An Activity Completion Duration based Checkpoint Selection Strategy for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems

Jinjun Chen, Yun Yang

CICEC – Centre for Internet Computing and E-Commerce
Faculty of Information and Communication Technologies
Swinburne University of Technology
PO Box 218, Hawthorn, Melbourne, Australia 3122
jchen@ict.swin.edu.au
yyang@ict.swin.edu.au

Abstract: In grid workflow systems, to verify fixed-time constraints efficiently at the run-time execution stage, some checkpoints are often selected so that we only need to conduct the fixed-time constraint verification at such checkpoints rather than at all activity points. However, the existing typical checkpoint selection strategies are inefficient and/or ineffective because they may incur some unnecessary verification by selecting some unnecessary checkpoints or omit some necessary verification by omitting some necessary checkpoints. Therefore, in this paper, based on the run-time activity completion duration, we develop a new checkpoint selection strategy that is more efficient and effective than the existing typical checkpoint selection strategies. The final comparison and quantitative evaluation further demonstrate this point.

1 Introduction

Grid workflow systems, which are evoking a high degree of interest aim to support modelling, redesign and execution of large-scale sophisticated e-science and e-business processes using the grid computing approach to enable the coordinated use of numerous distributed and heterogeneous resources [Ab04, Ca03, Cy04, Fo02]. In Open Grid Services Architecture (OGSA), a grid workflow can be defined as the automation of a Grid process, in whole or part, during which documents, information or data are passed from one grid service to another for action, according to a set of procedural rules [Am04, Cy04, CY05, Hu03]. Conceptually, a grid workflow is a collection of activities, and the dependencies between activities that define their execution orders and form four basic control structures: sequential, parallel, selective and iterative [Cy04]. These activities are implemented and executed by corresponding grid services. The whole working process of a grid workflow system can be divided into three stages: build-time, run-time instantiation and run-time execution. At the build-time stage, grid workflow specifications are defined by deploying some grid workflow definition languages such as

Grid Services Flow Language (GSFL), Abstract Grid Workflow Language (AGWL), Service Workflow Language (SWFL) and Grid Workflow Execution Language (GWEL) [Cy04, FPV04, Hu03, KWL02]. At the run-time instantiation stage, grid workflow instances are created, and especially grid services specified in the build-time definition documents are discovered by an instantiation service that is a high-level grid service [Cy04, KWL02]. At the run-time execution stage, the grid workflow instances are executed, and the execution is coordinated between grid services by the grid workflow engine that itself is a high-level grid service, hence automatically grid aware [Cy04, Hu03, KWL02].

To control the temporal correctness of the grid workflow specification and execution, fixed-time constraints are often set [CY04, EPR99, MO99]. A fixed-time constraint at an activity is an absolute time value by which the activity must be completed. For example, a climate modelling grid workflow must be finished by the scheduled time [Ab04], say 9:00pm, so that the weather forecasting can be broadcasted at a later time, say 10:30pm, on the same day. Here, 9pm is a fixed-time constraint.

After the fixed-time constraints are set, the temporal verification is conducted to check whether they are all consistent. At the build-time and run-time instantiation stages, the temporal verification is static because of no any specific execution times. For each fixed-time constraint, we conduct its verification once only with the consideration of all covered activities. Therefore, we need not decide at which activities we should conduct the fixed-time constraint verification. At the run-time execution stage, the activity completion duration is uncertain, which may affect the consistency of a fixed-time constraint. Hence, we may need to verify a fixed-time constraint many times at different activities. However, conducting the verification at every activity is not efficient because we may not have to do so at some activities such as those activities which can be completed within the allowed time interval. Therefore, a question is prompted which is: "where should we conduct the fixed-time constraint verification?". The activities at which we conduct the fixed-time constraint verification are called *checkpoints* [CYC04, De03, MO99, ZCP01]. Correspondingly, a research field comes into the picture whose topic is checkpoint selection strategies [CYC04, De03, MO99, ZCP01].

So far, some typical checkpoint selection strategies have been proposed. [De03] takes every activity as a checkpoint. We denote this strategy as CSS_1 (*CSS*: Checkpoint Selection Strategy). [ZCP01] sets checkpoints at the start time and end time of each activity and each flow. We denote this strategy as CSS_2. [MO99] takes the start point of a workflow instance as a checkpoint and adds a checkpoint after each decision activity is executed. We denote this strategy as CSS_3. [MO99] also mentions another checkpoint selection strategy: user-defined checkpoints. We denote this strategy as CSS_4. [CYC04], as our previous work, proposes a checkpoint selection strategy based on the run-time activity completion duration. We denote this strategy as CSS_5. However, since the activity completion duration is uncertain, we may not have to conduct the fixed-time constraint verification at some activities. Therefore, CSS_1 and CSS_2 are inefficient. Similarly, CSS_3 and CSS_4 are inefficient either as we may not have to conduct the fixed-time constraint verification at the start activity or the decision activities or the user-defined activities. In addition, CSS_3 and CSS_4 are ineffective because we will omit the verification which should be conducted at some other activities. As to CSS_5, in [CYC04], a fixed-time constraint only has two states: CC (Conventional Consistency) and CI (Conventional Inconsistency) and CSS_5 is based on it. However, according to the analysis in [CY05b], in the grid workflow systems, a fixed-time constraint should have four states: SC (Strong Consistency), WC (Weak Consistency), WI (Weak Inconsistency) and SI (Strong Inconsistency). CC corresponds to SC and CI is divided into WC, WI and SI. Based on CSS_5, we can judge whether we should take an activity as a checkpoint for CC or CI verification. However, since CSS_5 is only based on CC and CI, we are not able to further consider whether we should take the activity as a checkpoint for the verification of WC, WI or SI. As a result, the corresponding verification is omitted. Hence, CSS_5 is ineffective either.

Regarding the above limitations of the existing typical checkpoint selection strategies, in this paper, we develop a new checkpoint selection strategy. The strategy is based on the relationship between the run-time activity completion duration and the four states: SC, WC, WI and SI, and can select the checkpoints dynamically along the grid workflow execution. The final comparison and quantitative evaluation further show that our strategy is more efficient and effective than the existing typical ones.

The remainder of the paper is organised as follows. Section 2 describes a timed grid workflow representation. Section 3 details our checkpoint selection strategy. Section 4 further shows the benefits of our strategy through a comparison and quantitative evaluation. Section 5 concludes our contributions and points out the future work.

2 Timed Grid Workflow Representation

Based on the directed graph concept, a grid workflow can be represented by a grid workflow graph, where nodes correspond to activities and edges correspond to dependencies between activities, called flows [CM00, EPR99]. Here we assume that the grid workflow is well structured. As far as the time is concerned, an activity is similar to a flow. Therefore, we use term "activity" to refer to the real activity as well as the flow. Based on [CY04, CY05a, EPR99, MO99], we denote the i^{th} activity of a grid workflow as a_i, the expected time from which the specification of the grid workflow *gw* will become effective as $Cie(gw)$. For each a_i, we denote its maximum duration, minimum duration, average duration, run-time start time, run-time end time and run-time completion duration as $D(a_i)$, $d(a_i)$, $Ave(a_i)$, $S(a_i)$, $E(a_i)$ and $Rcd(a_i)$ respectively. $Rcd(a_i)$ covers the queuing delay, synchronisation delay, network latency and so on caused at a_i. If there is a fixed-time constraint at a_i, we denote it as $FTC(a_i)$ and its value as $ftv(a_i)$. If there is a path from a_i to a_j ($j \geq i$), we denote the maximum duration, minimum duration, average duration, run-time real completion duration between them as $D(a_i, a_j)$, $d(a_i, a_j)$, $Ave(a_i, a_j)$ and $Rcd(a_i, a_j)$ respectively [EPR99, MO99]. Normally we have $d(a_i) \leq Ave(a_i) \leq D(a_i)$ and $d(a_i, a_j) \leq Ave(a_i, a_j) \leq D(a_i, a_j)$. For convenience, we consider one execution path in the grid workflow without losing generality. As to a selective or parallel structure, for each branch, it is an execution path. For an iterative structure, from the start time to the end time, it is still an execution path. Therefore, for the selective/parallel/iterative structure, we can also apply the results achieved from one execution path. Hence, from a_i to a_j, $D(a_i, a_j)$, $d(a_i, a_j)$ and $Ave(a_i, a_j)$ is equal to the sum of activity maximum durations, minimum durations and average durations respectively.

Besides the above time attributes, in [CY05b], we have defined the four consistency states: SC, WC, WI and SI. Because we will develop a new checkpoint selection strategy based on them, we should present a summary of their definitions. In addition, for the comparison between CSS_5 and our strategy in Section 4, we also need to summarise the definitions of CC and CI. However, since the checkpoint concept is related to the run-time execution stage, we only summarise the definitions of the stage below. The definitions of other stages and detailed discussion can be referred to [CYC04, CY05b].

Definition 1. At the run-time execution stage, at checkpoint a_p which is either before or at a_i, $FTC(a_i)$ is said to be of SC when $Rcd(a_1, a_p)+D(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1)$.

Definition 2. At the run-time execution stage, at checkpoint a_p which is either before or at a_i, $FTC(a_i)$ is said to be of WC when $Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_i)$.

Definition 3. At the run-time execution stage, at checkpoint a_p which is either before or at a_i, $FTC(a_i)$ is said to be of WI when $Rcd(a_1, a_p) + d(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$,

Definition 4. At the run-time execution stage, at checkpoint a_p which is either before or at a_i ($p \leq i$), $FTC(a_i)$ is said to be of SI when $ftv(a_i)-S(a_1)<Rcd(a_1, a_p)+d(a_{p+1}, a_i)$.

Definition 5. At the run-time execution stage, at checkpoint a_p which is either before or at a_i, $FTC(a_i)$ is said to be of CC when $Rcd(a_1, a_p)+D(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1)$.

Definition 6. At the run-time execution stage, at checkpoint a_p which is either before or at a_i, $FTC(a_i)$ is said to be of CI when $ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+ D(a_{p+1}, a_i)$

For clarity, we further depict SC, WC, WI, SI, CC and CI in Figure 1.

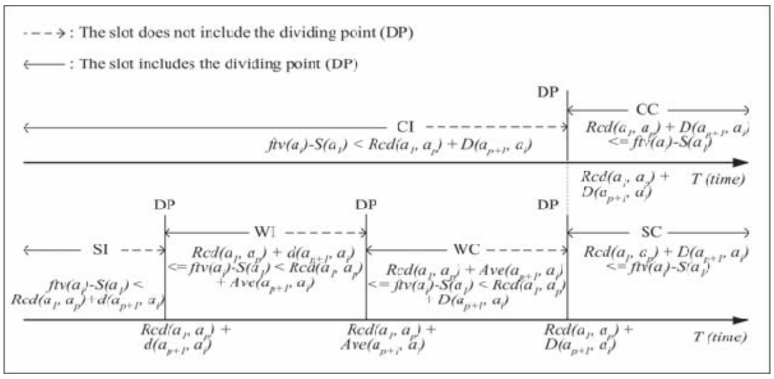

- - -→ : The slot does not include the dividing point (DP)
←—— : The slot includes the dividing point (DP)
DP
CC
CI
$ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + D(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + D(a_{p+1}, a_i) <= ftv(a_i)-S(a_1)$
$Rcd(a_1, a_p) + D(a_{p+1}, a_i)$
T (time)
SI
WI
WC
SC
$ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + d(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + d(a_{p+1}, a_i) <= ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) <= ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + D(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + D(a_{p+1}, a_i) <= ftv(a_i)-S(a_1)$
$Rcd(a_1, a_p) + d(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$
$Rcd(a_1, a_p) + D(a_{p+1}, a_i)$

Figure 1: Definitions of SC, WC, WI and SI vs definitions of CC and CI at run-time execution stage

3 A New Checkpoint Selection Strategy

According to [CY05b, HA00], WI and SI will be adjusted to SC or WC by the corresponding exception handling. Therefore, along the grid workflow execution, at an activity point, before the execution of the activity, all fixed-time constraints will be of either SC or WC. However, after the execution of the activity, the uncertain activity completion duration may affect their consistency. Therefore, we now firstly discuss the relationship between the activity completion duration and SC, WC, WI & SI. Then, we present our checkpoint selection strategy based on the relationship.

3.1 Relationship between Activity Completion Duration and SC, WC, WI & SI

At the run-time execution stage, at activity point a_p, its completion duration is $Rcd(a_p)$. We discuss its relationship with SC, WC, WI and SI by deriving some theorems that will form the basis for presenting our checkpoint selection strategy.

Theorem 1. At activity point a_p, if $D(a_p) < Rcd(a_p)$, 1) all previous WC fixed-time constraints cannot be of SC and 2) may be of WC, WI or SI; and 3) previous SC fixed-time constraints may be of SC, WC, WI or SI.

Proof: 1) Suppose $FTC(a_m)$ is of WC before the execution of a_p ($p \leq m$), according to Definition 2, we have $ftv(a_m)-S(a_1) < Rcd(a_1, a_{p-1})+D(a_p, a_m)$. If $D(a_p) < Rcd(a_p)$, then, we have: $ftv(a_m)-S(a_1) < Rcd(a_1, a_{p-1})+D(a_p, a_m) = Rcd(a_1, a_{p-1})+ D(a_p)+D(a_{p+1}, a_m) < Rcd(a_1, a_{p-1})+ Rcd(a_p)+D(a_{p+1}, a_m) = Rcd(a_1, a_p)+D(a_{p+1}, a_m)$. Hence, we have:

$$ftv(a_m)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_m) \tag{1}$$

However, for $FTC(a_m)$ to be of SC, we must ensure (2) below holds.

$$Rcd(a_1, a_p)+D(a_{p+1}, a_m) \leq ftv(a_m)-S(a_1) \tag{2}$$

Obviously, (2) does not hold. Hence, $FTC(a_m)$ cannot be of SC after a_p's execution.

2) From $D(a_p) < Rcd(a_p)$, we have: $Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) = Rcd(a_1, a_{p-1}) + Ave(a_p) + Ave(a_{p+1}, a_m) \leq Rcd(a_1, a_{p-1}) + D(a_p) + Ave(a_{p+1}, a_m) < Rcd(a_1, a_{p-1}) + Rcd(a_p) + Ave(a_{p+1}, a_m) = Rcd(a_1, a_p) + Ave(a_{p+1}, a_m)$. Hence, we have:

$$Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) \tag{3}$$

Meanwhile, because $FTC(a_m)$ is previously of WC, we have:

$$Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) \leq ftv(a_m)-S(a_1) \tag{4}$$

However, from (3) and (4), we can not judge whether (5) below holds.

$$Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) \leq ftv(a_m)-S(a_1) \tag{5}$$

If (5) holds, $FTC(a_m)$ is of WC again. However, depending on specific $Rcd(a_p)$, (5) may or may not hold. Similarly, we may or may not have $Rcd(a_1, a_p) + d(a_{p+1}, a_m) \leq ftv(a_m)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_m)$, and we also may or may not have $ftv(a_m)-S(a_1) < Rcd(a_1, a_p)+d(a_{p+1}, a_m)$. Therefore, according to Definitions 2, 3 and 4, depending on specific $Rcd(a_p)$, $FTC(a_m)$ may be of WC, WI or SI.

3) Suppose $FTC(a_n)$ is of SC before the execution of a_p ($p \leq n$), according to Definition 1, we have:

$$Rcd(a_1, a_{p-1})+D(a_p, a_n)\leq ftv(a_n)-S(a_1) \quad (6)$$

From $D(a_p) < Rcd(a_p)$, we have: $Rcd(a_1, a_{p-1}) + D(a_p, a_n) = Rcd(a_1, a_{p-1})+ D(a_p) + D(a_{p+1}, a_n) < Rcd(a_1, a_{p-1})+ Rcd(a_p) + D(a_{p+1}, a_n) = Rcd(a_1, a_p) + D(a_{p+1}, a_n)$. Hence, we have:

$$Rcd(a_1, a_{p-1}) + D(a_p, a_n) < Rcd(a_1, a_p) + D(a_{p+1}, a_n) \quad (7)$$

However, from (6) and (7), we can not judge whether (8) below holds.

$$Rcd(a_1, a_p) + D(a_{p+1}, a_n) \leq ftv(a_n)-S(a_1) \quad (8)$$

If (8) holds, $FTC(a_n)$ is of SC again. However, depending on specific $Rcd(a_p)$, (8) may or may not hold. Similarly, we may or may not have $Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_i)$ or $Rcd(a_1, a_p) + d(a_{p+1}, a_i) \leq ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$ or $ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+d(a_{p+1}, a_i)$. Therefore, according to Definitions 1, 2, 3 and 4, depending on specific $Rcd(a_p)$, after the execution of a_p, $FTC(a_n)$ may be of SC, WC, WI or SI.

In overall terms, the theorem holds. ▌

Theorem 2. At activity point a_p, if $Rcd(a_p) \leq D(a_p)$, 1) all previous SC fixed-time constraints are still of SC; and 2) previous WC fixed-time constraints may be of SC, WC, WI or SI.

Proof: 1) Suppose $FTC(a_n)$ is of SC before the execution of a_p ($p\leq n$), according to Definition 1, we have $Rcd(a_1, a_{p-1}) + D(a_p, a_n)\leq ftv(a_n)-S(a_1)$. If $Rcd(a_p) \leq D(a_p)$, then we have: $Rcd(a_1, a_p)+D(a_{p+1}, a_n)= Rcd(a_1, a_{p-1})+Rcd(a_p)+D(a_{p+1}, a_n) \leq Rcd(a_1, a_{p-1}) + D(a_p)+D(a_{p+1}, a_n) = Rcd(a_1, a_{p-1})+D(a_p, a_n) \leq ftv(a_n)-S(a_1)$. Hence, we have:

$$Rcd(a_1, a_p)+D(a_{p+1}, a_n) \leq ftv(a_n)-S(a_1) \quad (9)$$

According to Definition 1, $FTC(a_n)$ is still of SC after the execution of a_p.

2) The proof is similar to 3) of Theorem 1 and consequently is omitted.

In overall terms, the theorem holds. ▌

Theorem 3. At activity point a_p, if $Rcd(a_p) \leq Ave(a_p)$, 1) all previous SC fixed-time constraints are still of SC; and 2) all previous WC fixed-time constraints are of either WC or SC; and 3) if they are still of WC, the status has been changed for better.

Proof: 1) The proof is similar to 1) of Theorem 2 and consequently is omitted.

2) Suppose $FTC(a_m)$ is of WC before the execution of a_p ($p\leq m$), according to Definition 2, we have:

$$Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) \leq ftv(a_m)-S(a_1) < Rcd(a_1, a_{p-1})+D(a_p, a_m) \quad (10)$$

If $Rcd(a_p) \leq Ave(a_p)$, then we have: $Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) = Rcd(a_1, a_{p-1}) + Rcd(a_p) + Ave(a_{p+1}, a_m) \leq Rcd(a_1, a_{p-1}) + Ave(a_p) + Ave(a_{p+1}, a_m) = Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) \leq ftv(a_m)-S(a_1)$. Hence, we have:

$$Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) \leq ftv(a_m)-S(a_1) \quad (11)$$

In addition, we also have: $Rcd(a_1, a_p)+D(a_{p+1}, a_m) = Rcd(a_1, a_{p-1})+Rcd(a_p)+D(a_p, a_m) \leq Rcd(a_1, a_{p-1})+ Ave(a_p)+D(a_p, a_m) \leq Rcd(a_1, a_{p-1})+D(a_p)+D(a_p, a_m) = Rcd(a_1, a_{p-1}) + D(a_p, a_m)$. Hence, we have:

$$Rcd(a_1, a_p)+D(a_{p+1}, a_m) \le Rcd(a_1, a_{p-1})+D(a_p, a_m) \tag{12}$$

However, from (10), (11) and (12) that we only have, we cannot judge whether (13) or (14) holds.

$$ftv(a_m)\text{-}S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_m) \tag{13}$$

$$Rcd(a_1, a_p)+D(a_{p+1}, a_m) \le ftv(a_m)\text{-}S(a_1) \tag{14}$$

In fact, depending on how much $Rcd(a_p)$ is less than $Ave(a_p)$, (13) or (14) may or may not hold. If (13) holds, then, with (11), we have:

$$Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) \le ftv(a_m)\text{-}S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_m) \tag{15}$$

According to Definition 2, (15) means that $FTC(a_m)$ is of WC. If (14) holds, according to Definition 1, $FTC(a_m)$ is already switched to be of SC after the execution of a_p.

3) If $Rcd(a_p) \le Ave(a_p)$, then we have: $Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) = Rcd(a_1, a_{p-1}) + Rcd(a_p) + Ave(a_{p+1}, a_m) \le Rcd(a_1, a_{p-1}) + Ave(a_p) + Ave(a_{p+1}, a_m) = Rcd(a_1, a_{p-1}) + Ave(a_p, a_m)$. Therefore, we have:

$$Rcd(a_1, a_p) + Ave(a_{p+1}, a_m) \le Rcd(a_1, a_{p-1}) + Ave(a_p, a_m) \tag{16}$$

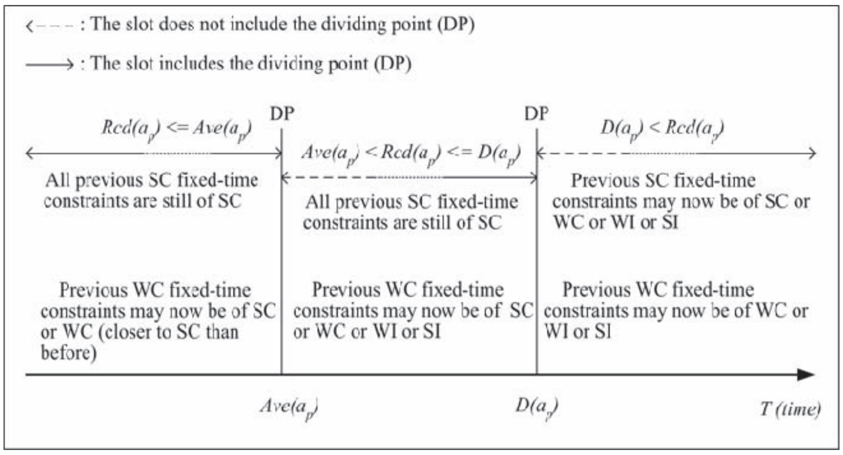
Correspondingly, we have:

$$[ftv(a_m)\text{-}S(a_1)]-[Rcd(a_1, a_{p-1})+Ave(a_p, a_m)] \le [ftv(a_m)\text{-}S(a_1)]-[Rcd(a_1, a_p)+Ave(a_{p+1}, a_m)] \tag{17}$$

(17) means that, after the execution of a_p, $FTC(a_m)$ is closer to SC than before. Therefore, the status of $FTC(a_m)$ has been changed for better.

In overall terms, the theorem holds. ▌

Based on Theorems 1, 2 and 3, we can derive the detailed relationship between the completion duration of a_p and SC, WC, WI & SI. For clarity, we depict it in Figure 2.



<--- : The slot does not include the dividing point (DP)
——> : The slot includes the dividing point (DP)
DP
DP
Rcd(a_p) <= Ave(a_p)
Ave(a_p) < Rcd(a_p) <= D(a_p)
D(a_p) < Rcd(a_p)
All previous SC fixed-time constraints are still of SC
All previous SC fixed-time constraints are still of SC
Previous SC fixed-time constraints may now be of SC or WC or WI or SI
Previous WC fixed-time constraints may now be of SC or WC (closer to SC than before)
Previous WC fixed-time constraints may now be of SC or WC or WI or SI
Previous WC fixed-time constraints may now be of WC or WI or SI
Ave(a_p)
D(a_p)
T (time)

Figure 2: Relationship between completion duration of a_p and SC, WC, WI & SI

3.2 Checkpoint Selection

According to the discussion in Section 3.1, we can draw the following conclusions:

- If $D(a_p) < Rcd(a_p)$, we have to verify all previous SC and WC fixed-time constraints. However, for previous WC fixed-time constraints, we need not verify whether they are of SC.

- If $Rcd(a_p) \leq D(a_p)$, we need not verify all previous SC fixed-time constraints. But we have to verify all previous WC fixed-time constraints.

- If $Rcd(a_p) \leq Ave(a_p)$, we need not verify all previous SC fixed-time constraints. As to previous WC fixed-time constraints, we borrow some conclusions from [CY05b] and we need not verify them either. In [CY05b], we already developed a method to adjust the WC fixed-time constraints to be of SC. According to Theorem 3, after the execution of a_p, the status of the previous WC fixed-time constraints is changed for better (even can be changed to SC). Therefore, we can still use the previous adjustment. Hence, we need not verify it.

Based on the above conclusions, we can decide whether or not we should take a_p as a checkpoint and correspondingly we derive a new checkpoint selection strategy. For convenience of the discussion, we denote it as CSS_{ACD} (Activity Completion Duration based Checkpoint Selection Strategy).

CSS_{ACD} is: *At activity a_p, if $D(a_p) < Rcd(a_p)$, we take it as a checkpoint for the verification of SC, WC, WI & SI of all previous SC fixed-time constraints, and for the verification of WC, WI & SI of all previous WC fixed-time constraints. If $Rcd(a_p) \leq D(a_p)$, we take a_p as a checkpoint for the verification of SC, WC, WI & SI of all previous WC fixed-time constraints. If $Rcd(a_p) \leq Ave(a_p)$, we do not take a_p as a checkpoint.*

According to CSS_{ACD}, we can further derive Algorithm 1 that combines the checkpoint selection process of CSS_{ACD} and the fixed-time constraint verification. Algorithm 1 is depicted on next page due to the space limit of this page.

By applying Algorithm 1, we can base the fixed-time constraint verification on CSS_{ACD}. As a result, we only need to conduct the fixed-time constraint verification at necessary activity points. And at such points, we only need to verify those fixed-time constraints that we should verify.

```
symbol Definitions:
ArraySC: an array of all previous SC fixed-time constraints that cover $a_p$;
ArrayWC: an array of all previous WC fixed-time constraints that cover $a_p$;
end Definitions
Input: ArraySC, ArrayWC, $S(a_1)$, $Rcd(a_p)$, $Ave(a_p)$, maximum, minimum and average durations of
        all activities involved in ArraySC and ArrayWC;
Output: checkpoint report and SC, WC, WI & SI report;
If ($D(a_p) < Rcd(a_p)$) then
    Output '$a_p$ is a checkpoint for the verification of SC, WC, WI and SI of all previous SC fixed-
            time constraints, and for the verification of WC, WI and SI of all previous WC fixed-
            time constraints'
while (not end of ArraySC) do
  // verify SC, WC, WI and SI of previous SC fixed-time constraints
        Select current fixed-time constraint from ArraySC, say $FTC(a_i)$ $(p \le i)$;
        if ($Rcd(a_1, a_p)+D(a_{p+1}, a_i) \le ftv(a_i)-S(a_1)$) then
          Output '$FTC(a_i)$ is of SC'
    else if ($Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WC' ;
    else if ($Rcd(a_1, a_p) + d(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WI' ;
    else if ($ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+d(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of SI' ;
    end if
end while
while (not end of ArrayWC) do //verify WC, WI and SI of previous WC fixed-time constraints
        Select current fixed-time constraint from ArrayWC, say $FTC(a_i)$ $(p \le i)$;
    if ($Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WC' ;
    else if ($Rcd(a_1, a_p) + d(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WI' ;
    else if ($ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+d(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of SI' ;
    end if
end while
else if ($Ave(a_p) < Rcd(a_p) \le D(a_p)$) then
    Output '$a_p$ is a checkpoint for the verification of SC, WC, WI and SI of all previous WC fixed-
            time constraints';
while (not end of ArrayWC) do
  // verify SC, WC, WI and SI of previous WC fixed-time constraints
        Select current fixed-time constraint from ArrayWC, say $FTC(a_i)$ $(p \le i)$;
        if ($Rcd(a_1, a_p)+D(a_{p+1}, a_i) \le ftv(a_i)-S(a_1)$) then
          Output '$FTC(a_i)$ is of SC';
    else if ($Rcd(a_1, a_p) + Ave(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+D(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WC' ;
    else if ($Rcd(a_1, a_p) + d(a_{p+1}, a_i) \le ftv(a_i)-S(a_1) < Rcd(a_1, a_p) + Ave(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of WI' ;
    else if ($ftv(a_i)-S(a_1) < Rcd(a_1, a_p)+d(a_{p+1}, a_i)$) then
          Output '$FTC(a_i)$ is of SI' ;
    end if
end while
else // $Rcd(a_p) < Ave(a_p)$
    Output '$a_p$ is NOT a checkpoint and there is no need of any fixed-time constraint verification';
end if
```

Algorithm 1: Checkpoint selection and fixed-time constraint verification based on CSS_{ACD}

4 Comparison and Quantitative Evaluation

In this section, we will evaluate our checkpoint selection strategy CSS_{ACD} by comparing it with other strategies: CSS_1, CSS_2, CSS_3, CSS_4 and CSS_5 which are stated in Section 1. Since CSS_1 is similar to CSS_2 as they both set checkpoints at every activity. And CSS_3 is similar to CSS_4 as they both define the checkpoints before the grid workflow execution. Therefore, we only analyse CSS_1, CSS_4, CSS_5 and CSS_{ACD}.

According to the discussion in Section 3, to compare CSS_1, CSS_4, CSS_5 and CSS_{ACD}, we should analyse the unnecessary and omitted fixed-time constraint verification based on them respectively. According to the definitions of the fixed-time constraint consistency in Section 2, the primary fixed-time constraint verification computation is focused on the sum of the maximum durations between two activities. Therefore, we take each computation of the maximum duration addition as a verification computation unit. Correspondingly, we analyse CSS_1, CSS_4, CSS_5 and CSS_{ACD} by comparing their unnecessary and omitted verification computation unit numbers. We denote the unnecessary and omitted verification computation unit numbers of CSS_1 as une_{css_1} and omi_{css_1} respectively, CSS_4 as une_{css_4} and omi_{css_4} respectively, CSS_5 as une_{css_5} and omi_{css_5} respectively, CSS_{ACD} as $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ respectively.

We consider a climate modelling grid workflow that may consist of hundreds and thousands of activities and must be time constrained so that the weather forecasting can be broadcasted on time [Ab04]. For simplicity, we focus on one fixed-time constraint in it, say $FTC(A)$. We suppose $FTC(A)$ covers N activities. Since in the real-world grid workflow systems, normally there are many grid workflow instances, we conduct the quantitative analysis in a statistical way. Therefore, we introduce possibility Q for an activity execution not exceeding its average duration ($0 \leq Q \leq 1$). For simplicity, we assume that each activity has the same Q. We now conduct the corresponding comparison between CSS_1 and CSS_{ACD} in Section 4.1, and between CSS_4 and CSS_{ACD} in Section 4.2, and between CSS_5 and CSS_{ACD} in Section 4.3. The comparison between CSS_1, CSS_4 and CSS_5 is beyond the scope of this paper as we are focusing on the development and discussion of CSS_{ACD}.

4.1 CSS_{ACD} vs CSS_1

According to the consistency definitions in Section 2 and the discussion of CSS_{ACD} in Section 3, we can derive the following computing equations:

$$une_{css_1} = Q*N^2, \quad omi_{css_1} = 0; \qquad une_{css_{ACD}} = 0, \quad omi_{css_{ACD}} = 0$$

Suppose $N = 20$, with Q changing, we list corresponding une_{css_1}, omi_{css_1}, $une_{css_{ACD}}$, and $omi_{css_{ACD}}$ in Table 1. The value selection of N does not affect our analysis because what we want is the trend of how une_{css_1}, omi_{css_1}, $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ are changing with Q changing.

Table 1: Comparison of unnecessary and omitted verification between CSS_{ACD} and CSS_1

Q	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
une_{css_1}	0	40	80	120	160	200	240	280	320	360	400
$une_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0
omi_{css_1}	0	0	0	0	0	0	0	0	0	0	0
$omi_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0

From the above computing equations and Table 1, we can see the following facts:

• omi_{css_1} and $omi_{css_{ACD}}$ are always 0. In fact, based on CSS_1, we conduct the fixed-time constraint verification at every activity. Therefore, there is no omitted verification. Based on CSS_{ACD}, we conduct the verification only at any activity whose completion duration is more than its average duration. According to Section 3, we should do so. Therefore, there is no omitted verification either. Therefore $omi_{css_1} = omi_{css_{ACD}}$ =0.

• $une_{css_{ACD}}$ is always 0. une_{css_1} is also 0 if Q=0. In fact, based on CSS_{ACD}, we conduct the verification at any activity whose completion duration is more than its average duration. And according to Section 3, we should do so. Therefore, $une_{css_{ACD}}$ is always 0. Based on CSS_1, we conduct the verification at every activity. If Q=0, the execution of every activity will exceed its average duration. According to Section 3, we should conduct the verification at every activity. Therefore, all verification based on CSS_1 becomes necessary. Hence, une_{css_1} =0.

• With Q increasing, une_{css_1} is increasing. This means that the more activities that can be completed with their average durations, the more amount of unnecessary verification based on CSS_1. However, the amount of the unnecessary verification based on CSS_{ACD} does not change and is always less than that based on CSS_1.

In summary, although CSS_1 does not omit any fixed-time constraint verification, it may cause some unnecessary verification. Especially, the more activities that can be completed within their average durations, the more amount of the unnecessary verification based on CSS_1. The unnecessary verification will eventually affect the overall verification efficiency. Hence, CSS_{ACD} is more efficient than CSS_1.

4.2 CSS_{ACD} vs CSS_4

We suppose there are M checkpoints defined by CSS_4. Then, according to the definitions in Section 2, we can derive the following computing equations for CSS_4:

$$une_{css_4} = Q*M*N \qquad omi_{css_4} = (1-Q)*(N-M)*N$$

We now take some specific values to see how they perform. Suppose N = 20 and M=3. The selection of the values does not affect our analysis because what we want is the trend of how une_{css_4} and omi_{css_4} are changing with Q changing. With Q changing, we list corresponding une_{css_4} and omi_{css_4} in Table 2. For clarity, we also list corresponding $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ in Table 2 although they have been listed in Table 1.

Table 2: Comparison of unnecessary and omitted verification between CSS_{ACD} and CSS_4

Q	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
une_{css_4}	0	6	12	18	24	30	36	42	48	54	60
$une_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0
omi_{css_4}	340	306	272	238	204	170	136	102	68	34	0
$omi_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0

From the above computing equations and Table 2, we can see the following facts:

- As discussed in Section 4.1, $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ are always 0.

- If Q=0, une_{css_4} =0. In fact, based on CSS_4, we conduct the verification at M predefined activities. If Q=0, the execution of every activity will exceed its average duration. Then, according to the discussion in Section 3, we must conduct the verification at every activity. Hence, all verification at M prescribed activities becomes necessary and consequently une_{css_4} =0.

- With Q increasing, une_{css_4} is increasing. This means that the more activities that can be completed within their respective average durations, the more amount of the unnecessary verification based on CSS_4.

- If Q=1, omi_{css_4} =0. In fact, if Q=1, all activities can be completed within their average activities, according to Section 3, we need not conduct any verification. Namely, we will not omit any verification by CSS_4. Hence, omi_{css_4} =0. This is the best case for omi_{css_4} .

- With Q decreasing, omi_{css_4} is increasing. This means that the more activities at which we should conduct the verification, the more omitted verification based on CSS_4.

In summary, on one hand, CSS_4 may cause some unnecessary fixed-time constraint verification. The more activities that can be completed within their respective average durations, the more amount of the unnecessary verification based on CSS_4. However, CSS_{ACD} does not incur any unnecessary verification. Since the unnecessary verification will eventually affect the overall verification efficiency, CSS_{ACD} is more efficient than CSS_4. On the other hand, CSS_4 may omit some necessary verification. The more activities where we should conduct the fixed-time constraint verification, the more omitted verification will be incurred. However, CSS_{ACD} does not omit necessary verification.

Since the omitted verification will eventually affect the overall verification effectiveness, CSS_{ACD} is more effective than CSS_4.

4.3 CSS_{ACD} vs CSS_5

According to the discussion in Section 3 and [CYC04], at a_i, if $D(a_i)<Rcd(a_i)$, then both CSS_{ACD} and CSS_5 take a_i as a checkpoint. Therefore, for the situation where $D(a_i)<Rcd(a_i)$, CSS_{ACD} and CSS_5 are the same in terms of the unnecessary and omitted verification computation. Hence, we need not consider the situation and for simplicity, we can assume that the possibility for the situation be 0. Then, we derive the following computing equations.

$$une_{css_5} = 0 \qquad omi_{css_5} = (1-Q)*N^2$$

We now take some specific values to see how they perform. We suppose $N = 20$. The value selection of N does not affect our analysis because what we want is the trend of how une_{css_5} and omi_{css_5} are changing with Q changing. With Q changing, we list corresponding une_{css_5} and omi_{css_5} in Table 3. For clarity, we also list corresponding $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ in Table 3 although they have been listed in Tables 1 and 2.

Table 3: Comparison of unnecessary and omitted fixed-time constraint verification between CSS_{ACD} and CSS_5

Q	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
une_{css_5}	0	0	0	0	0	0	0	0	0	0	0
$une_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0
omi_{css_5}	400	360	320	280	240	200	160	120	80	40	0
$omi_{css_{ACD}}$	0	0	0	0	0	0	0	0	0	0	0

From the above computing equations and Table 3, we can see the following facts:

- As discussed in Section 4.1, $une_{css_{ACD}}$ and $omi_{css_{ACD}}$ are always 0.

- une_{css_5} is always 0. In fact, according to CSS_5, we only conduct the fixed-time constraint verification at any activity whose completion duration is more than its maximum duration. And according to the discussion in Section 3, at an activity, if its completion duration is more than its average duration, we should conduct the verification. According to Section 2, the activity average duration is not more than the maximum duration. Hence, all verification based on CSS_5 is necessary. Hence $une_{css_5}=0$.

- If Q=1, $omi_{css_5}=0$. In fact, if Q=1, all activities can be completed within their respective average durations, according to Section 3, we need not conduct any verification. Namely, we will not omit any verification by CSS_5. Hence, $omi_{css_5}=0$.

•With Q decreasing, omi_{css_5} is increasing. This means that the more activities whose completion durations are more than their respective average durations and less than or equal to their respective maximum ones, the more omitted verification based on CSS_5.

In summary, CSS_5 may omit some necessary fixed-time constraint verification. The more activities whose completion durations are more than their respective average durations and less than or equal to their respective maximum durations, the more necessary verification will be omitted. However, CSS_{ACD} does not omit necessary verification. Since the omitted fixed-time constraint verification will eventually affect the overall verification effectiveness, CSS_{ACD} is more effective than CSS_5.

5 Conclusions and Future Work

In this paper, based on the analysis of the limitations of the existing typical checkpoint selection strategies, and the analysis of the relationship between the run-time activity completion duration and SC (Strong Consistency), WC (Weak Consistency), WI (Weak Inconsistency) and SI (Strong Inconsistency), a new checkpoint selection strategy named CSS_{ACD} has been developed. CSS_{ACD} selects the checkpoints dynamically along the grid workflow execution. The final comparison and quantitative evaluation have shown that, compared to the existing typical checkpoint selection strategies, our strategy CSS_{ACD} is more efficient and effective. With these contributions, we can further investigate some problems such as temporal exception handling when a fixed-time constraint is violated at a checkpoint, and the mechanisms by which we can predict future possible checkpoints.

Acknowledgements

The work reported in this paper is partly supported by Swinburne Vice Chancellor's Strategic Research Initiative Grant 2002-2004. It is also partly supported by the National Natural Science Foundation of China under grant No.60273026 and grant No.60273043.

References

[Ab04] Abramson, D. et.al.: An Atmospheric Sciences Workflow and Its Implementation with Web Services. In Proc. of the 4th International Conference on Computational Science, Part Ⅰ, Springer Verlag, LNCS 3036, , Krakow, Poland, June 2004; S. 164-173.

[Am04] Amin, K. et.al.: A Client-controllable Grid Workflow. In Proc. of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04), Hawaii, Jan. 2004; S. 210-219.

[Ca03] Cao, J. et.al.: GridFlow: Workflow Management for Grid Computing. In Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo, May 2003; S. 198-205.

[Cy04] Cybok, D.: A Grid Workflow Infrastructure. In Proc. of Workflow in Grid Systems Workshop in GGF10, 2, Berlin, Germany, Mar. 2004.

[CY04] Chen, J.; Yang, Y.: Temporal Dependency for Dynamic Verification of Temporal Constraints in Workflow Systems. In Proc. of the 3rd International Conference on Grid and Cooperative Computing, Springer-Verlag, LNCS 3251, Oct. 2004; S. 1005-1008.

[CYC04]Chen, J.; Yang, Y.; Chen, T.Y.: Dynamic Verification of Temporal Constraints on-the-fly for Workflow Systems. In Proc. of the 11th Asia-Pacific Software Engineering Conference (APSEC2004), IEEE CS Press, Busan, Korea, Nov./Dec. 2004; S. 30-37.

[CY05a]Chen, J.; Yang, Y.: Temporal Dependency for Dynamic Verification of Fixed-date Constraints in Grid Workflow Systems. In Proc. of the 7th Asia Pacific Web Conference, Springer-Verlag, LNCS 3399, Mar. 2005; S. 820-831.

[CY05b]Chen, J.; Yang, Y.: Multiple Consistency States of Fixed-time Constraints in Grid Workflow Systems. Technical Report, Faculty of ICT, Swinburne University of Technology, Mar. 2005, http://www.it.swin.edu.au/personal/yyang/papers/2005TR-Jchen-1.pdf.

[CM00] Chinn, S.; Madey, G.: Temporal Representation and Reasoning for Workflow in Engineering Design Change Review. IEEE Transactions on Engineering Management, 2000; 47(4), S. 485-492.

[De03] Deelman, E. et.al.: Mapping Abstract Complex Workflows onto Grid Environments. Journal of Grid Computing, 2003; 1(1), S. 9-23.

[EPR99]Eder, J.; Panagos, E.; Rabinovich, M.: Time Constraints in Workflow Systems. In Proc. of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99), Springer Verlag, LNCS 1626, June 1999; S. 286-300.

[FPV04]Fahringer, T; Pllana, S.; Villazon, A.: A-GWL: Abstract Grid Workflow Language. In Proc. of the 4th International Conference on Computational Science, Part Ⅲ, Springer Verlag, LNCS 3038, Krakow, Poland, June 2004; S. 42-49.

[Fo02] Foster, I. et.al.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In Proc. of the 5th Global Grid Forum Workshop (GGF5), Edinburgh, Scotland, July 2002.

[HA00] Hagen, C.; Alonso, G.: Exception Handling in Workflow Management Systems. IEEE Transactions on Software Engineering, 2000; 26(10), S. 943-958.

[Hu03] Huang, Y: JISGA: A JINI-BASED Service-Oriented Grid Architecture. The International Journal of High Performance Computing Applications, 2003; 17(3), S. 317-327.

[KWL02]Krishnan, S.; Wagstrom, P.; Laszewski, G.V.: GSFL: A Workflow Framework for Grid Services. Technical Report, Argonne National Laboratory, Argonne, U.S.A., 2002.

[MO99] Marjanovic, O.; Orlowska, M.E.: On Modeling and Verification of Temporal Constraints in Production Workflows. Knowledge and Information Systems, 1999; 1(2), S. 157-192.

[ZCP01]Zhuge, H.; Cheung, T.; Pung, H.: A Timed Workflow Process Model. The Journal of Systems and Software, 2001; 55(3), S. 231-243.