

## A probabilistic strategy for temporal constraint management in scientific workflow systems<sup>‡</sup>

Xiao Liu<sup>1,\*</sup>, Zhiwei Ni<sup>2</sup>, Jinjun Chen<sup>1</sup> and Yun Yang<sup>1</sup>

<sup>1</sup>*Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Melbourne 3122, Australia*

<sup>2</sup>*Institute of Intelligent Management, School of Management, Hefei University of Technology, Hefei, Anhui 230009, People's Republic of China*

### SUMMARY

In scientific workflow systems, it is critical to ensure the timely completion of scientific workflows. Therefore, temporal constraints as a type of QoS (Quality of Service) specification are usually required to be managed in scientific workflow systems. Specifically, temporal constraint management includes two basic tasks: setting temporal constraints at workflow build-time and updating temporal constraints at workflow run-time. For constraint setting, the current work mainly adopts user-specified temporal constraints without considering the system performance. Hence, it may result in frequent temporal violations which deteriorate the overall workflow execution effectiveness. As regards constraint updating, although not well investigated, so far is in fact of great importance to workflow management tasks such as workflow scheduling and exception handling. In this paper, with a systematic analysis of the above issues, we propose a probabilistic strategy for temporal constraint management which utilizes a novel probability-based temporal consistency model. Specifically for constraint setting, a negotiation process between the client and the service provider is designed to support the setting of coarse-grained temporal constraints and then automatically derive the fine-grained temporal constraints; for constraint updating, the probability time deficit/redundancy propagation process is proposed to update run-time fine-grained temporal constraints when workflow execution is either ahead of or behind the schedule. The effectiveness of our strategy is demonstrated through a case study on an example scientific workflow process in our scientific workflow system. Copyright © 2011 John Wiley & Sons, Ltd.

Received 20 September 2010; Accepted 21 February 2011

**KEY WORDS:** scientific workflow system; workflow QoS; temporal constraints; temporal constraint setting; temporal constraint updating; probabilistic strategy

### 1. INTRODUCTION

Scientific workflow is a new special type of workflow that often underlies many large-scale complex e-science applications such as climate modelling, structural biology and chemistry, medical surgery or disaster recovery simulation [1–3]. Real-world scientific processes normally stay in a temporal context and are often time constrained to achieve on-time fulfilment of certain scientific or business targets. Otherwise, the usefulness of its execution results will be severely deteriorated. For example, a daily weather forecast scientific workflow has to be finished before the broadcasting

\*Correspondence to: Xiao Liu, Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Melbourne 3122, Australia.

†E-mail: xliu@swin.edu.au

‡The initial work was published in the Proceedings of 6th International Conference on Business Process Management (BPM2008), Lecture Notes in Computer Science, vol. 5240, pp. 180–195, September 2008 Milan, Italy.

of the weather forecast program everyday at, for instance, 6:00pm. Furthermore, due to scientific research requirements, scientific workflows are usually deployed on high-performance computing infrastructures, e.g. peer-to-peer, cluster, grid and cloud computing, to deal with a large number of data-intensive and computation-intensive activities [4–9]. Therefore, as an important dimension of workflow QoS (Quality of Service) constraints, temporal constraints are often set to ensure satisfactory efficiency of scientific workflow executions [10–13].

In traditional business workflows, workflow systems usually maintain an overall deadline (a global temporal constraint for the entire workflow instance) and several milestones (local temporal constraints for some important workflow segments) [12, 14, 15]. Most business workflows involve a number of tasks which require the execution and decision making by human resources. Since the performance of human resources is normally difficult to be predicted and controlled [16, 17], it is neither effective nor realistic to set too many temporal constraints along the business workflow processes. In the real world, most business workflows are also partially controlled by human managers. Therefore, a user-defined global deadline and several milestones are normally enough for human managers who can execute dynamical control over workflow executions to ensure on-time completion based on their own experiences [18].

In contrast to business workflow systems, scientific workflow systems are designed to be highly automatic to conduct large-scale scientific processes [2]. Instead of human managers, scientific workflows are controlled by workflow execution engines where predefined scheduling and exception handling strategies are implemented to control the underlying high-performance computing resources [19–23]. For example, in many scientific computing environments such as grid and cloud computing, resources are shared and competed by many users. Resources such as clusters and supercomputers usually maintain a job queue of their own and are managed by local schedulers rather than being under the full control of specific workflow execution engines outside the organization [9, 19, 24]. Therefore, to meet specific QoS requirements in scientific workflows, hierarchical scheduling is often employed [25]. For hierarchical scheduling, the central scheduler in the workflow execution engine is responsible for controlling the workflow execution based on the global QoS constraint and assigning workflow segments to local schedulers with local QoS constraints. Each local scheduler is responsible for scheduling activities in a workflow segment onto one single resource or multiple resources owned by one organization. Therefore, to facilitate hierarchical scheduling and many other tasks for delivering satisfactory temporal QoS in scientific workflow systems, besides a global temporal constraint, a large number of local temporal constraints are required. In order to maintain these temporal constraints, the issue of temporal constraint management is brought up in scientific workflow systems.

Specifically, temporal constraint management in scientific workflow systems includes two basic tasks: setting temporal constraints at build-time and updating temporal constraints at run-time. Here, to illustrate the requirement for these two tasks, we take workflow temporal verification as an example. As an important means to deliver satisfactory temporal QoS, many efforts have been dedicated to workflow temporal verification in the recent years. Different approaches for checkpoint selection and dynamic temporal verification are proposed as scientific workflow functionalities to improve the efficiency of temporal verification with the given temporal constraints [11, 26–28]. However, with the assumption that temporal constraints are predefined, most work focuses on run-time temporal verification while neglecting the fact that efforts put at run-time will be mostly in vain without build-time setting of high-quality temporal constraints [29]. The reason is obvious since the purpose of temporal verification is to identify the potential violations of temporal constraints to minimize the exception handling cost. Therefore, if temporal constraints are themselves of low quality, temporal violations are highly expected regardless of the efforts put on temporal verification. Meanwhile, with the assumption that temporal constraints are unchanged during workflow run-time, the task of run-time updating temporal constraints is neglected. However, to support the many run-time functionalities such as temporal verification and exception handling on temporal violations, local temporal constraints should be updated dynamically according to real activity durations (the global temporal constraint which serves as a type of QoS contract between clients

and service providers should normally stay unchanged unless new contract is signed). As will be discussed later in Section 2, both the time deficit (the time delay between the execution time and the temporal constraint) and the time redundancy (the time saving between the execution time and the temporal constraint) should be propagated to subsequent activities to support the time-related decision making process [30]. Therefore, setting build-time temporal constraints and updating run-time temporal constraints are the two basic tasks in temporal constraint management in scientific workflow systems.

Temporal constraints mainly include three types, i.e. upper bound, lower bound and fixed-time. An upper bound constraint between two activities is a relative time value so that the duration between them must be less than or equal to it. A lower bound constraint between two activities is a relative time value so that the duration between them must be greater or equal to it. A fixed-time constraint at an activity is an absolute time value by which the activity must be completed. As discussed in [11], conceptually, a lower bound constraint is symmetrical to an upper bound constraint and a fixed-time constraint can be viewed as a special case of upper bound constraint whose start activity is exactly the start activity of the whole workflow instance, hence they can be treated similarly. In the scientific workflow area, upper bound constraints are often used as a general case to facilitate research investigation [26, 28]. Therefore, in this paper, we focus on upper bound constraints only.

In this paper, our target is to investigate and address the issue of temporal constraint management in scientific workflows. Specifically, as mentioned above, there are two basic tasks for temporal constraint management: *setting temporal constraints* and *updating temporal constraints*. The task of setting temporal constraints is to assign a set of coarse-grained and fine-grained upper bound constraints to scientific workflows at workflow build-time. Here, *coarse-grained constraints* refer to those assigned to the entire workflow instance or workflow sub-processes, while *fine-grained constraints* refer to those assigned to individual activities. The task of updating temporal constraints is to update fine-grained temporal constraints according to real activity durations at workflow run-time.

To address the above issues, in this paper, a probabilistic strategy for temporal constraint management in scientific workflow systems is proposed. Our strategy utilizes a novel probability-based temporal consistency model where workflow activity durations are modelled as random variables with their structure weight. Here, the duration of a specific activity is defined as the time period from the submission of the activity until its completion [31]. Based on system historic data, the structure weight of a specific activity is defined according to its contribution to the completion time of the entire workflow instance. Its value is specified as the choice probability or statistic iteration times associated with the workflow path where the activity belongs. The basic idea for the structure weight is illustrated in Section 3 with the weighted joint distribution of four basic Stochastic Petri Nets [32, 33]-based building blocks, i.e. sequence, iteration, parallelism and choice. Our strategy supports an iterative and interactive negotiation process between the client (e.g. a user) and the service provider (e.g. a workflow system) through either a time-oriented or a probability-oriented fashion for setting coarse-grained upper bound temporal constraints. Thereafter, fine-grained temporal constraints associated with each activity can be propagated automatically. Our strategy also provides a probability time deficit propagation process and a probability time redundancy propagation process to update fine-grained temporal constraints in an automatic fashion at run-time. The effectiveness of our strategy is further demonstrated by a weather forecast scientific workflow in our scientific workflow management system.

The remainder of the paper is organized as follows. Section 2 presents a motivating example and the problem analysis. Section 3 proposes a novel probability-based temporal consistency model which facilitates our probabilistic strategy for temporal constraint management in scientific workflow systems. Section 4 presents the negotiation-based probabilistic strategy for setting temporal constraints at build-time. Section 5 presents the probability time deficit and time redundancy propagation process for updating temporal constraints at run-time. Section 6 demonstrates both the setting and updating processes by a case study with the motivating example to verify the effectiveness of our strategy. Section 7 introduces the implementation of the strategy in our scientific

workflow system. Section 8 presents the related work. Finally, Section 9 addresses our conclusions and points out the future work.

## 2. MOTIVATING EXAMPLE AND PROBLEM ANALYSIS

In this section, we introduce a weather forecast scientific workflow to demonstrate the motivation for temporal constraint management in scientific workflow systems. In addition, two basic requirements for temporal constraint management are presented.

### 2.1. Motivating example

The entire weather forecast workflow contains hundreds of thousands of data-intensive and computation-intensive activities [34]. The major data-intensive activities include the collection of meteorological information, e.g. surface data, atmospheric humidity, temperature, clouds area and wind speed from satellites, radars and ground observatories at distributed geographic locations. These data files are transferred via various kinds of networks. Computation-intensive activities mainly consist of solving complex meteorological equations, e.g. meteorological dynamics equations, thermodynamic equations, pressure equations, turbulent kinetic energy equations and so forth which require high-performance computing resources. Owing to the space constraints, it is not possible to present the whole forecasting process in detail. Here, we only focus on one of its segments for radar data collection. The graphic notations for Stochastic Petri Nets are illustrated in Figure 1 and the example workflow segment is depicted in Figure 2.

The classical Petri Net is a directed bipartite graph. It contains two types of nodes called places (represent conditions) and transitions (represent events/activities) which are connected via arcs (represent control flows). Stochastic Petri Net is a type of high-level Petri Net which extends with timing and probability features [32]. The notations of constraint start and constraint end represent the start point and the end point of a temporal constraint, respectively. The notation of structure weight represents the structure weight for the duration of an activity as defined in Section 3. The notation of duration distribution represents the distribution model for the duration of an activity.

For simplicity, we denote these activities in our example scientific workflow segment as  $X_1$  to  $X_{12}$ . The workflow process structures are composed of four Stochastic Petri Nets-based building blocks, i.e. a choice block for data collection from two radars at different locations (activities  $X_1 \sim X_4$ ), a compound block of parallelism and iteration for data updating and preprocessing (activities  $X_6 \sim X_{10}$ ), and two sequence blocks for data transferring (activities  $X_5, X_{11}, X_{12}$ ).

It is evident that the duration of these scientific workflow activities is highly dynamic in nature due to the data complexity and the computation environment. However, to ensure that the weather forecast is broadcast on time, every scientific workflow instance must be completed within a

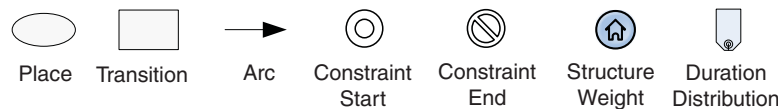


Figure 1. Graphic notations for Stochastic Petri Nets.

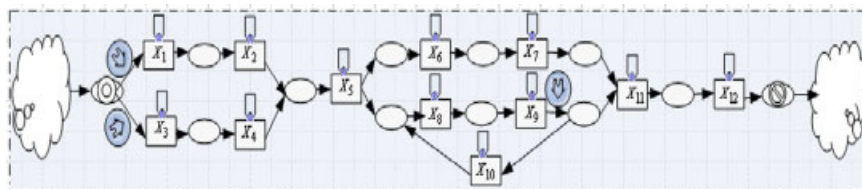


Figure 2. Example scientific workflow segment.

specific time duration. Therefore, a set of temporal constraints must be set to monitor the workflow execution time. For our example workflow segment, to ensure that the radar data can be collected on time and transferred for further processing, at least one overall upper bound temporal constraint is required. However, a coarse-grained temporal constraint is not effective enough to control fine-grained workflow execution time, e.g. the completion time of each workflow activity. It is evident that without the support of local enforcements, the overall workflow duration can hardly be guaranteed. For example, we set a two hour temporal constraint for this radar data collection process. But due to some technical problems, the connection to the two radars is broken and blocked in a state of retry and timeout for more than 30 min whereas its normal duration should be far less. Therefore, the two-hour overall temporal constraint for this workflow segment will probably be violated since its subsequent activities normally require more than 90 min to accomplish. However, no actions were taken due to the ignorance of the fine-grained temporal constraints. The exception handling cost for compensation of this time deficit, e.g. workflow rescheduling and recruitment of additional resources, is hence inevitable. Similar problems also take place in hierarchical workflow scheduling. If we only set a two-hour upper bound temporal constraint for the whole radar data collection process, it is difficult for a local scheduler to allocate a suitable time slot for activities  $X_6 \sim X_{10}$  in order to complete the data updating and pre-processing segment on time. That is why we also need to set fine-grained temporal constraints to each activity. Specifically, for this example workflow segment, at least one overall coarse-grained temporal constraint, and ideally, 12 fine-grained temporal constraints for activities  $X_1$  to  $X_{12}$  are required to be set.

Meanwhile, at workflow run-time, fine-grained temporal constraints need to be updated. For example, the local temporal constraint for the data collection sub-process (activities  $X_1 \sim X_4$ ) is 10 min and the local temporal constraint for the data updating and preprocessing (activities  $X_6 \sim X_{10}$ ) sub-process is 60 min given the overall 2-h constraint for the whole workflow segment. If at workflow run-time, due to some unexpected technical problems, the actual duration for the data collection sub-process is 30 min, then at activity  $X_5$ , a time deficit of 20 min will be detected. In order to resolve such a temporal violation, the durations of the subsequent activities  $X_6 \sim X_{12}$  need to be decreased to compensate such a 20-min delay. Normally, an exception handling strategy such as a local workflow rescheduling strategy will be triggered to tackle the occurring temporal violation. Therefore, the original temporal constraints for activities  $X_6 \sim X_{12}$  need to be updated. The new temporal constraints are required to facilitate the central scheduler to allocate faster resources or to facilitate local schedulers to assign closer time slots to decrease the queuing time. For another example, if the actual duration for activities  $X_6 \sim X_{10}$  is 30 min, then at activity  $X_{11}$ , we can detect that not only is the 20-min delay compensated but also there occurs a 10-min redundancy. In such a case, the original temporal constraints for data transferring activities  $X_{11} \sim X_{12}$ , for instance 10 min, can be increased to 20 min. In such a case, one possible strategy is that the priority of the data transferring activities is decreased so that the network can first meet the requirements of other urgent tasks. Another possible strategy is that the system manager chooses another network with less bandwidth so that the cost for data transfer can be reduced. With the former strategy, the overall temporal QoS of the scientific workflow system can be improved. With the latter strategy, the cost for scientific workflow execution can be reduced. Therefore, it is also important to update the local temporal constraints so as to fully utilize the time redundancy.

## 2.2. Problem analysis

From the illustration of the motivating example, it is evident that temporal constraint management plays an important role in scientific workflow systems. However, setting and updating temporal constraints are not straightforward tasks. Many factors such as workflow structures, system performance and user requirements should be taken into consideration. Here, we present the basic requirements for temporal constraint management by analyzing two criteria for high-quality temporal constraints.

- (1) *Temporal constraints should be well balanced between user requirements and system performance.* It is common that clients often suggest coarse-grained temporal constraints based

on their own interest while with limited knowledge about the actual performance of workflow systems. With our example, it is not rational to set a 60-min temporal constraint to the segment which normally needs 2 h to finish. Therefore, user-specified constraints are normally prone to causing frequent temporal violations. To address this problem, a negotiation process between the client and the service provider who is well aware of the system performance is desirable to derive balanced coarse-grained temporal constraints that both sides are satisfied with.

- (2) *Temporal constraints should facilitate both overall coarse-grained control and local fine-grained control.* As analyzed above, this criterion actually means that temporal constraint management should support both coarse-grained temporal constraints and fine-grained temporal constraints. Specifically, the task of setting build-time temporal constraints includes setting both coarse-grained temporal constraints (an overall deadline for the entire workflow instance and local temporal constraints for local workflow segments) and fine-grained temporal constraints (temporal constraints for individual workflow activities). However, although the overall workflow process is composed of individual workflow activities, coarse-grained temporal constraints and fine-grained temporal constraints are not in a simple relationship of linear culmination and decomposition. Meanwhile, it is impractical to set or update fine-grained temporal constraints manually for a large number of activities in scientific workflows. Since coarse-grained temporal constraints can be obtained through the negotiation process, the problem for setting fine-grained temporal constraints is how to automatically derive them based on the coarse-grained temporal constraints. Similarly, the problem for updating fine-grained temporal constraints is also how to automatically propagate the time deficit/redundancy in an efficient fashion.

To conclude, the basic requirements for temporal constraint management in scientific workflow systems can be put as: at build-time, effective negotiation for setting coarse-grained temporal constraints and automatically derive fine-grained temporal constraints; at run-time, automatically propagate time deficit/redundancy for updating local temporal constraints. To our best knowledge, the problem of temporal constraint management in scientific workflows has so far not been systematically investigated.

### 3. PROBABILITY-BASED TEMPORAL CONSISTENCY MODEL

In this section, we propose a novel probability-based temporal consistency model which utilizes the weighted joint distribution of workflow activity durations to facilitate temporal constraint management in scientific workflow systems.

#### 3.1. Weighted joint normal distribution for workflow activity durations

To define the weighted joint distribution of workflow activity durations, we first present two assumptions on the probability distribution of activity durations.

##### *Assumption 1*

The distribution of activity durations can be obtained from workflow system logs through statistic analysis [35]. Without losing generality, we assume that all the activity durations follow the normal distribution model, which can be denoted as  $N(\mu, \sigma^2)$ , where  $\mu$  is the expected value,  $\sigma^2$  is the variance and  $\sigma$  is the standard deviation [36].

##### *Assumption 2*

The activity durations are independent of each other.

For convenience of analysis, Assumption 1 chooses normal distribution to model the activity durations without losing generality. If most of the activity durations follow non-normal distribution, e.g. Uniform distribution, Exponential distribution, lognormal distribution or Beta distribution

[37], the idea of our strategy can still be applied in a similar way given different joint distribution models. However, we will leave the detailed investigation of different distribution models as our future work. Furthermore, as is commonly applied in the area of system simulation and performance analysis, Assumption 2 requires that the activity durations be independent of each other to facilitate the analysis of joint normal distribution. For those which do not follow the above assumptions, they can be treated by normal transformation and correlation analysis [36], or moreover, they can be ignored first when calculating joint distribution and then added up afterwards.

Furthermore, we present an important formula, *Formula 1*, of joint normal distribution.

*Formula 1*: If there are  $n$  independent variables of  $X_i \sim N(\mu_i, \sigma_i^2)$  and  $n$  real numbers  $\theta_i$ , where  $n$  is a natural number, then the joint distribution of these variables can be obtained with the following formula [36]:

$$Z = \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_n X_n = \sum_{i=1}^n \theta_i X_i \sim N\left(\sum_{i=1}^n \theta_i \mu_i, \sum_{i=1}^n \theta_i^2 \sigma_i^2\right) \tag{1}$$



Based on this formula, we define the weighted joint distribution of workflow activity durations as follows.

*Definition 1 (Weighted joint distribution)*

For a scientific workflow process  $SW$  which consists of  $n$  activities, we denote the activity duration distribution of activity  $a_i$  as  $N(\mu_i, \sigma_i^2)$  with  $1 \leq i \leq n$ . Then the weighted joint distribution is defined as

$$N(\mu_{SW}, \sigma_{SW}^2) = N\left(\sum_{i=1}^n w_i \mu_i, \sum_{i=1}^n w_i^2 \sigma_i^2\right)$$

where  $w_i$  stands for the weight of activity  $a_i$  that denotes the choice probability or iteration times associated with the workflow path where  $a_i$  belong.

The weight of each activity with different workflow structures is illustrated through the calculation of weighted joint distribution for basic Stochastic Petri Nets-based building blocks, i.e. sequence, iteration, parallelism and choice. These four building blocks consist of basic control flow patterns and are widely used in workflow modelling and structure analysis [14, 32]. Most workflow process models can be easily built by their compositions, and similarly for the weighted joint distribution of most workflow processes. Here, as introduced in Section 2, Stochastic Petri Nets-based modelling is employed to incorporate time and probability attributes with additional graphic notations, e.g.  stands for the probability of the path and  stands for the normal duration distribution of the associated activity. For simplicity, we illustrate with two paths for the iteration, parallelism and choice building blocks, except the sequence building block which has only one path by nature. However, the results can be effectively extended to more than two paths in a similar way.

(1) *Sequence building block*. As depicted in Figure 3, the sequence building block is composed of adjacent activities from  $a_i$  to  $a_j$  in a sequential relationship which means that the successor activity will not be executed until its predecessor activity is finished. The structure weight for each

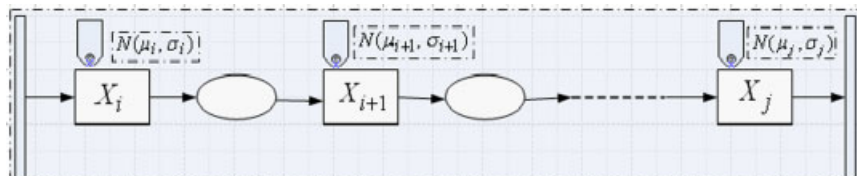


Figure 3. Sequence building block.

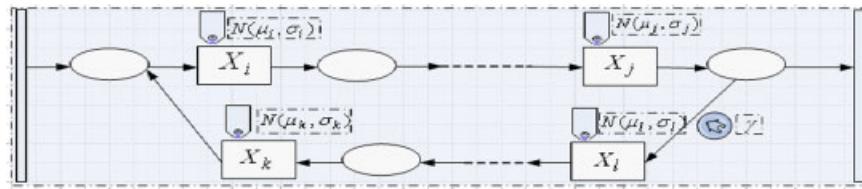


Figure 4. Iteration building block.

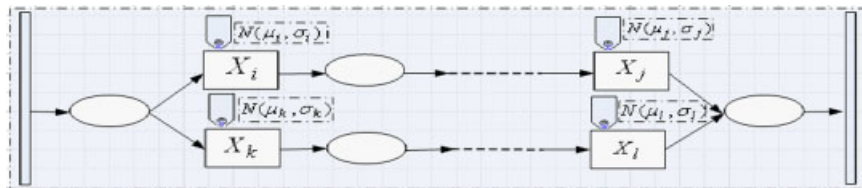


Figure 5. Parallelism building block.

activity in the sequence building block is 1 since they only need to be executed once. Therefore, according to *Formula 1*, the weighted joint distribution is

$$Z = \sum_{k=i}^j X_k \sim N \left( \left( \sum_{k=i}^j \mu_k \right), \left( \sum_{k=i}^j \sigma_k^2 \right) \right)$$

(2) *Iteration building block*. As depicted in Figure 4, the iteration building block contains two paths which are executed iteratively until certain end conditions are satisfied. Without the context of run-time workflow execution, it is difficult, if not impossible, to obtain the number of iteration times at workflow build-time. Therefore, in practice, the number of iteration times is usually estimated with the mean iteration times or with some probability distribution models such as normal, uniform or exponential distribution. In this paper, we use the mean iteration times to calculate the weighted joint distribution in the iteration building block. The major advantage for this simplification is to avoid the complex joint distribution (if exists) of activity durations (normal distribution) and the number of iteration times (may be normal or other non-normal distribution) in order to facilitate the setting of temporal constraints at build-time in an efficient fashion [36]. Here, to be consistent with the Stochastic Petri Nets, we assume that the probability of meeting the end conditions for a single iteration is  $\gamma$  (i.e. the mean iteration times is  $1/\gamma$ ) as denoted by the probability notation. Therefore, the lower path is expected to be executed for  $1/\gamma$  times and hence the upper path is executed for  $(1/\gamma)+1$  times. Accordingly, the structure weight for each activity in the iteration building block is the expected execution times of the path it belongs. Therefore, the weighted joint distribution here is

$$Z = ((1/\gamma)+1) \left( \sum_{p=i}^j X_p \right) + (1/\gamma) \left( \sum_{q=k}^l X_q \right) \sim N \left( ((1/\gamma)+1) \left( \sum_{p=i}^j \mu_p \right) + (1/\gamma) \left( \sum_{q=k}^l \mu_q \right), ((1/\gamma)+1)^2 \left( \sum_{p=i}^j \sigma_p^2 \right) + (1/\gamma)^2 \left( \sum_{q=k}^l \sigma_q^2 \right) \right)$$

(3) *Parallelism building block*. As depicted in Figure 5, the parallelism building block contains two paths which are executed in parallel. Since the activity durations are modelled by normal distributed variables, the overall duration time of the parallelism building block is equal to the distribution of the maximum duration of the two parallel paths. However, to calculate the exact distribution of the maximum of two random variables is a complex issue [38] which requires fundamental knowledge on statistics and non-trivial computation cost. Therefore, in practice, approximation

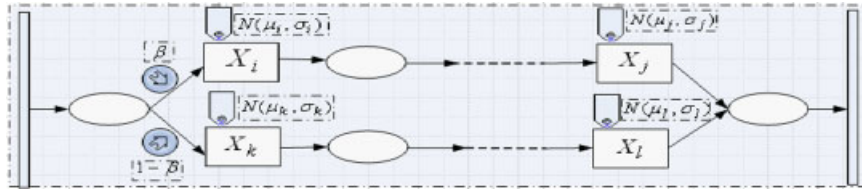


Figure 6. Choice building block.

is often applied instead of using the exact distribution. Since the overall completion time of the parallelism building block is dominated by the path with the longer duration [32], in this paper, we define the joint distribution of the parallelism building block as the joint distribution of the path with a larger expected duration, i.e. if  $\sum_{p=i}^j \mu_p \geq \sum_{q=k}^l \mu_q$  then  $Z = \sum_{p=i}^j \mu_p$ , otherwise  $Z = \sum_{q=k}^l \mu_q$ . Accordingly, the structure weight for each activity on the path with longer duration is 1 while on the other path it is 0. Therefore, the weighted joint distribution of this block is

$$Z = \begin{cases} \sum_{p=i}^j X_p \sim N\left(\sum_{p=i}^j \mu_p, \sum_{p=i}^j \sigma_p^2\right) & \text{if } \sum_{p=i}^j \mu_p \geq \sum_{q=k}^l \mu_q \\ \sum_{q=k}^l X_q \sim N\left(\sum_{q=k}^l \mu_q, \sum_{q=k}^l \sigma_q^2\right) & \text{otherwise} \end{cases}$$

(4) *Choice building block.* As depicted in Figure 6, the choice building block contains two paths in an exclusive relationship which means that only one path will be executed at run-time. The probability notation denotes that the probability for the choice of the upper path is  $\beta$  and hence the choice probability for the lower path is  $1 - \beta$ . In the real world,  $\beta$  may also follow some probability distributions. However, similar to the iteration building block, in order to avoid the complex joint distribution,  $\beta$  is estimated by the mean probability for selecting a specific path, i.e. the number of times that the path has been selected divided by the total number of workflow instances. Accordingly, the structure weight for each activity in the choice building block is the probability of the path it belongs. Therefore, the weighted joint distribution is

$$Z = \beta \left( \sum_{p=i}^j X_p \right) + (1 - \beta) \left( \sum_{q=k}^l X_q \right) \sim N \left( \beta \left( \sum_{p=i}^j \mu_p \right) + (1 - \beta) \left( \sum_{q=k}^l \mu_q \right), \beta^2 \left( \sum_{p=i}^j \sigma_p^2 \right) + (1 - \beta)^2 \left( \sum_{q=k}^l \sigma_q^2 \right) \right)$$

Note that the purpose of presenting the weighted joint normal distribution of the four basic building blocks is twofold. The first fold is to illustrate the definition of structure weight for workflow activity durations. The second is to facilitate the efficient calculation of weighted joint normal distribution of scientific workflows or workflow segments at build-time by the composition of the four basic building blocks. Furthermore, following the common practice in the workflow area [32, 39], approximations have been made to avoid calculating complex joint distribution. Since it is not the focus of this paper, the discussion on the exact distribution of these complex joint distribution models can be found in [36, 38].

### 3.2. Probability-based temporal consistency model

The weighted joint distribution enables us to analyze the completion time of the entire workflow from an overall perspective. Here, we need to define some notations. For a workflow activity  $a_i$ , its maximum duration, mean duration and minimum duration are defined as  $D(a_i)$ ,  $M(a_i)$  and  $d(a_i)$  respectively. For a scientific workflow  $SW$  which consists of  $n$  activities, its build-time

upper bound temporal constraint is denoted as  $U(SW)$ . In addition, we employ the ‘ $3\sigma$ ’ rule which has been widely used in statistical data analysis to specify the possible intervals of activity durations [37]. The ‘ $3\sigma$ ’ rule depicts that for any sample coming from normal distribution model, it has a probability of 99.73% to fall into the range of  $[\mu - 3\sigma, \mu + 3\sigma]$  which is a systematic interval of three standard deviations around the mean where  $\mu$  and  $\sigma$  are the sample mean and sample standard deviation, respectively. The statistic information can be obtained through scientific workflow system logs through statistical analysis [35]. Therefore, in this paper, we define the maximum duration, the mean duration and the minimum duration as  $D(a_i) = \mu_i + 3\sigma_i$ ,  $M(a_i) = \mu_i$  and  $d(a_i) = \mu_i - 3\sigma_i$  respectively. Accordingly, samples from the scientific workflow system logs which are above  $D(a_i)$  or below  $d(a_i)$  are hence discarded as outliers. The actual run-time duration at  $a_i$  is denoted as  $R(a_i)$ . Now, we propose the definition of probability-based temporal consistency which is based on the weighted joint distribution of activity durations. Note that, since temporal constraint management includes both setting temporal constraints at build-time and updating temporal constraints at run-time, our probability-based temporal consistency model also includes both definitions for build-time temporal consistency and run-time temporal consistency.

*Definition 2 (Probability-based temporal consistency Model)*

At build-time stage,  $U(SW)$  is said to be:

- (1) Absolute Consistency (AC), if  $\sum_{i=1}^n w_i(\mu_i + 3\sigma_i) < U(SW)$ ;
- (2) Absolute Inconsistency (AI), if  $\sum_{i=1}^n w_i(\mu_i - 3\sigma_i) > U(SW)$ ;
- (3)  $\alpha\%$  Consistency ( $\alpha\%C$ ), if  $\sum_{i=1}^n w_i(\mu_i + \lambda\sigma_i) = U(SW)$ .

At run-time stage, at a workflow activity  $a_p (1 < p < n)$ ,  $U(SW)$  is said to be:

- (1) Absolute Consistency (AC), if  $\sum_{i=1}^p R(a_i) + \sum_{j=p+1}^n w_j(\mu_j + 3\sigma_j) < U(SW)$ ;
- (2) Absolute Inconsistency (AI), if  $\sum_{i=1}^p R(a_i) + \sum_{j=p+1}^n w_j(\mu_j - 3\sigma_j) > U(SW)$ ;
- (3)  $\alpha\%$  Consistency ( $\alpha\%C$ ), if  $\sum_{i=1}^p R(a_i) + \sum_{j=p+1}^n w_j(\mu_j + \lambda\sigma_j) = U(SW)$ .

Here  $w_i$  stands for the weight of activity  $a_i$ ,  $\lambda (-3 \leq \lambda \leq 3)$  is defined as the  $\alpha\%$  confidence percentile with the cumulative normal distribution function of  $F(\mu_i + \lambda\sigma_i) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\mu_i + \lambda\sigma_i} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \bullet dx = \alpha\% (0 < \alpha < 100)$ . As depicted in Figure 7, different from the conventional multiple temporal consistency model where only four discrete coarse-grained temporal consistency states are defined [11, 26], in our temporal consistency model, every probability temporal consistency state is represented by a unique probability value and they together compose a Gaussian curve the same as the cumulative normal distribution [37]. Therefore, they can effectively support the requirements of both coarse-grained control and fine-grained control in scientific workflow systems as discussed in Section 2.2. The probability consistency states outside the confidence percentile interval of  $[-3, +3]$  are with continuous values infinitely approaching

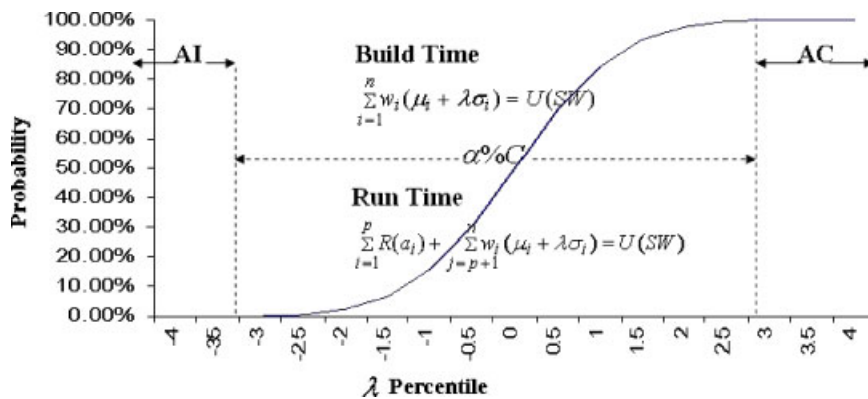


Figure 7. Probability-based temporal consistency.

0 or 100% respectively. However, since there are scarce chances (i.e.  $1 - 99.73\% = 0.17\%$ ) that the probability temporal consistency state will fall outside this interval, we name them absolute consistency (AC) and absolute inconsistency (AI) in order to distinguish them from others.

The purpose of probability-based temporal consistency is to facilitate the management of temporal constraints in scientific workflow systems. The advantage of the novel temporal consistency model mainly includes three aspects. First, clients normally cannot distinguish between qualitative expressions such as weak consistency and weak inconsistency due to the lack of background knowledge, and deteriorating the negotiation process for setting coarse-grained temporal constraints at build-time. In contrast, a quantitative temporal consistency state of 90 or 80% makes much more sense. Second, it is better to model activity duration as random variables instead of static time attributes in system environments with highly dynamic performance to facilitate statistic analysis. Third, to facilitate the setting of fine-grained temporal constraints at build-time and the updating of fine-grained temporal constraints at run-time, continuous states-based temporal consistency model where any fine-grained temporal consistency state is represented by a unique probability value is required rather than discrete multiple states-based temporal consistency models where temporal consistency states are represented by coarse-grained qualitative expressions. Therefore, in this paper, we propose the novel probability-based temporal consistency model.

#### 4. SETTING BUILD-TIME TEMPORAL CONSTRAINTS

In this section, we present our negotiation-based probabilistic strategy for setting temporal constraints at build-time. The strategy aims to effectively produce a set of coarse-grained and fine-grained temporal constraints which are well balanced between user requirements and system performance. As depicted in Table I, the strategy requires the input of process model and system logs. It consists of three steps, i.e. calculating weighted joint distribution, setting coarse-grained temporal constraints and setting fine-grained temporal constraints. We illustrate them accordingly in the following subsections.

##### 4.1. Calculating weighted joint distribution

The first step is to calculate weighted joint distribution. The statistic information, i.e. activity duration distribution and activity weight, can be obtained from system logs by statistical analysis [32, 35]. Afterwards, given the input process model for the scientific workflow, the weighted joint distribution of activity durations for the entire scientific workflow and workflow segments

Table I. Negotiation-based probabilistic setting strategy.

| Probabilistic strategy for setting temporal constraints |  |
|---|--|
| Overview  | Input: Process model and system logs for scientific workflow<br>Method: Probabilistic setting strategy<br>Output: Coarse-grained upper bound constraints and fine-grained upper bound constraints                      |
| Step 1: Calculating weighted joint distribution         | Obtain the statistic information (activity duration distribution and activity weight) from workflow system logs; calculate the weighted joint distribution of the workflow by the composition of basic building blocks |
| Step 2: Setting coarse-grained constraints              | Set coarse-grained temporal constraints through either the time-oriented or probability-oriented negotiation process based on weighted joint distribution and the probability-based temporal consistency               |
| Step 3: Setting fine-grained constraints                | Set fine-grained temporal constraints based on the same probability consistency as the coarse-grained temporal constraints obtained in Step 2  |

can be efficiently obtained by the composition of the four basic building blocks as illustrated in Section 3.1.

4.2. Setting coarse-grained temporal constraints

The second step is to set coarse-grained upper bound temporal constraints at build-time. Based on the four basic building blocks, the weighted joint distribution of an entire workflow or workflow segment can be obtained efficiently to facilitate the negotiation process for setting coarse-grained temporal constraints. Here, we denote the obtained weighted joint distribution of the target scientific workflow (or workflow segment)  $SW$  as  $N(\mu_{SW}, \sigma_{SW}^2)$  where  $\mu_{SW} = \sum_{i=1}^n w_i \mu_i$  and  $\sigma_{SW} = \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}$ . Meanwhile, we assume that the minimum threshold for the probability consistency is  $\beta\%$  which implies that client's acceptable bottom-line probability, namely the confidence for timely completion of the workflow instance; and the maximum threshold for the upper bound constraint is  $\max(SW)$  which denotes client's acceptable latest completion time. The actual negotiation process can be conducted in two alternative ways, i.e. time-oriented way and probability-oriented way.

The time-oriented negotiation process starts with the client's initial suggestion of an upper bound temporal constraint of  $U(SW)$  and the evaluation of the corresponding temporal consistency state by the service provider. If  $U(SW) = \mu_{SW} + \sigma_{SW}$  with  $\lambda$  as the  $\alpha\%$  percentile, and  $\alpha\%$  is below the threshold of  $\beta\%$ , then the upper bound temporal constraint needs to be adjusted, otherwise the negotiation process terminates. The subsequent process is the iteration that the client proposes a new upper bound temporal constraint which is less constrained than the previous one and the service provider reevaluates the consistency state, until it reaches or is above the minimum probability threshold.

In contrast, the probability-oriented negotiation process begins with the client's initial suggestion of a probability value of  $\alpha\%$ , the service provider evaluates the execution time  $R(SW)$  of the entire workflow process  $SW$  by the sum of all activity durations as  $\sum_{i=1}^n w_i(\mu_i + \lambda \sigma_i)$ , where  $\lambda$  is the  $\alpha\%$  percentile. If  $R(SW)$  is above the maximum upper bound constraint of  $\max(SW)$  for the client, the probability value needs to be adjusted, otherwise the negotiation process terminates. The following process is the iteration that the client proposes a new probability value which is lower than the previous one and the service provider reevaluates the workflow duration, until it reaches or is lower than the upper bound constraint.

As depicted in Figure 8, with the probability-based temporal consistency, the time-oriented negotiation process is normally where increasing upper bound constraints are proposed and evaluated with their temporal probability consistency states until the probability is above the client's bottom-line confidence, while the probability-oriented negotiation process is normally where decreasing temporal probability consistency states are proposed and estimated with their upper bound constraints until the constraint is below the client's acceptable latest completion time. In the

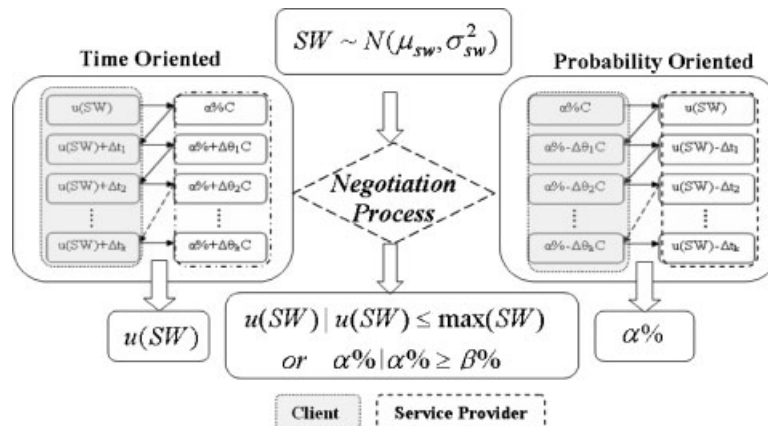


Figure 8. Negotiation process for setting coarse-grained.

real practice, the client and service provider can choose either of the two negotiation processes, or even interchange dynamically if they want. However, on the one hand, for clients who have some background knowledge about the execution time of the entire workflow or some of the workflow segments, they may prefer to choose the time-oriented negotiation process since it is relatively easier for them to estimate and adjust the coarse-grained constraints. On the other hand, for clients who have no enough background knowledge, the probability-oriented negotiation process is a better choice since they can make the decision by comparing the probability values of temporal consistency states with their personal bottom-line confidence values.

### 4.3. Setting fine-grained temporal constraints

The third step is to set fine-grained temporal constraints. In fact, this process is straightforward with the probability-based temporal consistency model. Since our temporal consistency actually defines that if all the activities are executed with the duration of  $\alpha\%$  probability and their total weighted duration equals their upper bound constraint, we say that the workflow process is  $\alpha\%$  consistency at build-time. For example, if the obtained probability consistency is 90% with the confidence percentile  $\lambda$  of 1.28 (the percentile value can be obtained from any normal distribution table or most statistic programs [36]), it means that all activities are expected for the duration of 90% probability. However, to ensure that the coarse-grained and fine-grained temporal constraints are consistent with the overall workflow execution time, the sum of weighted fine-grained temporal constraints should be approximate to their coarse-grained temporal constraint. Otherwise, even if the duration of every workflow activity satisfies its fine-grained temporal constraint, there is still a good chance that the overall coarse-grained temporal constraints will be violated, i.e. the workflow cannot complete on time. Therefore, based on the same percentile value, the fine-grained temporal constraint for each activity is defined with *Formula 2* to make them consistent with their overall coarse-grained temporal constraint.

*Formula 2:* For a scientific workflow or workflow segment  $SW$  which has a coarse-grained temporal constraint of  $U(SW)$  with  $\alpha\%$  consistency of  $\lambda$  percentile, if  $SW$  consists of  $n$  workflow activities with  $a_i \sim N(\mu_i, \sigma_i^2)$ , the fine-grained upper bound temporal constraint for activity  $a_i$  is  $U(a_i)$  and can be obtained with the following formula:

$$u(a_i) = \mu_i + \lambda \sigma_i \times \left( 1 - \left( \frac{\sum_{i=1}^n w_i \sigma_i - \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}}{\sum_{i=1}^n \sigma_i} \right) \right) \quad (2)$$

Here,  $\mu_i$  and  $\sigma_i$  are obtained directly from the mean value and standard deviation of activity  $a_i$  and  $\lambda$  denotes the same probability with the coarse-grained temporal constraint. Based on *Formula 2*, we can claim that with our setting strategy, the sum of weighted fine-grained temporal constraints is approximately the same as their overall coarse-grained temporal constraint. Here, we present a theoretical proof to verify our claim.

#### Proof

Assume that the distribution model for the duration of activity  $a_i$  is  $N(\mu_i, \sigma_i^2)$ , hence with *Formula 1*, the coarse-grained constraint is set to be of  $u(SW) = \mu_{SW} + \lambda \sigma_{SW}$ , where  $\mu_{SW} = \sum_{i=1}^n w_i \mu_i$  and  $\sigma_{SW} = \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}$ . As defined in *Formula 2*, the sum of weighted fine-grained constraints is  $\sum_{i=1}^n w_i u(a_i) = \sum_{i=1}^n w_i (\mu_i + \lambda \sigma_i \times (1 - (\sum_{i=1}^n w_i \sigma_i - \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}) / \sum_{i=1}^n \sigma_i))$ . Evidently, since  $w_i$  and  $\sigma_i$  are all positive values,  $\sum_{i=1}^n w_i \sigma_i \geq \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}$  holds and  $\sum_{i=1}^n \sigma_i$  is normally big for a large-size scientific workflow  $SW$ , hence the right-hand side of the equation can be extended and what we get is  $\sum_{i=1}^n w_i (\mu_i + \lambda \sigma_i \times (1 - A))$  where  $A$  equals  $(\sum_{i=1}^n w_i \sigma_i - \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}) / \sum_{i=1}^n \sigma_i$ . Therefore, it can be expressed as  $\sum_{i=1}^n w_i u(a_i) = \sum_{i=1}^n w_i \mu_i + \lambda \sum_{i=1}^n w_i \sigma_i - \Delta t_1$  (*Equation 1*) where  $\Delta t_1 = \sum_{i=1}^n w_i A$ . Meanwhile, since  $\sum_{i=1}^n w_i \sigma_i \geq \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2}$ , thus  $\sum_{i=1}^n w_i \mu_i + \lambda \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2} \leq \sum_{i=1}^n w_i \mu_i + \lambda \sum_{i=1}^n w_i \sigma_i$ . Therefore, it can be expressed as  $u(SW) = \sum_{i=1}^n w_i \mu_i +$

$\lambda\sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2} = \sum_{i=1}^n w_i \mu_i + \lambda \sum_{i=1}^n w_i \sigma_i - \Delta t_2$  (Equation II), where  $\Delta t_2$  equals  $\lambda(\sum_{i=1}^n w_i \sigma_i - \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2})$ . Furthermore, if we denote  $(\sum_{i=1}^n w_i \sigma_i - \sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2})$  as B, then we can have  $\Delta t_1 = (\sum_{i=1}^n w_i / \sum_{i=1}^n \sigma_i)B$  and  $\Delta t_2 = \lambda B$ . Since in real-world scientific workflows,  $(\sum_{i=1}^n w_i / \sum_{i=1}^n \sigma_i)$  is smaller than 1 due to  $\sum_{i=1}^n \sigma_i$  it is normally much bigger than  $\sum_{i=1}^n w_i$ , meanwhile,  $\lambda$  is a positive value smaller than 1 (1 means a probability consistency of 84.13% which is acceptable for most clients) [36],  $\Delta t_1$  and  $\Delta t_2$  are all relatively small positive values compared with the major component of Equations (1) and (2). Evidently, we can deduce that  $\sum_{i=1}^n w_i u(a_i) = \sum_{i=1}^n w_i \mu_i + \lambda \sum_{i=1}^n w_i \sigma_i - \Delta t_1 \approx \sum_{i=1}^n w_i \mu_i + \lambda \sum_{i=1}^n w_i \sigma_i - \Delta t_2 = u(WS)$ . Therefore, the sum of weighted fine-grained temporal constraints is approximately the same as the coarse-grained temporal constraint and thus our claim holds.  $\square$

## 5. UPDATING RUN-TIME TEMPORAL CONSTRAINTS

In this section, we propose our probabilistic updating strategy for run-time temporal constraints. At the scientific workflow run-time, build-time temporal constraints need to be updated according to run-time activity durations. As depicted in Table II, our probabilistic updating strategy consists of two major steps including calculating the probability time deficit/redundancy and updating fine-grained constraints. We illustrate them accordingly in the following subsections.

### 5.1. Calculating the probability time deficit/redundancy

Owing to none or limited context knowledge, it is difficult, if not impossible, to determine the concrete workflow structure of a workflow instance at build-time. Therefore, we utilize structure weights based on system historic data to facilitate the setting of temporal constraints. In contrast, at the scientific workflow run-time, there is normally some context knowledge available which can be used to determine the previous unknown information such as the execution path which will be selected in a choice building block. In such a case, the previously specified choice probability becomes ineffective. For instance, the probability needs to be modified to 1 for the selected path and 0 for others at run-time. Therefore, at the scientific workflow run-time, activity structure weights are subject to change according to the run-time execution results. However, for the workflow activities of those workflow paths which have not been determined yet, their structure weights are still effective as at build-time.

The task of updating the run-time temporal constraint is to automatically propagate the time deficit/redundancy to update fine-grained temporal constraints. Therefore, the time deficit/redundancy needs to be calculated first before the propagation process. Here, the effective

Table II. Probabilistic updating strategy.

| Probabilistic strategy for updating temporal constraints    |   |
|---|---|
| Overview  | Input: Build-time coarse-grained and fine-grained upper bound constraints, run-time activity durations<br>Method: Probabilistic updating strategy<br>Output: Updated run-time fixed-time coarse-grained and fine-grained temporal constraints |
| Step 1: Calculating the probability time deficit/redundancy | Based on the run-time activity durations, calculating the probability time deficit/redundancy   |
| Step 2: Updating fine-grained constraints                   | Time deficit/redundancy propagation process for activities on run-time workflow critical path<br>Time deficit/redundancy propagation process for activities on run-time workflow non-critical paths   |

workflow segment for time deficit/redundancy propagation is from the next activity point to the last activity point of the coarse-grained temporal constraint which covers the current activity point. During the effective workflow segment, as analyzed above, there are probably some workflow paths which have been determined but others that have not. Therefore, the issue of estimating the execution time at run-time is very different from its build-time counterpart since there is a mixture of determined and non-determined workflow paths. To solve such an issue, we define the run-time workflow critical path which can facilitate the estimation of the duration for the effective workflow segment.

*Definition 3 (Run-time Workflow Critical Path)*

Within the effective workflow segment for time deficit/redundancy propagation, the run-time workflow critical path is defined as the longest execution path from the start node to the end node of the workflow segment. Specifically, for those workflow paths which have been determined, all the activities are included; for those workflow paths which have not been determined, only the activities of the longest path are included. Here, the longest path is the path which has the maximum mean duration. For calculating the probability time deficit/redundancy, the choice probability for those longest paths in the previous choice building blocks is changed to 1.

Based on the definition of run-time workflow critical path, the probability time deficit and the probability time redundancy are defined as follows.

*Definition 4 (Probability Time Deficit)*

Given a scientific workflow or workflow segment  $SW$  with an upper bound constraint of  $U(SW)$ , at activity point  $a_p$ , let  $U(SW)$  be of  $\alpha\%C$  with the percentile of  $\lambda_\alpha$  which is below the threshold of  $\beta\%$  with the percentile of  $\lambda_\beta$  ( $\beta\%$  is the initial probability temporal consistency state agreed upon by clients and service providers at build-time through the negotiation process). Then the probability time deficit of  $U(SW)$  at  $a_p$  is defined as  $PTD(U(SW), a_p) = R(a_1, a_p) + (\sum_{k \in CriticalPath} w_k U(a_k)) - U(SW)$ , where  $w_k$  and  $U(a_k)$  are the structure weight and the fine-grained upper bound temporal constraint for activity  $a_k$ , respectively.

The probability time deficit is defined to measure the occurring time deficit at the activity point given the upper bound temporal constraint which is set on the last activity of a scientific workflow or workflow segment. In order to ensure on-time completion of scientific workflows and workflow segments, the probability time deficit needs to be propagated to decrease the subsequent fine-grained temporal constraints. As illustrated in Section 2.1 with the motivating example, in some cases, if the expected activity durations exceed their fine-grained temporal constraints, some exception handling strategies such as workflow rescheduling and resource recruitment [40, 41] may be triggered so as to avoid the possible violations of coarse-grained temporal constraints.

*Definition 5 (Probability Time Redundancy)*

At activity point  $a_p$ , let  $U(SW)$  be of  $\alpha\%C$  with the percentile of  $\lambda_\alpha$  which is above the threshold of  $\beta\%$  with the percentile of  $\lambda_\beta$ . Then the probability time redundancy of  $U(SW)$  at  $a_p$  is  $PTR(U(SW), a_p)$  which is equal to  $U(SW) - R(a_1, a_p) + (\sum_{k \in CriticalPath} w_k U(a_k))$ , where  $w_k$  and  $U(a_k)$  are the structure weight and the fine-grained upper bound temporal constraint for activity  $a_k$ , respectively.

The probability time redundancy is defined to measure the time redundancy at the current activity point given the upper bound temporal constraints. In order to save the execution cost or improve the overall temporal QoS, probability time redundancy needs to be propagated to increase the subsequent fine-grained temporal constraints. As illustrated in Section 2.1 with the motivating example, in some cases, if the fine-grained temporal constraints are large enough compared with the expected activity durations, activities can be re-allocated to less expensive resources to save the execution cost, or be postponed intentionally to decrease the queuing time of other urgent activities so as to improve the overall temporal QoS in the scientific workflow systems.

### 5.2. Updating fine-grained temporal constraints

After the probability time deficit/redundancy has been obtained, the next step is to propagate it to the subsequent fine-grained temporal constraints. Note that since the probability time deficit/redundancy is defined based on the critical path, the probability time deficit/redundancy should only apply to the fine-grained temporal constraints of those activities on the critical path. However, the fine-grained temporal constraints of those activities which are on the non-critical paths should also be updated. The reason can be explained as follows. For choice building blocks (which have not been determined), those non-critical paths still have the probability to be executed and hence require the update of fine-grained temporal constraints. For parallelism building blocks, those non-critical paths also need to be updated in case the durations of those non-critical paths exceed that of the critical path, i.e. non-critical paths may become the critical path, when large time deficits occur on non-critical paths. Similar situations may also occur on those sequence and iteration building blocks on the non-critical paths. Therefore, we not only need to update the fine-grained temporal constraints for the activities on the critical path, but also those for the activities on the non-critical paths. To address such an issue, in our strategy, we first conduct the probability time deficit/redundancy propagation process for activities on the critical path. Afterwards, based on the propagation results, the fine-grained temporal constraints for activities on the non-critical paths can be updated accordingly.

#### (1) Probability Time Deficit/Redundancy Propagation Process for Activities on Critical Path

To ensure fairness among subsequent activities, the probability time deficit/redundancy quota is defined which is based on the ratio of the mean time redundancy (i.e. the difference between the maximum and mean activity durations) with the mean activity durations. Given the current activity point  $a_p$ , the effective range for time deficit/redundancy propagation is from the next activity point  $a_{p+1}$  to the last activity, e.g.  $a_{p+m}$ , of the coarse-grained temporal constraint which covers  $a_p$ . Here, we denote the critical path in the effective range as *Critical Path*. Since the probability time deficit/redundancy is defined based on the critical path, the coefficient for the deficit quota of each activity on the critical path is hence defined as

$$\frac{\frac{D(a_i)-M(a_i)}{M(a_i)}}{\sum_{i \in \text{Critical Path}} \frac{D(a_i)-M(a_i)}{M(a_i)}} = \frac{\frac{\mu_i+3\sigma_i-\mu_i}{\mu_i}}{\sum_{i \in \text{Critical Path}} \frac{\mu_i+3\sigma_i-\mu_i}{\mu_i}} = \frac{\sigma_i/\mu_i}{\sum_{i \in \text{Critical Path}} \sigma_i/\mu_i}.$$

Therefore, given the probability time deficit  $PTD(a_p)$  or probability time redundancy  $PTR(a_p)$  at activity point  $a_p$ , the time deficit quota  $PTDQ(a_i)$  or time redundancy quota  $PTRQ(a_i)$  propagated to *Critical Path* are defined with *Formula 3* and *Formula 4* respectively:

$$PTDQ(a_i) = \frac{PTD(a_p) * \frac{w_j \sigma_j / \mu_j}{\mu_j} \sum_{j \in \text{Critical Path}} w_j \sigma_j / \mu_j}{w_i} \quad (3)$$

$$PTRQ(a_i) = \frac{PTR(a_p) * \frac{w_j \sigma_j / \mu_j}{\mu_j} \sum_{j \in \text{Critical Path}} w_j \sigma_j / \mu_j}{w_i} \quad (4)$$

Given the time deficit quota  $PTDQ(a_i)$  or time redundancy quota  $PTRQ(a_i)$  for  $a_i$ , and the build-time upper bound fine-grained temporal constraint  $U(a_i)$  for  $a_i$ ,  $U(a_i)$  is updated according to *Formula 5* or *Formula 6*, respectively.

$$F(a_i) = U(a_i) - PTDQ(a_i) \quad (5)$$

$$F(a_i) = U(a_i) + PTRQ(a_i) \quad (6)$$

(2) *Probability Time Deficit/Redundancy Propagation Process for Activities on Non-critical Paths*

Here, the basic idea is to apply the probability time deficit/redundancy quota of the longest path to the other paths. The motivation of applying the same probability time deficit/redundancy quota to non-critical paths can be explained as follows. Since the critical path is the longest path which has the maximum mean duration among all choice or parallel paths in the choice or parallelism building blocks, its probability time deficit quota will be the maximum one among all the paths if we calculate the probability time deficit quota for all the paths according to *Definition 3*. Similarly, the time redundancy quota of the longest path will be the minimum according to *Definition 4*. Therefore, in such a condition, if time deficit occurs, the sum of the updated fine-grained temporal constraints for all the activities on the non-critical paths will compensate for the time deficit since the maximum probability time deficit quota is propagated. Similarly, if time redundancy occurs, the sum of the updated fine-grained temporal constraints for all the activities on the non-critical paths will not exceed the coarse-grained temporal constraints since the minimum probability time redundancy quota is propagated. For example, if we assume that the mean duration for the longest path in a choice building block is 100 min and its probability time deficit quota is 10 min, then the probability time deficit quota of the other paths, e.g. a path with its mean duration of 60 min will be less than 10 min, e.g. 5 min, according to *Definition 3*. However, based on our probability time deficit propagation process, the probability time deficit quota for those non-longest paths is also set as 10 min. Therefore, given our method, whether at run-time the longest path or the other non-longest paths are selected, the sum of the updated fine-grained temporal constraints can compensate for the occurring probability time deficit since the maximum probability time deficit quota of 10 min has already been propagated. Another example similar is that if the probability time redundancy quota for the longest path is 5 min, then according to *Definition 4*, the probability time redundancy quota for the other non-longest path will be larger than 5 min e.g. 8 min. However, based on our probability time redundancy propagation process, the probability time redundancy quota for those non-longest paths is also set as 5 min, i.e. the same as the longest path. Therefore, at run-time whether the longest path or the other non-longest paths are selected, the sum of the updated fine-grained temporal constraints will not exceed the coarse-grained temporal constraints since only the minimum probability time redundancy quota of 5 min has been propagated.

Specifically, the probability time deficit/redundancy propagation process for activities on non-critical paths is described as follows:

After the fine-grained temporal constraints of the activities on the critical path have been updated, the following issue is to update the fine-grained temporal constraints of the activities on non-critical paths. Here, we assume that the longest path in the choice or parallelism building block is denoted as  $LP$ . The fine-grained temporal constraints of activities on  $LP$  have been updated and the sum of their probability time deficit and time redundancy quota are denoted as  $PTDQ(LP)$  and  $PTRQ(LP)$  respectively. Here, the probability time deficit/redundancy quota for the other non-longest paths are defined the same as that of the longest path as shown in *Formula 7* and *Formula 8*, respectively.

$$PTDQ(P_i | P_i \neq LP) = PTDQ(LP) \quad (7)$$

$$PTRQ(P_i | P_i \neq LP) = PTRQ(LP) \quad (8)$$

After that, the probability time deficit/redundancy quota for those activities on the other non-longest paths is defined the same as in *Formula 4/Formula 5*, and their fine-grained temporal constraints are updated according to *Formula 6/Formula 7*.

### 5.3. Updating frequency and overhead

Note that the probability time deficit/redundancy propagation processes need to be conducted numerous times in order to update fine-grained temporal constraints. However, although the computation cost for a single propagation process is trivial, it is unnecessary to update fine-grained temporal constraints every time when a minor time deficit/redundancy takes place. In practice,

there are two alternative ways to update fine-grained temporal constraints in a batch fashion. The first one is to set a time deficit/redundancy threshold. Accordingly, the propagation process will only be conducted when the accumulated deficit/redundancy exceeds the threshold. For example, if the threshold is set as 10 min, then the propagation process will be conducted if and only if the accumulated deficit/redundancy exceeds the 10 min threshold. An alternative way is to set a fixed size for workflow activities so that the propagation process will only be conducted at those activity points with a distance of the fixed size in between. For example, if the fixed size for workflow activities is set as 20, then the propagation process will be conducted only on those activities such as the 20th, 40th, 60th and so on. Besides the above two intuitive yet practical methods, some sophisticated strategies which can choose specific activity points to conduct certain actions such as some work in temporal checkpoint selection could be referred [11, 28]. However, since it is not the focus of this paper, we will leave it as our future work.

As regards the overhead of the updating process, the major overhead is the calculation of the probability time deficit/redundancy. However, since all the required information despite the run-time durations of completed activities such as the activity duration distribution model, the structure weight and the workflow run-time critical path is either already available or can be easily obtained by simple computation based on the build-time setting results, with a moderate updating frequency, the overhead for updating the fine-grained temporal constraint is acceptable.

## 6. CASE STUDY

In this section, we evaluate the effectiveness of our probabilistic strategy for temporal constraint management by further illustrating the motivating example introduced in Section 2.1. The process model is the same as depicted in Figure 1. Since our strategy consists of build-time setting temporal constraints and run-time updating temporal constraints, the evaluation also includes two consecutive parts.

Here, we first illustrate our probabilistic strategy for build-time setting temporal constraints. As presented in Table I, the first step is to calculate the weighted joint distribution. Based on statistical analysis and the '3 $\sigma$ ' rule, the normal distribution model and its associated weight for each activity duration are specified through statistical analysis of accumulated system logs. As the detailed specification of the workflow segment depicted in Table III, the weighted joint distribution of each building block can be derived instantly with their formulas proposed in Section 4. We obtain the weighted joint distribution as  $N(6190, 217^2)$  with seconds as the basic time unit.

The second step is the negotiation process for setting an overall upper bound temporal constraint for this workflow segment. Here, we first illustrate the time-oriented negotiation process. We assume that the client's bottom-line confidence of the probability consistency state is 80%. The client starts to propose an upper bound temporal constraint of 6250 s, based on the weighted joint distribution of  $N(6190, 217^2)$  and the cumulative normal distribution function, the service provider can obtain the percentile as  $\lambda=0.28$  and reply with the probability of 61% which is lower than the threshold of 80%. Hence the service provider advises the client to relax the temporal constraint. Later, for example, the client proposes a series of new candidate upper bound temporal constraints one after another, e.g. 6300s, 6360s and 6380s, and the service provider replies with 69%, 78% and 81% as the corresponding temporal consistency states. Since 81% is higher than the 80% minimum threshold, therefore, through the time-oriented negotiation process, the final negotiation result could be an upper bound temporal constraint of  $6190+0.88*217=6380$  s with a probability consistency state of 81% where 0.88 is the 81% probability percentile. As regards the probability-oriented negotiation process, we assume that the client's acceptable latest completion time is 6400s. The client starts to propose a probability temporal consistency state of 90%, based on the weighted joint distribution of  $N(6190, 217^2)$  and the cumulative normal distribution function, the service provider reply with an upper bound temporal constraint of 6468 s which is higher than the threshold. Afterwards, for example, the client proposes a series of new candidate probability temporal consistency states one after another, e.g. 88, 85 and 83%, and the service provider replies

Table III. Specification of the workflow segment.

| Workflow activities               |      |          |        | Joint distribution  |  |
|-----------------------------------|------|----------|--------|---|--|
| Activity                          | Mean | Variance | Weight | Building blocks   | Weighted joint distribution                              |
| Activity $X_1$ ;                  | 105  | 225      | 0.67   | Choice. The probability for the upper path is   | Mean = $0.67 * (105 + 223) + 0.33 * (256 + 358) = 422$   |
| Activity $X_2$                    | 223  | 289      | 0.67   |   |  |
| Activity $X_3$                    | 256  | 529      | 0.33   | 66.7%; the lower path is 33.3%  | Variance = $0.67^2(225 + 289) + 0.33^2(529 + 400) = 331$ |
| Activity $X_4$                    | 358  | 400      | 0.33   |   |  |
| Activity $X_5$                    | 558  | 784      | 1      | Sequence  | Mean = 558, Variance = 784                               |
| Activity $X_6$                    | 650  | 1089     | 0      | Parallelism and iteration.  | Mean = $5 * (125 + 285) + 4 * 594 = 4426$                |
| Activity $X_7$                    | 230  | 225      | 0      | The probability for a single iteration is 25%   | Variance = $5^2 * (64 + 1444) + 4^2 * 484 = 45444$       |
| Activity $X_8$                    | 125  | 64       | 5      |   |  |
| Activity $X_9$                    | 285  | 1444     | 5      |   |  |
| Activity $X_{10}$                 | 594  | 484      | 4      |   |  |
| Activity $X_{11}$                 | 661  | 529      | 1      | Sequence  | Mean = $661 + 123 = 784$ ;                               |
| Activity $X_{12}$                 | 123  | 64       | 1      |   | Variance = $529 + 64 = 593$                              |
| Overall weight joint distribution |      |          |        | Mean = $422 + 558 + 4426 + 784 = 6190$ ;<br>Variance = $331 + 784 + 45444 + 593 = 47152$<br>The overall weighted joint distribution for the workflow segment $\Rightarrow N(6190, 217^2)$ |  |

Table IV. Setting results.

|   |   |                    |
|---|---|--------------------|
| Overall weight joint distribution                         | $N(\mu_{SW}, \sigma_{SW}^1) = N(6190, 217^2)$ , |                    |
| Coarse-grained upper bound temporal constraint            |   |                    |
| $u(SW) = 6380s$ with 81% consistency and $\lambda = 0.88$ |   |                    |
| Fine-grained upper bound temporal constraints             |   |                    |
| $U(X_1) = 108s$   | $U(X_2) = 227s$                                 | $U(X_3) = 261s$    |
| $U(X_4) = 362s$   | $U(X_5) = 564s$                                 | $U(X_6) = 657s$    |
| $U(X_7) = 233s$   | $U(X_8) = 127s$                                 | $U(X_9) = 293s$    |
| $U(X_{10}) = 599s$  | $U(X_{11}) = 666s$                              | $U(X_{12}) = 125s$ |

with 6445, 6415 and 6397s as the corresponding temporal consistency states. Since 6397s is lower than the 6400s maximum threshold, through probability-oriented negotiation process, the final negotiation result could be an upper bound temporal constraint of 6397s with a probability temporal consistency state of 83%. Evidently, from this example, with the result of 6380 and 6397s obtained through two different negotiation processes, we can confirm that the setting process is effective regardless of which kind of negotiation process is adopted. Furthermore, the final coarse-grained temporal constraints obtained are normally similar if the decision maker is the same client. The setting result of the time-oriented negotiation process is presented in Table IV.

The third step is to set the fine-grained temporal constraints for each workflow activity with the obtained overall upper bound constraint. As we mentioned in Section 4, the probability-based temporal consistency defines that the probability for each expected activity duration is the same as the probability consistency state of the workflow process. Therefore, taking the result obtained through the time-oriented negotiation process for illustration, since the coarse-grained temporal constraint is 6380s with a probability consistency state of 81%, according to Formula 2, the fine-grained temporal constraints for each activity can be obtained instantly. Since  $\sum_{i=1}^n w_i \sigma_i = 412$ ,  $\sqrt{\sum_{i=1}^n w_i^2 \sigma_i^2} = 217$  and  $\sum_{i=1}^n \sigma_i = 250$ , the coefficient here is  $1 - (412 - 217) / 250$  which equals to 0.22. Therefore, for example, the fine-grained upper bound temporal constraint for activity  $X_1$  is  $(105 + 0.88 * \sqrt{225} * 0.22) = 108s$  and the constraint for activity  $X_{12}$  is  $(123 + 0.88 * \sqrt{64} * 0.22) = 125s$ . The detailed results are presented in Table IV.

Table V. Updating results.

| Probability time deficit  |                      | $PTD(U(SW), X_5)=200$ |
|---|----------------------|-----------------------|
| <i>PTDQ</i> for activities on run-time workflow critical path     |                      |                       |
| $PTDQ(X_8)=10.4s$   | $PTDQ(X_9)=21.6s$    | $PTDQ(X_{10})=6s$     |
| $PTDQ(X_{11})=5.5s$   | $PTDQ(X_{12})=10.5s$ |                       |
| <i>PTDQ</i> for activities on run-time workflow non-critical path |                      |                       |
| $PTDQ(X_6)=80s$   | $PTDQ(X_7)=104s$     |                       |
| Updated fine-grained temporal constraints                         |                      |                       |
| $U(X_6)=577s$   | $U(X_7)=129s$        | $U(X_8)=116.6s$       |
| $U(X_9)=271.4s$   | $U(X_{10})=593s$     | $U(X_{11})=660.5s$    |
| $U(X_{12})=114.5s$  |                      |                       |

Now we further illustrate our strategy for run-time updating temporal constraints. Here, assume that the fixed size for workflow activities is set as 5, then the updating process will be conducted on  $X_5$  and  $X_{10}$ . Here, we first take activity  $X_5$  as an example. If the second radar is selected, i.e. the lower path in the first choice building block is selected, and activity durations for  $X_3$ ,  $X_4$  and  $X_5$  are 248, 445 and 600s, respectively. Here, the effective range for updating temporal constraints is from  $X_6$  to  $X_{12}$  and the critical path is easily identified as  $(X_8, X_9, X_{10}, X_{11}, X_{12})$  from the results shown in Table III. Therefore, at  $X_5$ , there occurs a probability time deficit of 200s since  $PTD(U(SW), X_5) = R(a_1, a_5) + \sum_{k \in \text{Critical Path}} w_k U(a_k) - U(SW) = 1293 + 5287 - 6380 = 200s$ . Therefore, we will first update the fine-grained temporal constraints for the activities on the critical path and then update those for the activities on the non-critical path, i.e.  $(X_6, X_7)$  in the sequence building block. The updating results are shown in Table V.

Based on *Formula 4*, it is easy to calculate the probability time deficit quota for each activity on the critical path. After that, the sum of the probability time deficit quota for the non-critical path, i.e.  $(X_6, X_7)$ , is directly set the same as its counterpart of the longest path  $(X_8, X_9, X_{10})$ , i.e. 184s. Therefore, we can obtain the probability time deficit quota for  $X_6$  and  $X_7$  as 80 and 104s, respectively. The fine-grained temporal constraints are hence updated according to *Formula 6*.

Here, in order to verify the effectiveness of the updated fine-grained temporal constraints, we test the sum of the estimated execution time to check whether it can compensate for the occurring 200s time deficit. Since  $R(a_1, a_5) = 1293s$ , and the sum of the fine-grained temporal constraints is equal to  $\sum_{i=6}^{12} w_i U(a_i) = 5087$ , hence the estimated execution time for the workflow segment is  $R(a_1, a_5) + \sum_{i=6}^{12} w_i U(a_i) = 6380s$  which is equal to the coarse-grained temporal constraint set at build-time as shown in Table IV. Therefore, the updated fine-grained temporal constraints can ensure that the occurring 200s time deficit can be compensated for by our probability time deficit propagation process.

Since the probability time redundancy propagation process is symmetrical to the probability time deficit propagation process as illustrated above, its evaluation is omitted here.

To conclude, the above demonstration of the setting and updating process evidently shows that our probabilistic strategy is effective for the management of temporal constraints in scientific workflow systems. It has met the two basic requirements proposed in Section 2: at build-time, effective negotiation for setting coarse-grained temporal constraints and automatically derive fine-grained temporal constraints; at run-time, automatically propagate time deficit and time redundancy for updating local temporal constraints. As regards the overhead of the setting process, the major overhead is the calculation of weighted joint normal distribution. However, as presented in Section 6, with the Stochastic Petri Nets-based modelling tool provided in our scientific workflow system, the weighted joint normal distribution can be obtained on-the-fly with the user's modelling process. The four basic building blocks can speed up both the modelling process and the calculation of weighted joint normal distribution for scientific workflows. After that, the negotiation process for setting coarse-grained temporal constraints is totally under the control of the stakeholders and the propagation process for setting fine-grained temporal constraints can be done instantly. As for the

overhead of the updating process, the major overhead is the calculation of the probability time deficit/redundancy. However, since all the required information despite the run-time durations of completed activities such as the activity duration distribution model, the structure weight and the workflow run-time critical path is either already available or can be easily obtained based on the build-time setting results, as demonstrated above; with the guarantee of effectiveness, the overhead of temporal constraint management with our probabilistic strategy is acceptable.

## 7. SYSTEM IMPLEMENTATION

In this section, we introduce the implementation of the setting strategy in our SwinDeW-G scientific workflow system.

### 7.1. SwinDeW-G scientific workflow system

SwinDeW-G (*Swinburne Decentralised Workflow for Grid*) is a peer-to-peer-based grid workflow system running on the SwinGrid (Swinburne service Grid) platform [8]. An overall picture of SwinGrid is depicted in Figure 9 (bottom plane).

SwinGrid contains many grid nodes distributed in different places. Each grid node contains many computers including high-performance PCs and/or supercomputers composed of a significant number of computing units. The primary hosting nodes include the Swinburne CS3 (Centre for Complex Software Systems and Services) Node, Swinburne ESR (Enterprise Systems Research laboratory) Node, Swinburne Astrophysics Supercomputer Node and Beihang CROWN (China R&D environment Over Wide-area Network) Node in China. They are running Linux, GT4 (Globus Toolkit) or CROWN grid toolkit 2.5, where CROWN is an extension of GT4 with more middleware, hence compatible with GT4. Besides, the CROWN Node is also connected to some other nodes such as those in Hong Kong University of Science and Technology and University of Leeds in U.K. The Swinburne Astrophysics Supercomputer Node is cooperating with PfC (Australian Platform for Collaboration) and VPAC (Victorian Partnership for Advanced Computing). Currently, SwinDeW-G is deployed at all primary hosting nodes as exemplified in the bottom of plane of Figure 9. In SwinDeW-G, a scientific workflow is executed by different peers that may be distributed at different grid nodes. As shown in Figure 9, each grid node can have a number of peers, and each peer can be simply viewed as a grid service. In the top plane of Figure 9, we show a sample of how a scientific workflow can be executed in the simulation environment.

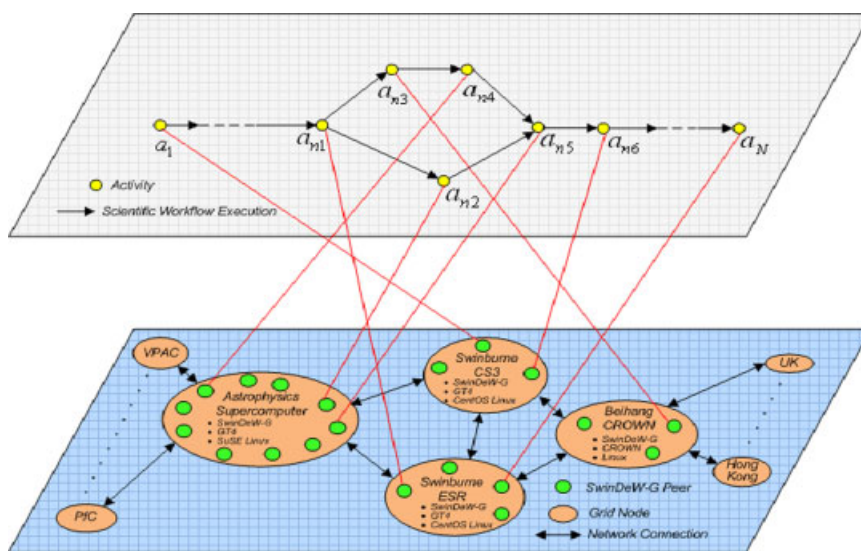







Figure 9. Overview of SwinDeW-G environment.

As an important reinforcement for the overall workflow QoS, temporal verification is being implemented in SwinDeW-G. It currently supports dynamic checkpoint selection and temporal verification at run-time [26, 27]. After the running of SwinDeW-G for a period of time, statistical analysis can be applied to accumulated system logs to obtain probability attributes [35, 37]. The probabilistic strategy for setting temporal constraints is being integrated into the scientific workflow modelling tool which supports Stochastic Petri Nets-based modelling, composition of building blocks, temporal data analysis, interactive and automatic setting of temporal constraints.

## 7.2. SwinDeW-G scientific workflow modelling tool

Our probabilistic strategy for temporal constraint management is being implemented into our SwinDeW-G scientific workflow system as integrated components. Specifically, the setting strategy is included in the SwinDeW-G modelling tool which is in charge of the modelling and setting of QoS constraints at workflow build-time, the updating strategy is included in the SwinDeW-G temporal adjustment tool which is in charge of maintaining temporal correctness at workflow run-time. As shown in Figure 10(a), the modelling tool adopts Stochastic Petri Nets with additional graphic notations, e.g.  for probability,  for activity duration,  for a sub-process,  for the start point and  for the end point of an upper bound temporal constraint, to support explicit representation of temporal information. It also supports the composition of four basic building blocks and user-specified ones.

The modelling tool supports temporal data analysis from workflow system logs. Temporal data analysis follows the '2 $\sigma$ ' rule and can generate the normal distribution model for each activity duration. The probability attributes for each workflow structure such as the choice probability and iteration times can also be obtained through statistical analysis on historic workflow instances from system logs. After temporal data analysis, the attributes for each activity, i.e. its mean duration,



Figure 10. SwinDeW-G Constraint Management Tool. (a) Temporal data analysis; (b) setting temporal constraints; and (c) updating temporal constraints.

variance, maximum duration, minimum duration and weight are associated with the corresponding activity and explicitly displayed to the client. Meanwhile, the weighted joint distribution of the target process is obtained automatically with basic building blocks. As shown in Figure 10(b), with our probability-based temporal consistency, the client can specify either an upper bound temporal constraint or a probability consistency state, and afterwards, the system will instantly reply with the corresponding probability state or the upper bound temporal constraint.

Based on the visualized results shown by a Gaussian curve (the cumulative normal distribution), the client can decide whether to accept or decline the results. If the client is not satisfied with the outcomes, he or she can specify a new value for evaluation until a satisfactory result is achieved. Evidently, the negotiation process between the client and the service provider is implemented as an interactive process between the system user and our developed program. After setting the coarse-grained temporal constraints, the fine-grained constraints for each activity are propagated automatically. These activity duration distribution models, coarse-grained and fine-grained temporal constraints are explicitly represented in the scientific workflow models and will be further deployed to facilitate the control of the overall workflow execution time by run-time temporal verification in scientific workflows. For a detailed discussion on how temporal constraints are employed in scientific workflow temporal verification refer to [27, 28].

At workflow run-time, fine-grained temporal constraints are updated according to run-time activity durations. As shown in Figure 10(c), based on run-time activity durations and the statistical information obtained at build-time, the run-time workflow critical path (as the execution path highlighted in the 'Run-Time Workflow Critical Path' interface) and the probability time deficit/redundancy (as shown in the 'Run-Time Activity Durations' interface) can be derived automatically. As a component in the run-time temporal adjustment tool, the probability time deficit/redundancy propagation process is normally conducted as a background program given the system-defined frequency (based on either a time deficit/redundancy threshold or a fixed size for workflow activities as addressed in Section 5.3) in an automatic fashion. Meanwhile, it can also be conducted as a foreground program to facilitate the decision making of system managers. When working as a foreground program, temporal constraints will be updated under manual control. Therefore, in our SwinDeW-G constraint management tool, the run-time fine-grained temporal constraints can be updated accordingly either in an automatic or manual fashion.

## 8. RELATED WORK

In this section, we review some related work on temporal constraints in workflow systems. The work in [30] presents the taxonomy of grid workflow QoS constraints which include five dimensions, i.e. time, cost, fidelity, reliability and security. Some papers have presented an overview analysis of scientific workflow QoS [6, 42, 43]. The work in [10] presents the taxonomy of grid workflow verification which includes the verification of temporal constraints. In a distributed environment such as a distributed soft real-time system, a task is usually divided into several subtasks to be executed in a specific order at different sites. Therefore, the issue of automatically translating the overall deadline into deadlines for the individual subtasks is investigated in [15]. Generally speaking, there are two basic ways to assign QoS constraints, one is activity-level assignment and the other is workflow-level assignment. Since the whole workflow process is composed of all individual activities, an overall workflow-level constraint can be obtained by the composition of activity-level constraints. On the contrary, activity-level constraints can also be assigned by the decomposition of workflow-level constraints [30]. However, different QoS constraints have their own characteristics and require in-depth research to handle different scenarios.

As shown in our probabilistic strategy, the primary information required for temporal constraint management includes the workflow process models, statistics of activity durations and the temporal consistency model. Scientific workflows require the explicit representation of temporal information, i.e. activity durations and temporal constraints to facilitate temporal verification. One of the classical modelling methods is the Stochastic Petri Nets [32, 33] which incorporates time and

probability attributes into workflow processes that can be employed to facilitate scientific workflow modelling. Activity duration, as one of the basic elements to measure the system performance, is of significant value to workflow scheduling, performance analysis and temporal verification [9, 19, 28, 39]. Most work obtains activity durations from workflow system logs and describes them by a discrete or continuous probability distribution through statistical analysis [32, 29]. As regard temporal consistency, traditionally, there are only binary states of consistency or inconsistency. However, as stated in [26], it argues that the conventional consistency condition is too restrictive and covers several different states which should be handled differently for the purpose of cost-effectiveness. Therefore, it divides conventional inconsistency into weak consistency, weak inconsistency and strong inconsistency and treats them accordingly. However, multiple-state-based temporal consistency model cannot support quantitative measurement of temporal consistency states and lacks the ability to support statistical analysis for constraint management. Therefore, in our preliminary work [29], a probability-based build-time temporal consistency model is presented to facilitate the setting of temporal constraints. Furthermore, in this paper, its run-time counterpart is presented to facilitate the updating of temporal constraints.

Temporal constraints are not well emphasized in traditional workflow systems. However, some business workflow systems accommodate temporal information for the purpose of performance analysis. For example, Staffware provides the audit trail tool to monitor the execution of individual instances [14] and the SAP business workflow system employs the workload analysis [44]. As regards the support of temporal constraints in scientific workflow systems, a survey was conducted by us based on some of the work reported in [10, 30]. Since workflow modelling is highly related to the specification of temporal constraints, the survey also concerns the two aspects of the modelling language and the modelling tool (language-based, graph-based or both) in addition to the three aspects of whether they support the specification of temporal constraints (the specification of temporal constraints in workflow models), the management of temporal constraints (i.e. the setting and updating of temporal constraints) and the temporal verification (the verification of temporal constraints). As shown in Table VI, among the 10 representative scientific workflow projects (ASKALON [45], CROWN [46], DAGMan [47], GridBus [48], JOpera [49], Kepler [50], SwinDeW-G [8], Taverna [51], Triana [52] and UNICORE [53]), most projects are using XML-like modelling language and support language-based or graph-based modelling tool. Therefore, in the modelling stage, a temporal constraint can either be inexplicitly specified as an element in the XML document or explicitly as a graphic component in the workflow template. As regards the representation of temporal constraints, the management of temporal constraints and the support

Table VI. A survey on the support of temporal constraints.

| Scientific workflow systems | Modelling language | Modelling tool                | Temporal constraint specification | Temporal constraint management | Temporal constraint verification |
|-----------------------------|--------------------|-------------------------------|-----------------------------------|--------------------------------|----------------------------------|
| ASKALON                     | AGWL               | Language-based<br>Graph-based | Supported                         | N/A                            | N/A                              |
| CROWN                       | GPEL               | Language-based                | N/A                               | N/A                            | N/A                              |
| DAGMsm                      | DAG Scripts        | Language-Based                | Supported                         | N/A                            | N/A                              |
| GriclBus                    | KWPL               | Language-based<br>Graph-based | Supported                         | N/A                            | N/A                              |
| JOpera                      | JVCL               | Language-based<br>Graph-based | Supported                         | N/A                            | N/A                              |
| Kepler                      | SDF                | Graph-based                   | Supported                         | N/A                            | N/A                              |
| SwinDeW-G                   | XPDL/BPEL          | Graph-based                   | Supported                         | Supported                      | Supported                        |
| Taverna                     | SCUFL              | Language-based<br>Graph-based | Supported                         | N/A                            | N/A                              |
| Triana                      | WSFL               | Language-based<br>Graph-based | N/A                               | N/A                            | N/A                              |
| UMCORE                      | BPEL               | Language-based                | N/A                               | N/A                            | N/A                              |

of temporal verification which we are most concerned with, only some of the projects such as ASKALON, DAGMan, GridBus, JOpera, Kepler, Taverna and SwinDeW-G clearly stated in their published literatures that temporal constraints are supported in their system QoS control or performance analysis. Yet, to our best knowledge, only SwinDeW-G has set up a series of strategies such as the probabilistic strategy for temporal constraint management [29] and the efficient checkpoint selection strategy to support dynamic temporal verification [28]. In summary, although temporal QoS has been recognized as an important aspect in scientific workflow systems, the work in this area, e.g. the specification of temporal constraints and the support of temporal verification, is still in its infancy [10].

## 9. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a probabilistic strategy for temporal constraint management in scientific workflow systems. A novel probability-based temporal consistency model which is defined by the weighted joint distribution of activity durations has been provided to support statistical analysis for temporal constraint management. Meanwhile, the weighted joint distribution of four Stochastic Petri Nets-based basic building blocks, i.e. sequence, iteration, parallelism and choice, has been presented to facilitate the efficient calculation of the weighted joint distribution of specific workflows and workflow segments by their compositions. Our temporal constraint management consists of a negotiation-based probabilistic strategy for setting temporal constraints at build-time and a probabilistic strategy for updating temporal constraints at run-time. The setting strategy aims to achieve a set of coarse-grained and fine-grained temporal constraints which are well balanced between the user requirements and system performance. With the probability-based temporal consistency, well-balanced overall coarse-grained temporal constraints can be obtained through either a time-oriented or probability-oriented negotiation process. Thereafter, fine-grained temporal constraints for each activity can be propagated instantly in an automatic fashion. The updating strategy is designed to effectively propagate occurring time deficit and time redundancy to update run-time fine-grained temporal constraints according to run-time activity durations. For such a purpose, the probability time deficit and the probability time redundancy are defined in this paper to effectively estimate the occurring time deficit and time redundancy along scientific workflow execution. Afterwards, the probability time deficit/redundancy propagation process is conducted automatically to update the run-time fine-grained temporal constraints.

A weather forecast scientific workflow has been first employed as a motivating example and then revisited as a case study to evaluate the effectiveness of our strategy. The evaluation results have shown that our strategy is capable of setting a set of coarse-grained and fine-grained temporal constraints. Meanwhile, the sum of weighted fine-grained temporal constraints is approximately the same as their coarse-grained temporal constraint, namely the coarse-grained and fine-grained temporal constraints are consistent with the overall workflow execution time. The evaluation results have also shown that our strategy can update the run-time fine-grained constraints effectively so that the occurring probability time deficit can be compensated for and the occurring time redundancy can be reasonably propagated among subsequent activities. The system implementation of our constraint management strategy has been demonstrated with the components in our SwinDeW-G scientific workflow Systems. To the best of our knowledge, this is the first work that has systematically analyzed and addressed the issue of temporal constraint management (including setting build-time coarse-grained and fine-grained temporal constraints, and updating run-time fine-grained temporal constraints) in scientific workflow systems.

In this paper, to simplify the statistical analysis without losing generality, the normal distribution model has been used to model the workflow activity durations. In the future, other representative probability distribution models such as uniform exponential will also be employed to represent the performance of different underlying services. Furthermore, our strategy will be investigated and modified accordingly to accommodate those more complex scenarios.

## ACKNOWLEDGEMENTS

This work is partially supported by the Australian Research Council under Linkage Grant LP0990393, the National High Technology Research and Development 863 Program of China under Grant No. 2007AA04Z116.

## REFERENCES

1. Deelman E, Gannon D, Shields M, Taylor I. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems* 2008; **25**(6):528–540.
2. Taylor IJ, Deelman E, Gannon DB, Shields M. *Workflows for e-science: Scientific Workflows for Grids*. Springer: Berlin, 2007.
3. Yu J, Buyya R. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Records* 2005; **34**(3):44–49.
4. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 2009; **25**(6):599–616.
5. Foster I, Kesselman C. *The Grid: Blueprint for a New Computing Infrastructure* (2nd edn ). Morgan Kaufmann: San Francisco, CA, 2004.
6. Martinez A, Alfaro FJ, Sanchez JL, Quiles FJ, Duato J. A new cost-effective technique for QoS support in clusters. *IEEE Transactions on Parallel and Distributed Systems* 2007; **18**(12):1714–1726.
7. Miller M. Cloud Computing: Web-based applications that change the way you work and collaborate online, Que, August, 2008.
8. Yang Y, Liu K, Chen J, Lignier J, Jin H. Peer-to-peer based grid workflow runtime environment of SwinDeW-G. *Proceedings of 3rd International Conference on e-Science and Grid Computing (e-Science07)*, Bangalore, India, December, 2007; 51–58.
9. Yu J, Buyya R. Workflow scheduling algorithms for grid computing. Computing and distributed systems laboratory, The University of Melbourne, Australia, May 31, 2007.
10. Chen J, Yang Y. A taxonomy of grid workflow verification and validation. *Concurrency and Computation: Practice and Experience* 2008; **20**(4):347–360.
11. Chen J, Yang Y. Adaptive selection of necessary and sufficient checkpoints for dynamic verification of temporal constraints in grid workflow systems. *ACM Transactions on Autonomous and Adaptive Systems* 2007; **2**(2): article 6.
12. Eder J, Panagos E, Rabinovich M. Time constraints in workflow systems. *The 11th International Conference on Advanced Information Systems Engineering (CAiSE99)*, Heidelberg, Germany, 1999; 286–300.
13. Zhuge H, Cheung T, Pung H. A timed workflow process model. *Journal of Systems and Software* 2001; **55**(3):231–243.
14. Aalst WMPvd, Hee KMV. *Workflow Management: Models, Methods, and Systems*. The MIT Press: Cambridge, 2002.
15. Kao B, Garcia-Molina H. Deadline assignment in a distributed soft real-time system. *IEEE Transactions on Parallel Distribution Systems* 1997; **8**(12):1268–1274.
16. Chang CK, Jiang H, Di Y. Time-line based model for software project scheduling with genetic algorithms. *Information and Software Technology* 2008; **50**:1142–1154.
17. Jørgensen M, Shepperd M. Systematic review of software development cost estimation studies. *IEEE Transaction on Software Engineering* 2007; **33**(1):33–53.
18. Liu X, Yang Y, Chen J, Wang Q, Li M. Achieving on-time delivery: A two-stage probabilistic scheduling strategy for software projects. *Proceedings of International Conference on Software Process: Trustworthy Software Development Processes*, Vancouver, B.C., Canada, 2009.
19. Chen J, Yang Y. Activity completion duration based checkpoint selection for dynamic verification of temporal constraints in grid workflow systems. *International Journal of High Performance Computing Applications* 2008; **22**(3):319–329.
20. Cooper K, Dasgupta A, Kennedy K, Koelbel C, Mandal A. New grid scheduling and rescheduling methods in the GrADS project. *Proceedings of 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, New Mexico, April, 2004; 199–206.
21. Russell N, Aalst VD, Hofstede AHMt. Exception handling patterns in process-aware information systems, BPMcenter.org2006.
22. Wang M, Kotagiri R, Chen J. Trust-based robust scheduling and runtime adaptation of scientific workflow. *Concurrency and Computation: Practice and Experience* 2009; **21**(16):1982–1998.
23. Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming* 2006; **14**(3-4):217–230.
24. Wei-Neng C, Jun Z. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 2009; **39**(1):29–43.
25. Wieczorek M, Prodan R, Hoheisel A. Taxonomies of the multi-criteria gridworkflow scheduling problem. *CoreGRID Technical Report Number TR-0106*, August 30, 2007.

26. Chen J, Yang Y. Multiple states based temporal consistency for dynamic verification of fixed-time constraints in grid workflow systems. *Concurrency and Computation: Practice and Experience* 2007; **19**(7):965–982.
27. Chen J, Yang Y. Temporal dependency based checkpoint selection for dynamic verification of fixed-time constraints in grid workflow systems. *Proceedings of 30th International Conference on Software Engineering (ICSE2008)*, Leipzig, Germany, 2008; 141–150.
28. Chen J, Yang Y. Temporal dependency based checkpoint selection for dynamic verification of temporal constraints in scientific workflow systems. *ACM Transactions on Software Engineering and Methodology*, to appear. Available at: <http://www.swinflow.org/papers/TOSEM.pdf> [1 March 2011].
29. Liu X, Chen J, Yang Y. A Probabilistic strategy for setting temporal constraints in scientific workflows. *Proceedings of 6th International Conference on Business Process Management (BPM2008)*, Milan, Italy, 2008; 180–195.
30. Yu J, Buyya R. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing* 2005; **3**:171–200.
31. Prodan R, Fahringer T. Overhead analysis of scientific workflows in grid environments. *IEEE Transactions on Parallel and Distributed Systems* 2008; **19**(3):378–393.
32. Aalst WMPvd, Hee KVM, Reijers HA. Analysis of discrete-time stochastic petri nets. *Statistica Neerlandica* 2000; **54**:237–255.
33. Bucci G, Sassoli L, Vicario E. Correctness verification and performance analysis of real-time systems using stochastic preemptive time petri nets. *IEEE Transactions on Software Engineering* 2005; **31**(11):913–927.
34. Center NMD. China meteorological data sharing service system. Available at: <http://cdc.cma.gov.cn/index.jsp> [1 March 2011].
35. Liu X, Chen J, Liu K, Yang Y. Forecasting duration intervals of scientific workflow activities based on time-series patterns. *Proceedings of 4th IEEE International Conference on e-Science (e-Science08)*, IN, U.S.A., December. 2008; 23–30.
36. Stroud KA. *Engineering Mathematics* (6th edn ). Palgrave Macmillan: New York, 2007.
37. Law AM, Kelton WD. *Simulation Modelling and Analysis* (4th edn). McGraw-Hill: New York, 2007.
38. Nadarajah S, Kotz S. Exact distribution of the max/min of two Gaussian random variables. *IEEE Transactions on Very Large Scale Integration Systems* 2008; **16**(2):210–212.
39. Son JH, Sun Kim J, Ho Kim M. Extracting the workflow critical path from the extended well-formed workflow schema. *Journal of Computer and System Sciences* 2005; **70**(1):86–106.
40. Russell N, Aalst WMPvd, Hofstede AHM. Workflow Exception Patterns. *Proceedings of 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, Berlin, Germany, 2006; 288–302.
41. Yu Z, Shi W. An adaptive rescheduling strategy for grid workflow applications. *Proceedings of 2007 IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2007)*, Long Beach, CA, U.S.A., March 2007; 115–122.
42. Hwang SY, Wang H, Tang J, Srivastava J. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Information Sciences* 2007; **177**(23):5484–5503.
43. Liangzhao Z, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering* 2004; **30**(5):311–327.
44. SAP Library. *Workflow System Administration*. Available at: [http://help.sap.com/saphelp\\_nw2004s/helpdata/en](http://help.sap.com/saphelp_nw2004s/helpdata/en) [1 March 2011].
45. ASKALON Project. Available at: <http://www.dps.uibk.ac.at/projects/askalon> [1 March 2011].
46. CROWN Project, *CROWN Portal*. Available at: <http://www.springerlink.com/content/w1m24586434640t6/> [1 March 2011].
47. DAGMan. *Condor Project*. Available at: <http://www.cs.wisc.edu/condor/> [1 March 2011].
48. GridBus Project. Available at: <http://www.gridbus.org> [1 March 2011].
49. JOpera Project. Available at: <http://www.iks.ethz.ch/jopera> [1 March 2011].
50. Kepler Project. Available at: <http://kepler-project.org/> [1 March 2011].
51. Taverna Project. Available at: <http://www.mygrid.org.uk/tools/taverna/> [1 March 2011].
52. Triana Project. Available at: <http://www.trianacode.org/> [1 March 2011].
53. UNICORE Project. Available at: <http://www.unicore.eu/> [1 March 2011].