An Analysis of Workflow Formalisms for Workflows with Complex Non-Functional Requirements

Laurens Versluis
Computer Systems
Vrije Universiteit
Amsterdam, The Netherlands
l.f.d.versluis@vu.nl

Erwin van Eyk
Distributed Systems Group
Technische Universiteit Delft
Delft, The Netherlands
E.vanEyk@atlarge-research.com

Alexandru Iosup
Computer Systems
Vrije Universiteit
Amsterdam, The Netherlands
a.iosup@vu.nl

ABSTRACT

Cloud and datacenter operators offer progressively more sophisticated service level agreements to customers. The Quality-of-Service guarantees by these operators have started to entail non-functional requirements customers have regarding their applications. At the same time, expressing applications as workflows in datacenters is increasingly more common. Currently, non-functional requirements (NFRs) can only be defined on entire workflows and cannot be changed at runtime, possibly wasting valuable resources. To move towards modifiable NFRs at the task level, there is a need for a formalism capable of expressing this. Existing formalisms do not support this level of granularity or are restricted to a subset of NFRs. In this work, we investigate the current support for NFRs in existing formalisms. Using a library containing workflows with and without NFRs, we inspect the capability of existing formalisms to express these requirements. Additionally, we create and evaluate five metrics to qualitatively and quantitatively compare each formalism. Our main findings are that although current formalisms do not support arbitrary NFRs per-task, the Directed Acyclic Graphs (DAGs) formalism is the most suitable to extend.

ACM Reference Format:
Laurens Versluis, Erwin van Eyk, and Alexandru Iosup. 2018. An Analysis of Workflow Formalisms for Workflows with Complex Non-Functional Requirements. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3185768.3186297

1 INTRODUCTION

Expressing applications and processes as workflows is a well-established practice in various disciplines, including scientific computing, big data processing, business process management, and cloud computing [16]. These diverse use cases for workflows lead to a correspondingly diverse set of job-specific constraints or *non-functional requirements (NFRs)* associated with parts of workflows. Examples of these NFRs for workflows are that a particular task needs to be computed in an on-premise database for legal or privacy reasons, a task with a specific deadline, or a particular task that needs to be computed on a server with a GPU. Although some workflow management systems allow users to define NFRs on workflows, such as deadlines [9], prior work indicates that just applying these NFRs on entire workflows still is wasteful [20]. This is an important problem as minimizing resource usage while still adhering to the quality demands of customers is increasing in importance. To address this problem, we aim to refine the granularity and apply NFRs at the task level. However, despite the popularity of workflows, there is no well-defined notion for expressing these NFRs on a task-based level. Prior work such as [1] have introduced the concept of task-based NFRs, yet only support specific requirements such as high-performance. Therefore, in this paper, we aim to analyze existing formalisms in order to assess their ability to express workflows with NFRs at the task-based level.

The concept of a workflow can be described as follows. *A workflow describes the necessary computational steps and their data needs, required to reach a certain goal* [7]. The goal of a workflow can range from achieving a certain system state to calculating specific data points. Given this concept, there are several encapsulated concepts.

First, a *task* represents a computation process or step. This process receives data as input through one or more incoming dependency links, processes it and, optionally, emitting the output data over its outgoing dependency links.

Second, a *dependency link* or *(data) constraint* represents the data transfer from one task A to another task B. Due to the expectation of the incoming data, task B depends on task A.

Third, each workflow has a clear start and end defined. It has one or more *starting tasks* or *entry tasks*. These tasks can be identified by having one or more outgoing dependency links, while not having any incoming dependency links. A workflow having multiple starting events may have a single task added to emphasize the initial start of the application as described in [12]. A workflow concludes with one or more *end tasks*. These tasks can be recognized by having one or more incoming dependency links, while not having any outgoing dependency links. A workflow is said to be completed once all its end tasks have finished their computation.

Based on these definitions, elementary workflow structures can be identified. Bharathi et al. [5] mention four basic workflow structures (excluding start and end tasks). These basic workflow structures are the building blocks of complex workflows.

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/10.1145/3185768.3186297

A workflow formalism is a formal grammar, which allows users to express the workflow concepts using consistent, machine-readable semantics. There are several reasons for establishing formalisms for workflows, including allowing users to work with common syntax that is parseable by workflow management systems. Existing workflow formalisms differ in various, often ambiguous aspects, such as the popularity, expressiveness, and semantics. For this reason, a comparison of these workflow formalisms with regards to extending one with NFRs is a non-trivial endeavor.

NFRs specify quality attributes that the workflow management system has to adhere to. These attributes often are detailed in the service level agreement (SLA) through service level objectives (SLOs). A resource and management system must continuously monitor if none of these SLOs is being violated. As cloud computing is moving towards higher-level resource abstractions, workflows with granular NFRs are becoming increasingly more relevant in order to express these abstract requirements for common cloud metrics. The SPEC CLOUD research group notes four important system properties: elasticity, performance isolation, availability, and operational risk [11]. To analyze these properties, several metrics are introduced per property. Many of these metrics can be expressed as NFRs. Elasticity is important while running any workload, as minimizing resource consumption while not affecting performance is important for both the cloud operator and customer. Availability is necessary when running e.g., a (business) critical job or a job consuming a lot of time to avoid expensive re-computation in the case of failure. Performance isolation is important in the face of e.g., parallelism and multi-tenancy. When serving multiple tenants, the cloud operator must ensure other tenants are not or minimally affected by a single tenant. Operational risk is necessary to classify and map risks such as contention or under- and overprovisioning.

In this work we assign a performance isolation, high-availability, and resource contention NFR to different applications to observe the expressiveness of existing formalisms.

Overall, our contribution is threefold:

(1) A quantitative investigation of the popularity of the various workflow formalisms in the computer science domain (Section 2).
(2) A library of six complex workflows of which three containing NFRs, including the mapping of these workflows to each of the selected popular workflows formalism (Section 3).
(3) A comparison of the most popular formalisms in Section 4, followed by the motivation of which formalism that should be chosen to extend.

2 SELECTION OF FORMALISMS FOR DEFINING COMPLEX WORKFLOWS IN DISTRIBUTED COMPUTING SYSTEMS

In this section we describe the method used to extract the most applied workflow formalisms currently used in computer systems literature. Our focus is on investigating the support of existing formalisms for defining complex workflows. The key idea is that by using a well-known formalism, we can leverage existing tools built around this formalism and architectures can be used. If a formalism requires modifications, creating an extension that has backwards-compatibility allows for existing workflows to be defined as well, which increases its traction and adoption.

Table 1: The number of papers per conference that explicitly mention one of the three most used formalisms surveyed.

Conference	Workflow	BPMN	Petri net	DAG
HPDC	2	0	0	0
NSDI	1	0	0	0
SOCC	0	0	0	0
ICPE	1	1	1	0
Cluster	20	0	0	5
OSDI	1	0	0	0
SIGMETRICS	0	0	0	0
CCgrid	50	1	2	15
ICPP	12	0	0	7
IEEECLOUD	21	1	2	12
IPDPS	8	0	0	5
Total	116 (100%)	3 (3%)	5 (4%)	44 (38%)

2.1 Method

To select literature on workflow formalisms, we perform a comprehensive survey of existing literature with an explicit focus on workflows. We first make a selection of conferences having topics in distributed computing systems. From these conferences, we select papers in the span of 2012 – 2016, that have the word "workflow" in either their title or abstract. We then manually investigate each paper that meets these criteria and keep track of explicit mentions of the use of workflow formalisms. Implicit mentions such as a workflow management system, which supports a certain workflow formalism, are not be taken into account.

The conferences we investigate are HPDC, NSDI, SOCC, ICPE, Cluster, OSDI, SIGMETRICS, CCGrid, ICPP, IEEECLOUD, and IPDPS. These conferences have been selected to reflect the current state-of-the-art, diverse approaches, and directions of research regarding scheduling workflows in (cloud) datacenters.

2.2 Results

In total, 116 papers meet the criteria defined in the previous section. Table 1 shows the distribution of these papers among the eleven surveyed conferences. Additionally, the three most used formalisms are outlined in the table: Business Process Model and Notation (BPMN), Petri net (PN), and Directed Acyclic Graph (DAG).

From Table 1 we observe the DAG formalism is by far the most applied. This supports statements made by previous work on DAG popularity in e.g., [16] and [25]. In the following sections, we will investigate the support for NFRs of these three formalism.

3 A LIBRARY OF COMPLEX WORKFLOWS WITH AND WITHOUT NON-FUNCTIONAL REQUIREMENTS

To investigate the characteristics of the selected formalisms, we need a library containing complex workflows both with and without NFRs. As shown by [13], the current state of cloud computing has made it economically and technically appealing to execute complex,

often scientific, workflows there rather than use self-managed dedicated HPC clusters. In this section we will introduce six workflows, consisting of three existing applications without NFRs and three fictive, yet representative workflows with NFRs. Next, we describe each formalism and map the six workflows on them. From these mappings, we assess whether the current formalisms are capable of expressing NFRs.

3.1 Workflows

Table 2: Characteristics of the BLAST, Montage and Epigenomics workflows.

Workflow	Domain	CPU	IO	Memory
BLAST	Bioinformatics	High	Mid	Mid
Montage	Astronomy	Low	High	Low
Epigenomics	Bioinformatics	High	Low	Low

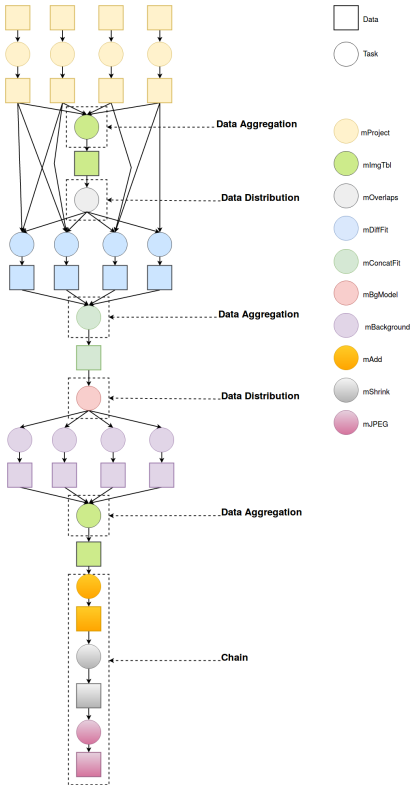

Figure 1: The Montage workflow structure.

First, we describe three real, complex workflows structures: BLAST (bioinformatics), Montage (astronomy), and Epigenomics (bioinformatics). These belong to the class of scientific workflows and have different characteristics, visible in Table 2.

The BLAST workflow is a bioinformatics application to perform rapid sequence comparisons. It can be applied in a variety of contexts including straight-forward DNA or protein sequence database searches [3]. BLAST is characterized by its high CPU and medium memory and IO consumption.

The Montage workflow is an astronomy application in which custom mosaics of the sky are created using a set of input images [4, 5, 27]. This workflow is characterized by its IO-intensive behaviour. In contrast to the BLAST and Epigenomics structures, entry nodes pass data to several nodes at later levels. Levels are defined by tasks that have the same distance from the entry task. Figure 1 visualizes the Montage workflow structure.

The Epigenomics workflow is a CPU intensive application to automate the execution of various genome sequencing operations [5, 18]. Different from BLAST and Montage, it features a high parallel chain structure with a single entry node.

Next, we introduce three fictive yet representative workflows with availability, performance isolation, and operational risk NFRs as specified in Section 1. To maintain the representativeness of the workflows, we select existing, real-world workflows and set NFRs on (some of) their tasks. The workflow structure and properties (e.g. task runtime) are left untouched.

The first workflow originates from the Big Data domain. The BTWorld workflow is a workflow of coupled MapReduce jobs that is derived from the BitTorrent network [8, 10]. In this workflow, the AH, TKSL, TKSG, TKHL, and TKHG queries are the most time consuming [10]. As such, it is preferred to keep the contention risk low. For example, the contention risk r_c defined in [11] is not allowed to exceed 0.9.

The second workflow is a modified version of the Epigenomics workflow. As Epigenomics features a highly parallel structure, performance isolation is important. As the wait time of the MapMerge step is determined by the slowest parallel chain, performance imbalance will affect the runtime. For example, in this scenario, a maximum performance imbalance of 10% can be set.

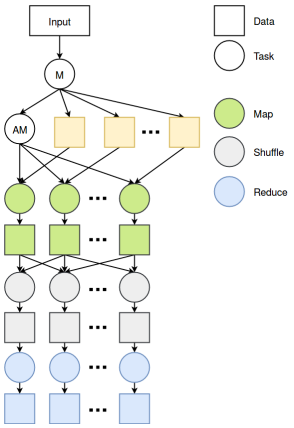

Figure 2: The MapReduce workflow structure.

The third workflow is a MapReduce workflow. This workflow consists of a master (M), application master (AM), and several map, shuffle, and reduce tasks. As the master monitors the several application masters (in this workflow there is one), and the application master monitors the other tasks, these two nodes require high availability. As availability is usually defined by uptime, an uptime guarantee of 99.99999% can be set for this application.

The other tasks do not require high availability as they can be restarted by the application master. The MapReduce workflow structure is visible in Figure 2.

Table 3: Mapping of the six complex workflows to the BPMN formalism.

Workflow	E	A	G	F	D	T	P
BLAST	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{A}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{M+R}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{B}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{C}}\}$	$\{\}$	$\{p_1\}$
Montage	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{D}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{N+S}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{E}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{F}}\}$	$\{\}$	$\{p_1\}$
Epigenomics	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{G}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{O+T}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{H}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{I}}\}$	$\{\}$	$\{p_1\}$
BTWorld	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{10}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{P+U}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{14}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{12}}\}$	$\{\}$	$\{p_1\}$
Mod. Epigenomics	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{G}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{O+T}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{H}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{I}}\}$	$\{\}$	$\{p_1\}$
MapReduce	$\{e_{start}, e_{end}\}$	$\{a_1, a_2, \ldots, a_{\mathbf{J}}\}$	$\{g_1, g_2, \ldots, g_{\mathbf{Q+V}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{K}}\}$	$\{d_1, d_2, \ldots, d_{\mathbf{L}}\}$	$\{\}$	$\{p_1\}$

3.2 Mapping to BPMN

The BPMN Version 2.0 (BPMN 2.0) is a visual standard for designing and modelling (business) workflows [1]. It is often used to design workflows at a high-level, targeted at human readability, and the most used business process model [21].

Definition 3.1. [26] defines a BPMN 2.0 workflow W as:

$$W =< E, A, G, F, D, T, P >$$

- E, the set of events.
- A, the set of activities.
- G, the set of gateways.
- F, the set of flows.
- D, the set of data.
- T, the set of artefacts.
- P, the set of swimlanes.

To outline each workflow in BPMN, we assume the following.

- BLAST contains A tasks, B constraints, and C units of data.
- Montage contains D tasks, E constraints, and F units of data.
- Epigenomics contains G tasks, H constraints, and I units of data (the same holds for the modified version).
- BTWorld contains 10 tasks, 14 constraints, and 12 units of data.
- MapReduce contains J tasks, K constraints, and L units of data.

In the context of BPMN 2.0, we assume that M, N, O, P, and Q tasks have multiple outgoing flows for BLAST, Montage, (modified) Epigenomics, BTWorld, and MapReduce respectively. Next, we assume R, S, T, U, and V tasks require a merge gate because of multiple incoming flows for BLAST, Montage, (modified) Epigenomics, BTWorld, and MapReduce respectively. The details for each workflow are presented in Table 3. As we can observe, each workflow has a similar structure in BPMN. All workflows have a single start and end event and are contained in one pool. The NFRs required in the modified Epigenomics, BTWorld, and MapReduce workflow cannot be expressed in the formalism.

3.3 Mapping to PN

Petri nets (PNs) are state-transition systems that extend elementary nets [19]. They are useful for describing and studying information processing systems that feature concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic characters [17].

Definition 3.2. [23] defines a Petri net PN as:

$$PN = (P, T, F)$$

- P is a finite set of places.

Table 4: The elements per workflow, modeled using PNs.

Workflow	P	T	F
BLAST	$\{P_1, P_2, \ldots, P_{\mathbf{A}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{B}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{M}}\}$
Montage	$\{P_1, P_2, \ldots, P_{\mathbf{D}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{E}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{N}}\}$
Epigenomics	$\{P_1, P_2, \ldots, P_{\mathbf{G}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{H}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{O}}\}$
BTWorld	$\{P_1, P_2, \ldots, P_{\mathbf{10}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{14}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{P}}\}$
Mod. Epigenomics	$\{P_1, P_2, \ldots, P_{\mathbf{G}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{H}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{O}}\}$
MapReduce	$\{P_1, P_2, \ldots, P_{\mathbf{J}}\}$	$\{t_1, t_2, \ldots, t_{\mathbf{K}}\}$	$\{f_1, f_2, \ldots, f_{\mathbf{Q}}\}$

Table 5: The elements per workflow, modeled using DAGs.

Workflow	V	E
BLAST	$\{v_1, v_2, \ldots, v_{\mathbf{A}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{B}}\}$
Montage	$\{v_1, v_2, \ldots, v_{\mathbf{D}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{E}}\}$
Epigenomics	$\{v_1, v_2, \ldots, v_{\mathbf{G}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{H}}\}$
BTWorld	$\{v_1, v_2, \ldots, v_{\mathbf{10}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{14}}\}$
Mod. Epigenomics	$\{v_1, v_2, \ldots, v_{\mathbf{G}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{H}}\}$
MapReduce	$\{v_1, v_2, \ldots, v_{\mathbf{I}}\}$	$\{e_1, e_2, \ldots, e_{\mathbf{J}}\}$

- T is a finite set of transitions.
- $F \subseteq (PxT) \bigcup (TxP)$ is a set of arcs (flow relations).

To outline the elements for each workflow using PNs, we assume the same information as in Section 3.2. In addition, we assume the BLAST, Montage, (modified) Epigenomics, BTWorld, and MapReduce workflows to have M, N, O, P, and Q arcs respectively. The outlines are visible in Table 4. From this table we observe that we cannot express that the BTWorld, modified Epigenomics workflow, and MapReduce workflows have NFRs.

3.4 Mapping to DAG

DAGs are frequently used to model (scientific) workflows [25]. They are used in many areas of computer science, including distributed systems. As the name implies, the graph dependency structure may not contain cycles, unlike BPMN and PNs. While the absence of support for loops is the biggest limitation [6], it simplifies the processing and scheduling of tasks in cloud environments. Note that tasks internally are allowed to use loops in their functions.

Definition 3.3. The definition of a DAG G is

$$G =< V, E >$$

- V is the set of vertices.
- E is the set of directed edges.

To outline the elements for each workflow using DAGs, we again assume the same information as in Section 3.2. The outline per workflow is given in Table 5. Just like BPMN and PN, we cannot express the NFRs using DAGs.

3.5 Results

Given the results in section 3, we observe all workflows show a similar structure for all formalisms. Yet none of the NFRs can be fully represented in the mappings. Therefore, We conclude that the current most used formalisms cannot express NFRs requirements.

4 A QUALITATIVE AND QUANTITATIVE COMPARISON OF EXISTING WORKFLOW FORMALISMS

Although the investigated workflow formalisms do not have sufficient support for NFRs, it is still possible to extend one of these formalisms in order to support NFRs. Extending one of these existing formalisms has many clear advantages over attempting to develop a completely novel formalism, such as existing tools, communities, and workloads. Especially in cloud environments, extending another, more ubiquitous formalism potentially increases the portability of existing workloads, as well as lowers the barrier to introduce interoperability between different cloud providers. Therefore, in the remainder of this chapter a quantitative, as well as, qualitative comparison is made between the three selected formalisms, BPMN, PN, and DAG, in order to determine which formalism to extend.

4.1 Method

In this section the method for comparing the workflow formalisms will be discussed. We introduce several metrics to quantitatively and qualitatively compare the selected formalisms. The overall method will consist of a table comparing each of the formalism on a combination of both qualitative and quantitative metrics. For each metric a description, argumentation regarding the relevance of the metric and, if applicable, what value is most preferred are provided.

Complexity In this context complexity is a measure as the perceived complexity of a given formalism. We use the cardinality of the set of symbols defined in each formalism as a measure of the complexity. Intuitively, a formalism that requires more abstract concepts or symbols to convey the same workflow is more difficult to understand by users and more error prone.

Utilization Related to the measure of complexity, is the measure of utilization. The utilization of a formalism is an indication of how well the formalism fits the context of workflows. We measure the utilization as the percentage of symbols used to model the Montage workflow compared to the total number of symbols that the formalism defines. Formalisms fitting the target applications will have a higher utilization, which is desired.

Supports loops The support for loops in workflows is a simple, yet impactful differentiator between formalisms. The support for loops allows workflows to be more expressive, at the expense of workflows becoming potentially non-deterministic. Due to the added complexity of having support for non-determinism in workflows, either full support or no support at all is preferred over 'limited' support.

Support for Non-Functional Requirements (SNFR) The main focus of this work is to establish a formalism to express task-level NFRs. Even though the surveyed formalism do not support SNFRs at a task-level as shown in section 3, there might be formalisms that already have a notion of SNFRs to extend. Similar to the support for loops, here we either prefer full support or no support at all. Having limited support will solely increase the complexity of the formalism, while not being expressive enough to express with the various NFRs.

Popularity In section 2 we have performed a comprehensive survey to inspect which formalisms are most applied. For this metric the results from that section will be reused as the percentage of usages of a certain workflow formalism in papers. The reason for this is that popularity is an important factor for the eventual support and adoption of the formalism. Extending a popular workflow formalism would also allow for compatibility and easy integration of existing tools, communities and workloads. Moreover, a popular formalisms tend to have more tools available. In this case a higher popularity is preferred.

Table 6: Comparison of BPMNs, PNs and DAGs using the metrics defined in Section 4.1.

Properties	BPMN	PN	DAG
Complexity	6	4	2
Utilization	$< 6\%$	100%	100%
Supports loops	Yes	Limited	No
SNFR	Limited	No	No
Popularity	4%	3%	38%

4.2 Results

The results of the workflow formalism comparison can be found in table 6. Overall, there is much diversity in characteristics between each formalism. The complexity metric shows that there is a distinct difference in complexity between the formalisms, where BPMN is the most complex. Moreover, the utilization metric shows that, although more complex, the actual appropriateness of BPMN for scientific workflows is low, where DAGs and PNs both do seem to be appropriate given their low complexity and high utilization. In the support for loops all of the investigated workflow formalisms take a different approach. The DAG workflow formalism does not have any support for loops, while BPMN does have complete support for it. As concluded in section 3 not a single workflow formalism currently has sufficient support for NFRs. BPMN does have some support for NFRs as noted in the comparison. However, the support is limited and is not extendible to all the NFRs we desire. Finally, where DAGs are very popular in the scientific domain, PNs and BPMNs are less common.

4.3 Discussion

These results of this comparison lead to a decision of which workflow formalism to extend. The BPMN formalism is needlessly complex, given that the utilization of the formalism symbols is very low. Although it supports loops and has very limited support for NFRs, this is not a winning argument. Introducing NFRs in this formalism would either increase the complexity even more or might break compatibility with existing BPMN workflows. Finally, it is clear that BPMN is not popular in the target domain.

The PN formalism is less complex, while having a higher utilization than BPMN. It does have limited loop support, which might complicate the introduction of NFRs. Additionally, like BPMN, PN is not popular in the target domain of cloud computing.

Therefore, the most optimal workflow formalism to extend seems to be DAGs. Similar to PNs it has a very low complexity, while being very applicable to the relevant workflows. It does not have support for loops and NFRs, which allows for easier extension in these areas, without having to break compatibility with existing use cases. Finally, the popularity of DAGs is a very strong argument in favor of this workflow formalism. As this would also imply that most existing tools and workflows engines have support for DAGs.

5 RELATED WORK

Prior works has attempted to extend formalisms to allow expressing NFRs. Bocciarelli et al. [1] propose an extension of BPMN to allow for NFRs. While mentioning the NFRs also targeted by MagnaData [14] e.g., availability, high performance, and security, their formalism does not support all of them. Tang et al. [22] propose an approach where tasks are marked as e.g., IO- or CPU-intensive. This allows schedulers to take into account tasks behavior and select more appropriate resources. A drawback using this approach is not being able to specify values. Different tasks may require different percentages of availability, which this approach does not support.

Abiteboul et al. [2] introduce a framework to compare data-driven formalisms using Views. Kim et al. mention a research methodology for comparing formalisms [15]. In particular, they argue the validity for users and analyst's ability to perform modeling tasks are central. Dependent and independent variables play a key role in this comparison. Task performance, perceived usefulness, comprehension, and discrepancy are mentioned as measures.

6 CONCLUSION AND ONGOING WORK

In this paper we outline basic workflow concepts and introduce a library of six complex workflows of which three containing non-functional requirements (NFRs). Using our library, we conclude that none of the surveyed formalisms is able to express NFRs at the granularity we desire. To investigate which formalism is the most suitable to extend, we create five metrics for quantitative and qualitative comparison. From the results of this comparison, we argue that the DAGs formalism is the most suitable workflow formalism to extend to allow for NFRs at the task-based level. DAGs feature the smallest set of construct, are extensively used in literature, and are supported by most workflow management systems.

In our ongoing work, we will focus on extending a workflow formalism to incorporate NFRs at a task-based level as specified in this work. Using this extended formalism, ongoing research will be conducted in several directions.

A future direction of research is towards modifications of NFRs at runtime. Modifications at runtime are necessary in dynamic systems, especially when conditions change. Research on the impact of these dynamic changes is necessary to understand the implication of such changes. Investigate changes required to current workflow engines to support a new formalism, possibly with dynamic NFRs, is another direction of research. This will allow conducting experiments in simulation or real world scenarios.

Another direction of future and ongoing research is into introducing DAG-based workflows along with NFRs to emerging fields within cloud computing, including the serverless computing paradigm and Function-as-a-Service (FaaS) model. As a part of the SPEC CLOUD research group we have found that both workflows and NFRs are promising future directions to explore for this paradigm [24]. The high level of abstraction and functional nature of this cloud model is a good fit for the abstraction of technical details of both workflows and NFRs. In the future, we want to explore this topic in depth, evaluating workflows supporting NFRs with, next to the common cloud metrics[11], other NFRs such as policies on how to deal with version upgrades.

REFERENCES

[1] 2011. A BPMN extension for modeling non functional properties of business processes, author=Bocciarelli, Paolo and others. In *DEVS Integrative M&S Symposium*.
[2] Serge Abiteboul et al. 2012. Comparing workflow specification languages: a matter of views. *TODS* 37 (2012).
[3] Stephen F Altschul et al. 1990. Basic local alignment search tool. *Journal of molecular biology* 215, 3 (1990).
[4] G Bruce Berriman et al. 2004. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *SPIE Astronomical Telescopes+ Instrumentation*.
[5] Shishir Bharathi et al. 2008. Characterization of scientific workflows. In *WORKS 2008. Third Workshop on*. IEEE.
[6] Jorge Cardoso et al. 2003. Workflow quality of service. In *Enterprise Inter-and Intra-Organizational Integration*. Springer.
[7] Ewa Deelman et al. 2015. Pegasus, a workflow management system for science automation. *FGCS* 46 (2015).
[8] Bogdan Ghit et al. 2014. Balanced resource allocations across multiple dynamic MapReduce clusters. In *SIGMETRICS*, Vol. 42. ACM.
[9] Robert Grandl et al. 2016. Altruistic Scheduling in Multi-Resource Clusters. In *OSDI*.
[10] Tim Hegeman et al. 2013. The BTWorld use case for big data analytics: Description, MapReduce logical workflow, and empirical evaluation. In *IEEE BigData*.
[11] Nikolas Herbst et al. 2016. Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics. *arXiv preprint arXiv:1604.03470* (2016).
[12] Alexey Ilyushkin et al. 2015. Scheduling workloads of workflows with unknown task runtimes. In *CCGRID*. IEEE.
[13] Alexey Ilyushkin et al. 2017. An Experimental Performance Evaluation of Autoscaling Policies for Complex Workflows. In *ICPE*.
[14] Alexandru Iosup. 2017. MagnaData: Massivizing Datacenter Scheduling to Bring All Data Services to All People. (2017). http://www.ds.ewi.tudelft.nl/~iosup/MagnaData/
[15] Young-Gul Kim et al. 1995. Comparing Data Modeling Formalisms. *Commun. ACM* 38, 6 (1995).
[16] Ji Liu et al. 2015. A survey of data-intensive scientific workflow management. *GC* 13, 4 (2015).
[17] Tadao Murata. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE* 77, 4 (1989).
[18] Maria Rodriguez et al. 2016. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *CCPE* (2016).
[19] Grzegorz Rozenberg et al. 1998. Elementary net systems. In *Lectures on Petri Nets I: Basic Models*.
[20] Siqi Shen et al. 2015. An Availability-on-Demand Mechanism for Datacenters. In *CCGrid*.
[21] Marigianna Skouradaki et al. 2015. On the road to benchmarking BPMN 2.0 workflow engines. In *ICPE*.
[22] Zhuo Tang et al. 2016. An optimized MapReduce workflow scheduling algorithm for heterogeneous computing. *SC* 72, 6 (2016).
[23] Wil M. P. van der Aalst. 1998. The Application of Petri Nets to Workflow Management. *JCSC* 8, 1 (1998).
[24] Erwin van Eyk et al. 2017. The SPEC cloud group's research vision on FaaS and serverless architectures. In *IWSC*.
[25] Marek Wieczorek et al. 2009. Towards a general model of the multi-criteria workflow scheduling on the grid. *FGCS* 25, 3 (2009).
[26] Peter YH Wong et al. 2008. A process semantics for BPMN. In *International Conference on Formal Engineering Methods*.
[27] Zhao Zhang et al. 2012. Design and analysis of data management in scalable parallel scripting. In *SC*.