathbb{N}$

Following the base assumption of the fraction is 1 for every $s \in \mathbb{N}$, it means that the slopes are equal in case of the numerator (in which the point of reference is the initial descriptor value at the time $t_0$) and the denominator (in which the change is measured from the previous value) too. Since the statement has to be true for every $s \in \mathbb{N}$, it is true in the case of $s = 1$ and $s = 2$. In this case, not only the means are equal but also the element of the summation. Starting from this point, if the fraction is 1, every appropriate element in the summation of the numerator and the denominator is equal. Additionally, the elements of the summation are equal too. Letting $\tan\alpha$ be the elements of the summation. The claim is negated to assume that there is at least one element $(\frac{\|v^{(i)} - v^{(i-1)}\|}{t_i - t_{i-1}})$ in the summation which differs from the others. Let it be indicated by $\tan\beta$ . If $\tan\beta$ is found in the numerator than it has to be found in the denominator too. But if the deviation is $\beta$ in the numerator, the deviation is $\gamma$ in the denominator and $\gamma \neq \beta$ since the two triangles (figure 7) are not similar. This result is in contradiction with the starting statement, that the elements of the summation in the numerator and in the denominator, are equal. It means that the slopes are equal and the two lines is coincident. QED




*7. Figure* The proof of the linearity

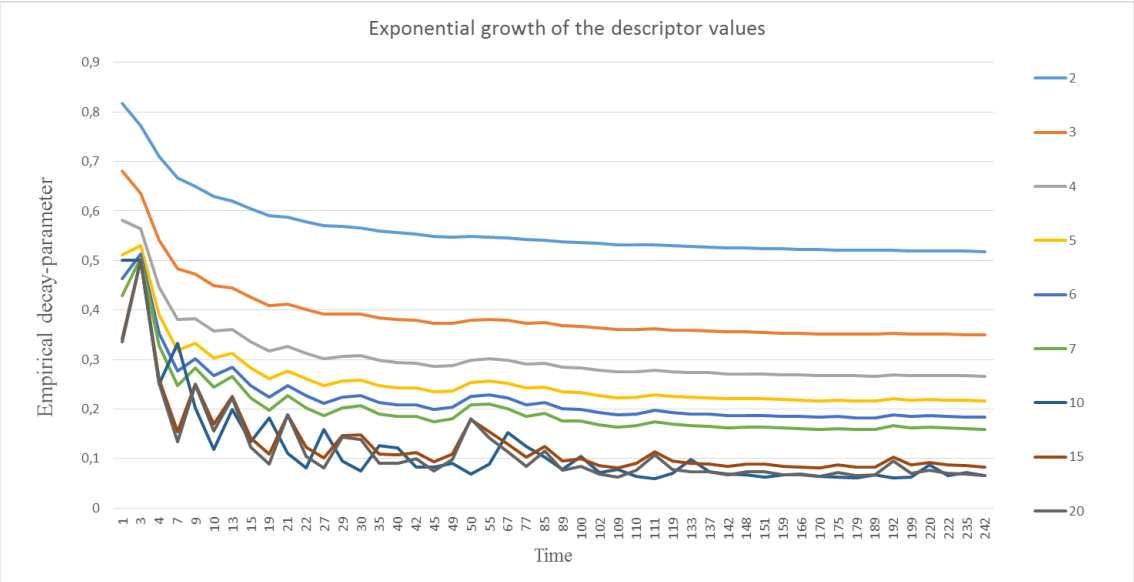
### 4.10.3 Exponantial and logarithmic change

The continuously increasing (or decreasing) change has three basic trends:

1. the exponential, in which the degree of the change is continuously increases
2. the logarithmic or radical, in which the degree of the change is continuously decreases
3. irregular increase

Exponential growth

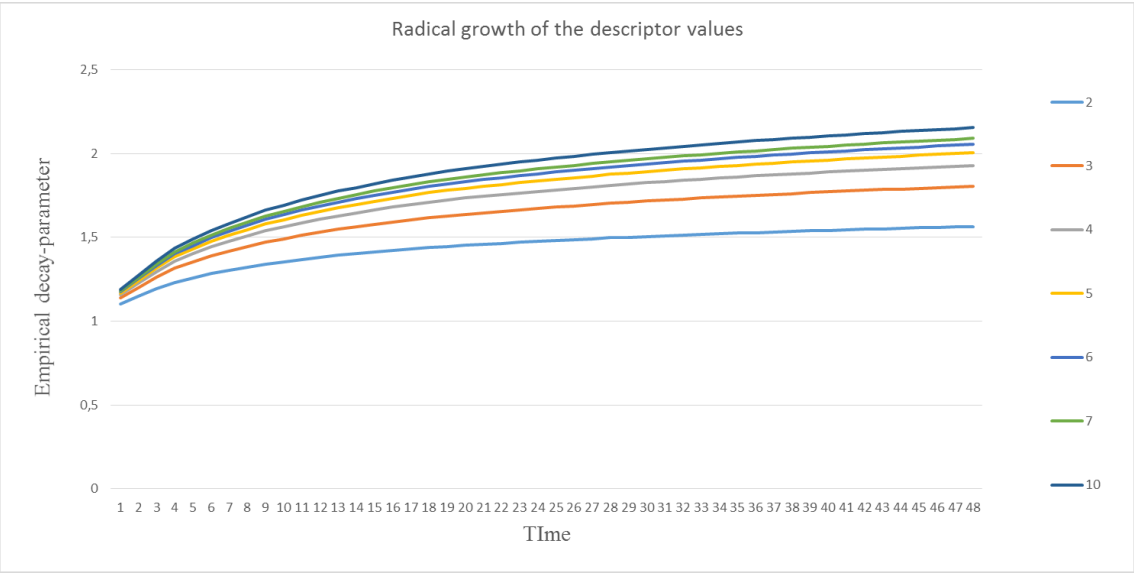
In time-dependent cases, the simulations were performed on the sample set in which the increase of the descriptor values follow the exponential function on different power (2., 3., 4., 5., 6., 7., 10., 15., 20). (Figure 8) The results showed that in case of a not too fast increase (second or third power) the decay function is a monotone decreasing function which has a well-recognizable characteristic. But fast increase (on 10. 15. and 20. power) “disorders” the curve and generates sharp breaks. The decay values remain below 1 and decrease.




*8. Figure* The time-dependent empirical decay-parameter in case of exponential growth of the descriptor based on 50 samples.

Radical growth:

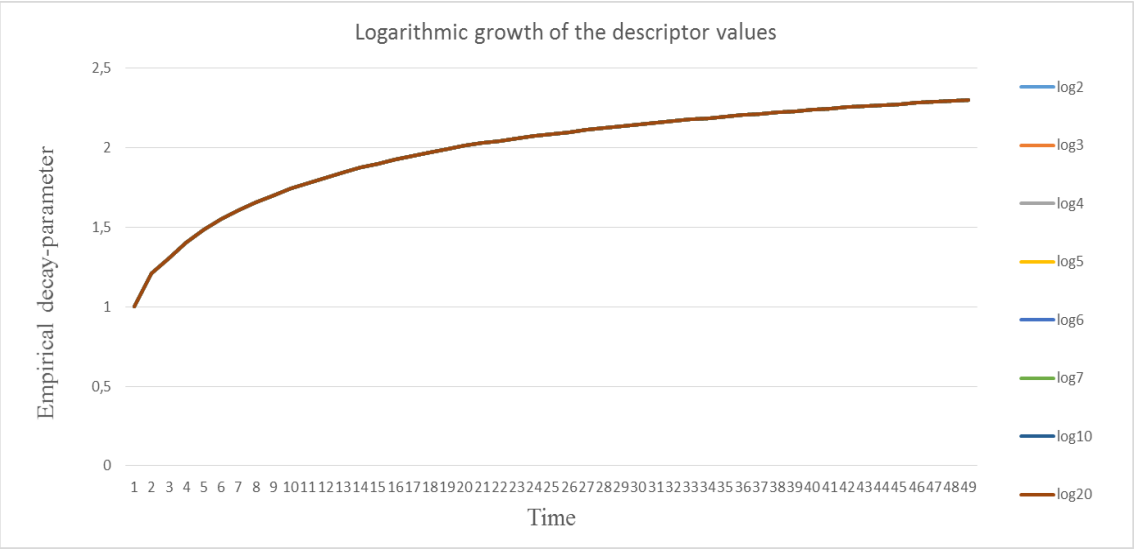
On figure 9, the radical growth of the descriptor values results monotone increasing, smooth curves and the values of the time-dependent, empirical decay-parameter remains above 1. The higher is the index of the radical function the steeper is the empirical decay. The different colors of the curves indicate the different index of the radical function.




*9. Figure* The time-dependent empirical decay-parameter in case of radical growth of the descriptor based on 50 samples

Logarithmic growth

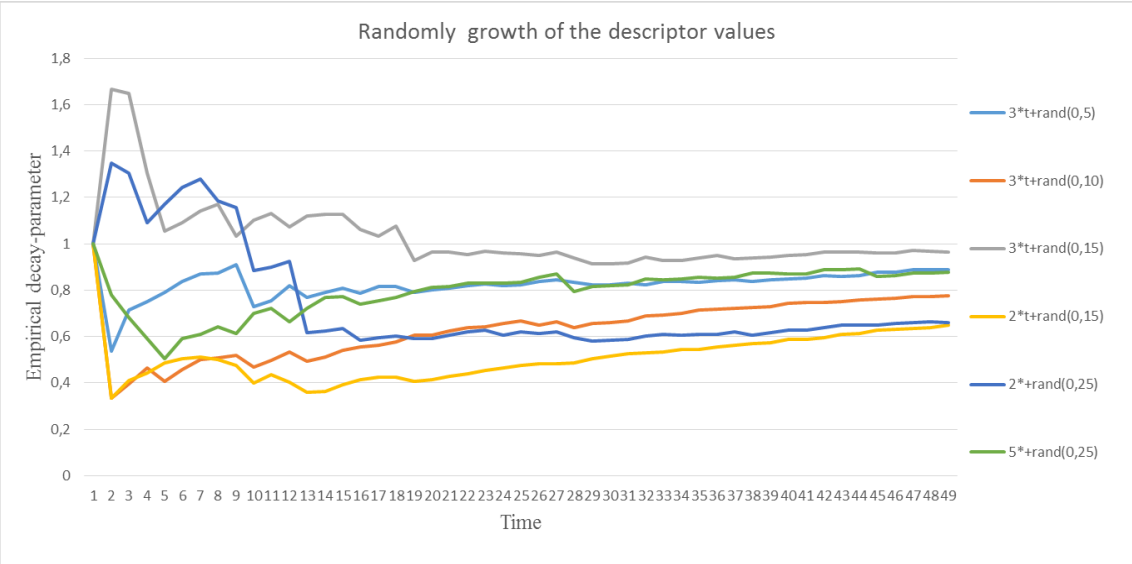
The logarithmic change (figure 10.) in the descriptor values shows a very interesting result; the empirical decay is in all cases the same independently from the base of the logarithmic function. The decay value starts from 1 and monotone increases similarly to the logarithmic curve.




*10. Figure* The time-dependent empirical decay-parameter in case of logarithmic growth of the descriptor based on 50 samples

Randomly growth

In the case of the randomly growth (figure 11) the time-dependent decay follow an especially changeable curve but with increasing size of the sample set the curve becomes smooth and approaches to 1.



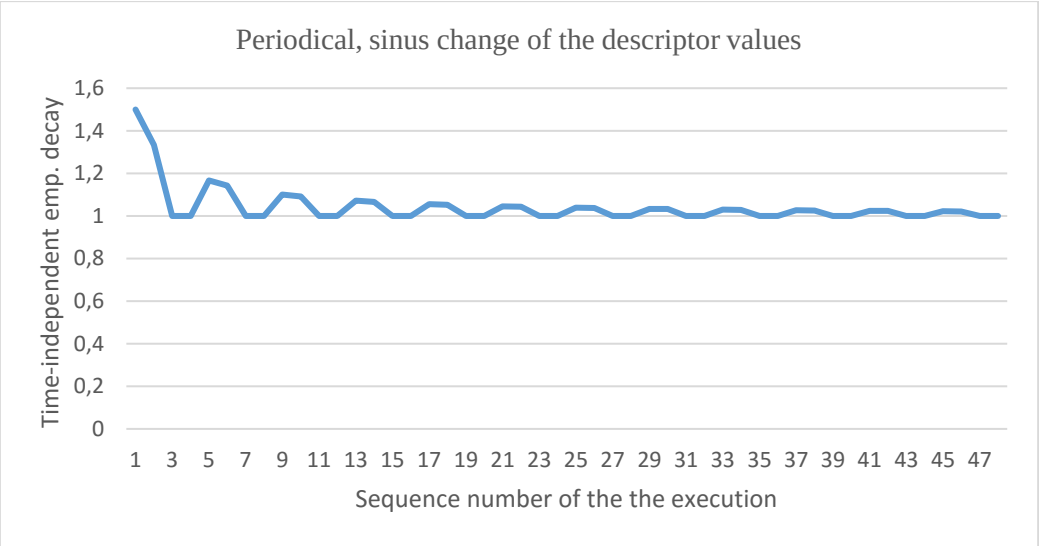

*11. Figure* The time-dependent empirical decay-parameter in case of randomly growth of the descriptor based on 50 samples

### 4.10.4 Fluctuation

The fluctuation of the descriptor value can be either periodical such as sinus or cosines, or randomly when the values randomly move in a predefined interval.

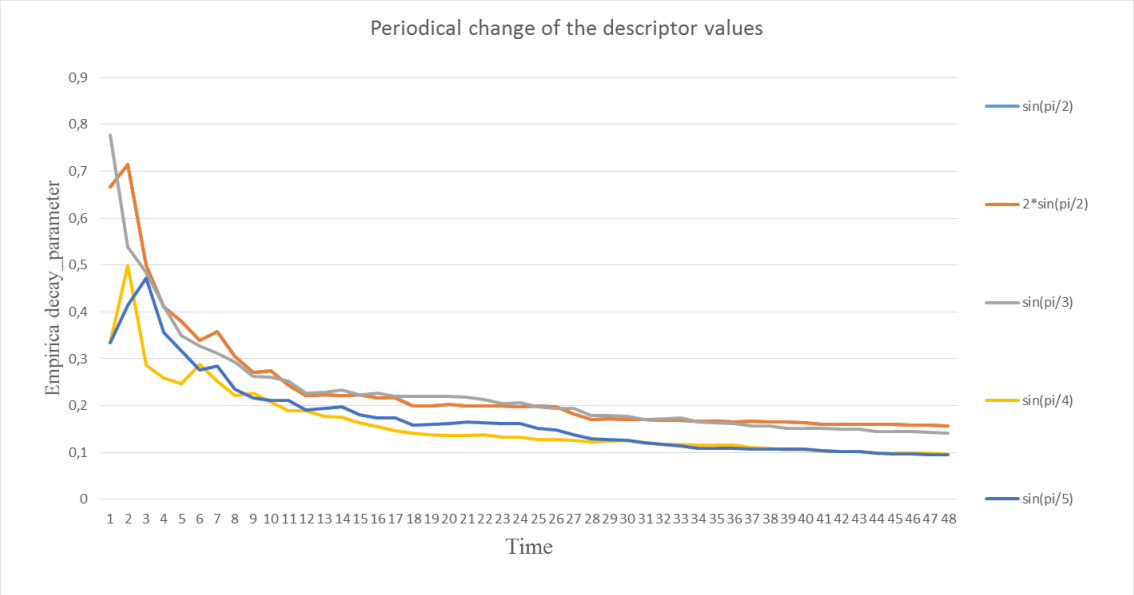
Periodical

In time-independent case (figure 12), if $decay_{emp}(\Delta v_i, s) \approx 1$ than the descriptor values fluctuate about a certain value (the expected value of the descriptors). If the change is periodical and the long of the period is multiple of the sample size, then $decay_{emp}(\Delta v_i, s) = 1$. The curve of the decay also follows an "almost" periodic change which has a minimum of 1, but with the increasing size of the sample set the "waves" become smaller.




*12. Figure* The time-independent emp. decay of the periodically changing descriptor values

In time-dependent case (figure 13), after a few time the decay curve continuously decreases and the values are small near to 0.

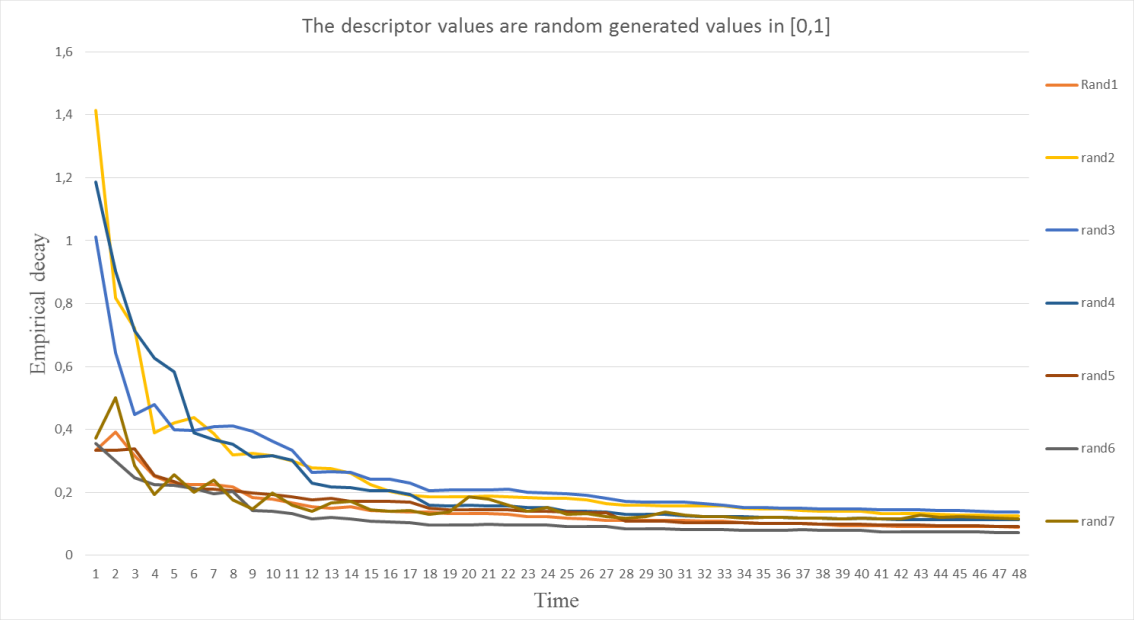



*13. Figure* The time-dependent empirical decay-parameter in case of sinus change of the descriptor based on 50 samples

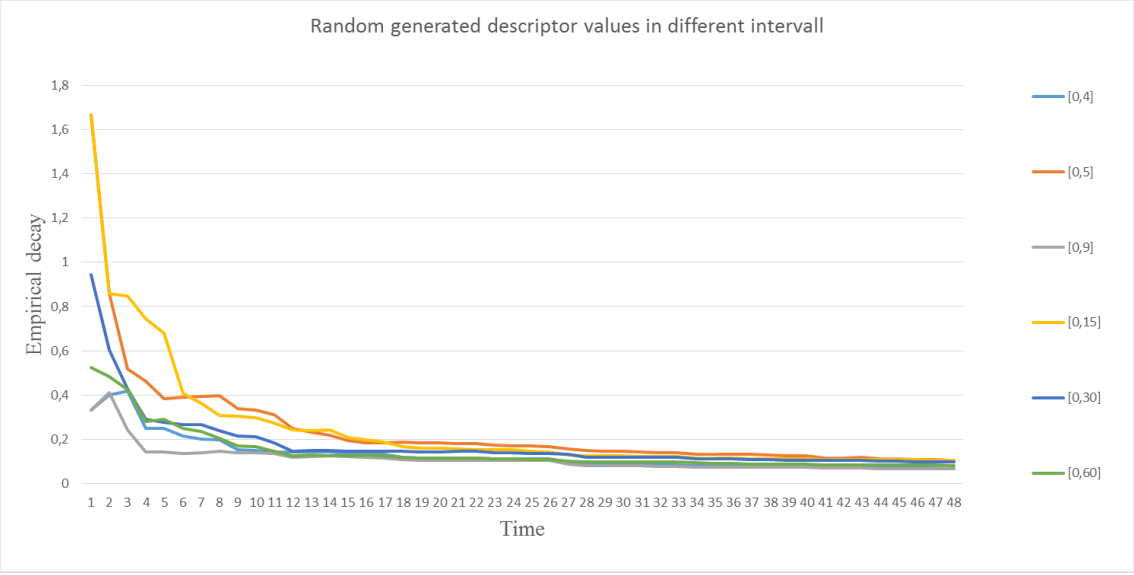
Non-periodical, randomly fluctuation

In time-dependent case the curves are similar to the periodical case. The simulations were performed on samples generated in different ways: Gaussian distribution in the [0,1] interval on

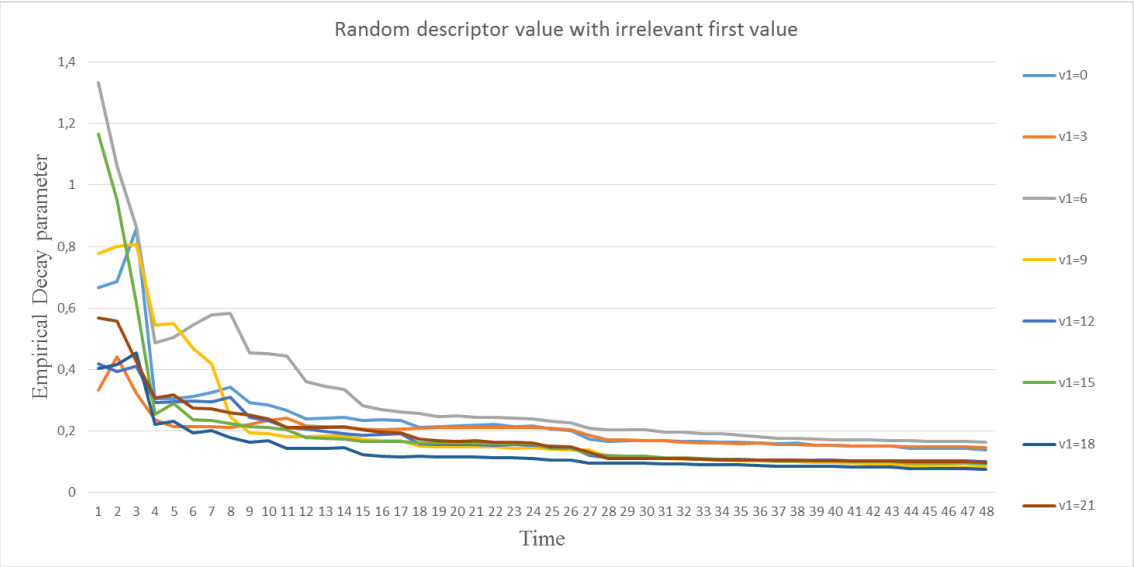
the set of real number (figure 14), Gaussian distribution in different interval on the set of integer (figure 15) and I also investigated the cases, when the first value is irrelevant (figure 16).




*14. Figure* The time-dependent empirical decay-parameter in case of random change in [0,1] interval based on 50 samples



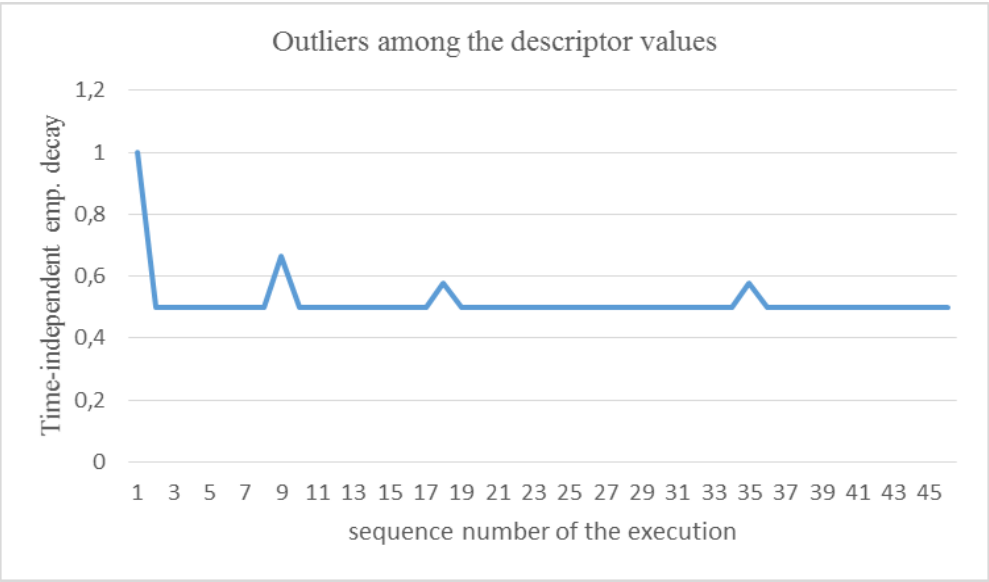

*15. Figure* The time-dependent empirical decay-parameter in case of random change in different interval based on 50 samples




*16. Figure* The time-dependent empirical decay-parameter in case of random change with irrelevant first value based on 50 samples

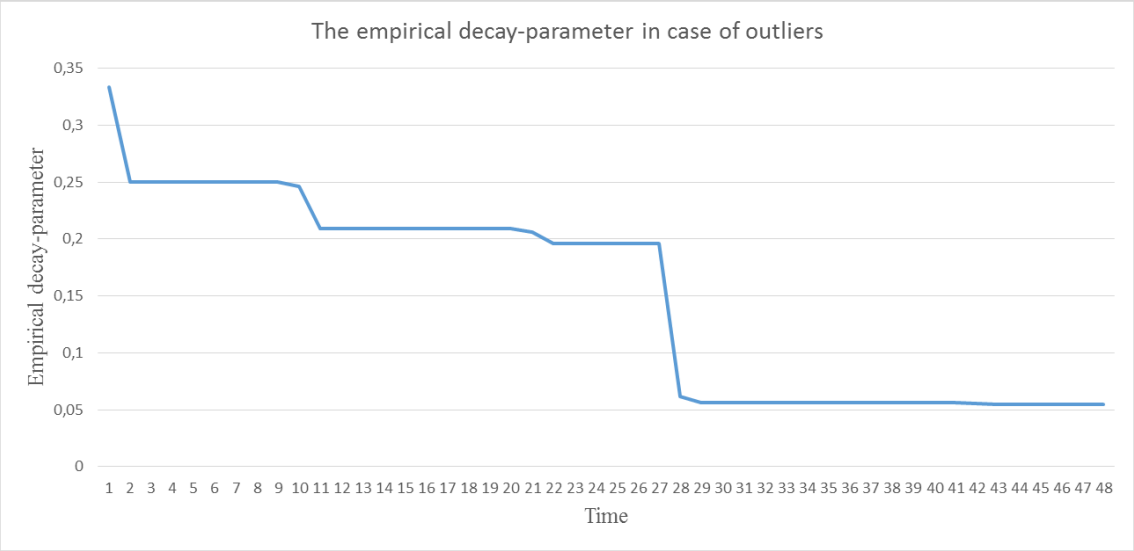
### 4.10.5 Outliers

In time-independent case (figure 17) the empirical decay clearly shows the outliers. If the $decay_{emp}(\Delta v_i, s) \neq 0$ than $\min decay_{emp}(\Delta v_i, s) = 0.5$ which means that the descriptor value does not change but there are some outliers among the values. At the outliers, the decay curve has a sharp break. If the first value is the outlier and the other values are same, the $decay_{emp}(\Delta v_i, s) = s - 2$




*17. Figure* The time-independnet emp. decay when outliers are among the descriptor values

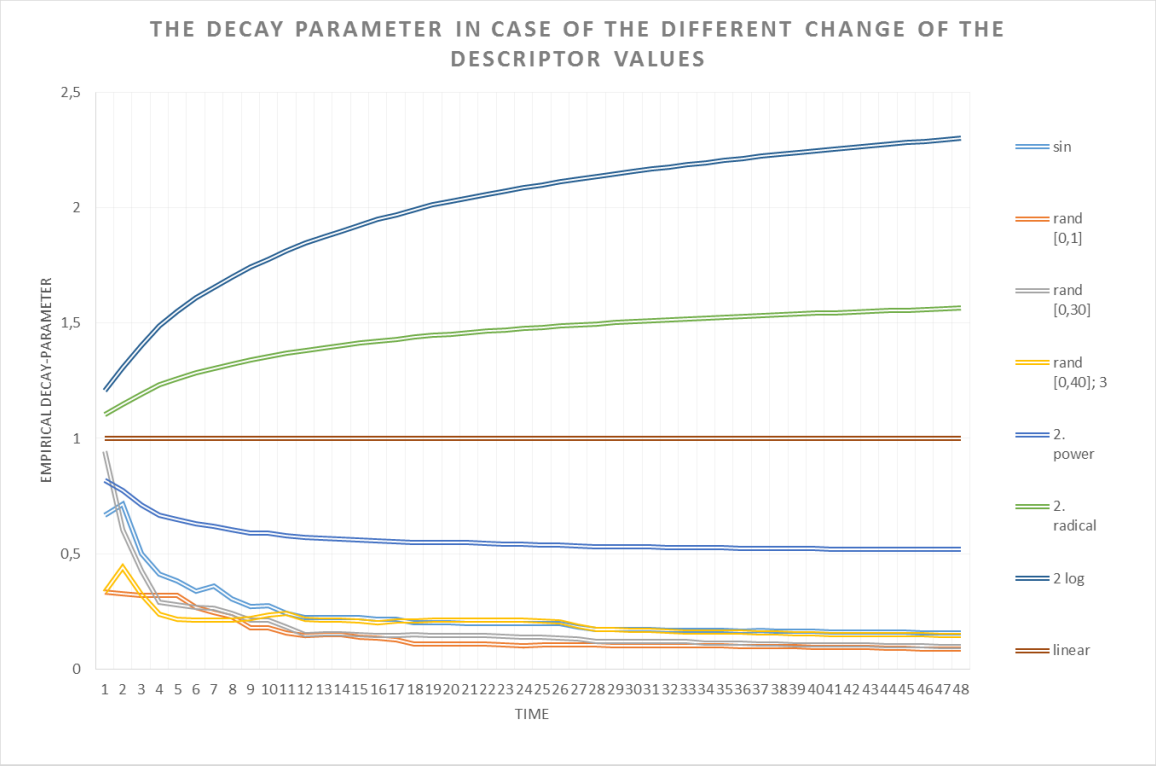
The time-dependent decay is a decreasing step function and the steps show the place of the outliers (figure 18).




*18. Figure* The time-dependent empirical decay-parameter in case of outliers based on 50 samples

Summarizing the results (figure 19), if the sample size is at least 30, the time-dependent empirical decay can unambiguously show:

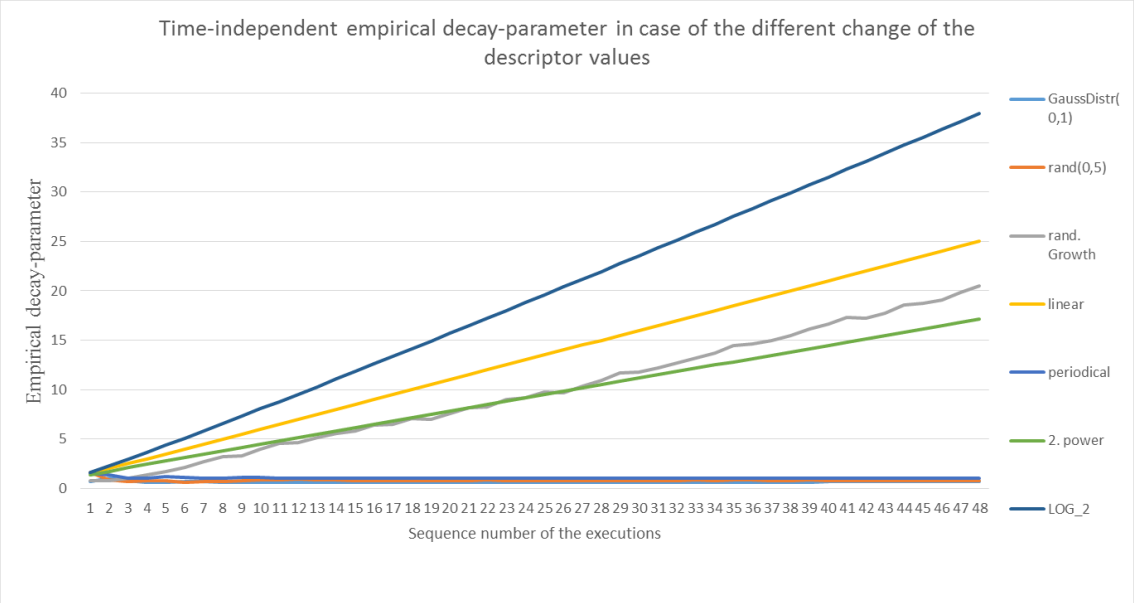
- it linearly diverges from the first descriptor value – $decay_{emp}^{time}(\Delta v_i, s) = 1$,
- it radically or logarithmically diverges from the first descriptor value – $decay_{emp}^{time}(\Delta v_i, s) > 1$,
- it exponentially diverges from the first descriptor value, if the change is not too fast. – The quadratic diverge: $\lim_{s\to\infty} decay_{emp}^{time}(\Delta v_i, s) = 0.5$
- the outliers – it is a step function
- the fluctuating change of the descriptors – the decay values approach to 0.



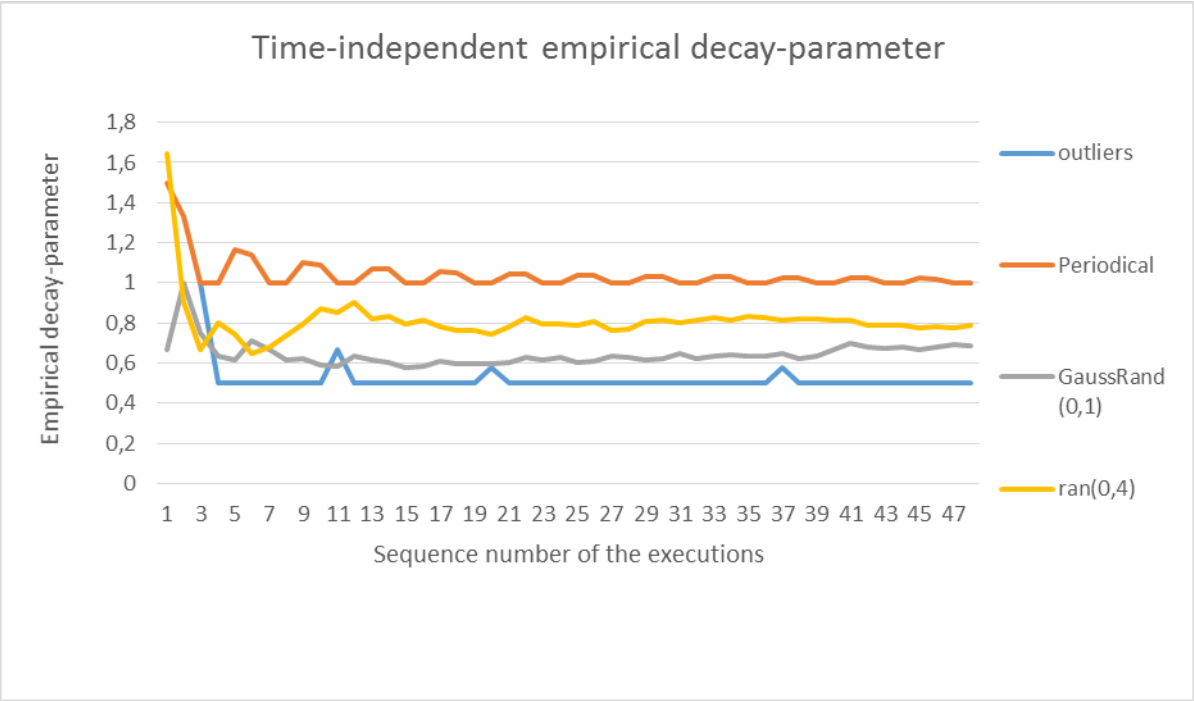

*19. Figure Summary chart about the time-dependent empirical decay in case of different change in the descriptor value based on 50 samples*

The time-independent empirical decay can unambiguously show (figure 20, 21):

- the linear diverge from the first descriptor value – $decay_{emp}(\Delta v_i, s) = 1 + \frac{s-2}{2}$
- the continuously diverge from the first descriptor value – $decay_{emp}(\Delta v_i, s) \gg 1$
- the outliers – $decay_{emp}(\Delta v_i, s) = 0.5$, if the first and the $s$-th descriptor value is not outlier.
- the randomly fluctuating change of the descriptors – $decay_{emp}(\Delta v_i, s) < 1$ or $decay_{emp}(\Delta v_i, s) \approx 1$
- the periodic fluctuating change of the descriptors – $decay_{emp}(\Delta v_i, s) = 1$, if the $s$ is multiple of the period, else $decay_{emp}(\Delta v_i, s) \approx 1$ and $decay_{emp}(\Delta v_i, s) > 1$




*20. Figure Summary chart about the time-independent empirical decay in case of different change in the descriptor value based on 50 samples*




*21. Figure Summary chart about the time-dependent empirical decay when the change is small*

Analyzing the empirical decay, if it shows a well-identified nature of the descriptor values, evaluation can be performed to replace the descriptor when it is unavailable.

## 4.11 Conclusion

In this section, I introduced the basic terms of my research, namely the descriptor-space and the decay-parameter. According to these expressions, I differentiated the theoretical and the empirical approaches. The theoretical descriptor-space contains all the descriptors (descriptor names) needed to reproduce a job. The theoretical decay-parameter describes the nature of the descriptors assuming an "a priori" knowledge – originated from the scientist or from the experiences related to other workflows – about the behavior of the descriptors. But the values of the descriptors can be assigned to them only in occasion of an execution. During more and more executions, the descriptor values originated from the different executions can be stored producing a sample-set and giving the possibilities to the further investigation. Based on this sample-set the empirical decay-parameter can be defined to identify the behavior of the descriptors in an empirical way, in cases of the time-dependent and the time-independent descriptors too. The empirical decay-parameter can clearly show the different types of the change in both cases.

Moreover, based on the descriptor-space I gave the mathematical definitions of the reproducible job and scientific workflow.

## 4.12 Novel scientific results (theses)

Thesis group 1: I have defined and extended the mathematical definition of the reproducible job and reproducible scientific workflow and I have determined the empirical and theoretical decay-parameters of the descriptors.

1. Téziscsoport: Meghatároztam majd kiterjesztettem a reprodukálhatóság matematikai definícióját és egzakt matematikai számítások mentén meghatároztam egy számítási feladat elméleti és tapasztalati romlási mutatóit.

Thesis 1.1
I have introduced the terms of the descriptor-space assigned to the jobs and the theoretical decay-parameter assigned to the descriptors, and I have determine with these two terms the definition of a reproducible job.

1.1 Altézis
Bevezettem a számítási feladatokhoz (job) rendelt deszkriptor-tér és a deszkriptorokhoz tartozó elméleti romlási-mutató fogalmát, melyek segítségével meghatároztam a reprodukálható számítási feladat definícióját.

Related publications: 1-B, 2-B, 3-B, 4-B, 5-B

Thesis 1.2
I have extended the definition of the reproducible job for the scientific DAG (directed acyclic graph) type workflows and based on the definition I have proved that if and only if a job is reproducible, than the scientific workflow is also reproducible.

1.2 Altézis:
Kiterjesztettem a reprodukálható számítási feladat definícióját irányított körmentes gráffal (DAG) reprezentálható tudományos munkafolyamat gráfokra és a definíciók alapján bebizonyítottam, hogy egy tudományos munkafolyamat akkor és csak akkor reprodukálható, ha benne minden job reprodukálható.

Related publications: 1-B, 4-B, 5-B

Thesis 1.3
Based on *s* previous executions of a deterministic job I have defined an empirical decay-parameter assigned to the descriptors of a given job in case of time-dependent and time-independent descriptors and I revealed the relationships between the behavior of the descriptors and the values of the decay-parameters.

1.3 Altézis

Definiáltam *s* futás alapján egy determinisztikus job deszkriptoraihoz rendelt tapasztalati romlási mutatót idő-függő és idő-független deszkriptorok esetére és

feltártam a deszkriptorok viselkedéseinek és a romlási mutatók értékeinek összefüggéseit.

Related publications: 2-B

# 5 INVESTIGATION OF THE EFFECT OF A CHANGING DESCRIPTOR

In this section, I investigated how one or more changing descriptor can influence the result of the job, how far the effect of a changing descriptor can spread and which part of a scientific workflow can be reproduced. Based on the empirical decay-parameter and the sample-set I determine the coverage of a changing descriptor and the reproducible part of the scientific workflow in other words the reproducible subworkflow. Further, I give the method to calculate the theoretical and the empirical probability of the reproducibility.

## 5.1 The impact factor of a changing descriptor for the result

After the behavior of a descriptor is determined, the effect for the result has to be investigated. The underlying question is does the variation of a descriptor influences the result of the job and if yes, in which way? The relationship between the descriptor value and the job result can be determined by calculating the correlation between their deviations. However, correlation can show first of all the linear relationship, the value of the correlation near to 0.5 can shows the relation which is not linear. If the correlation is near to 0, the change of the descriptor value does not connect with the change of the job result.

$$cor\big(\delta(v_i),\delta(R)\big)=\frac{\sum_{j=2}^{S}\left(\delta_{1,j}(v_i)-\overline{\delta_{v_l}}\right)\left(\delta_{1,j}(R_i)-\overline{\delta_R}\right)}{\sqrt{\left[\sum_{i=0}^{S-1}\left(\delta_{j-1,j}(v_i)-\overline{\delta_{v_l}}\right)^2\right]\left[\sum_{i=0}^{S-1}\left(\delta_{j-1,j}(R_i)-\overline{\delta_R}\right)^2\right]}} \tag{5.1.1}$$

If there are many descriptor in the descriptor-space which has non-zero decay-parameter, the correlation cannot be investigated independently in the case of the different descriptor, thus the multi-variate correlation has to be calculated in the following way:

$$cor\big(\delta(v_i),\delta(R)\big)=\frac{\sum_{j=2}^{S}\left(\delta_{1,j}(v_{i_1})-\overline{\delta_{v_l}}\right)\left(\delta_{1,j}(v_{i_2})-\overline{\delta_{v_l}}\right)\ldots\left(\delta_{1,j}(v_{i_L})-\overline{\delta_{v_l}}\right)\left(\delta_{1,j}(R_i)-\overline{\delta_R}\right)}{\sqrt{\left[\sum_{i=0}^{S-1}\left(\delta_{j-1,j}(v_{i_1})-\overline{\delta_{v_l}}\right)^2\right]\ldots\left[\sum_{i=0}^{S-1}\left(\delta_{j-1,j}(v_{i_L})-\overline{\delta_{v_l}}\right)^2\right]\left[\sum_{i=0}^{S-1}\left(\delta_{j-1,j}(R_i)-\overline{\delta_R}\right)^2\right]}} \tag{5.1.2}$$

where L indicates the number of the changing descriptors.

If the distance metric of the result consists of more component, the correlation has to be calculated for every component independently.

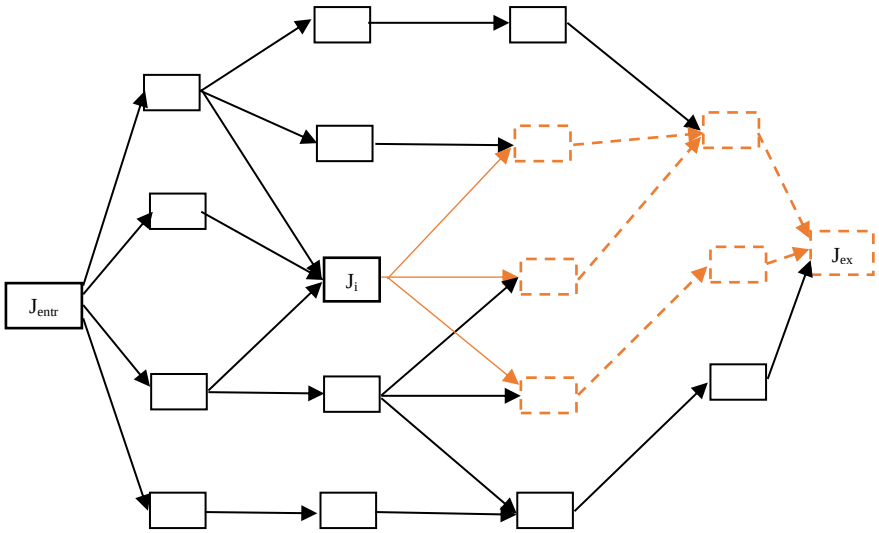
When the impact of a changing descriptor for the job result proves true, this descriptor can prevent to reproduce the job, I call the descriptor crucial descriptor.

## 5.2 Partially reproducible scientific workflows

In this subsection, I deal with the question which part of the scientific workflow is affected by a descriptor which has a non-zero decay parameter. It may be important to determine the reproducible part of the workflow or which part can prevent the reproducibility and to inform the scientist about this fact. In order to formalize the problem, I have introduced some terms and definitions. The first is the forward subworkflow belonged to a given job.

Definition (D6): The *forward subworkflow of a job* $J_i$ is a subgraph of the workflow graph where the entry job is $J_i$ and the exit job is the exit job of the original workflow graph. (Figure 22)

Notation: $SubWF_{J_i}^{forward}$




*22. Figure* The forward sub-workflow of a job $J_i$

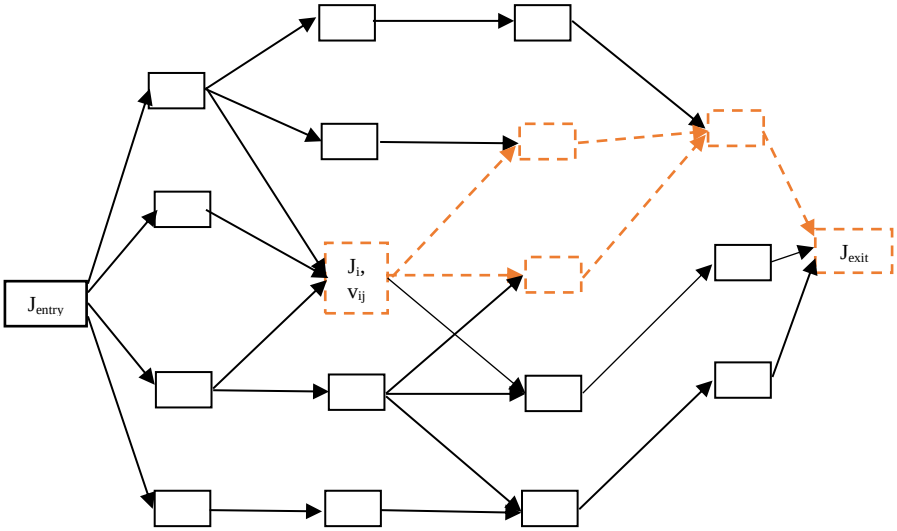
In this way, the coverage of a descriptor also can be defined. Let the $J_i$ is a job of the scientific workflow and $v_{ij}$ is a descriptor of the job $J_i$.

Definition (D9): The *coverage of a descriptor* $v_{ij}$ (*descriptor coverage*) is a forward subworkflow containing that jobs, which are influenced by this descriptor. (figure 23)

Notation: $cvrg(J_i, v_{ij}) = \{J_k \in V \mid Y_k \text{ depends on } v_{ij}\}$

In other words, if a descriptor value changes the results of the jobs contained by the *descriptor coverage* is also changes.

The coverage of a descriptor does not necessarily contain all the path between the given job and the exit job. It can be occurred, that certain successors are affected by the varying descriptor but the others are not.




*23. Figure* The coverage of the descriptor $v_{ij}$

## 5.3 Determination of the descriptor coverage

After the definition has introduced, it is necessary that the coverage of a given descriptor can be determined. Assuming $S$ previous execution of the workflow a sample set $S_{J_i}^{subforward}$ can be created. Let the $J_1, J_2, \dots, J_C$ are the jobs of the $SubWF_{J_i}^{forward}$, where the $C$ is the number of the job in the $SubWF_{J_i}^{forward}$, the $R_1, R_2, \dots, R_C$ are the result of these jobs and the $v_{ij}$ is the descriptor of the job $J_i$. The sample set contains the $v_{ij}$ and the $R_1, R_2, \dots, R_C$ originated from the $S$ previous execution:

$$S_{J_i}^{subforward} = \begin{Bmatrix} v_{ij}^{(1)} & R_1^{(1)} & R_2^{(1)} & \cdots & R_C^{(1)} \\ v_{ij}^{(2)} & R_1^{(2)} & R_2^{(2)} & \cdots & R_C^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{ij}^{(S)} & R_1^{(S)} & R_2^{(S)} & \cdots & R_C^{(S)} \end{Bmatrix} \tag{5.3.1}$$

based on the sample set the empirical correlation matrix can be computed as follows:

$$M_{cor} = \begin{pmatrix} cor\left(\delta(v_{ij}), \delta(v_{ij})\right) & cor\left(\delta(v_{ij}), \delta(R_1)\right) & cor\left(\delta(v_{ij}), \delta(R_2)\right) & \dots & cor\left(\delta(v_{ij}), \delta(R_C)\right) \\ cor\left(\delta(R_1), \delta(v_{ij})\right) & cor\left(\delta(R_1), \delta(R_1)\right) & cor\left(\delta(R_1), \delta(R_2)\right) & \dots & cor\left(\delta(R_1), \delta(R_C)\right) \\ cor\left(\delta(R_2), \delta(v_{ij})\right) & cor\left(\delta(R_2), \delta(R_1)\right) & cor\left(\delta(R_2), \delta(R_2)\right) & \dots & cor\left(\delta(R_2), \delta(R_C)\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ cor\left(\delta(R_C), \delta(v_{ij})\right) & cor\left(\delta(R_C), \delta(R_1)\right) & cor\left(\delta(R_C), \delta(R_2)\right) & \dots & cor\left(\delta(R_C), \delta(R_C)\right) \end{pmatrix} \tag{5.3.2}$$

The coverage of the given descriptor can be determined based on the first row of the correlation matrix. The non-zero values can show the influenced jobs. Based on the coverage the non-reproducibility rate index can be computed which says which part of the scientific workflow is not reproducible.

## 5.4 The reproducibility rate index

With help of the descriptor coverage the reproducible subworkflow can be determined and the reproducibility rate index can be introduced (RRI). If a scientific workflow contains only one job ($J_i$) which has only one descriptor with non-zero decay-parameter ($v_{ij}$) and the $Cvrg(v_{ij})$ can be determined, the reproducibility rate index (RRI) can be expressed as:

$$RRI_{single} = \frac{|Cvrg(v_{ij})|}{|V|} \tag{5.4.1}$$

where $|V|$ and $\left|Cvrg(v_{ij})\right|$ denote the number of the jobs in the workflow graph.

If there are more descriptors in the whole workflow which have non-zero decay-parameter the union of the descriptor coverages has to be determined. The union of a collection of subworkflow is the subworkflow of all distinct jobs in the collection. In this way the expression (5.4.1) can be extended as follows:

$$RRI_{multiple} = \frac{\left|\cup_{i=1}^{L} \cup_{j=1}^{M_i} Cvrg(v_{ij})\right|}{|V|} \tag{5.4.2}$$

where the $L$ $(L \leq N)$ is the number of the jobs which have descriptor with non-zero decay-parameter and $M_i$ ($M_i \leq K_i \; \forall i \in [1, N]$) is the number of the non-zero decay-parameter in the job $J_i$.

## 5.5 Determination of the reproducible subworkflow

Based on the descriptor coverage the reproducible subworkflow of the given SWf can be determined by omitting the coverages of the crucial descriptors. (Figure 24)

```
G(V,E)==SWF;
for i=1 to N
    for j=1 to Ki
        if decay(vij)≠0 than determine SubWF_Ji^forward
          m== "the number of jobs in SubWF_Ji^forward„
          calculate M_corr(vij)_(m+1)x(m+1);
          cvrg(vij)=={};
          for k=1 to m+1
            if M_corr_1_k≠0 than cvrg(vij)==
     cvrg(vij)UJk
            V=V\Jk,
            endif;
        endfor;
        endif;
    endfor;
endfor;
```

*24. Figure The pseudo code of the determination of the rperoducible part of the SWf*

### 5.6 Reproducibility by substitution

In this subsection we investigate the case when one or more of the descriptors is continuously changing in time and they influence the job result. It can occur for example when a job get input from a database which continuously collects more and more data from a sensor networks, consequently the database is also greater and greater. Another example may be the operation related descriptors which based on some actual state of the system. Beyond the special tool developed for this purpose may be necessary to store this descriptor value, the value can be change continuously concerning to the state of the system, such as time or free available memory. Typically if a job has a descriptor described above, it cannot be reproduced. The ultimate goal is to give method, which helps to solve this issue by substituting the descriptor or the job result or by evaluating the deviation of the result based on the changing descriptor. An existing relationship between the deviation of the descriptor value and the deviation of the job result can give the possibility to substitute or evaluate the deviation of the result when the original descriptor value is changed or unavailable. The method, and the parameters of the method can be also stored in the repositories together with the scientific workflow and if the re-execution of the workflow fail because of that descriptor, the result still can be reproduced.

In order to achieve my goal I introduced another two terms, namely the substitutional and the approximative reproducibility referring to that case, in which the decay parameter of the descriptors changes in time but the variation of the result can be determined or evaluated based on the variation of the given descriptor. Two options can be differentiated: the first one is that the variation of the descriptor value is known and it can be described by a mathematical function; and

the second one is that the variation of the descriptor value is unknown but an approximation can be performed which fit to the curve of the change. In both cases a sample set is necessary which is based on provenance information originated from *S* previous executions.

Definition (D.5.6.1): The $J_i$ job is *reproducible by substitution*, if the descriptor space $D_{J_i} = \{v_{i1}, v_{i2}, \dots, v_{iK_i}\}$ is known, $k \in [1, 2, \dots, K_i]: Vary_{ik}(\Delta t, v_{ik})$ and based on the function $Vary_{ik}(\Delta t, v_{ik})$ a new $Vary_i^*(\Delta t, v_{ik})$ can be unambiguously determined which give the variation of the result depending on the given descriptor.
In other words if $JOB_i(t_0, v_{i1}, v_{i2}, \dots, vary_{ik}(t_0, v_{ik}), \dots, v_{iK_i}) = R_i(t_0)$ than

$$JOB_i(t_0 + \Delta t, v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots, v_{iK_i}) = Vary_i^*(\Delta t, v_{ik}) = R_1(\Delta t)$$

Notation: $JOB_i^{substitute}(Vary_i^*(\Delta t, v_{ik}))$

Definition (D.6.2): The $J_i$ job is *reproducible by approximation*, if the descriptor space $D_{J_i} = \{v_{i1}, v_{i2}, \dots, v_{iK_i}\}$ is known, $k \in [1, 2, \dots, K_i]: Vary_{ik}(\Delta t, v_{ik})$ and based on the function $Vary_{ik}(\Delta t, v_{ik})$ an approximator $\Psi_{ik}(\Delta t, \delta(v_{ik}))$ can be determined to evaluate the deviation of the job result.
In other words if $JOB_i(t_0, v_{i1}, v_{i2}, \dots, vary_{ik}(t_0, v_{ik}), \dots, v_{iK_i}) = R_1(t_0)$ than
$JOB_i(t_0 + \Delta t, v_{i1}, v_{i2}, \dots, vary_{ik}(\Delta t, v_{ik}), \dots, v_{iK_i}) \approx \widetilde{R_\iota}(t_0) + \Psi_{ik}(\Delta t, \delta(v_{ik}))$
Notation: $JOB_i^{appro}(\Psi_{ik}(\Delta t, v_i))$

The definition D.6.1 says that the result of the job can be determined exactly by a function of the $v_{ik}$ descriptor while in the second case (D.6.2) an approximator can be found to estimate the deviation of the result based on the deviation of the given descriptor.

## 5.7 Determination of the substitutional and the approximation function

Since the crucial descriptors can belong to different types, the approximation method which evaluate the change of the descriptor or the deviation of the result also can be various. The relationship between the crucial descriptor and the result can follow different types of function such as linear, quadratic or higher order, exponential, logarithmic or trigonometric etc.

The substitutional function, if exist, can be determined based on the empirical decay-parameter. The investigation of the empirical decay-parameter presented in section 4.10 showed the evaluability of the changing descriptor.

To find an approximator which can evaluate the deviation of the job result depends on the impact factor defined in the subsection 5.1. The correlation between the descriptor and the deviation of the result can determine the evaluability of the deviation. The simplest case, when the correlation is near to 1, since the linear relationship can be simply evaluated by, for example linear regression. If the correlation less than 1, non-linear evaluation has to be found. Applying the empirical decay-parameter also for the job result, the nature of the result also can be identified which can help to find the appropriate approximator. If the correlation is near to 0 there is no relationship between the change of the descriptor and the deviation of the job thus this crucial descriptor cannot be compensated with approximation

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.

## 5.8 Reproducible scientific workflows with the given probability

In this section I introduce a probability value assigned to the descriptors to determine how likely the value is changed or unavailable. This probability can be generated based on the theoretical decay-parameter, if the user knows the nature of the descriptor or it can be originated in empirical way based on the sample set. With help of the probability value, the probability of the reproducibility can be determined which is an essential information for the scientists on one hand during the design phase when the scientific workflow is in progress to become reproducible, on the other hand, when the scientist intend to reuse a workflow from a repository.

Many investigations revealed the problem caused by volatile third party resources (Zhao & al, 2012), when the reproducibility of workflows became uncertain. The third party services or any external resources can be unavailable during the years. If the decay of the resources and the probability distribution function can be identified and we can determine its probability distribution function we can predict the behavior of the workflow on occasion of a re-execution at a later time. Sometimes the users may have to know the chance of the reproducibility of their workflow. Assuming that the probability distribution of the third party service is known or assumable we can inform the users about the expected probability of the reproducibility.

## 5.9 Theoretical probability

To formalize the problem, first, we have separated the *M*i descriptors of a given job Ji which depend on external or third party resources and its decay-parameter, which is a probability distribution function given as follows: $F_{i1}(t), F_{i2}(t), \dots, F_{iM_i}(t)$. The rest of the descriptors have

zero decay-parameter. In this case, at time t0, a given descriptor's value $v_{ij}(d_{ij})$is available with a given probability (for the sake of the easier comprehensibility hereafter we omitted the i index referred to the *i*-th job of a given scientific workflow):

$$F_1(t_0) = p_1^{(t_0)}, F_2(t_0) = p_2^{(t_0)}, \dots, F_M(t_0) = p_M^{(t_0)} \quad (5.9.1)$$

Let us assign to the job Ji a state vector $y_i = (y_{i1}, y_{i2}, \dots, y_{iM_i}) \in \{0,1\}^{M_i}$, in which the $y_{ij} = 1$ , if the jth descriptor of the job Ji is unavailable. In this way the probability of a given yi state vector can be computed as follows:

$$p(y) = \prod_{j=1}^{M} p_j^{y_j} \left(1 - p_j\right)^{1-y_j} \quad (5.9.2)$$

In addition a time interval can be given during which the descriptor is available with a given probability P.

Since we assume the independency of the descriptors the cumulative distribution function of the availability referred to the job Ji can be written as follows:

$$F_i(t) = \prod_{j=1}^{M} F_{ij}(t) \quad (5.9.3)$$

Based on the cumulative distribution the probability of the reproducibility can be determined in the following way:

$$P_{theo}(JOB_i^{repro}(x < t)) = 1 - \prod_{j=1}^{M} F_{ij}(x) \quad (5.9.4)$$

$$P_{theo}(SWF^{repro}(x < t)) = \prod_{n=1}^{N} \left(1 - \prod_{j=1}^{M_i} F_{nij}(x)\right) \quad (5.9.5)$$

where *N* is the number of the jobs and $M_i$ is the number of the descriptors referred to the job Ji which has the decay-parameter determined by the probability distribution function.

## 5.10 Empirical probability

Based on the sample set many useful information can be collected about the descriptors. The probability of their change or unavailability may be also an important characteristic of the scientific workflows which can support the reproducibility analysis and also the scientist's community to create or reuse a reproducible workflows. Therefore based on the previous executions of the SWf, the relative incidence of the change/unavailability can be assigned for

every descriptor. In this way, assuming the independency of the descriptors, the probability of the descriptor-space-changing can be calculated as follows:

$$P_{emp}(D_i \; is \; changed) = \prod_{j=1, p_j \neq 0}^{K_i} p_{ij}^{emp} \tag{5.10.1}$$

where $p_{ij}^{emp}$ is the relative incidence of that the descriptor value $j$-th in the job $J_i$ is changed or unavailable.

In the expression (6.2.1) only the crucial descriptors assist, which can influence the job result. If a descriptor value did not change at all, its relative incidence is 0. Consequently, if the change of the descriptor-space means the non-reproducibility, the probability of the reproducibility of a job can be written as:

$$P_{emp}(\mathrm{JOB}_i^{\,repro}) = 1 - \prod_{j=1, p_j \neq 0}^{K_i} p_{ij}^{emp} \tag{5.10.2}$$

Assuming the independency, the expression (6.2.2) can be easily extended for scientific workflows:

$$P_{emp}(\mathrm{SWF}^{repro}) = \prod_{n=1}^{N} \left(1 - \prod_{j=1, p_j \neq 0}^{K_i} p_{nij}^{emp}\right) \tag{5.10.3}$$

If the independency cannot be assumed, the expression (6.2.2) has to calculate for coverage of the crucial descriptors and then the independency of the coverage can be assumed.

## 5.11 Conclusion

In this section, I investigated the effect of the changing descriptors for the job result and for the forward sub-workflow. With help of the sample set and the distance-metric interpreted on the descriptor values and on the results, I determined the relationship between the deviation of the descriptor values and the deviation of the results. Calculating the empirical correlation between these deviations, the type of the relationship can be identified. Knowing the relationship, the reproducing of the job can be replaced by the evaluation of the deviation when the critical descriptors are not available or the reproduction cannot be performed. Moreover, with help of the empirical correlation, the coverage of a critical descriptor can be determined. Based on the coverage, the reproducible part of the scientific workflow also can be given.

Additionally, the probability of the reproducibility also is an important information to help the scientist to decide whether a workflow is worth reuse or not. If the theoretical decay-parameter

and the probability distribution function is given, the theoretical probability can be calculated assuming the independency of the descriptors and the jobs. Else, the empirical probability should be used. Obviously, the assumption of the independency limits the number of the workflows which fulfil this requirement thus further investigation should be planned in the future work.

## 5.12 Novel scientific results (theses)

Thesis group 2: Based on simulations and on the empirical decay-parameters I have investigated and determined the behavior, the coverage of the changing descriptors and the feasible approximation of the result deviation.

2. Téziscsoport: Szimulációk segítségével megvizsgáltam és meghatároztam egy számítási feladat változó deszkriptorainak viselkedését, hatókgráfját és a számítási feladat eredményén értelmezett eltérés közelíthetőségét.

Thesis 2.1
Based on a sample set originated from *s* previous executions I have defined and realized a method to determine that subgraph of a given scientific (DAG) workflow, in which the job results are influenced by a given descriptor.

2.1 Altézis
Kidolgoztam egy eljárást, mellyel ismert deszkriptor-tér esetén és *s* futásból származó provenance adatból nyert mintahalmaz alapján meghatározható egy tudományos munkafolyamat gráf azon részgráfja, melyben egy adott számítási feladat deszkriptorának hatása észlelhető.

Related publications: 1-B,

Thesis 2.2
I have introduced the term reproducibility rate index (RRI) to calculate how big part of the scientific workflow is reproducible and I have developed a method to determine the reproducible sub-graph of a partially reproducible scientific workflow represented by a DAG.

2.2 Altézis
Bevezettem egy reprodukálhatósági arányszámot (RRI), amely meghatározza, hogy a tudományos munkafolyamat mekkora részben reprodukálható és kidolgoztam egy eljárást a DAG-gal reprezentálható, részlegesen reprodukálható tudományos munkafolyamat reprodukálható részgráfjának meghatározására.

Related publications: 1-B, 7-B

Thesis 2.3

I have defined the impact factor term of a changing descriptor set based on *s* previous executions, and I have determined the feasible approximation of the result deviation.

2.3 Altézis
Definiáltam a változó deszkriptor halmaz, *s* futás alapján számított *impakt faktorának* fogalmát és szimulációk alapján meghatároztam az eredmény változásának közelíthetőségét.

Related publications: 1-B, 2-B

Thesis 2.4
Based on the theoretical decay-parameter and the empirical probability calculated according to the *s* previous job executions, I have defined and proved the theoretical and the empirical probability of the reproducibility concerning to a given scientific workflow assuming that the descriptors and the jobs are independent.

2.4 Altézis
Az elméleti romlási mutató és az s futásból számított tapasztalati valószínűség segítségével definiáltam és bizonyítottam egy tudományos munkafolyamat reprodukálhatóságának elméleti és tapasztalati valószínűségét abban az esetben, amikor a deszkriptorok és a számítási feladatok egymástól függetlenek.

Related publications: 2-B, 5-B

# 6 THE REPRODUCIBILITY METRICS

In this section I give the metrics of the reproducibility, which helps to measure the cost of the reproducibility, in other words, how extra-cost – it can be extra computation, extra reproducing time or extra storage to store the descriptor values or other parameters – is necessary to reproduce the scientific workflow. In order to achieve this goal, I introduce a so-called repair-cost assigned to the descriptors. In this way, the Average Reproducibility Cost (ARC) can be calculated and that how likely the reproducibility cost is over then a predefined threshold. I call this probability Non-reproducibility Probability (NRP). Since the computation complexity of this metrics exponentially grows with the size of the descriptor-space I give evaluation methods to be able to calculate them in polynomial time. Finally, I classify the scientific workflows from the point of view of the reproducibility.

## 6.1 The "repair-cost"

The ultimate goal of my research is to make the workflow reproducible in sense that I intend to set out different methods which help reproduce an originally non-reproducible job. In other words, if a job has a time-based operation-related descriptor which make doubtful or questionable the re-execution of the job a method is required to eliminate or replace it and reproduce the job without the descriptor in dispute. For example, according to a descriptor which has the decay-parameter determined by a *vary* function, the value of the descriptor may be evaluated or even the result of the descriptor may be evaluated as well. In certain cases, the job cannot be made reproducible in any circumstance, only a given subworkflow of the original workflow can be reproduced or even only the probability of the reproducibility can be determined. To be able formalize and measure this extra work needed to reproduce a job I assigned to every descriptor a cost index which is a real number in the interval [0, 1]. The cost-index can refer to extra time, computation or storage, etc.

The following table represents the descriptor-space extended by the cost index and the probability:

## 6.2 The reproducibility metrics

| Descriptor's name | Descriptor's value | Decay-parameter | Cost | Cost probability |
|---|---|---|---|---|
| $d_1$ | $v_1 = v_1(t)$ | $decay(v_1)$ | $c_1$ | $p_1$ |
| $d_2$ | $v_2 = v_2(t)$ | $decay(v_2)$ | $c_2$ | $p_2$ |
| … | … | … | … | |
| $d_K$ | $v_K = v_K(t)$ | $decay(v_K)$ | $c_K$ | $p_K$ |

*4. Table:* The extended descriptor-space of a given job

It may be important to inform the user about the conditions of reproducibility of his workflow or even the cost of the reproducibility. Introducing the cost-index assigned to the descriptors the question may be what will be the expected cost to reproduce the scientific workflow. Since only the probabilities of the cost are available, the exact computation is not possible but the expected cost can be computed. Additionally it can be also determined how likely the reproducibility cost is over a predefined threshold in other words whether the reproducibility cost worth the "invested work" or not. Conversely I determined two measures: the Average Reproducibility Cost (ARC) and the Non-Reproducibility Probability (NRP).

## 6.3 Average Reproducibility Cost

In order to perform the computation of the ARC, I introduced some additional expression. Based on the descriptor space we can create a binary state vector of the job $J_i$:

$$\mathrm{y}_i = (y_{i1}, y_{i2}, \dots, y_{iK_i}) \in \{0,1\}^{K_i} \tag{6.3.1}$$

in which the $y_i = 1$ with probability $p_i$ , if the $i$th descriptor value of the job $J_i$ is unavailable or changed but with the help of the cost assigned to it the job can be reproduced. In this way the probability of a given y state vector can be computed as follows:

$$p(\mathrm{y}) = \prod_{j=1}^{M} p_j^{y_j} \left(1 - p_j\right)^{1-y_j} \tag{6.3.2}$$

The cost of a state vector is the following:

$$g(\mathrm{y}) = \sum_{i=1}^{K} c_i \tag{6.3.3}$$

The ARC assigned to the job Ji expressed as

$$ARC_{J_i} = \sum_{y \in Y} g(\mathrm{y}) p(\mathrm{y}) \quad (6.3.4)$$

and the ARC assigned to the scientific workflow SWF expressed as

$$ARC_{SWF} = \sum_{j=1}^{N} E\left(g(\mathrm{y})\right) \quad (6.3.5)$$

## 6.4 Non-reproducibility Probability (NRP)

When the overall cost of making the workflow reproducible is greater than a predefined C cost, generally the reproducibility do not worth the time and the cost to perform it. In other words in that case the workflow is not reproducible. If the users are informed about this fact, they have the possibility to modify their workflow or to apply other virtualization tools (Virtual Machin).

The NRP of a given job Ji is expressed as

$$P(g(\mathrm{y}_i) > C) = \sum_{Y: g(\mathrm{y}) > C} p(\mathrm{y}_i) \quad (6.4.1)$$

where C is a given level of the reproducibility cost and

and the NRP of a sientific workflow SWF is expressed as

$$NRP_{SWF} = \prod_{j=1}^{N} P(g\left(\mathrm{y}_{J_j}\right) > C) \quad (6.4.2)$$

The mathematical model described in the previous subsection is similar to the model of the network reliability analysis which investigate the availability and the reliability of a communication network infrastructure such as SDH, IP or ATM.. In that model the network component such as switches, routers etc. are represented by an $N$ dimensional vector $\mathrm{y} \in Y = \{0,1\}^N$, where $N$ is the number of the network components and the vector element $y_i = 0$, if the $i$-th network component is operational and $y_i = 1$ with the probability $p_i$, if the $i$-th network component is malfunctioning. Additionally, a measure of loss is given by $g(\mathrm{y})$ ($g: Y \to \mathbb{R}$), which expresses the loss of system performance due to a failure scenario represented by vector y. The two main reliability measures are the following:

1. $\mathrm{E}g(\mathrm{y}) = \sum_{\mathrm{y} \in Y} g(\mathrm{y}) p(\mathrm{y})$ (6.1)
2. $P(g(\mathrm{y}) > C) = \sum_{\mathrm{y}: g(\mathrm{y}) > C} p(\mathrm{y})$ (6.2)

where $C$ is a given level of degradation in performance.

This two measure can be translated to my approach of the reproducibility such as Average Reproducibility Cost (ARC) and the Non-Reproducibility Probability (NRP). Concerning to the

reproducibility the network components are the descriptors and the measure of loss is the repair cost. A descriptor is "malfunctioning" if the descriptor value is changed or unavailable. ARC means the expected reproducibility cost which is necessary to make the scientific workflow reproducible. If the process of making the workflow reproducible is over a predefined threshold, the reproducibility do not worth the "invested work" namely the extra cost which is provided by the method and extra tools needed to the reproducibility.

It follows from the definitions of ARC and NRP is clear that exact computation bases on calculating $g(\mathrm{y})$ for each possible binary state vector, which entails $2^N$ computations. Since the number of the descriptors referred to a single job can fall into the range from a few hundred to a couple of thousand even in point of the whole scientific workflow which typically has hundreds of jobs, 'taking a full walk' in the state space for calculating the reproducibility measures is clearly out of reach. Therefore, I have to calculate ARC and NRP approximately by using an estimation function, which is based on only a few samples $\{(\mathrm{y}_i, g(\mathrm{y}_i)),\ i = 1, \dots, S\}$ taken from the state space. Of course, the underlying question is how to find the most optimal or typical samples which furnish the most accurate estimation despite the small number of samples. There are many classical methods of the reliability analysis, which define the method of sampling and the estimation of the $\mathrm{E}(g(\mathrm{y}))$ is performed based on the samples:

a) The Monte Carlo method, which is based on a random samples
b) The importance sampling, which tries to tailor the sample-taking procedures to the most relevant samples
c) The Stratified sampling, which accelerates the Monte Carlo simulation by grouping the samples into different classes.

Another approach is the estimation by transforming method. The main idea in this method to find an appropriate transformation which maps the loss function $g(\mathrm{y})$ into a function $f(\mathrm{y}, \mathrm{w})$ which lends itself for easy statistical calculations. The vector w denotes the free parameters which can be subject to learning in order to fit the curve $f(\mathrm{y},\mathrm{w})$ to the specific loss function $g(\mathrm{y})$.

Namely, the evaluation is done in three steps:

1. generating a sample set $\tau^{(N)} = \{(\mathbf{y}_k, g(\mathbf{y}_k)), k = 1,..., N\}$
2. finding $\mathrm{w}_{\mathrm{opt}}$ by minimizing the approximation error over the sample set, where the approximation error is defined as follows:

$$w_{opt} = \min_{w} \sum_{k=1}^{N} \left(g(\mathbf{y}_k) - f(\mathbf{y}_k, w)\right)^2$$

3. calculating the expected value $\mathrm{E}\left(f(\mathbf{y}, \mathbf{w}_{opt})\right)$ by analytical tools.

This method proves to be a viable alternative to the classical statistical estimations if the learning algorithm is not too complex and if it does not require an over-excessively large training set to obtain a good approximation.

## 6.5 Evaluation of the Average Reproducibility Cost

The average reproducibility cost (ARC) is a good starting point to inform the scientists about the reproducibility conditions of their workflow. In view of ARC they can consider the possibilities and the thought of a possible modification. After all in certain cases the size of the descriptor-space can be enormous in addition it can exponentially increase. If this computation is sticky or a real-time reply is needed an evaluation can be applied.

The universal approximation capabilities of neural networks have been well documented by several papers (Hornik & al., 1989), (Hornik & al., 1990), (Hornik, 1991) Therefore, it seems plausible to construct $f(\mathrm{y},\mathrm{w})$ as a neural network. In order to fulfil the condition which enforces the analytical calculation of $E(f(\mathrm{y},\mathrm{w}))$ the choice fell on Radial Basis Function (RBF) networks:

$$f(\mathbf{y},\mathbf{w}) = \sum_{k=1}^{K} w_k \varphi\left(\left\|\mathbf{y}-\mathbf{y}_k\right\|\right) \tag{6.5.1}$$

where φ is similar to the Gaussian density function:

$$\varphi\left(\left\|\mathbf{y}-\mathbf{y}_k\right\|\right) = e^{\frac{-\sum\left(y_i-y_i^{(k)}\right)}{2\sigma}} \tag{6.5.2}$$

where σ is the deviation of $y_\mathrm{i}$ probability variables.

In this way the $g(\mathrm{y})$ function can be evaluated by $f(\mathrm{y},\mathrm{w})$ and the expected value can be estimated as follows:

$$ARC = E(g(\mathrm{y})) \approx E(f(\mathrm{y},\mathrm{w})) \tag{6.5.3}$$

The training set contains of all the state vectors where only one component is 1 and the others are 0. In this way the size of the training set is K, the number of the descriptors for a given job is:

$$\tau^{(K)} = \left\{\left(\mathrm{y}_1, g(\mathrm{y}_1)\right), \left(\mathrm{y}_2, g(\mathrm{y}_2)\right), \ldots, \left(\mathrm{y}_\mathrm{K}, g(\mathrm{y}_\mathrm{K})\right)\right\} \tag{6.5.4}$$

Based on the training set the optimal weights $\mathrm{w}_{\mathrm{opt}}$ can be determined minimizing the following mean square error:

$$\mathrm{w}_{opt} = \min_{w} \sum_{i=1}^{K} (g(\mathrm{y}_i) - f(\mathrm{y}_\mathrm{i},\mathrm{w}))^2 \tag{6.5.6}$$

Applying the radial basis function the approximator can be determined in the following way:

$$f(\mathrm{y}) = \sum_{i=1}^{K} w_i \varphi(\|\mathrm{y} - \mathrm{y}^{(i)}\|) \tag{6.5.7}$$

where $\varphi(\|\mathrm{y} - \mathrm{y}^{(j)}\|) = e^{\frac{-\sum_{i=1}^{K}(y_i - y_i^{(k)})}{2\sigma}}$.

The expected value of the *f* approximator can be calculated in the following way:

$$\begin{aligned} E(f(\mathrm{y})) &= E\left(\sum_{i=1}^{K} w_i\, \varphi(\|\mathrm{y} - y^{(i)}\|)\right) = \sum_{i=1}^{K} w_k\, E\left(\varphi(\|y - y^{(i)}\|)\right) \\ &= \sum_{i=1}^{K} w_k\, E\left(e^{\frac{-\sum_{j=1}^{K}(y_j - y_j^{(i)})}{2\sigma}}\right) = \sum_{i=1}^{K} w_k\, E\left(\prod_{j=1}^{K} e^{\frac{-\left(y_j - y_j^{(i)}\right)}{2\sigma}}\right) \\ &= \sum_{i=1}^{K} w_k \prod_{j=1}^{K} E\left(e^{\frac{-\left(y_j - y_j^{(i)}\right)}{2\sigma}}\right) \\ &= \sum_{i=1}^{K} w_k \prod_{j=1}^{K} \left(p_j e^{\frac{-\left(1 - y_j^{(i)}\right)^2}{2\sigma}} + (1 - p_j) e^{\frac{-\left(y_j^{(i)}\right)^2}{2\sigma}}\right) \end{aligned}$$

In this way the ARC can be calculated for every job in a scientific workflow, furthermore the $ARC_{swf}$ can be calculated for the whole workflow summarizing the $ARC_{job}$. Figure (figure 25) shows the pseudo code of the algorithm.

```
ARC_eppr=0
for i=1 to N
     get J_i
     generate τ_Ji^(Ki)={y_i, g(y_i)}
     calculate w_opt
     calculate f(y,w_opt)
     calculate E(f(y,w_opt))
     ARC==ARC+E(f(y,w_opt))
end
```

*25. Figure:* The pseudo code of the estimation of the ARC

## 6.6 The upper bound of the unreproducibility probability

In probability theory, the theory of large deviations is concerned with the study of probabilities of rare events. In Large Deviation Theory, the Chernoff bound gives exponentially decreasing

bounds on tail distributions of sums of independent random variables. Assuming the independency of the descriptor it can be applied to give a sharper upper bound of the NRP.

In the case when the cost-function is a linear function of the binary variables $y_i$ we can apply the Chernoff-bound methods and give an upper bound to the probability defined by the equation is

$$P(g(y) > C) = P\left(\sum_{i=1}^{K} w_i y_i > C\right) = \phi(C) < e^{(\sum_{i=1}^{K} \mu_i(w_i s) - sC)} \quad (6.6.1)$$

The functions $\mu_i(s)$ are the logarithmic momentum generator functions of the random variables $w_i y_i$:

$$\mu_i(s) = logEe^{w_i y_i s} \quad (6.6.2)$$

where $s: \sum_{i=1}^{K} \frac{d\mu_i(s)}{ds} = C$

This functions can be easily calculated as the following:

$$\mu_i(s) = logEe^{w_i y_i s} = \log(p_i e^{w_i s} + (1 - p_i)) \quad (6.6.3)$$

If the cost-function is not linear it has to be approximated by a linear function:

$$g(\mathbf{y}) \approx f(\mathbf{y}) = \sum_{i=1}^{K} w_i y_i \quad (6.6.4)$$

In this way, the evaluation is similar to the evaluation of the ARC using the capability of the neural networks. A training set must be generated. It should contain all the state vector which has only one element with the value of 1, and all the others are 0. In this way, the size of the training set is equal to *K*, where the *K* is the number of the descriptor in a given job. Based on the training set the optimal $w_i$ values can be calculated by minimizing the mean square error in the following way:

$$\mathbf{w}_{opt}: \min_{w} \frac{1}{K} \sum_{i=1}^{K} \left(g(\mathbf{y}_i) - \sum_{j=1}^{K} w_j y_{ij}\right)^2 \quad (6.6.5)$$

The minimization is based on solving the linear equation system.

In this way the approximator φ(C) function can be calculated as:

$$P(g(\mathbf{y}) > C) < \varphi(C) = e^{\sum_{i=1}^{K} \mu_i(w_i s) - sC} \quad (6.6.6)$$

```
NRP_eppr=1
for i=1 to N
     get J_i
     generate τ_Ji^(Ki)={y_i, g(y_i)}
     calculate w_opt
     calculate f(y,w_opt)
     calculate ϕ(C)
     NRP==NRP* ϕ(C)
end
```

*26. Figure:* The pseudo code of the estimation of the NRP

When the overall cost function of the scientific workflow is greater than a predefined *C* cost, generally the reproducibility does not worth the time and the cost to perform it. In other words, in that case the workflow is not reproducible. If the users are informed about this fact, they have the possibility to modify their workflow or to apply other virtualization tools.

## 6.7 Classification of scientific workflows based on reproducibility analysis

Analyzing the decay parameters of the descriptors we can classify the scientific workflows. First, we can separate the workflows which decay-parameters for all the jobs are zero. These workflows are reproducible at any time and any circumstance since they do not have dependencies. Than we can determine those ones which can influence the reproducibility of the workflow in other words which have non-zero decay parameter(s). Six groups have been created:

| **decay-parameter** | **cost** | **category** |
|---|---|---|
| *decay(v)=0* | *cost = 0* | reproducible |
| *decay(v)* is unknown | -- | non-reproducible |
| *decay(v)* is unknown, the descriptor value cannot be stored | *cost = $C_1$* | reproducible with extra cost |
| *decay(v) = F(t)* | *cost = $C_2$* | reproducible with probability *P* |
| *decay(v)* = vary(t,v) | *cost = $C_3$* | approximately reproducible |

*5. Table* The classification of the scientific workflow

### *6.7.1 Reproducible workflows*

The first group represents the reproducible workflows. In this case, all the decay-parameters of all the jobs belonged to a workflow are zero. These workflows are reproducible and they can be

executed and re-executed at any time and any circumstance since they are not influenced by dependencies.

### *6.7.2 Reproducible workflow with extra cost*

There are workflows, which have operation related descriptors which are unknown in normal circumstance, but with the help of additional resources or tools these dependencies can be eliminated. For example, if a computation is based on random generated value, this descriptor's value is unknown. In this case with the help of an extra, operation system level tool we can capture the return value of the system call and we can store it in the provenance database. Another example is when a virtualization tool, such as a virtual machine have to be applied to make the workflow reproducible.

### *6.7.3 Approximetly reproducible workflows*

In certain cases the workflow execution may depend on some continuously changing resource. For example there are continuously growing databases which get the data from sensor networks without intermission. If the computation of a workflow use some statistical parameters of this database, the statistical values never will be the same. Moreover the descriptor can be operation-related descriptor which is based random value or time or other parameter referred to the state of the system and the values are captured by the appropriate tools. In this case the appropriate descriptor's value of the given job may change on occasion of every re-execution, consequently the reproducibility of this workflow could be failed.

In this case, analyzing the change of the descriptor value and the effect for the result, in certain cases the relationship and an estimation method can be determined to replace the descriptor value or even the result of the job. On occasion of a later re-execution, if reproducing is not possible or the crucial descriptor is unavailable, this evaluating method can be applied and an evaluated result can be done.

### *6.7.4 Reproducible workflows with a given probability*

Many investigations revealed the problem caused by volatile third party resources, when the reproducibility of workflows became uncertain. The third party services or any external resources can be unavailable during the years. If there are no method to handle or eliminate this dependency, the probability of the reproducibility can be determined based on the theoretical decay-parameter (if the availability of the service can be given by the user or by the third party) or based on the

sample set in empirical way. Sometime the users may have to know the chance of the reproducibility of a workflow for example when they look for one in the repositories. Assuming that the probability distribution of the third party service is known, assumable or evaluable information can be provided to the users about the expected probability of the reproducibility.

### *6.7.5 Non-reproducible workflows*

There is no method to make the workflow reproducible. In this case the scientific workflow may have too many dependencies or it probably contains very complex non-deterministic job or jobs.

### *6.7.6 Partially reproducible workflows*

If a workflow has a crucial descriptor which influence the reproducibility and there are no method to compensate or eliminate this descriptor, the job containing this descriptor become non-reproducible. However, it does not mean, that the whole workflow is also non-reproducible. Determining the coverage of that crucial descriptor the reproducible part of the SWF can be identified. The reproducible part of the SWF also can be in any group listed above.

## 6.8 Conclusion

In this section I defined the metrics of the reproducibility, the ARC and the NRP. I determined the expected cost for making a workflow reproducible and we also gave an efficient adaptive evaluation method for the ARC. The method is very useful in the continuously changing environment in which scientific workflows are mostly enacted. Further I determined the probability that how likely the reproducibility cost is greater than a predefined threshold and I also gave an upper limit for the probability of making a workflow reproducible with a cost greater than a predefined C threshold. The analysis was bounded on the special cases when the cost function is linear or can be approximated by a linear function. The advantage of this evaluation is the simply computation but it provides a rough estimation. The future work may extend our evaluations on higher order approximations as well.

Finally, I investigated the possible types of the scientific workflows from the point of view of their reproducibility. The basis of the analysis is the decay-parameter which describes the type and the measure of the change of the descriptor's values. According to this parameter we determined a cost function which means the "work" required to reproduce the given job or workflow. In this way, the classification of the scientific workflows can be given and how they

can be reproduced in a later time. In the different categories, I set up methods to make the workflows reproducible or we gave the probability and the extra cost of the reproducibility.

## 6.9 Novel scientific results (theses)

Thesis group 3: I defined two metrics of the reproducibility and I determined approximations to evaluate them in polynomial time if the exact calculation is not possible in real-time.

3. Téziscsoport: Definiáltam a reprodukálhatóság költségének mérőszámait és polinomiális lépésszámú közelítő eljárást határoztam meg ezek becslésére abban az esetben, amikor a pontos számítás nem lehetséges valós időben.

Thesis 3.1

I have introduced the term of the repairing cost-index assigned to the computational job descriptors, which gives the ability to determine the reproducibility metrics of the DAG type scientific workflow:, namely the Average Reproducibility Cost (ARC) and the Non-Reproducibility Probability (NRP) values

3.1 Altézis

Bevezettem a számítási feladat deszkriptoraihoz rendelt javítási költség fogalmát, melynek segítségével meghatároztam az irányított körmentes gráfokkal reprezentálható tudományos munkafolyamatok reprodukálhatósági mértékeit, a reprodukálhatóság átlagos költségét (ARC) és a reprodukálhatatlansági valószínűséget (NRP).

Related publications: 3-B, 4-B, 5-B

Thesis 3.2

I have determined a real time computable method to evaluate in polynomial time the ARC of a DAG type scientific workflow in case the descriptors are independent.

3.2 Altézis

Meghatároztam egy valós időben számolható polinomiális lépésszámú közelítő eljárást a DAG tudományos munkafolyamatok átlagos reprodukálhatósági költségének (ARC) becslésére abban az esetben, amikor a deszkriptorok egymástól függetlenek.

Related publications: 4-B

Thesis 3.3

I have determined a real time computable method to calculate upper estimates in polynomial time the NRP value of a scientific workflow, when the descriptors and jobs are independent and the g(y) cost function is the linear function of the $y_i$ binary variables.

3.3 Altézis

Valós időben számolható, polinomiális lépésszámú felső becslést határoztam meg a reprodukálhatatlansági valószínűségre abban az esetben, amikor a $g(\mathbf{y})$ költségfüggvény az $y_i$ bináris változók lineáris függvénye, valamint a deszkriptorok és a jobok egymástól függetlenek.

Related publications: 3-B

Thesis 3.4

Based on the decay-parameters and the cost index I have categorized from the reproducibility perspective the scientific DAG-type workflows.

3.4 Altézis

A tapasztalati romlási mutató és a deszkriptorok javítási költsége alapján osztályoztam a DAG-gal reprezentálható tudományos munkafolyamatokat reprodukálhatósági szempontból.

Related publications: 5-B

# 7 PRACTICAL APPLICABILITY OF THE RESULTS

Based on this research I designed two extra modules of the WSPGRADE/gUSE to reproduce an in other way non-reproducible SWf. It performs an pre-analysis phase before re-execute a SWf based on the descriptor space to determine in which way the SWf can be reproduced and which extra tools (evaluation tool, descriptor value capture or extra storage) is required. After the re-execution an post analysis phase perform an estimation (if necessary) and updates the provenance database with the appropriate parameters needed to evaluation.

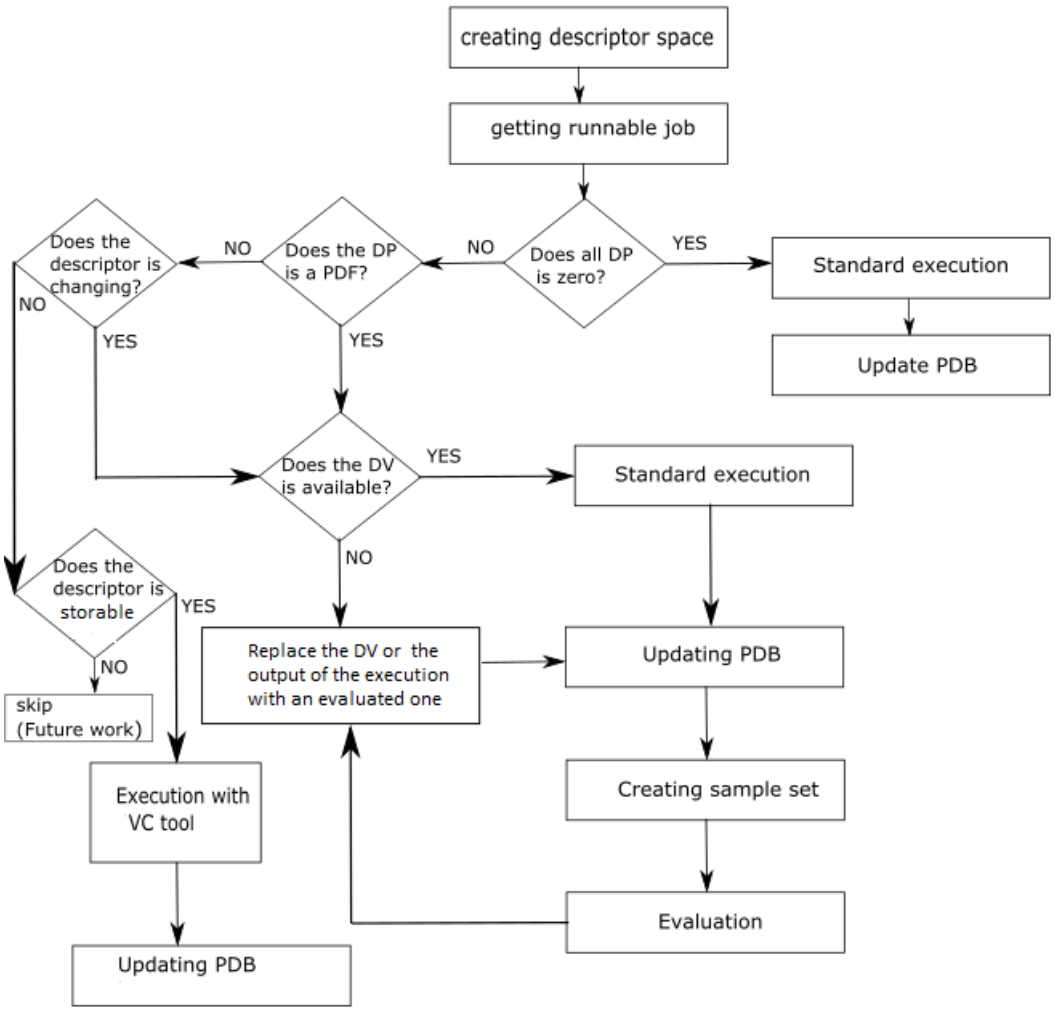
The process of reproducibility-analysis

Based on the descriptor's space the pre-analyzer performs a classification of the jobs of the given Wf. Depending on the classification, the job can be executed in three ways:

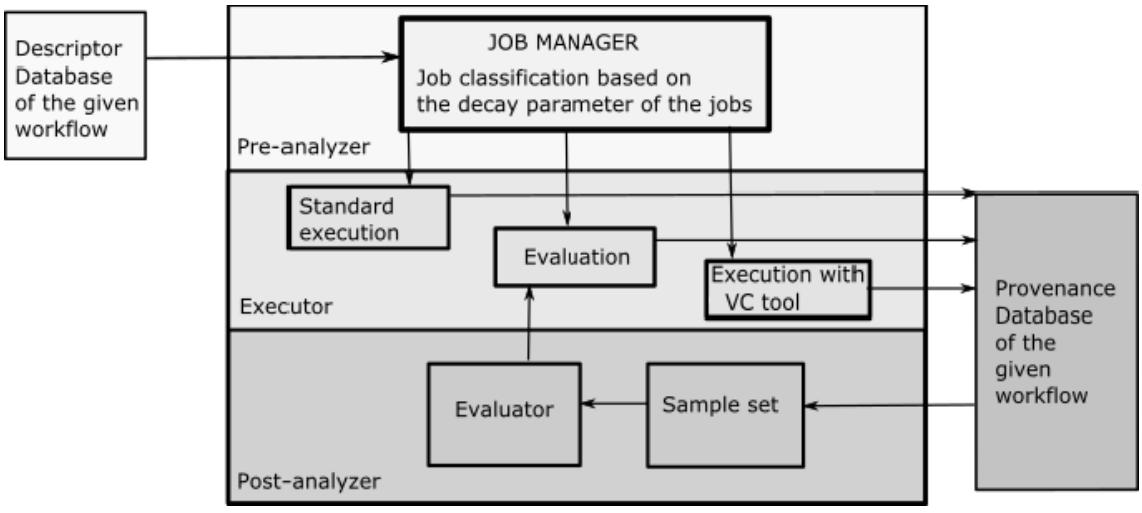
1. Standard execution, if all the decay parameters are zero.
2. Replacing the execution with evaluation, if there are changing descriptor values in the descriptor-space and their availabilities are changing in time.
3. Execution with descriptor value capture (VC) tool, if the execution of the job is based on operation related descriptor value or the value cannot be stored due to the

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 26, 27)




*27. Figure* The flowchart of the reproducing process




*28. Figure* The block diagram of the reproducing process

# 8 CONCLUSION

During the last decades the e-science widely gather ground among the scientific communities. Thanks to the high performance computing and to the parallel and distributed systems the classical analytical experiments conducted in the laboratories are taken over by the data and compute intensive in-silico experiments. The steps of these experiments are chained to a so called scientific workflow. An essential part of the scientific method is to repeat and reproduce the experiments of other scientists and to test the outcomes themselves even in a different execution environment. A scientific workflow is reproducible, if it can be re-executed without failures and gives the same result as the first time. In this approach the failures do not mean the failures of the Scientific Workflow Management System (SWfMS) but the correctness and the availability of the inputs, libraries, variables etc. The different users for different purposes may be interested in reproducing of the scientific workflow. 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. A reproducible workflow can be shared in repositories and it can become useful building blocks that can be reused, combined or modified for developing new experiments.

In this dissertation I investigated the requirements of the reproducibility and I set out methods which can handle and solve the problem of changing or missing descriptors to be able to reproduce a – in other way – non-reproducible scientific workflow. In order to achieve this goal I formalized the problem and based on provenance database I introduced the term of the descriptor-space which contains all the necessary component (call descriptor) to reproduce a job. Concerning to the descriptors I defined the theoretical and the empirical decay-parameter which describe the change of the descriptor in time-dependent and time-independent cases as well. Additionally, with the help of the decay parameters the crucial descriptors – which can influence or even prevent to reproduce a SWf – can be identified. Based on provenance database I created a sample set referred to a job which contains the descriptors of the job originated from the previous executions. Analyzing the empirical decay-parameter based on the sample set the relation can be determined between the change of the descriptor values and the empirical decay-parameter. Our goal was to find methods which can help to compensate the changing nature of the descriptors and which can help to perform evaluation to make the scientific workflow reproducible by replacing the missing values with simulated ones. In addition I determined the impact of a descriptor which says how the descriptor influences the result of a given job. The sample set also can help to determine the probability of the reproducibility and the reproducible part of a given SWf. Since the basis of our analysis is the decay-parameter, according to it I assigned to every descriptor a cost-index which means the “work” required to reproduce a given job or workflow. In this way I introduced two

measures of the reproducibility: the Average Reproducibility Cost and the Non-reproducibility Probability. The first one determines the expected value of the cost to reproduce a – on other way – non-reproducible SWf. The other measure is the Non-reproducibility Probability which gives how likely the reproducibility cost is greater than a predefined C threshold. The analyses was bounded on the special cases when the cost function is linear or can be approximated by a linear function. Finally I classified the scientific workflows from the reproducibility perspective and I determined the reproducible, partial reproducible, reproducible by substitution, reproducible with probability *p* and the non-reproducible scientific workflows.

During the design phase the results of this investigation can help the scientists to analyze the crucial descriptors of their workflow which can prevent to reproduce it. Additionally, storing this information, statistics and evaluation methods together with the workflows in the repositories, can provide a useful tool to support the reusability of the SWf making it reproducible and the scientists to find the most adequate (in sense of reproducibility) workflow to reuse.

## 8.1 Future research directinos

As a further extension of my research I plan to investigate scientific workflows represented by non-DAGs. These cyclic graph may contain execution loops which results recursive workflows. Moreover, the evaluability of the two reproducibility metrics, ARC and NRP can be investigated without assuming the independency of the descriptors.
First and foremost an implementation of the extension (mentioned in section 9) should be carried out in WSPGRADE/gUSE scientific workflow management system developed by MTA SZTAKI.

# 9 BIBLIOGRAPHY


Afgan, E. (2016). Afgan, E., Baker, D., van den Beek, M., Blankenberg, D., Bouvier, D., Čech, M., ... & Grüning, B. "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update." . *Nucleic acids research gkw343*.

Altintas. (2004). I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows". *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM '04)* (pp. 423–424). New York, NY, USA,: IEEE Computer Society.

Altintas, I. (2006). Altintas, I., Barney, O., Jaeger-Frank, E. "Provenance collection support in the kepler scientific workflow system". *In International Provenance and Annotation Workshop* (pp. 118-132). Springer Berlin Heidelberg.

Balaskó, Á., & al., e. (2013). Building science gateways by utilizing the generic WS-PGRADE/gUSE workflow system. *Computer Science*, 307-325.

Barker, A., & Hemet, J. V. (2007). Scientific workflow: a survey and research directions. In *Parallel Processing and Applied Mathematics* (pp. 746-753). Springer Berlin Heidelberg.

Bechhofer, S., & al, e. (2010). Research objects: Towards exchange and reuse of digital knowledge. . *The Future of the Web for Collaborative Science*.

Belhajjame. (2015). Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., ... & Klyne, Using a suite of ontologies for preserving workow-centric research objects. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Belhajjame, K., & al., e. (2012). *Workflow-centric research objects: First class citizens in scholarly discourse, Proceedings of Sepublica*, 1-12.

Benabdelkader, A. (2011). *A provenance approach to trace scientific experiments on a grid infrastructure; E-Science (e-Science); IEEE 7th INternational Conf. on IEEE*, (pp. 134-141).

Benabdelkader, A. (2014). *Provenance Manager: PROV-man an Implementation of the PROV Standard, Provenance Taskforce.* Budapest.

Bowers, S. (2008). Bowers, S., McPhillips, T. M., Ludäscher, B. "Provenance in collection-oriented scientific workflows". *Concurrency and Computation: Practice and Experience, 20(5),* , 519-529.

Brazma, A., & al, e. (2011). Minimum information about a microarray experiment (MIAME)—toward standards for microarray data. , 29(4),. *Nature genetics, 29*(4), 365-371.

Brooks, C. (2008). Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains). Technical Report No. UCB/EECS-2008-37.

Bucklew, J. A., & Sadowsky, J. S. (1993). A contribution to the theory of Chernoff bounds. *IEEE Transactions on Information Theory, 39*(1), 249-254.

Chapman, B., & Chang, J. (2000). "Biopython: Python tools for computational biology." . *ACM Sigbio Newsletter*, (pp. 15-19).

Chirigati, F., D, S., & Freire, J. (2013). Using Provenance to Support Computational Reproducibility. *TaPP*.

Clifford, B., & al., e. (2008). Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience, 20*(5).

Costa, F., & al., e. (2013). Capturing and Querying workflow runtime provenance with prov: a practical aproach. *Proceedings of the Joint EDBT/ICDT 2013 Workshop;* (pp. 282-289). ACM.

Cruz, S., & al., e. (2009). Toward a Taxonoomy of Provenance in Scientific Workflow Management Systems. *2009 Congress on Services-I.* (pp. 259-266). IEEE.

Davidson, S. B., & Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 1345-1350). ACM.

Davison, A. (2012, july). Automated Capture of Experiment Context for Easier Reproducibility in Computational Research . *Computing in Science & Engineering, 14/ 4*, 48–56.

Deelman. (2005). E. Deelman, G. Singh, M. H. Su et al., "Pegasus: a framework for mapping complex scientific workflows onto distributed systems," , vol. 13, no. 3,. *Scientific Programming*, 219–237.

Deelman, E., & al, e. (2009). *Workflows and e-Science: An overview of workflow system features and capabilities; Future Generation Computer Systems*, 528-540.

Deelman, E., & Gil, Y. (2006). *Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges, e-Science*, 144.

D-PROV. (n.d.). *D-PROV: Extending the PROV Provenance Model with Workflow Structure.*

Feitelson, D. G. (2015). From repeatability to reproducibility and corroboration. *ACM SIGOPS Operating Systems Review 49(1),* (pp. 3-11).

Freire, & al., e. (2014). Reproducibility Using VisTrails. *[Online]*. *Available: http://citeseerx.ist.psu.edu/viewdoc/download doi:10.1.1.369.9566*.

Freire, J., & al., e. (2012). Computational reproducibility: state-of-the-art, challenges, and database research opportunities. *Proceedings of the 2012 ACM SIGMOD international conference on management of data* (pp. 593-596). ACM.

Gentleman, R. (2004). Gentleman R, Carey V, Bates D, Bolstad B, Dettling M, ...& Zhang J: Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol. 5: R80-10.1186/gb-2004-5-10-r80*.

Gesing, S., & al., e. (2014). Workflows in a dashboard: a new generation of usability. *In proceedings of 9th Workshop on Workflows in Support of Large-Scale Science (WORKS), IEEE*, 82-93.

Gil. (2011). Gil, Y., Ratnakar, V., Kim, J., González-Calero, P. A., Groth, P., Moody, J., & Deelman, E. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems, 26(1)*, 62-72.

Gil, Y., & al, e. (2006). Report on the 2006 NSF Workshop on Challenges of Scientific Workflow; .

Gil, Y., & al, e. (2007). *Examining the challenges of scientific workflows; Ieee computer 40 (12),* (pp. 26-34).

Goble, C., & al., e. (2010). myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research, 38*(2), W677-W682.

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

Groth, P., & al., e. (2009). Pipeline-centric provenance model. *in proceedings of the 4th Workshop on Workflows in Support of Large-scale Science*, (p. 4).

gUSE. (n.d.). *User's Guide: http://guse.hu/about/home.* SZTAKI LPDS.

Hettne, K., & al, e. (2012). Best Practices for Workflow Design: How to Prevent Workflow Decay. *SWAT4LS.*

Hornik, K. (1991). *Approximation Capabilities of Multilayer Feedforward Networks, Neural Networks, 4*, 251-257.

Hornik, K., & al., e. (1989). *Multilayer Feedforward Networks are Universal Approximators; Neural Networks, 2*, 359-366.

Hornik, K., & al., e. (1990). *Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks, Neural Networks, 3*, 251-257.

J.Freire, & al., e. (2011). Exploring the coming repositories of reproducible experiments: Challenges and opportunities. *Proceedings of the VLDB Endowment*, (pp. 9-27).

Jordan, D. (2007). Web Services Business Process Execution Language, Version 2.0 (WS-BPEL 2.0),.

Kim, J. (2008). Kim, J., Deelman, E., Gil, Y., Mehta, G., & Ratnakar, V. "Provenance trails in the wings/pegasus system" . *Concurrency and Computation: Practice and Experience, 20(5), ,* 587-597.

Koop, D., & al, e. (2011). A Provenance-Based Infrastructure to Support the Life Cycle of Executable Papers. *Procedia Computer Science*, 648-657.

Koop, J., & al, e. (2013). Enabling Reproducible Science with VisTrails", arXiv preprint arXiv:1309.1784, 2013. *arXiv preprint arXiv:1309.1784,*.

Korolev, A., & al., e. (2014). PROB: A tool for Tracking Provenance and Reproducibility of Big Data Experiments. *Reproduce'14. HPCA, 11*, 264-286.

Ludäscher, B., & al, e. (2009). *Scientific process automation and workflow management, Scientific Data Management: Challenges, Existing Technology, and Deployment, Computational Science Series*, 476–508.

Ludäscher, B., & al, e. (2009). Scientific process automation and workflow management; Scientific Data Management: Challenges, Existing Technology, and Deployment. *Computational Science Series*, 476-508.

Mates, P., & al., e. (2011). Crowdlabs: Social analysis and visualization for the sciences." . *International Conference on Scientific and Statistical Database Management.* (pp. 555-564). Springer Berlin Heidelberg.

Mesirov. (2010, january). Accessible Reproducible Research. *Science, 327*(5964), 415-416.

MIAME. (n.d.). http://fged.org/projects/miame/.

Missier, S., & al., e. (2013). Provenance and data differencing for workflow reproducibility analysis. *Concurrency and Computation: Practice and Experience.*

Oinn, & al, e. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics, 20*(17), 3045-3054.

Oinn, T., & al., e. (2006). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Concurrency Computation Practice and Experience, 18*(10), 1067-1100.

OPM. (n.d.). OPM: http://openprovenance.org/.

Peng, R. D., & al., e. (2011, december). Reproducible Research in Computational Science. *Science, 334*(6060), 1226-1227.

PGRADE. (n.d.). *User's Guide: http://sourceforge.net/projects/guse/_les/3.7.4/Documentation.* SZTAKI LPDS.

Piccolo, S. R., & Frampton, A. B. (2015). Tools and techniques for computational reproducibility. . *In: bioRxiv, 022707*.

PROV. (n.d.). http://www.w3.org/TR/prov-overwiew/.

Roure, D., & al, e. (2011). Towards the preservation of scientific workows. *Proceedings of 8th International Conference on Preservation of Digital Objects.* iPRES.

Santana-Perez, I., & Perez-Hernandez, M. (2015). Towards Reproducibility in Scientific Workows: An Infrastructure-Based Approach. *Scientific Programming*, 11.

Simmhan, Y., & al., e. (2006). Performance evaluation of the karma provenance framework for scientific workflows. In *Provenance and Amotation of Data* (pp. 222-236). Springer.

Stajich. (2002). Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., ... & Lehväslaiho, H. "The Bioperl toolkit: Perl modules for the life sciences" . *Genome research, 12(10),* , 1611-1618.

Stodden, V., & al., e. (2013). *Setting the default to reproducible. computational science research. SIAM News, 46, 4-6.* computational science research. SIAM News, 46, 4-6.

Talia, D. (2013). Workflow Systems for Science: Concepts and Tools. *ISRN Software Engineering*.

Taverna. (2009). *The Taverna projekt; http://www.taverna.org.uk.*

Taylor. (2004). I. Taylor, M. Shields, I. Wang, and O. Rana, "Triana, applications within Grid computing and peer to peer environments,". *Journal of Grid Computing, 1*, 199-217.

Taylor, J. (2005). I. Taylor, M. Shields, I. Wang, and A. Harrison, "Visual grid workflow in Triana,". *Journal of Grid Computing, 3*(3-4), 153-169.

Wang, Y. (2005). A New Grid Workflow Description Language. *A New Grid Workflow Description Language,* (pp. 257-260).

Wolstencroft, K. (2013). Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., ... & Bhagat, J; The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic acids research.*

Woodman, H., & al., e. (n.d.). Achieving reproducibility by combining provenance with service and workflow versioning. *Proceedings of the 6th workshop on Workflows in support of large-scale science* , (pp. 127-136).

Yu, J., & Buyya, R. (2005). *A taxonomy of scientific workflow systems for grid computing, ACM Sigmod Record, 34(3),* (pp. 44-49).

Yu, J., & Buyya, R. (2005). A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing*, 171-200.

Zhao, J., & al, e. (2012). Why workflows break—Understanding and combating decay in Taverna workflows. *E-Science (e-Science), 2012 IEEE 8th International Conference on,* (pp. 1-9).

## Own Publications Pertaining to Theses


1-B A. Bánáti, P. Kacsuk, M. Kozlovszky: Reproducibility analysis of scientific workflows; Acta Politechnica Hungarica, accepted, unpublished

2-B A. Bánáti, P. Kacsuk, M. Kozlovszky; Investigation of the Descriptors to make the Scientific Workflows reproducible; CINTI 2016 - 17th IEEE International Symposium on Computational Intelligence and Informatics. Budapest, Hungary, (IEEE Computational Intelligence Society)

3-B A. Bánáti, P. Kárász, P. Kacsuk, M. Kozlovszky: Evaluating the Average Reproducibility Cost of the Scientific Workflows, In: International Symposium on Intelligent Sytems and Informatics (SISY), 2016

4-B A. Bánáti, P. Kacsuk, M. Kozlovszky, M. Evaluating the Reproducibility cost of the scientific workflows Applied Computational Intelligence and Informatics (SACI), 2016 IEEE 11th Jubilee International Symposium on. IEEE, 2016

5-B A. Bánáti, P. Kacsuk, M. Kozlovszky; Classification of Scientific Workflows Based on Reproducibility Analysis; 39th International Convention on Information and Communication Technology, Electronics and Microelectronics: MIPRO 2016. Opatia, Rijeka: Croatian Society for Information and Communication Technology Electronics and Microelectronics (MIPRO'16),

6-B A. Bánáti, P. Kacsuk, M. Kozlovszky; Minimal sufficient information about the scientific workflows to create reproducible experiment; 19th IEEE International Conference on Intelligent Engineering Systems: INES 2015. Bratislava, 2015.09.03-2015.09.05. Bratislava: IEEE, 2015. pp. 189-194.

7-B A. Bánáti, P. Kacsuk, M. Kozlovszky; Four level provenance support to achieve portable reproducibility of scientific workflows; 38th International Convention on Information and Communication Technology, Electronics and Microelectronics: MIPRO 2015. Opatia, may, 2015. Rijeka: Croatian Society for Information and Communication Technology Electronics and Microelectronics (MIPRO'15), pp. 241-244.

8-B Eszter Kail, Anna Bánáti, Péter Kacsuk, Miklós Kozlovszky; Dynamic workflow support in gUSE; 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO'14). Opatija, Croitia, 2014.05.26-2014.05.30. Rijeka: IEEE, 2014. pp. 369-374.

9-B Eszter Kail, Anna Bánáti, Péter Kacsuk, Miklós Kozlovszky; Provenance based adaptive and dynamic workflows; CINTI 2014 - 15th IEEE International Symposium on Computational Intelligence and Informatics. Budapest, Hungary, 2014.11.19-2014.11.21. (IEEE Computational Intelligence Society) pp. 215-219.

10-B Eszter Kail, Anna Bánáti, Péter Kacsuk, Miklós Kozlovszky: Provenance Based Runtime Manipulation and Dynamic Execution Framework for Scientific Workflows, in Scientific Bulletin of The Politehnica University of Timisoara, 2016, Vol: 61(75) No: 1