Dealing with Logical Failures for Collaborating Workflows*

R. Müller, E. Rahm
University of Leipzig, Germany

Abstract. Logical failures occurring during workflow execution require the dynamic adaptation of affected workflows. The consequences such a dynamic adaptation may have for collaborating workflows have not yet been investigated sufficiently. We propose a rule-based approach for dynamic workflow adaptation to deal with logical failures. In our approach, workflow collaboration is based on agreements specifying the delivery time and quality of objects a workflow expects from its collaboration partners. Our mechanisms decide which collaborating workflows have to be informed when a dynamic adaptation is performed. In particular, we estimate the temporal and qualitative implications a dynamic adaptation has for collaboration partners. Because of the automated handling of logical failures, we expect that our approach significantly improves the robustness and correctness of collaborating workflows. The approach has been developed in the context of collaborative workflow-based care for cancer patients.

1 Introduction

Failure situations during workflow execution are usually classified into *system* failures and *logical* failures [24]. System failures cover exceptional situations such as malfunctions of operating system components or database servers. Logical failures refer to application-specific exceptional situations for which the control and data flow of a workflow is not adequate anymore. For example, a workflow supporting a physician during a cancer chemotherapy may become inadequate because the patient suddenly shows a drug allergy. In this case, structural adaptations such as dropping a drug activity may become necessary to cope with the new situation. Previous work on dealing with system failures has often been based on advanced transactional models [13]. Logical failures have been addressed in the field of dynamic workflow management [23,12,4,7].

So far little work has dealt with logical failures affecting *collaborating workflows*. Workflow collaboration usually means that a workflow provides a result for another workflow within a specific time interval or quality range. Thus, a dynamic adaptation of the providing workflow may imply that this result cannot be delivered timely anymore or only with reduced quality. Generally, collaborating workflows are processed by *different* workflow systems located at separate organizational units so that one side usually has no detailed knowledge about logical failures and dynamic adaptations occurring to workflows at the other side. Therefore, a general mechanism is needed that informs collaboration partners in an appropriate way when a relevant dynamic adaptation has been performed for a workflow.

Collaborating workflows are necessary in many application domains, e.g. for e-business, banking or medical treatments. In Fig. 1 we illustrate an example that we will also refer to in subsequent sections. It originates from collaborative cancer treatment. In the shown example, we assume that a workflow system at the department of internal medicine supports the physicians w.r.t. the chemotherapy of a patient, while another workflow system at the radiotherapy department supports tasks such as the preparation, performance and aftercare of radiotherapy procedures. Both workflows depend on each other in order to allow for a coordinated treatment of a patient. For example, a treatment may consist of a two weeks che-

* Supported by the GERMAN RESEARCH ASSOCIATION (DFG) under grant number Ra 497/12-1

O. Etzion and P. Scheuermann (Eds.): CoopIS 2000, LNCS 1901, pp. 210–223, 2000.

motherapy and parallel units of supporting radiotherapy every two days (for radiotherapy, the patient has ambulance appointments at the radiological department). If a logical failure such as an unexpected allergy w.r.t. the drug VINCRISTIN occurs, this may require dynamic adaptations of the chemotherapy workflow such as deleting the VINCRISTIN node (Fig. 1). This adaptation may impact the radiotherapy workflow. As a *temporal* implication the chemotherapy workflow may be finished earlier so that a radiotherapy unit may be started earlier. As a *qualitative* implication, additional radiotherapy units may become necessary to compensate the dropped drug being essential for tumor remission. Thus deletion of nodes in one workflow can make it necessary to insert additional nodes in a collaborating workflow.

To address logical failures and their impacts on collaborating workflows, we are currently developing the workflow management system AGENTWORK at the University of Leipzig. Main characteristics of AGENTWORK described in this paper are as follows:

First, AGENTWORK allows to specify at what time a workflow expects which results from other workflows. In particular, tolerance limits and constraints for delivery times and result qualities can be specified. Second, based on our previous work [21] we use a rule-based approach for dynamic workflow adaptation when logical failures occur.

Third, as a main contribution of this paper, we provide a model that enables a workflow system to decide which collaborating workflows have to be informed in what manner when a dynamic adaptation is performed. In particular, we propose a *predictive* strategy estimating whether constraints for delivery times or result qualities will be violated due to the dynamic adaptation. In this way we inform collaborating partners *in time* so that they can prepare themselves w.r.t. consequences of the logical failure. For example, if the chemotherapy adaptation in Fig. 1 implies that the required total amount of drug dosages cannot be applied anymore, we inform the radiological department as soon as possible and *before* the chemotherapy has finished. This allows preparing, for instance, additional radiotherapy units to compensate the reduced chemotherapy which would not have been possible without a predictive approach. *Temporal* implications of an adaptation are determined by estimating the duration that will be needed to execute the dynamically adapted workflow, and by compar-

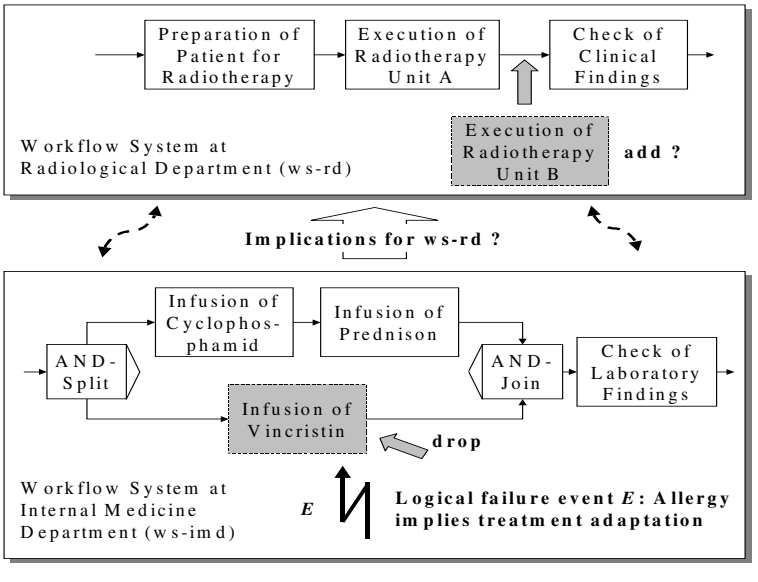

Fig. 1. Example of two collaborating workflows in a medical domain. CYCLOPHOSPHAMID, PREDNISON and VINCRISTIN are cancer drugs.

ing it with originally fixed time constraints. If time constraints are expected to be violated, affected collaboration partners are informed immediately. For determining *qualitative* implications we will introduce so-called *quality-measuring objects*. Such objects have already been used for quality control in data warehousing [15] but, to our knowledge, not yet in the workflow area. In our context, quality-measuring objects are numerical objects of a workflow's data flow that measure the quality of a result provided by a workflow. Such objects are used to decide whether an adaptation decreases the quality of a collaboration result below a specified tolerance limit so that collaboration partners have to be informed. In our medical example, the drug dosage applied to a patient can serve as a quality-measuring object as it is an important measure for the degree of tumor remission. In financial applications, price and credit limits may represent such quality objects.

Fourth, AGENTWORK aims at automating as much as possible of this process. This is desirable especially in large-scale workflow environments with many collaborating workflows running concurrently. By a high degree of automation, we expect to reduce the probability that collaboration partners are not informed timely about relevant logical failures and dynamic workflow adaptations.

The paper is organized as follows: After a discussion of related work in section 2, section 3 gives an overview of the AGENTWORK system. Sections 4 introduces our workflow and collaboration model; section 5 explains the rule-based approach for handling logical failures. In section 6 we outline the approach to determine the temporal and qualitative implications of workflow adaptation for collaborating workflows. Finally, we summarize and discuss future work.

2 Related Work

Collaborative workflow research has focussed on aspects such as interoperability frameworks, collaboration management infrastructures and workflow-oriented event notification systems [1,5,9]. However, not much work has been done so far to cope with failure management in inter-workflow collaboration scenarios.

For example, in [11] an approach is described for event-based communication between processes interacting in a consumer-producer relationship. A dependency graph maintains which processes have been triggered by which events. If a process *P* fails, the system derives from the dependency graph which processes depend on *P* and sends exception events to them. Notified processes then perform an abort or compensation-based partial rollback. The possibility that affected processes may continue after a dynamic adaptation is not investigated by the authors.

In WIDE [6], workflow collaboration is specified via SEND and RECEIVE nodes by which workflows synchronously can exchange information about results. Failures leading to a workflow starvation (i.e. a receiver workflow waits in vain for a result) or a deadlock (i.e. two workflows in vain wait for results from each other) are handled as follows: Either an alternative control flow path that already has been specified at definition time is executed when a waiting threshold expires, or the conflict is resolved manually in an ad-hoc manner. A more detailed communication protocol informing that because of a logical failure a result will be provided *later* or *with reduced quality* is not supported. In particular, it is not investigated how to adapt a receiver workflow dynamically so that it can better cope, for example, with a result of reduced quality.

In [18], unexpected terminations of workflows or workflow activities in e-commerce scenarios are addressed. In case of such a termination, a gateway protocol informs collaborating workflows about the termination reason and the state of the failed workflow or activity. Furthermore, collaborating workflows may be informed about a modification of an already agreed-on *price* for the service or product that is going to be provided by the workflow affected by the failure. To determine the price modification due to a termination, exception rules can be assigned to agreements specifying under which termination circumstances

which price modifications shall be applied. However, the approach does not cover failures *not* leading to a workflow or activity termination but for example to the dynamic dropping or adding of activities. Furthermore, considering only price modifications is not appropriate for many non-commercial domains such as collaborative medical care.

Recently, also approaches from artificial intelligence have been proposed for workflow failure handling. For example, in [14] business processes are modeled in terms of *agents, services*, *(re-)negotiations* and *service failures*. In [17], exception handling agents detect and resolve workflow exceptions using the *heuristic classification* method [8]. However, both approaches do not address how failures are resolved in means of structural process adaptations and how the consequences for collaboration partners can automatically be derived. Other approaches addressing constraint violations during plan execution [3,22] also do not to the best of our knowledge, specifically address the temporal and qualitative consequences of plan adaptations for collaboration partners.

3 Layers and Components of AGENTWORK

We are currently developing the AGENTWORK prototype to support rule-based dynamic workflow adaptation in order to deal with logical failures. It consists of three architectural layers (Fig. 2). A *communication layer* based on CORBA is responsible for communication with other workflow systems, databases, users, etc. The *workflow definition and execution layer* supports the definition and execution of workflows. The *layer for logical failure handling* provides three agents to cope with logical failures:

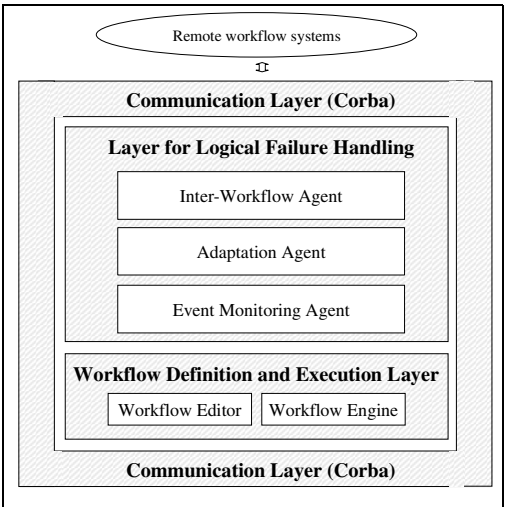

Fig. 2. Layers and components of AGENTWORK.

- The *event monitoring agent* decides which events in the workflow environment constitute logical failure events.
- The *adaptation agent* adapts affected workflows. For example, it removes or inserts activities so that the workflow can better cope with the new situation caused by the logical failure event.
- The *inter-workflow agent* determines whether a dynamic workflow adaptation has any implications for other workflows collaborating with this workflow.

How these tasks are achieved will be explained in the sequel. AGENTWORK is currently applied to the domain of cancer treatment within the HEMATOWORK project [19] at the University of Leipzig.

4 Workflow Model

We now briefly outline our approach for defining workflows and workflow collaboration. In particular we introduce temporal and qualitative collaboration agreements. AGENTWORK definitions are based on an object-oriented meta model mainly consisting of a class hierarchy for cases, events, activities and agents. A *Case* object represents a person or institution for which an enterprise or organization provides its services (such as a patient or a customer). Objects of class *Event* represent anything that occurs w.r.t. a case and therefore may impact workflows running for this case. The *Event* subclass *Activity* is used for events that do

not only happen to a case but are actively performed (e.g. a drug infusion). Activities are performed by *Agent* objects, such as physicians, clerks or application programs. The agents of the layer for logical failure handling are also members of this class.

4.1 Workflows and Activities

Workflows are defined based on the sketched meta model. Activities of a workflow are represented by *activity nodes*; the control flow is specified by *edges* and *control nodes*. AGENTWORK provides control node types for conditional branching (node types OR-SPLIT/OR-JOIN), for parallel execution (AND-SPLIT/AND-JOIN) and loops (LOOP-START/LOOP-END). For every split node or LOOP-START node there must be exactly one closing join node or LOOP-END node. The data flow is represented by data flow edges. Internal data flow edges specify the data flow between nodes within one workflow. External data flow edges specify the data flow between workflow nodes and external data sources such as databases or user interfaces.

An activity node has an associated *activity definition* to specify what has to be done when the control flow reaches this node. In Fig. 3 an activity definition using the *Activity* subclass *Drug-Infusion* has been assigned to an activity node specifying that the patient has to receive a VINCRISTIN infusion with a dosage of 2 mg. Furthermore, it is specified that the agent performing this activity must be a physician.

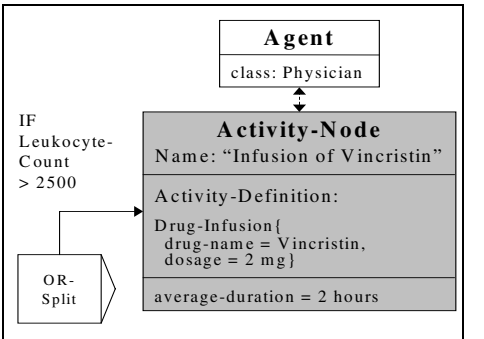

Fig. 3. Workflow definition part.

As a shorthand, we use the terms *A-activity* and *A-node* to denote an activity resp. activity node based on the activity definition *A*.

To an activity definition *A*, meta information about the execution duration of *A*-activities can be assigned. This may be the minimal, maximal or average duration (e.g. in Fig. 3 it is specified that the average duration of such an infusion is 2 hours). In addition, the workflow engine measures the execution durations for each activity type. These measurements are used to calculate and to continuously refine the *average, maximal* and *minimal* duration of activities of a specific type. These measurements allow restricting the use of pre-specified duration information to the first phase of an AGENTWORK installation.

As usual, the term workflow refers to an instantiation of a workflow definition executed by the workflow engine. In AGENTWORK, a workflow runs for exactly one case (e.g. patient or customer) but for one case several workflows may be executed concurrently.

4.2 Workflow Collaboration

In AGENTWORK, workflow collaboration is specified on a *communication level* by defining when a workflow has to exchange which information to which other workflow system. Note that we communicate with a *workflow system* and not directly with its *workflows*, as a workflow modeler at one site usually does not have knowledge about the structure of workflows at another site. Thus it is the task of the receiving workflow system as our collaboration partner to propagate information messages to those workflows that are affected.

Workflow communication is specified by COMM-OUT and COMM-IN nodes and inter-workflow objects. A COMM-OUT node specifies when information has to be send to some collaboration partner. A COMM-IN node specifies when information is expected to be received from some collaboration partner. The details are specified by *inter-workflow objects* assigned to these nodes. Such an inter-workflow object if of the structure *(ws, o: Class, cs)*

where *ws* identifies the collaborating workflow system that shall receive or is expected to send information.

- *o* is an object of class *Class* which is exchanged between collaboration partners and which contains or describes a product or service.
- *cs* is a *Case* object and describes the case to which *o* belongs (e.g. a patient treated collaboratively or an industrial customer). The receiving workflow system uses *cs* to identify the affected workflows.

In Fig. 4 we extend the workflows of Fig. 1 with such communication nodes and inter-workflow objects. For the lower workflow, the COMM-OUT node and its associated inter-workflow object specify that a *Chemo-Report* object *c* has to be sent to the radiological workflow system *ws-rd* after the inspection of several laboratory findings. *imd-pat* identifies the patient to whom the report belongs. Vice versa, the upper workflow at *ws-rd* contains a COMM-IN node with an inter-workflow object stating that before the radiotherapy preparation a *Chemo-Report* object is expected from *ws-imd* w.r.t. the patient *rd-pat* treated by the radiotherapy workflow. Based on attribute values of *rd-pat*, *ws-rd* can determine which inter-workflow object received from *ws-imd* belongs to which of its workflows. To a COMM-IN or COMM-OUT node several inter-workflow objects can be assigned, and a workflow may contain an arbitrary number of COMM-IN or COMM-OUT nodes.

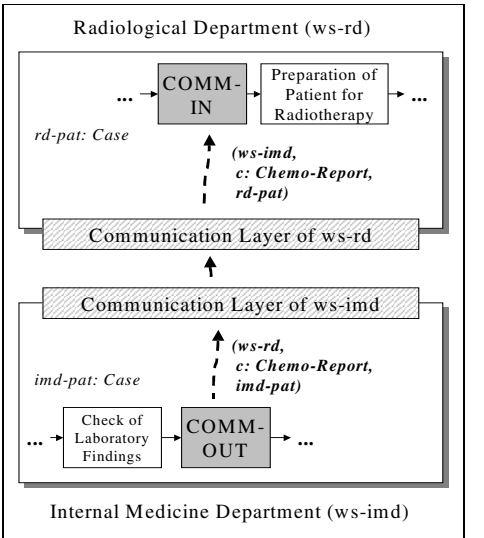

Fig. 4. Workflow communication example.

COMM-OUT and COMM-IN nodes are processed as follows:

- When a COMM-OUT node is reached, for each of its inter-workflow objects *(ws, o: Class, cs)* the tuple *(o: Class, cs)* is sent to *ws*. This is done asynchronously, i.e. the workflow is continued after the send operation. If the path with the COMM-OUT node shall not be continued until *ws* sends a reply, this can be specified by placing a COMM-IN node directly after the COMM-OUT node.
- Vice versa, when a COMM-IN node is reached with an inter-workflow object *(ws, o: Class, cs)*, the engine checks whether such an object *o* for case *cs* has already been received from *ws*. If yes, the engine forwards *o* to all activity nodes of the respective workflow that need *o*. If no, the engine waits for *o* until a deadline is exceeded and then sends a reminder to *ws*.

By default COMM-OUT and COMM-IN nodes are executed when they are reached during workflow execution. In addition, at workflow start time or during execution we can assign absolute (calendar) time points to them to specify when information has to be send *to* or when it is expected *from* a collaboration partner (e.g. send information on 20th July 2000, 6 p.m.). For example, for a long-term workflow covering two phases of a treatment it may be useful to dynamically assign absolute time points fixing the intended delivery times w.r.t. the remaining COMM-OUT nodes not before the second phase is entered. Such absolute (calendar) time points can be manually assigned or can automatically be derived by estimating the execution duration of the path(s) leading to the communication node and by adding this duration to the workflow's actual execution time point (similar to [10]). A combination could consist of a manual assignment and an automated check estimating whether this absolute time point is realistic w.r.t. the workflow definition. We describe such estimation algo-

rithms in more detail in section 6 where they are used for determining temporal implications of workflow adaptations. Relative time points can also be assigned to a COMM-OUT node (e.g. send information 3 weeks after workflow start), but are converted to absolute time points by the system.

Furthermore, we assign so-called temporal and quality collaboration agreements to communication nodes in order to specify which deviations, e.g. caused by logical failures, are tolerable and which not.

Temporal collaboration agreements: To a COMM-OUT node two thresholds *acc-threshold* (*acc* for acceleration) and *delay-threshold* of the form *(v, time-unit)* can be assigned (with $v \geq 0$ and *time-unit* $\in$ {*sec, min, hour, day, week, ...*}). The semantics is:

- If *acc-threshold* or *delay-threshold* is left unspecified, an acceleration resp. delay is viewed as irrelevant.
- If an absolute time point *atp* (such as 20th July 2000, 6.00 p.m.) has been assigned to the COMM-OUT node, *acc-treshold* and *delay-threshold* specify that the workflow containing this COMM-OUT node should send its information within the interval
  [atp - acc-threshold, atp + delay-threshold]
  (e.g. within [20th July 2000 - 2 days, 20th July 2000 + 3 days]). Whenever a dynamic adaptation implies that this will not be possible anymore, the collaboration partner has to be informed.
- If *no* absolute time point has been assigned to the COMM-OUT node, *acc-treshold* and *delay-threshold* refer to the relative change in the execution time due to a workflow adaptation. Let d_{before} denote the execution time that would have been needed to reach the COMM-OUT *before* the adaptation, and d_{after} the execution time that will be needed to reach the COMM-OUT *after* the adaptation. The collaboration partner then has to be informed if:
  $d_{before} - d_{after} >$ *acc-threshold* (workflow accelerated by more than *acc-threshold)* or
  $d_{after} - d_{before} >$ *delay-threshold* (workflow delayed by more than *delay-threshold).*

We emphasize that the described semantics of these thresholds serve the specific purposes of logical failure handling. For handling deadlines and temporal thresholds for workflow execution not "disturbed" by dynamic adaptations we refer to [16,10].

Qualitative collaboration agreements: A collaboration partner often expects that a result provided by its partner will arrive not only in time but also with a certain quality. To express this AGENTWORK allows to assign *quality constraints* to inter-workflow objects. For example, the chemotherapy report object *c* of Fig. 4 may have different subsections for the applied drugs, for clinical findings and for laboratory findings. Then, by assigning a quality constraint to *c* such as

c.subsection-for-applied-drugs = Mandatory AND
c.subsection-for-laboratory-findings = Mandatory

both collaboration partners could fix the agreement that in the report at least the subsections for the applied drugs and the laboratory findings have to be filled out as otherwise the radiotherapy workflow cannot continue because important patient data are missing.

Even more, in many domains the quality of a result can be expressed by a numerical threshold value. For example, the weighted sum of the report's drug dosages describes the quality of the chemotherapy as it closely correlates to the degree of tumor remission[†]. The collaboration partners then could also assign a quality constraint such as

c.weighted-sum-of-drug-dosages > 100 mg

to the transferred report *c*. If this constraint is violated because some drugs had to be dynamically dropped from the chemotherapy workflow, the radiological department has to be informed as it may be necessary to dynamically add some radiotherapy units to compensate the reduced chemotherapy. Generally, we will refer to a numerical object that is used to mea-

[†] The sum is weighted as the different drugs have a different strength w.r.t. tumor remission.

sure the quality of a result as a so-called *quality-measuring object*. Non-medical examples for quality-measuring objects and constraints on them could be price ranges for e-business interactions or credit limits for banking applications.

Determining how dynamic adaptations may influence such a quality-measuring object requires additional quality-related meta knowledge w.r.t. workflow *activities*. Therefore, in AGENTWORK quality transformation rules can be assigned to an activity definition *A* stating how *A*-activities transform a quality-measuring object. For example, the activity definition of Fig. 3

A := Drug-Infusion{drug-name = VINCRISTIN, dosage = 2 mg}

can be augmented by the quality transformation rule

c.weighted-sum-of-drug-dosages += 2 mg

to account for the respective drug dosage increase. Based on this meta knowledge, qualitative implications of adaptations can then be determined as we show in section 6.

5 Logical Failures and Intra-Workflow Adaptation

To handle logical failure events, we use *event-condition-action rules* of the structure

WHEN event WITH condition THEN control action

Such a rule specifies in its event-condition part which event constitutes a logical failure. The action part states which *control action* has to be performed for workflow activities to cope with the failure event. Table 1 lists the supported control actions. *A* and *B* denote activity definitions, *cs* again denotes a case (e.g. a patient or customer).

Control Action	Meaning
drop(A,cs)	For *cs*, *A*-activities must not be executed anymore.
replace(A,B,cs)	For *cs*, every *A*-activity execution is to be replaced by a *B*-activity.
check(A,cs)	For *cs*, every execution of an *A*-activity has to be checked by a user.
add(A,cs)	For *cs* an *A*-activity has *additionally* to be executed exactly *once*.

Table 1. AGENTWORK Control Actions

The *check(A,cs)* control action is used when there is not enough knowledge available to decide whether an *A*-activity has become inadequate or not for *cs*. When a *check(A,cs)* control action is triggered, control is shifted to a user who has to specify whether the activity should, e.g., be dropped or replaced.

An example for a rule triggering a control action is the following (*A* denotes the activity definition *Drug-Infusion{drug-name = VINCRISTIN}*, *Hemato-Findings(pat-id, parameter, value)* is a table collecting blood-related patient data):

WHEN INSERT ON Hemato-Findings REFERENCING NEW AS h **(*)**
WITH h.parameter = Leukocyte-Count AND h.value < 1000
THEN drop(A, h.pat-id)

This rule expresses that if a patient has a leukocyte count less than 1000, VINCRISTIN infusions have to be dropped for this patient. With rules such as **(*)**, AGENTWORK can monitor any application environment events that may impact workflows.

When a new event *E* occurs, the following steps are performed: First, the event monitoring agent checks whether *E* constitutes a logical failure. *E* is classified as failure event if at least one control action is triggered by rules such as **(*)**. Second, if a control action *ca(A,cs)* has been triggered, affected workflows running for *cs* are determined. Concerning *drop*, *replace* and *check,* a workflow is affected if it contains at least one *A*-node in the remaining control flow. Concerning *add*, the workflow the user has selected for the new *A*-node is affected. An affected workflow then is interrupted. With N_E we denote the *interruption node set* which is the set of nodes which were either in execution or prepared for execution (i.e. the predeces-

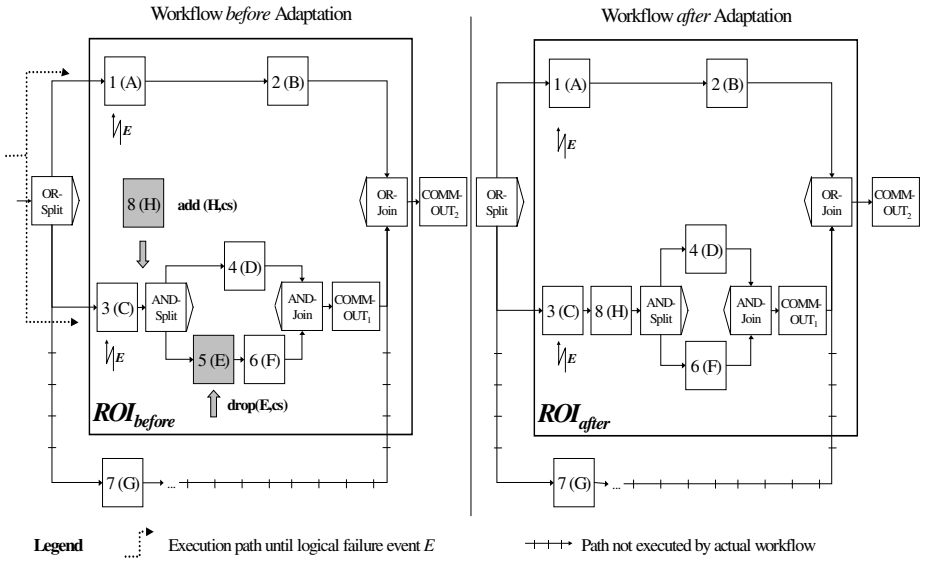

Fig. 5. The regions of interest ROI_{before} and ROI_{after}. Letters denote activity definitions.

sor nodes have already committed) at the interruption moment. The cardinality of N_E may be > 1, if the workflow is interrupted during parallel execution (i.e. after an AND-SPLIT or a non-exclusive OR-SPLIT). In Fig. 5, N_E consists of nodes 1 and 3.

Third, after the interruption the *adaptation agent* translates *ca(A,cs)* into node operators adapting the control and data flow. By default, *all A*-nodes in the remaining control flow are handled, e.g. all *A*-nodes would be dropped from the remaining control flow if *ca(A,cs)* = *drop(A,cs)*. If the adaptation shall be restricted only to a *part* of the remaining control flow, a user can graphically select the workflow part to which the adaptation operations shall be applied exclusively. A new node which has to be added because of an *add* control action is – by default – inserted directly after a node of N_E. If such a new node shall be inserted somewhere later in the control flow, this has to be specified by a user. In Fig. 5, the adaptation agent has dropped an *E*-node (5) from the control flow because of a *drop(E,cs)* control action and has added a new *H*-node (8) because of an *add(H,cs)* control action.

AGENTWORK also supports the *semi-automated* determination of an appropriate part to which the adaptation shall be restricted. This is mainly achieved by the possibility to assign a valid time interval to the action part of a failure rule, such as:

WHEN event WITH condition THEN drop(VINCRISTIN,cs)
VALID-TIME [now, now + 3 days]

stating that the drug VINCRISTIN shall be dropped only for the next three days (starting from "now" which is the moment when the control action was triggered). The adaptation agent estimates which part of the remaining control flow will be executed during the next three days, and applies the adaptations operation only to this part.

The *consistency* of adapted workflows is achieved as follows: First, the consistency of failure handling rules is ensured by rule consistency algorithms as proposed in [2]. This avoids that for example *drop(A,cs)* and *add(A,cs)* are triggered for the same workflow at the same time. Second, consistency constraints reject adaptations leading, for example, to an activity node for which input objects are not provided by data flow edges. Third, only authorized users such as senior physicians may contribute to adaptations.

As we want to concentrate on the *inter*-workflow aspects of workflow adaptation, we re-

fer to [21] for further details w.r.t. our intra-workflow adaptation approach.

6 Managing Inter-Workflow Implications of Adaptations

Before an adapted workflow is continued, the *inter-workflow agent* is invoked if the workflow contains COMM-OUT nodes in its remaining control flow. This agent has to determine whether the adaptation affects any collaborating workflow. Principally, it operates as follows:

First, it determines the so-called **region of interest** *ROI* (Fig. 5). This is the workflow region which starts at the interruption node set N_E and contains *all* adapted workflow parts. If all adaptations from N_E occurred within a sequence of activities not belonging to a split/join region, *ROI* simply ends with the last node (having the largest distance to N_E) that was dropped, replaced or added. If the last adaptation took place within a split/join region (e.g. between an OR-SPLIT and an OR-JOIN node), *ROI* ends *beyond* these last adapted parts at the first AND-JOIN or OR-JOIN node *joining all reachable paths starting from nodes of* N_E even if one of these paths has not been adapted at all. This is necessary, as especially the *temporal* influence of an adaptation often can only be determined by considering *all reachable paths* starting at N_E up to this joining node. For example, in Fig. 5 for COMM-OUT$_2$ the temporal influence of the shown adaptation requires considering the path $1 \rightarrow 2$ although it has not been adapted. This is because the execution duration of path $1 \rightarrow 2$ may be longer than that of the adapted paths. Therefore, in Fig. 5 *both* paths starting at N_E have to be considered up to the closing OR-JOIN.

In the sequel, ROI_{before} and ROI_{after} denote the region of interest *before* resp. *after* the adaptation. They may cover the whole remaining control and data flow or only a part of it (e.g. if the user manually has restricted the adaptation to such a part). As the control actions of Table 1 only affect *activity* nodes, ROI_{before} resp. ROI_{after} contain the same set of communication nodes. We write *ROI* instead of ROI_{before} or ROI_{after} if the distinction between the region before and after the adaptation is irrelevant.

Second, the relevant **temporal and qualitative implications** of the adaptation are determined. This is mainly done by estimating and comparing the execution durations and "qualities" of the regions ROI_{before} and ROI_{after}. Whether the entire regions ROI_{before} and ROI_{after} are considered or only parts of them depends on the location of the COMM-OUT nodes. This will be discussed below. The *execution duration* of a workflow region is estimated on the base of several path duration algorithms of [20]. The *quality* of ROI_{before} and ROI_{after} is determined and measured by using quality constraints and quality-measuring objects (section 4.2). If temporal thresholds or qualitative constraints are violated for an inter-workflow object because of an adaptation, the affected collaborating workflow system is directly informed.

Third, when the workflow is continued, the **actual execution is monitored** by the inter-workflow agent. For example, unexpected delays of activity executions or external data accesses caused by system failures can be detected by this monitoring, and the inter-workflow agent can refine its estimations. If necessary, the collaboration partners are informed about such refined estimations.

A workflow system *ws* can handle temporal or qualitative effects about which it was informed as follows: It may itself have failure rules stating how to react on constraint violations caused by (remote) logical failures of a collaborating workflow system. Alternatively, users of both workflow systems together may decide how *ws* should react.

6.1 Determining Temporal Implications

We first sketch the principal algorithm of determining temporal implications. Then, we describe how execution durations are estimated.
The principal algorithm is as follows (Fig. 6):

1. *Handling nodes within ROI*: For each COMM-OUT node within *ROI* the subregion $SROI_{after}$ is determined leading from the interruption node set to the COMM-OUT node. For example, for COMM-OUT_1 in Fig. 5 $SROI_{after}$ consists of nodes 3, 8, 4, 6 and the AND-SPLIT/AND-JOIN nodes. After this, it is estimated how long it will take to execute $SROI_{after}$.

If an absolute calendar time point is assigned to the COMM-OUT node, it is then checked whether the estimated execution duration of $SROI_{after}$ violates a temporal threshold for any inter-workflow object *(ws, o: Class, cs)* of the COMM-OUT node. If yes, *ws* is informed (left branch of Fig. 6). If *no* absolute time point is assigned to the COMM-OUT node, the duration that would have been needed to execute $SROI_{before}$ has to be estimated, too. Then, it is checked whether there is a mismatch between the durations of $SROI_{before}$ and $SROI_{after}$ violating any temporal thresholds of an inter-workflow object *(ws, o: Class, cs)* of the COMM-OUT node. If yes, *ws* is informed (middle branch of Fig. 6).

2. *Handling nodes beyond ROI*: For COMM-OUT nodes beyond *ROI*‡ – such as COMM-OUT_2 in Fig. 5 – the same procedure is performed as for COMM-OUT nodes within *ROI* with the only difference that we now consider the entire region *ROI* instead of a subregion *SROI* (right branch of Fig. 6). In particular, it is estimated how long it will take to execute ROI_{after}. Then, for every inter-workflow object of one of the COMM-OUT nodes beyond *ROI* we check whether any temporal constraints are violated. For example, if an absolute time point *atp* has been assigned to a COMM-OUT node beyond *ROI* it may be that already the duration of ROI_{after} makes it impossible to reach the node before *atp* + *delay-threshold*. If such a constraint is violated, the affected workflow systems are immediately informed. Obviously, for the duration estimations w.r.t. $ROI_{after/before}$ the results w.r.t. $SROI_{after/before}$ can be re-used.

Note that estimating the time duration of *ROI* does not take into account the execution time of nodes that have to be executed between the end of *ROI* and the COMM-OUT node.

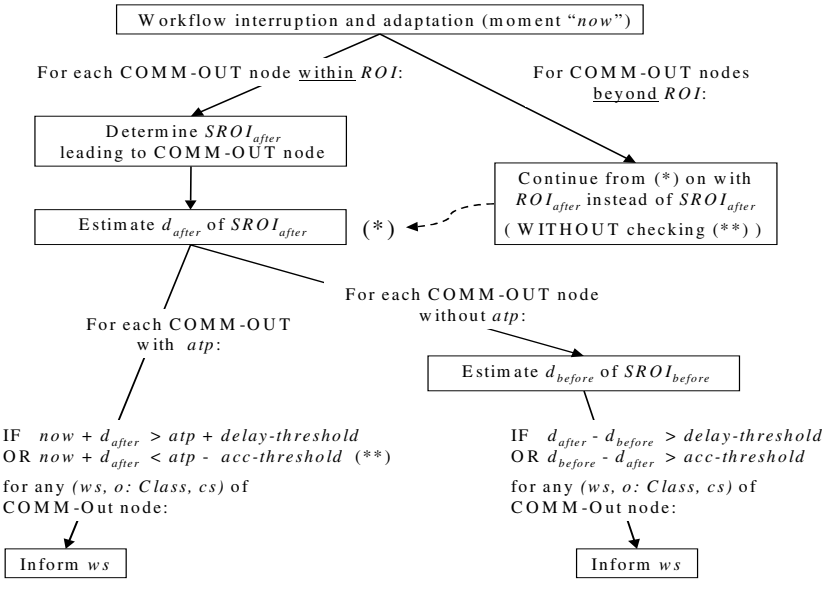

Fig. 6. Algorithm checking temporal implications for collaborating workflows (*atp* = absolute time point, $d_{before/after}$ = execution duration before/after adaptation). The acceleration check (**) is not done for a COMM-OUT node beyond *ROI* as there may be further nodes between *ROI* and the COMM-OUT node.

‡ Note that ROI_{before} and ROI_{after} do not differ w.r.t. communication nodes.

For a COMM-OUT node with *no* absolute time point this is irrelevant as the algorithm then only has to check whether there is a relative temporal mismatch w.r.t. the durations *before* and *after* the adaptation. As nodes beyond *ROI* have *not* been adapted, this mismatch is entirely determined by *ROI*. For a COMM-OUT node with an absolute time point the nodes between it and *ROI* are relevant, but the COMM-OUT node may be far away from *ROI* so that estimating the time durations between *ROI* and such a node would be inherently imprecise. Yet, by considering the changed duration of *ROI* we already enable an early notification for workflow adaptations with significant impact on execution times (e.g. we detect if already the duration of ROI_{after} makes it impossible to reach the COMM-OUT node in time). For those COMM-OUT nodes outside *ROI* not affected by the change of the *ROI* execution time, we switch to a continuous monitoring approach. That is, after the workflow adaptation it is continuously checked for these COMM-OUT nodes whether their timing constraints can still be met. By combining the predictive approach with continuous monitoring we achieve early notifications even when only partial or imprecise execution time estimates are possible. The overhead for the additional checks is considered acceptable because first the number of affected COMM-OUT nodes is limited and second workflow adaptations due to logical failures should be relatively rare.

We now sketch how the execution duration of workflow regions is estimated. In the current implementation, estimations are based on the *average* duration of activity executions. *Worst*-case durations using the *maximal* activity duration are viewed as too pessimistic, as too often delays would be predicted which would not really occur.

The duration of a **sequence** is estimated by iteratively summing up the average activity durations. Control edges are assumed to have a *negligible* duration. The duration of an **AND-SPLIT/AND-JOIN region** (such as nodes 4, 5, 6 in ROI_{before} in Fig. 5) is the maximum of the estimated durations of all its paths.

If an **OR-SPLIT/OR-JOIN region** is discovered, it is tried either to predict which of the paths starting at the OR-SPLIT will qualify for execution when the workflow will be continued, or the maximum of all paths is taken as a worst-case estimation (as not all paths may be executed actually). A prediction may be possible if the workflow interruption has occurred close to the OR-SPLIT, e.g. when data needed for the decision which paths will qualify for execution is already available. For example, at an OR-SPLIT one path may qualify for execution if the patient has liver metastases, the other one if the patient has *no* liver metastases. If it is definitely known at adaptation time that the patient has liver metastases, then only the duration of the metastases path would have to be taken into account. Furthermore, instead of worst-case estimations one could also determine (and later on execute) the path with the *smallest* duration or *highest* quality, if constraints otherwise would be violated. However, as OR splits usually contain data-related conditions (such as *IF Leukocyte-Count > 2500* in Fig. 3), this often is not appropriate or at least requires some user intervention.

The duration of a **loop** is estimated by determining the duration of the loop's body sequence and then by trying to estimate how often the loop will be iterated. AGENTWORK supports three principal ways for the latter (used with decreasing priority): First, the engine records all loop executions and calculates an average number of iterations for every loop (analogously to the monitoring of activity executions). Second, at workflow definition time heuristic information about the average number of iterations can be specified (such as "On average, the radiotherapy unit of type *A* has to be repeated three times until the tumor vanishes"). Third, it is tried to resolve the loop's termination condition in a similar way as it is done w.r.t. OR splits (i.e. by inspecting whether necessary data is already available).

Duration information w.r.t. **data flow edges** (e.g. for accessing external data sources) is obtained by measuring the average duration of such data accesses.

The described mechanisms estimate the duration of workflow regions in a primarily *average*-based manner. If an OR-SPLIT can not be resolved and therefore the maximum of all

(average) path durations has to be taken, this estimation *locally* becomes a worst-case estimation (but not as much as it would be the case when using the maximal duration for *all* workflow activities). In case nothing can be said about a loop, the region estimation remains incomplete but may nevertheless be helpful as, for example, relevant delays w.r.t. the rest of the region may still be predictable and can be communicated to collaboration partners.

6.2 Determining Qualitative Implications

For qualitative implications, the inter-workflow agent uses a modification of the temporal algorithm of Fig. 6 restricted to COMM-OUT nodes *within ROI*. Instead of estimating the *durations* of regions such as $SROI_{after}$ it determines the *qualitative* effects of the activities by using the quality transformation rules of section 4.2. Then, it checks whether the derived quality w.r.t. the adapted workflow violates any *quality* constraints (instead of temporal) of an inter-workflow object. The restriction to COMM-OUT nodes *within ROI* is made as AGENTWORK views qualitative estimations for COMM-OUT nodes far away from the adapted workflow region as being inherently imprecise.

If no qualitative implications can be determined because of missing quality-oriented meta knowledge, the inter-workflow agent at least informs the collaboration partners which activities have been dropped or added due to the dynamic adaptation. However, as this requires that the activities performed by one workflow are *meaningful* for the collaboration partner, AGENTWORK views this only as an "emergency solution".

The temporal and qualitative estimation approaches work if an AGENTWORK workflow is "well-formed" in the sense that an object *o* sent by a COMM-OUT node is only produced by predecessor nodes of this COMM-OUT node. For Fig. 5 this means that the objects sent by COMM-OUT_1 may only be provided by nodes 3, 5, 6, 4 (before adaptation) resp. 3, 8, 4, 6 (after adaptation) or predecessors of 3, but not by node 1 or 2. If, for example, node 2 would contribute to an object of COMM-OUT_1 (and therefore, for example, could delay the execution of COMM-OUT_1 or influence qualitative constraints) then COMM-OUT_1 would have to be placed behind the OR-JOIN node. However, the algorithm can easily be extended to cope also with "cross-over" data flow edges (e.g. a data flow edge transferring data from node 1 to COMM-OUT_1 in Fig. 5): For each COMM-OUT node within *ROI* the subregion $SROI_{after}$ has to be recursively extended by all nodes contributing to the objects send by the COMM-OUT node. Syntactically, these are the nodes from which internal data flow edge paths are leading to the COMM-OUT node.

7 Summary and Discussion

We have introduced a model to deal with logical failures and dynamic adaptations for collaborating workflows. If a workflow is adapted, the temporal and qualitative implications for collaborating workflows are automatically determined. Relevant failures causing agreed-on temporal and qualitative constraints to be violated are immediately communicated to affected collaborating workflow systems. By this approach, we expect that the frequency of failure situations inducing local workflow adaptations but *not* reported timely to affected collaboration partners can be reduced significantly.

Our approach is based on knowledge such as activity durations, quality-measuring objects and quality transformation rules. We expect that in many application domains – due to the increasing importance of quality management and quality assurance – the duration and quality transformation „behavior" of activities can be obtained quite exactly. In particular, in most real-world collaboration scenarios, there will be at least one object (such as a document) containing information which in some way measures the quality of products or results provided by the collaboration partners. Our approach does not need significantly more specifications than other workflow failure handling approaches where, for example, compensating activities have to be defined for every „normal" activity [13]. Of course, it is not possible to derive *all* implications of a logical failure automatically because the internal structure of

workflows can be arbitrarily complex. If the most significant problem cases can be handled this would already be of great value.

Currently, we are completing the implementation of the sketched approach within the HEMATOWORK project. We are also evaluating the applicability of the approach in different application domains like electronic inter-business workflows. The approach shall also be extended by using additional activity meta information (such as the resources required to execute an activity) and by determining the respective implications of adaptations. Furthermore, we plan empirical studies on the quality of temporal estimations for real-world workflows.

References

1. Georgakopoulos, D., Schuster, H., Cichocki, A., Baker, D.: Managing Process and Service Fusion in Virtual Enterprises. Information Systems 24 (1999) 429-456
2. Baralis, E.: Rule Analysis. In: Paton, N. (ed.): Active Rules in Database Systems. Springer (1999) 51-67
3. Blum, A., Furst, M.L.: Fast Planning Through Planning Graph Analysis. Artificial Intelligence 90 (1997) 281-300
4. Borgida, A., Murata, T.: Tolerating Exceptions in Workflows: a Unified Framework for Data and Processes. WACC (1999) 59-68
5. Bussler, Ch.: Workflow Interoperability Classification and its Implication to Workflow Management System Architectures. EDBT Workflow Workshop (1998): 45-54
6. Casati, F.: Semantic Interoperability in Interorganizational Workflows. WACC Workshop on Cross-Organizational Workflows (1999)
7. Chiu, D.K.W., Li, Q., Karlapalem, K.: A Meta Modeling Approach to Workflow Management System Supporting Exception Handling. Information Systems 24 (1999) 159-184
8. Clancey, W.J.: Heuristic Classification. Artificial Intelligence 25 (1985) 289-350
9. Dogac, A., Kalinichenko, L., Özsu, T., Sheth, A. (eds.): Workflow Management Systems and Interoperability. Springer (1998)
10. Eder, J., Panagos, E., Rabinovich, M.: Time Constraints in Workflow Systems. CAiSE (1999) 286-300
11. Hagen, C., Alonso, G.: Beyond the Black Box: Event-based Inter-Process Communication in Process Support Systems. ICDCS (1999) 450-457
12. Heinl, P., Horn, S., Jablonksi, S., Neeb, J., Stein, K., Teschke, M.: A Comprehensive Approach to Flexibility in Workflow Management Systems. WACC (1999) 79-88
13. Jajodia, S., Kerschberg, L. (eds.): Advanced Transaction Models and Architectures. Kluwer (1997)
14. Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., Odgers, B.: Autonomous Agents for Business Process Management. Journal of Applied Artificial Intelligence 14 (2000) 145-189
15. Jeusfeld, M.A., Quix, C., Jarke, M.: Design and Analysis of Quality Information for Data Warehouses. ER (1998) 349-362
16. Kafeza, K., Karlapalem, K: Temporally Constrained Workflows. ICSC (1999) 246-255
17. Klein M., Dellarocas C.: A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. Journal of Computer-Supported Collaborative Work 9 (2000) 399-412
18. Ludwig, H: Termination Handling in Inter-Organisational Workflows - An Exception Management Approach. Workshop on Parallel and Distributed Processing (1999) 122 - 129
19. Müller, R., Heller, B.: A Petri Net-based Model for Knowledge-based Workflows in Distributed Cancer Therapy. EDBT Workflow Workshop (1998): 91-99
20. Marjanovic, O., Orlowska, M.E.: On Modeling and Verification of Temporal Constraints in Production Workflows. Knowledge and Information Systems 1 (1999) 157-192
21. Müller, R., Rahm, E.: Rule-Based Dynamic Modification of Workflows in a Medical Domain. German Database Conference (BTW). Springer (1999). http://dol.uni-leipzig.de
22. Schwalb, E., Dechter, R.: Processing Disjunctions in Temporal Constraint Networks. Artificial Intelligence 93 (1997) 29-61
23. Reichert, M., Dadam, P.: ADEPT$_{FLEX}$ - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Information Systems 10 (1998) 93-129
24. Worah, D., Sheth, A.: Transactions in Transactional Workflows. In [13]: 3-34