Memorial
University of Newfoundland

Computer Science

TECHNICAL REPORT #2009-01

Supporting Process Control in Business Collaborations

by

K. Vidyasankar

In conjunction with:
Samuil Angelov, Jochem Vonk, Paul Grefen
School of Industrial Engineering
Eindhoven University of Technology

Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada A1B 3X5

July 2009

Supporting Process Control in Business Collaborations

Samuil Angelov[1], Krishnamurthy Vidyasankar[2], Jochem Vonk[1], Paul Grefen[1]

[1]School of Industrial Engineering, Eindhoven University of Technology
{s.angelov, j.vonk, p.w.p.j.grefen}@tue.nl
[2]Department of Computer Science, Memorial University of Newfoundland
vidya@mun.ca

Abstract

Business collaborations have become highly dynamic and flexible. "Black-box" services are gradually replaced by services where service providers not only expose the underlying processes but also allow some monitoring of their executions. In this paper, we explore the next step where the consumers can exercise some control over the process execution. We specify a set of control primitives that can be used to exert control on activities before, during and after their execution and on the complete process itself. Our design approach allows the consumer to observe the state of execution of the process and its activities, control their execution and decide how to deal with unsuccessful executions. We describe the support that must be provided in the internal process specification of the service provider for the control options. We illustrate our approach with an example from the healthcare domain.

Table of Contents

1 Introduction

Traditionally, the agreement for business collaboration is specified in a contract. Nowadays, to stay competitive, companies engage in highly dynamic and complex business collaborations. This dynamism and complexity requires improvement of the efficiency and effectiveness of business collaborations, which has led to the usage of information technology to support the contracting process and to the transformation of paper contracts into electronic contracts [1]. E-contracts contain the terms and conditions of the collaboration agreed by the parties in a digital, machine interpretable format. In process intensive collaborations (e.g., service delivery), e-contracts specify explicitly the processes agreed by the parties. Explicit, formal process specification allows dynamic coupling of the systems of providers and consumers for the duration of the collaboration.

With the advances in information technology, process providers started offering the possibility for consumers to monitor the process execution, thereby allowing them to quickly react and adapt their processes to the context [2]. To further improve their services, providers can offer the consumers (limited) control over the execution of the agreed processes. The benefits for the consumer are obtaining a more flexible, potentially more efficient and effective service [3]. For the provider, this opportunity can mean obtaining a competitive advantage over similar services offered by others.

In this paper, we present an approach for supporting process control in process intensive business collaborations. We identify control primitives that are valuable for a process consumer and which a provider can offer. We discuss how these primitives can be supported internally by the provider. The approach is illustrated with an example from the healthcare domain.

2 Background

In business collaborations, in order to allow counterparties to be aware of the activities that will be performed by the party and to monitor the states of these activities during the business collaboration, providers share relevant parts of their processes with consumers. However, private processes should not be directly disclosed as they may reveal company sensitive information or may contain activities that will be irrelevant for the counterparty.

In [4], a three level framework for process specification is proposed, distinguishing external, conceptual, and internal process levels. At the conceptual level, the process specification is technology independent, specifying the process that will be performed by the party. Process specifications at the external level contain the activities that will be disclosed to an external party. At the internal level, the process specification reflects changes to the conceptual process specification that are driven by the specific technology used by the company. In this paper, we abstract from the technological side and consider process specifications defined at the conceptual and external levels.

Activities on the external level are derived by hiding and aggregating activities from the conceptual level. An external activity contains all conceptual activities between the starting conceptual activity and ending conceptual activity for this external activity and each conceptual activity is part of one external activity (see [5] for a detailed description of the rules for deriving an external level process specification based on a conceptual process specification).

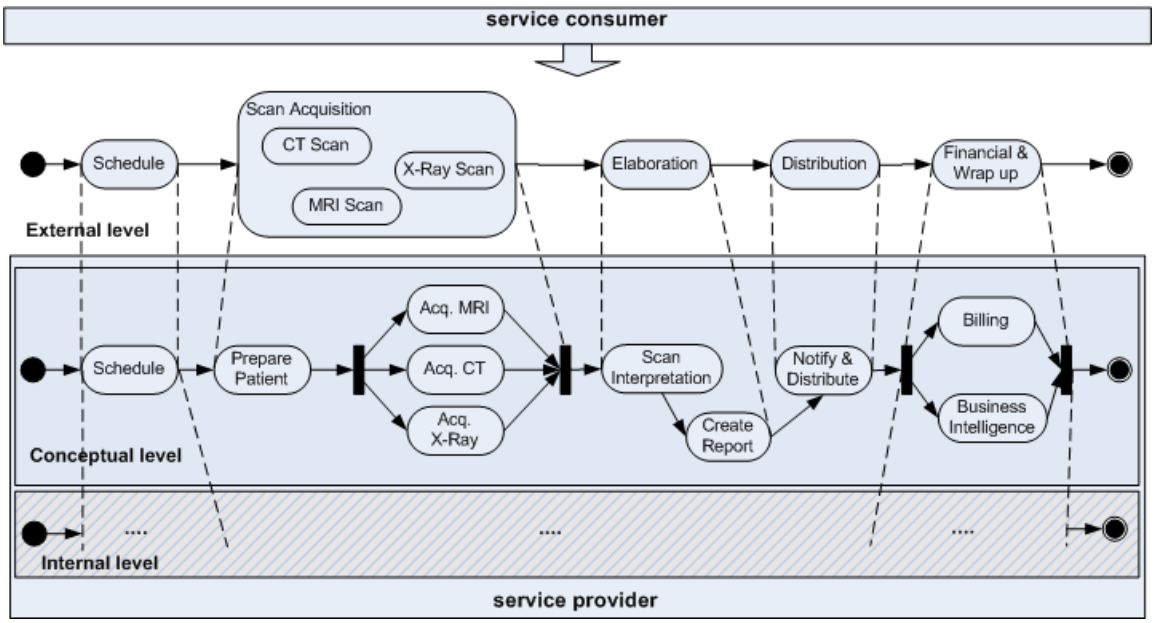

Fig. 1. The teleradiology process

We use a sample case throughout the paper. The case describes a teleradiology process about the acquisition and interpretation of medical scans of patients, and results in a report that a medical specialist, who ordered the scan, can use to base his diagnosis and treatment on. Fig. 1 shows the

simplified teleradiology process (extensively described in [6]). The process starts by scheduling the patient. At the scheduled time, the required scans are acquired, after which an interpretation report is created and distributed to the service client. The process ends after financing has been handled. In Fig. 1, we represent the process at the external and conceptual levels (as already discussed we do not pay attention to the internal level in this paper). The dashed lines represent the mapping of external activities to conceptual activities, e.g., the 'Elaboration' activity actually consists of two conceptual activities.

The three-level framework presented in [4] discusses explicitly intermediate levels of aggregations at the conceptual and internal levels. These intermediate levels can be defined to allow "smoother" transitions between the levels. However, it does not pay explicit attention to the case when several aggregation levels can be defined at the external level. In this work, we extend the three-level framework by introducing different levels of process aggregation at the external level as well. This extension is driven by the need to sometimes provide more detailed information for the activities performed within a highly aggregated external activity. The information allows the service provider to offer limited controls over its process on both coarse-grained activities as well as on finer-grained level. Furthermore, we allow decompositions of a highly aggregated external activity without providing details on the control flow between these activities. This extension is driven by our goal to allow the definition of process controls that can influence the optional execution of activities and their order of execution. For example, in Fig. 1, the "scan acquisition" activity is decomposed into three sub-activities, i.e., CT Scan, MRI Scan, and X-Ray Scan. This lower level of detail gives the consumer additional information on the scanning activities and allows the provider to offer to the consumer controls over them (e.g., pausing some of them) as well as over the aggregated "Scan acquisition" activity. The lack of control flow allows the provision of controls for example for choosing the order of execution of these scans and their optional execution.

3 Specification and Support of Activity Controls

We call an activity at the external level a Visibility point (VP). The service provider supplies information to the service consumer on a push or pull fashion about the states of the VPs during the process execution (called messaging and polling in [7]). The consumer is informed about the start and completion of each VP [7]. In this paper, we discuss the new paradigm in which the process provider may allow the consumer to exert certain control on a VP. We call VPs in which the consumer may exert control Interference Points (IPs). The control primitives that are offered to the process consumer are called Interference options (I-options). Different I-options may be available at different IPs. A request from the service consumer for the exertion of an I-option is called an I-request.

3.1 Specification of I-options

Process control by a service consumer at the external level has been briefly addressed in [3]. This publication was a main source of inspiration for our initial work on the definition of I-options. Additionally, we investigated existing work on business process flexibility [8], [9]. Flexibility of a process indicates the adaptability of a process to changes in the environment during its execution. The changes can be caused for instance by an I-request. After elaborating and extending the results from [3] and adapting the results from [8], [9] to the context of cross-organizational collaborations, we have defined the list of I-options presented in Table 1.

Table 1. List of the I-options

I-option	Comments
Group 1	
START	The execution of an activity is started.
DELAY/PROCEED	Starting execution of an activity is delayed/continued.
SKIP	The execution of a non-started activity is skipped.
Group 2	
PAUSE/CONTINUE	The execution of a started activity is paused/resumed.
CANCEL	The execution of a started activity is terminated. Partial results from the execution of the activity remain.
PART-RESET	The execution of a started activity is stopped and the activity is put back in its ready state without undoing any of the work that has been performed.
PART-UNDO	The execution of a started activity is stopped, what has been done is undone, and the activity is put back in its ready state.
Group 3	
RESET	An activity that has ended is put back in its ready state. Results from previous executions are not undone.
UNDO	An activity that has ended is put back in its ready state, after the results from the previous executions are undone.

In Table 1, we group the I-options in three groups, i.e., I-options that can be invoked before the execution of an activity (group 1), I-options that may be invoked during the execution of an activity (group 2), and I-options that can be invoked after the execution of an activity (group 3).

Each I-option is parameterized upon invocation. A parameter common for all I-options is the activity(s) to which the I-option is applied. Other parameters, e.g., time (for DELAY, PAUSE) may be provided. Based on the basic I-options defined in Table 1, complex I-options (combinations of several I-options) can be defined. Several complex I-option examples are listed in Table 2. The complete list of complex I-options, including a matrix presenting how they are derived, is presented in the appendix.

Table 2. Sample list of complex I-options

Complex I-options	**Constituent I-options**	**Comments**
POSTPONE	DELAY+PROCEED	The execution of an activity is postponed.
RESTART	PART-RESET+START	A started activity is stopped and is started from the beginning.
PART-REDO	PART-UNDO+START	A started activity is stopped, undone, and started again.
TERMINATE	PART-UNDO+SKIP	A started activity is stopped and undone. The control flow is passed to the next activity.
RETRY	RESET+START	An ended activity is started from the beginning.
REDO	UNDO+START	An ended activity is undone and started again.

An I-request leads to a change in the state of a VP. In Fig. 2, we present the state model of a VP and the I-options that trigger the state changes. The model serves three purposes. First, we use it to clarify the I-options and illustrate their impact on the activity states. Second, we use it for the definition of the requirements on the support of the I-options at the conceptual level (see Section 3.2). Third, the model will serve as a main tool for the definition and system support of I-options (see Section 5).

A state change of a VP caused by an I-request should be reflected with the corresponding state change in the conceptual process (e.g., when a VP is paused, the corresponding conceptual activity(s) should be paused as well). However, the execution of an I-request at the conceptual level may require time. Thus, an activity at the external level must have states that represent these times of transition assuring consistency between the external and conceptual process levels. We call these transition states "ING" states, e.g., PAUSING, and show them in grey in Fig. 2. The transitions that do not have an “ING” state take place synchronously at the external and conceptual levels (see Section 3.2).

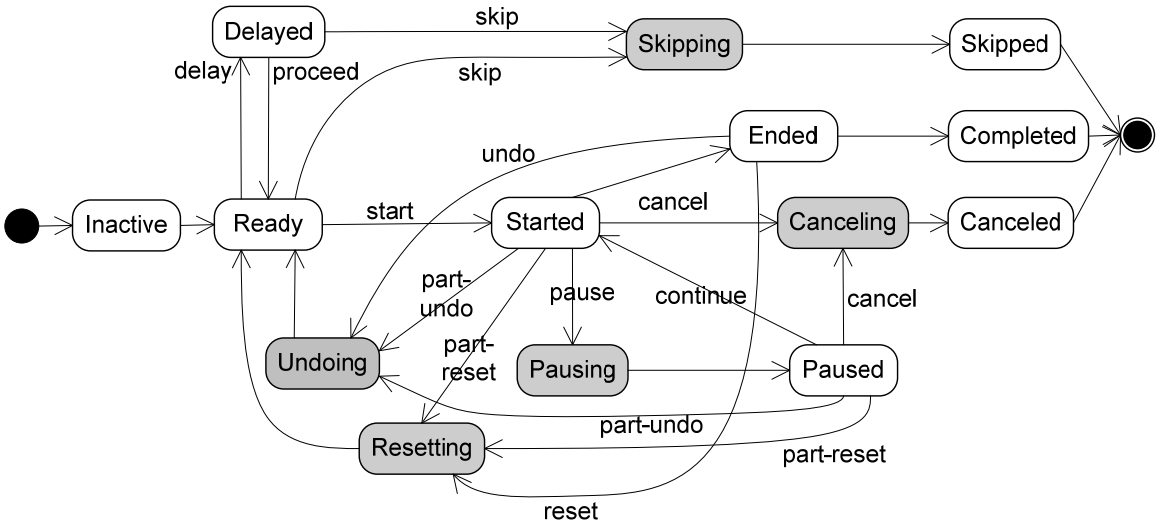

Fig. 2. The state model for the visibility points

3.2 Support of the I-options at the Conceptual Level

I-options require appropriate support at the conceptual level. The complexity of the solution for the I-options support depends on the activity state model that is used at the conceptual level. The support for an I-option will be trivial if each conceptual activity implements a state model that supports all non-“ING” states as shown in Fig. 2. However, typically, the activity state model supported at the conceptual level would be more limited (see e.g., [10]), which complicates the support for the I-options. Currently, no commonly agreed upon activity state model exists, so we use the most limited activity state model, which considers a conceptual activity as an isolated (i.e., non-interruptible) activity. Thus, a conceptual activity has three states, i.e., INACTIVE, STARTED, COMPLETED (see Fig. 3).

Fig. 3. The state model for a conceptual activity

Our approach for defining the support at the conceptual level in the case of isolated activities is based on the introduction of “wait” states for the support of I-options. A “wait” state has two functions. First, it is used to provide time to the consumer to invoke an I-option (as transitions between activities occur instantaneously). Second, it is used as an “activity” that “pauses” the conceptual level process execution (as conceptual activities in our state model are non-interruptible). Direct introduction of a “wait” state in the restricted state model that we have accepted for conceptual activities is not possible as in this model an activity has no intermediate states between its “inactive” and “completed” states which are required by a “wait” state (e.g., to handle input providing information on when to leave the wait state). However, existing activity state models (e.g., [12], [11],

[10]) provide limited or highly-sophisticated intermediate activity states that allow the introduction and support of a "wait" state. That is why we can introduce the wait state to the restricted state model of conceptual activities and assume that the underlying system will be able to support it. Its support will vary depending on the concrete choices made on the conceptual activity state model in the underlying system.

To explain the support for the I-options, we use x_i to denote the i^{th} activity at the external level, and $c_{i1},\ldots,c_{in}$ to denote the first and the last conceptual activities in the segment of activities mapped to x_i. Note that in case of parallel execution of several first (last) activities, the first (last) conceptual activity c_{i1} (c_{in}) represents a set of concurrently executing activities. The support of complex I-options at the conceptual level is not discussed as it can be directly derived on the basis of the support defined for the I-option primitives.

Group 1 I-options

For each I-option from group 1, a "wait" state preceding c_{i1} must be introduced to give the consumer some time to exert the I-option. This wait state represents the READY state of x_i.

DELAY(x_i), PROCEED(x_i), START(x_i): The "wait" state that is introduced (see Fig. 4) is automatically entered when x_i is ready to be executed and gives the consumer some time to exert the DELAY control. If DELAY is requested, the time of the wait state is extended until a PROCEED request arrives (or a time-out in case no PROCEED is requested, to avoid a deadlock situation). Otherwise, the execution proceeds with the execution of c_{i1}. If a START I-option is provided, the "wait" state is left only when START I-request arrives.

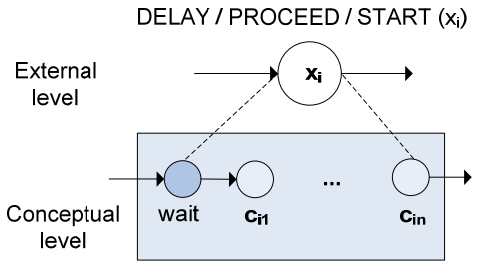

Fig. 4. Support of "DELAY"/ "START"

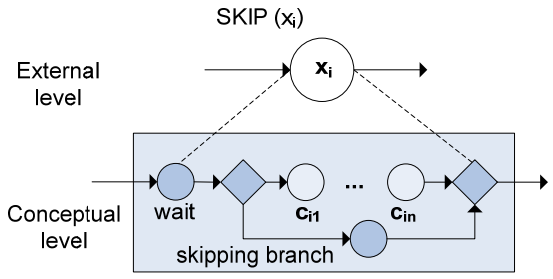

Fig. 5. Support of "SKIP"

SKIP(x_i): The invocation of this I-option requires in addition to the wait state, the introduction of a "split" construct before c_{i1} and a "merge" construct after c_{in} (the WCP-4 and WCP-5 patterns[1]). During the transition period, x_i is in state SKIPPING. The transition can be direct or certain activities might have to be executed during the skipping (see Fig. 5).

[1] In [13], workflow control patterns are presented under the abbreviation WCP, followed by their number.

Group 2 I-options

PAUSE(x_i), CONTINUE(x_i): These I-options require the introduction of a "wait" state at one or more places in the conceptual process model, in which the process can be paused (see Fig. 6). The external activity is in state PAUSING until this state is reached. Note that to avoid process execution deadlocks, it should be guaranteed the 'Paused' state (caused by the PAUSE I-option) leads, at some point in time, to one of the other possible states.

CANCEL(x_i): The invocation of this I-option requires the implementation of a cancellation construct on the conceptual level (WCP-19). The cancellation construct can be provided at several points between c_{i1} ... c_{in} allowing several points for internal reaction to a CANCEL (see Fig. 7). During a cancellation, x_i is in state CANCELLING.

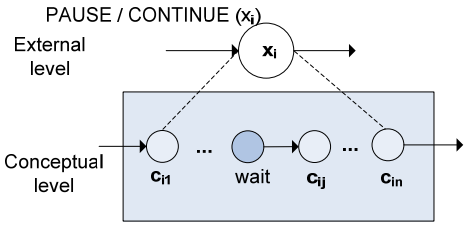

Fig. 6. Support of "PAUSE"

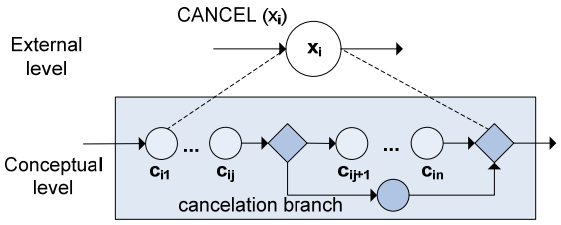

Fig. 7. Support of "CANCEL"

PART-RESET(x_i): Similar to the cancel I-option, a "split" is necessary at the conceptual level to "implement" this I-option (see Fig. 8). The control flow after the reset is passed to c_{i1}. During the transition between the started and ready states, activity x_i is in RESETTING state. During the resetting one or more internal activities required for the reset may be executed (their optional character is depicted by presenting the activity in Fig. 8 with a dashed line).

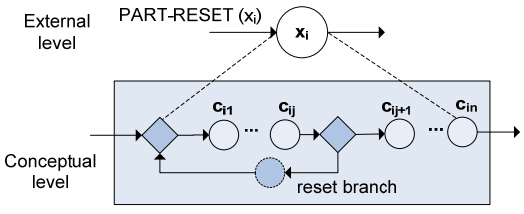

Fig. 8. Support of "PART-RESET"

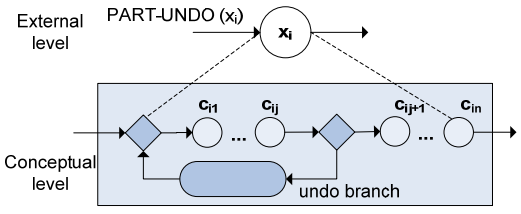

Fig. 9. Support of "PART-UNDO"

PART-UNDO(x_i): Two approaches can be used to support the (part-)undo I-option. The first solution is comparable to the handling of the reset I-option and consists of explicit conceptual level undo point(s) at which the undo will be handled by executing activities that will undo the work done in the activities that have been executed. We use an oval in Fig. 9 to depict the potentially more complex nature of an undo activity. This is a very rigid way of handling the undo and requires pre-

specifying each undo-branch for each conceptual level undo point (which becomes complex quickly in the case of choices or loops in the process).

The second approach makes use of transaction management support (see, e.g., [14], [15]). Two well-known transaction mechanisms are atomicity [16] and compensation [17]. If the entire conceptual level subprocess (corresponding to the external activity) is designated as being atomic, this means that either all activities (not the ones excluded through choices in the process execution) will be performed, or none at all. This is ensured by the transaction support. For the undo I-option, this implies that the conceptual process can simply be aborted and the transaction support undoes what has been done. Compensation on the other hand is used to semantically undo the work that has been done. At the time that an undo I-option is exerted, the transaction support will analyze what has been executed, construct a new process model consisting of activities that undo the work done in the original activities, and have that process model executed by the process engine, after which control is passed back from the transaction engine to the process engine, which will find that conceptual process is in the ready state again (and so the external activity must be placed in the ready state as well, following the undoing state).

Group 3 I-options

Similar to the I-options from Group 1, for each I-option from group 3, a "wait" state after c_{in} must be introduced to give the consumer some time to exert the I-option. This wait state represents the ENDED state of x_i.

RESET(x_i): To support the invocation of the RESET I-option, a loop construct around c_{i1}, .., c_{in} has to be defined (WPC-21). The loop construct is preceded by a "wait" state (see Fig. 10).

UNDO(x_i): To support the invocation of the UNDO I-option, a construct analogous to the construct for the RESET I-option is required. In contrast to the reset branch, the undoing branch contains one or more undoing activities (see Fig. 11) that are pre-specified, or the undoing can be taken care of by a transaction engine as explained previously for the part-undo I-option.

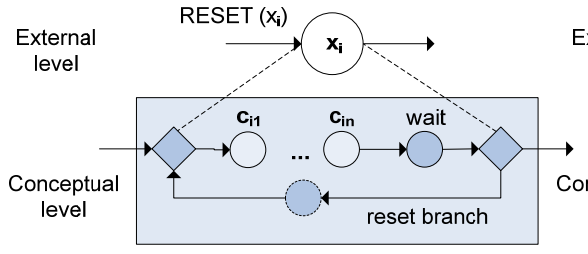

Fig. 10. Support of "RESET"

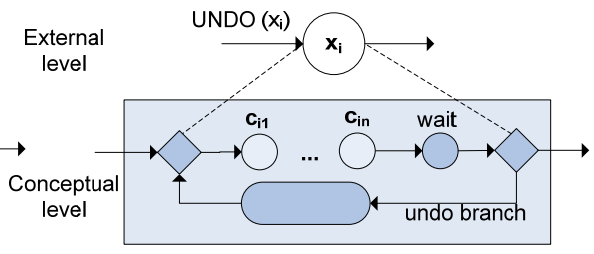

Fig. 11. Support of "UNDO"

In cases of parallelism at the conceptual level, the I-option will be effectuated when each parallel running branch reaches a state that supports the I-option. If such a state cannot be reached the I-option

cannot be carried out. An example for the case of a PAUSE I-option is shown in Fig. 12. In this example, two possible "cuts" through the parallel conceptual branches exist (a "cut" is formed by the line passing through the wait states of each branch). A PAUSE can be effectuated in each of the two "cuts" shown in this example.

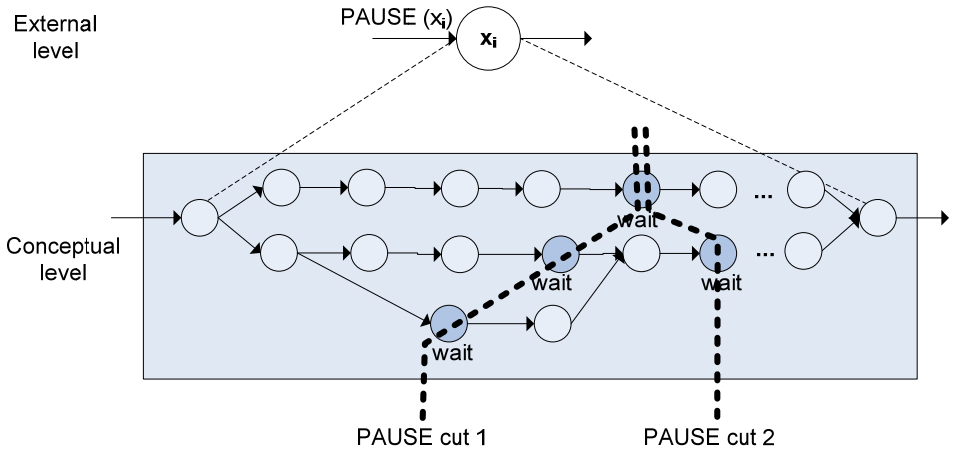

Fig. 12. Support of "PAUSE" in case of parallelism

3.3 Examples for I-options

In this section, we illustrate our approach using our running case as an example (see Section 2). As can be seen in Fig. 13, three I-options are specified for the teleradiology service: RETRY and DELAY/PROCEED. The consumer receives a copy of the scan after it has been made. The 'retry' I-option for the 'X-Ray Scan' activity allows consumers who are not satisfied with the result of the X-ray scan acquisition to request the X-ray scan to be taken again.

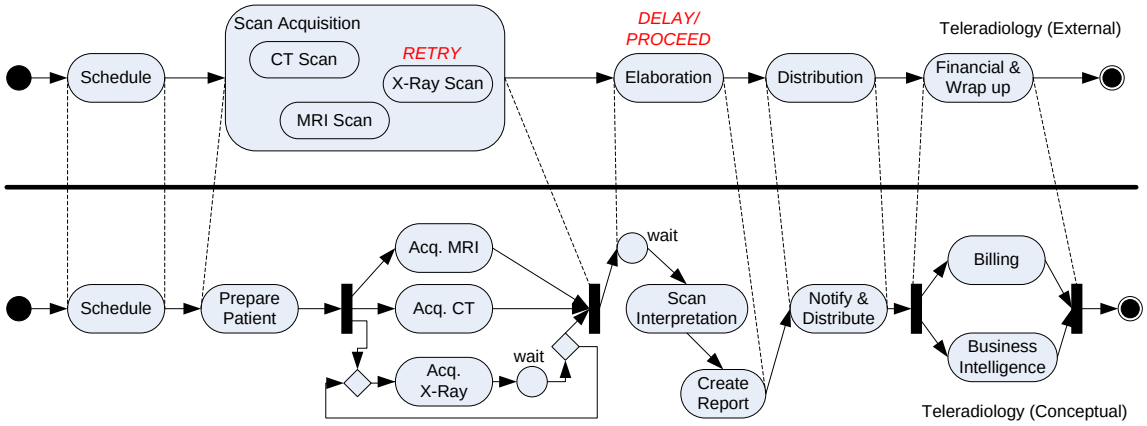

Fig. 13. I-options in the Teleradiology example

The DELAY/PROCEED I-options allow the consumer to delay (if desired) the elaboration of the report until the load at the service provider becomes low, resulting in smaller charges for the service.

Of course if the load is low (and hence price is low) by the time of the start of the activity or the case is urgent, the consumer would not exert this control. The financial consequences of exerting an I-option under the different conditions should be included in the e-contract.

4 Specification and Support of Higher-Level Controls

In this section, we look into control primitives that can be defined and invoked on sets of VPs. Process I-options (PI-options) are control primitives that can be defined over a complete process, i.e., all VPs within one external process. Segment I-options (SI-options) are control primitives that can be defined over a part of a complete external process specification.

4.1 Specification of PI-options

The process concept can be seen as an aggregation of the activities at the external level to a single high-level activity at the external level. Consequently, a PI-option can be viewed as an I-option defined on an activity (the process). That is why the set of PI-options is equivalent to the set of I-options that we have defined in Section 3.1. Furthermore, the state model of a "process" activity is the same as for a VP (see Fig. 2). In Table 3, we list the PI-options and explain their semantics at a process level. Note that also complex PI-options exist, similar to the complex I-options (see Table 4).

Table 3. List of the PI-options

PI-option	Comments
Group 1	
START	The execution of a process is started.
DELAY/PROCEED	Starting execution of a process is delayed/continued.
SKIP	The execution of a non-started process is skipped.
Group 2	
PAUSE/CONTINUE	The execution of a started process is paused/resumed.
CANCEL	The execution of a started process is terminated. Partial results from the execution of the process remain.
PART-RESET	The execution of a started process is stopped and the process is put back in its ready state without undoing any of the work that has been performed.
PART-UNDO	The execution of a started process is stopped, what has been done is undone, and the activity is put back in its ready state.
Group 3	
RESET	A process that has ended is put back in its ready state. Results from previous executions are not undone.
UNDO	A process that has ended is put back in its ready state, after the results from the previous executions are undone.

4.2 Support of the PI-options at the Conceptual Level

As PI-options are I-options that are defined over a "process" activity, the requirements for their support at the conceptual level are analogous to those for the I-options. The equivalence of the support

for the PI- and I-options at the conceptual level means that any PI-option can be represented as one or several I-options.

In the case of a sequential external process specification, Group 1 and Group 3 Pi-options can be defined by their respective I-option equivalences. In the example presented in Fig. 14, the DELAY(X) PI-option can be represented as a DELAY(X1) I-option. Group 2 PI-options can be represented as one or more I-options, as there can be multiple points in the conceptual process specification where support for a PI-option is provided. In the example presented in Fig. 14, the PAUSE (X) PI-option can be represented as the PAUSE(X2) and PAUSE(X3) I-options (as there are two constructs at the conceptual level that support PAUSE (X)).

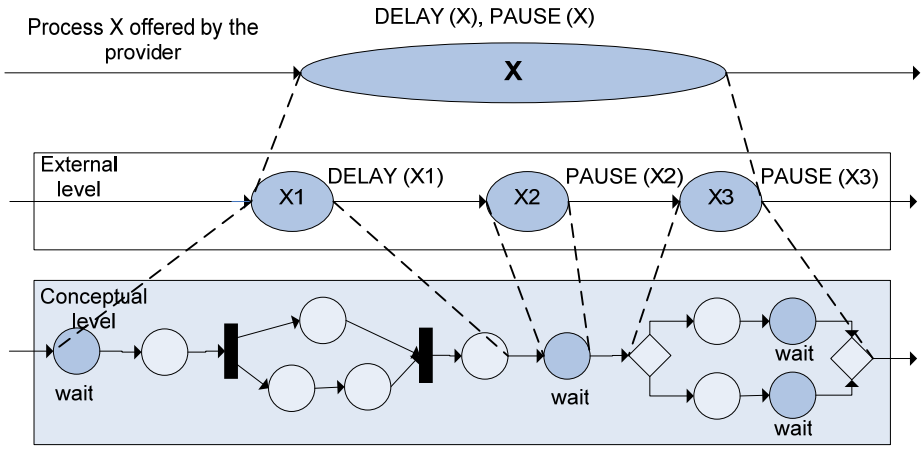

Fig. 14. Relationship between PI- and I-options in the case of a sequential external process

In the case of a PI-option that can be invoked during the parallel executions of branches at the external level (see Fig. 15), the PI-option is equivalent to the *simultaneous* invocation of the respective I-options in each of the parallel branches. In the example in Fig. 15, invocation of PAUSE(X) is equivalent to the simultaneous invocation PAUSE(X2) and PAUSE(X3) I-options. The definition of the PAUSE(X2) and PAUSE(X3) I-options cannot be equated to the PAUSE(X) PI-option as the invocation of only one of the I-options (e.g., PAUSE(X2)) will lead to the execution of the other VP (i.e., X3), while the invocation of PAUSE (X) will pause both X2 and X3.

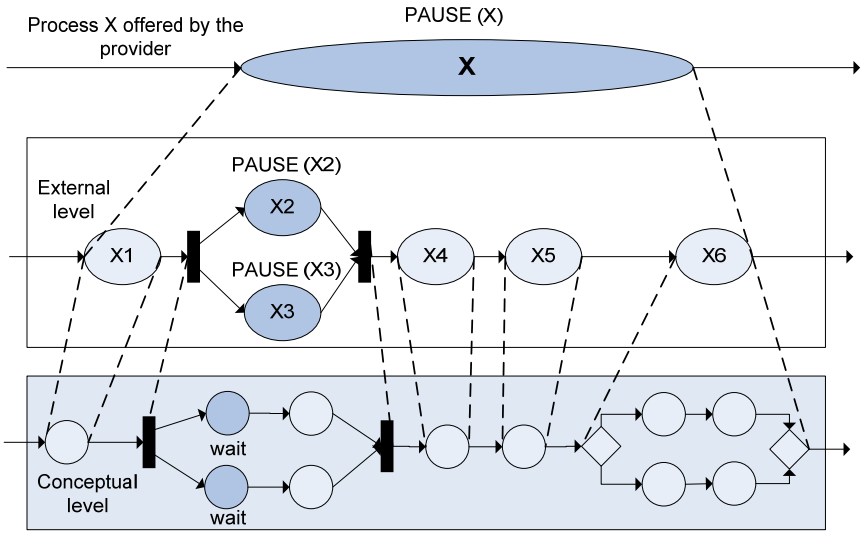

Fig. 15. Relationship between PI- and I-options in the case of parallelism

Several variations for an advanced support may be considered for PI-options. We illustrate this possibility with the PAUSE/CONTINUE PI-options in the Teleradiology process (see Fig. 1). We will assume that a "wait" state is provided in the conceptual level before the "Prepare Patient" activity. If a PAUSE request comes before the execution of "Scan Acquisition" started the execution of the process will pause there. Let us assume that the consumer does not issue a "CONTINUE" within a "reasonable" time (affecting the scheduling of the patient fixed in the preceding activity). Then the provider may undo the "Schedule" activity, restarting the process. This example indicates that a PAUSE PI-request may necessitate a partial backward recovery of the execution in the conceptual level. Such advanced interpretations of a PI-option that deviate from their standard definition should be explicitly stated in the contract between the provider and the consumer, guaranteeing clarity on the effects of the invocation of a PI-option on the consumer side.

4.3 Examples for PI-options

Referring to the Teleradiology example, a PART-RESET PI-option can be offered in the event that scheduling is not satisfactory (e.g., the dates scheduled for the scans are too late for the needs of the consumer). Invocation of this PI-option would enable starting the process all over (see Fig. 16). A CANCEL PI-option can be offered to terminate the entire process if the customer believes that no acceptable scheduling can be achieved and the process should be terminated.

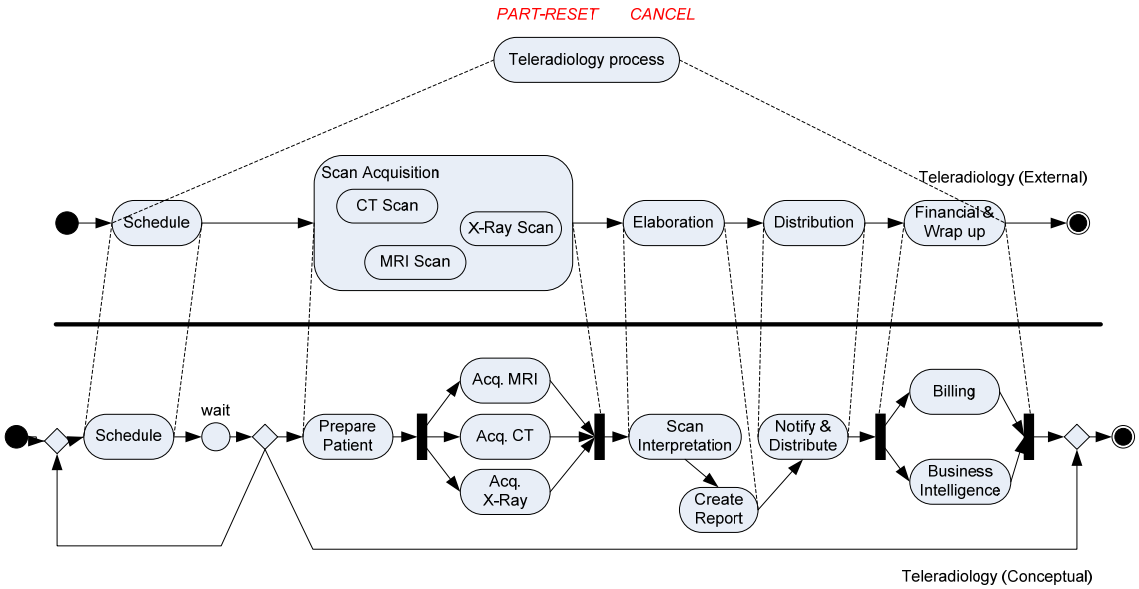

Fig. 16. PI-options in the Teleradiology example

4.4 Specification of SI-options

We call a set of external activities over which a control primitive is defined a “segment” and the control primitive a SI-option. Analogously to I- and PI-options, we distinguish three groups of SI-options. We do not define equivalent SI-options for all the I-options as certain I-options affect only a single conceptual activity and hence do not have semantics on a segment level. That is why we do not define SI-options like PAUSE, START, etc. However, new control primitives that operate over sets of activities can be defined (i.e., CHOICE and ORDER). In Table 4, we list the set of SI-options that we have identified and provide explanations on their semantic.

We note that SI-options like CANCEL, PART-RESET, PART-UNDO, RESET, UNDO allow the user to request partial forward recovery or partial backward recovery of the process execution.

Table 4. List of the SI-options

SI-option	Comments
Group 1	
SKIP	The execution of a non-started segment is skipped.
CHOICE	From the set of activities in the segment, the consumer chooses an activity(s) that will be executed.
ORDER	The consumer selects the order of execution of the activities in the segment.
Group 2	
CANCEL	The execution of a started segment is terminated. Partial results from the execution of the segment remain.
PART-RESET	The execution of a started segment is stopped and is put back in its ready state without undoing any of the work that has been performed.
PART-UNDO	The execution of a started segment is stopped, what has been done is undone, and the segment is put back in its ready state.
Group 3	
RESET	An ended segment is put back in its ready state without undoing any of the work that has been performed.
UNDO	An ended segment is undone, and the segment is returned to its ready state.

In Fig. 17, we present the state model of a segment. The model differs from the state model of a VP in the new controls that are introduced for a segment and in the controls that are removed due to their irrelevance in the context of segments.

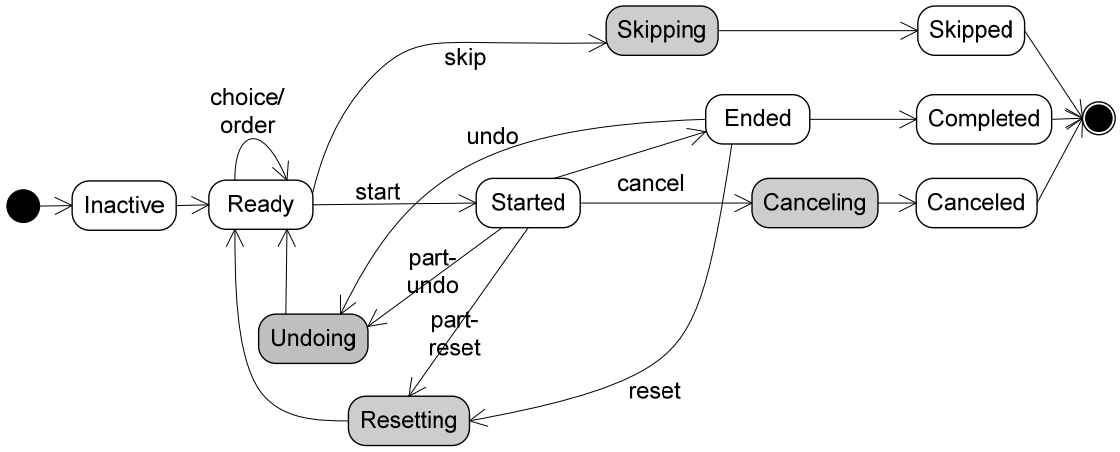

Fig. 17. The state model for a segment

Different sets of the SI-options can be defined over different segments in a process. To avoid the definition of SI-options that lead to non-deterministic process states we only allow SI-options that do not cross over an AND or OR splits and their respective joins (AND and OR joins). The reason to

define this constraint is that SI-options are affecting sequences of activities. When the effects of a SI-option cross over a split or a join construct the behaviour expected from the other branches running in parallel is unclear. For example, as illustrated in Fig. 18, skipping a branch and some activities following the merge construct (SKIP (x3, x4)) leaves open the question of whether activity x2 should still be executed and if it is executed whether x4 has to be executed after its completion. The problem is analogous to the problem occurring in the case of using "non-symmetrical" process constructs (AND split followed by an OR join) where, because of inconsistent modelling, multiple execution paths over an activity following a join occur [17].

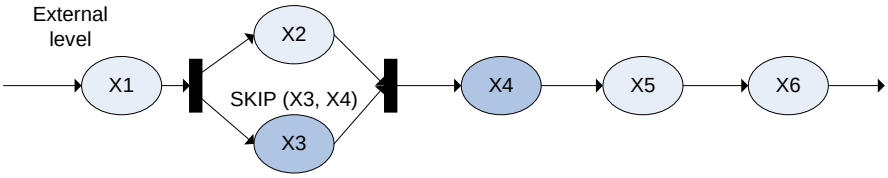

Fig. 18. An example for disallowed SI-option

4.5 Support of the SI-options at the Conceptual Level

A segment can be represented as an aggregated external activity and a SI-option defined over a segment can be seen as an I-option defined over the aggregated activity. Thus, we can use the results for the support required at the conceptual level for the corresponding I-option. The invocation of the CHOICE(x_i,...,x_j) and ORDER(x_i,...,x_j) SI-options requires the implementation of an exclusive choice construct (WCP-4 and WCP-6) or of a partial- or free-order construct (WCP-17 and WCP-40), respectively, see [13] for an explanation of these constructs.

4.6 Examples for SI-options

In this example, shown in Fig. 19, the ORDER and RESET SI-options are defined on the Scan acquisition segment. Using the ORDER SI-option, the consumer can state the order of their execution. The conceptual process allows for the three activities to be performed in any order defined with the I-request. The RESET SI-option allows the consumer to perform all scanning activities again (e.g., if there was a change in the patient's condition during the scanning). The SKIP SI-option is defined over the Reporting segment (an aggregated external activity consisting of the Elaboration and Distribution external activities). It allows the consumer to skip the reporting process in case the report is not needed any more.

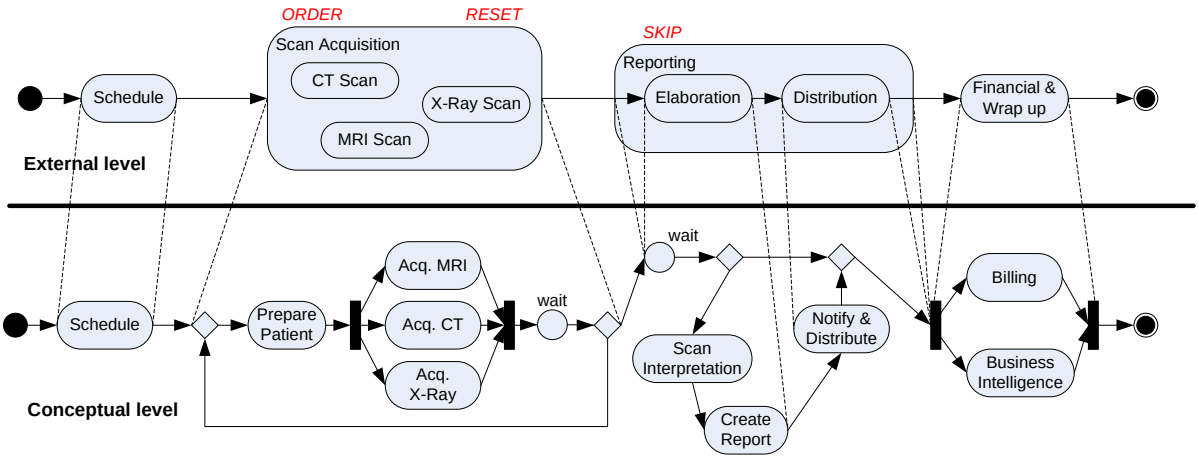

Fig. 19. SI-options in the Teleradiology example

5 Specification and System Support of I-, PI-, and SI-options

A process designer first defines the conceptual process specification. After applying aggregation and customization techniques [5] the external specification is derived. Based on the company policy, the process designer will define a set of I-options for the external level and if necessary will make adaptations on the conceptual specification that guarantee the support of the I-options.

A consumer may invoke an I-option whenever it is available. Several, mutually exclusive I-options can be available for the same activity (e.g., SKIP(x_i) and DELAY(x_i)). Different approaches to handle multiple, potentially inconsistent I-requests are possible: allowing multiple invocations and checking them for consistency (queuing of invocations), accepting one invocation and ignoring subsequent invocations, etc. The system providing control to the consumer should handle this. Our approach to the support for handling of I-requests is that I-options (PI-options, SI-options) can be invoked only sequentially. That is, after an I-request, a response should be received before issuing another I-request. This enables checking consistency of the I-requests in a simple manner. We suggest using the state model for the visibility points, given in Fig. 2, for consistency checks. The state model defines the possible I-options that can be offered by the provider for each possible state of a visibility point. At any state, only the options allowed in that state from the complete set of I-options initially defined by the provider can be offered to the consumer. Sequential execution of I-requests enables transitions in the diagram to occur deterministically. The approach for consistency checks for SI- and PI-options is analogous.

Constraints similar to those for the normal workflow execution may also be specified in I-requests. For example, a DELAY can be requested for a certain fixed period of time, until a particular agent for the activity (a particular technician for MRI-scan) is available, etc. Responsibility for ensuring these constraints can be delegated to the workflow execution engine itself.

6 Related Work

Control over a process in a service was initially explored in the CrossFlow project [3]. The CrossFlow approach was, however, rather ad hoc and a limited set of controls was devised. Mapping control primitives from the contract-level process to the internal process was determined by the possibilities of the underlying WfMS. Part of the work in process flexibility is concerned with controlling process executions [8], [9]. However, it is only focused on intra-organizational processes. In this paper, relevant flexibility possibilities have been extended and adapted for the collaborative settings.

E-contracting research deals with the process of establishment and enactment of electronic contracts and determining their content to support digital collaborations between parties [1], [2], [19]. The work presented in this paper complements the research in e-contracting by extending e-contracts with the allowed I-options.

Collaboration types range from simple service outsourcing to forming complex, dynamic business networks. Web services are a de facto standard to offer services to parties [20], [21]. Web services, however, are black-box services that do not expose the process contained within. In [22], an extension is presented, which ‘opens up’ web services through different interfaces, one of which is the control interface to allow for process control to the service consumer. However, details on how to effectuate this control are not given.

7 Conclusions and Future Work

In this paper, we define a set of interference options, called I-options that can be given to a consumer organization to exert control over the process performed by a provider organization in the business relationship. To support these I-options on the internal process, we present a mapping from the external process to the private conceptual process. The approach is illustrated with a real-life scenario from the healthcare domain. The mapping from conceptual to internal process specification is system specific and is therefore considered outside the scope of this paper. Through the I-options offered, a provider organization has an additional mechanism to distinguish its services from those of other service providers. For a service consumer, I-requests offer an increased flexibility in service executions.

Analogous to process execution control, a service provider may offer control over the usage and production of resources at activities. Example resource controls are selection of resources and selection of resource parameters (resource time allocation, resource quality). We see this as an interesting direction for future work.

References

[1] S. Angelov and P. Grefen, "The business case for B2B e-contracting," in Janssen, M., Sol, H. G., and Wagenaar, R. W. (eds.) *Proceedings of the 6th international conference on Electronic commerce, Delft, The Netherlands, 25-27 October* New York: ACM Press, 2004, pp. 31-40.

[2] L. Xu, "Monitoring multi-party contracts for e-business." University of Tilburg, 2004.

[3] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: cross-organizational workflow management in dynamic virtual enterprises," *International Journal of Computer Systems Science and Engineering*, vol. 15, no. 5, pp. 277-290, 2000.

[4] P. Grefen, H. Ludwig, and S. Angelov, "A three-level framework for process and data management of complex e-services," *International Journal of Cooperative Information Systems*, vol. 12, no. 4, pp. 487-531, 2003.

[5] R. Eshuis and P. Grefen, "Constructing customized process views," *Data & Knowledge Engineering*, vol. 64, no. 2, pp. 419-438, 2008.

[6] J. Vonk, T. Wang, P. Grefen and M. Swennenhuis. An Analysis of Contractual and Transactional Aspects of a Teleradiology Process. WP 263. 2008. Eindhoven, Eindhoven University of Technology. Beta Working Papers.

[7] A. Norta, "Exploring Dynamic Inter-Organizational Business Process Collaboration." Technische Universiteit Eindhoven, 2007.

[8] M. H. Schonenberg, R. S. Mans, N. Russel, N. Mulyar, and W. M. P. v. d. Aalst, "Process Flexibility: A Survey of Contemporary Approaches," in Dietz, J. L. G., Albani, A., and Barjis, J. (eds.) *Advances in Enterprise Engineering I* Berlin Heidelberg: Springer, 2008, pp. 16-30.

[9] N. Mulyar, "Patterns for Process-Aware Information Systems: An Approach Based on Colored Petri Nets." Technische Universiteit Eindhoven, 2009.

[10] D. Hollingsworth. The Workflow Reference Model. TC00-1003. 1995. Workflow Management Coalition. Workflow Management Coalition Documents.

[11] N. Russel, W. M. P. v. d. Aalst, A. Hofstede and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. Pastor, O. and Falcao e Cunha, J. Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05). 3520, 216-232. 2005. Berlin, Springer-Verlag. Lecture Notes in Computer Science.

[12] IBM Corporation; IBM WebSphere Process Server Documentation; 2009. Available at: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.612.doc/doc/bpc/cactivity_state.html

[13] N. Russel, A. Hofstede, W. M. P. v. d. Aalst and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM-06-22. 2006. BPM Center. BPM Center Reports.

[14] P. Grefen and J. Vonk, "A taxonomy of transactional workflow support," *International Journal of Cooperative Information Systems*, vol. 15, no. 1, 2006.

[15] S. Jajodia, L. Kerschberg (eds.); Advanced Transaction Models and Architectures Dordrecht: Kluwer, 1997.

[16] J. Gray, "Notes on Data Base Operating Systems," *Operating Systems, An Advanced Course* London: Springer-Verlag, 1978, pp. 393-481.

[17] H. Garcia-Molina and K. Salem, "Sagas," *SIGMOD*, vol. 16, no. 3, pp. 249-259, 1987.

[18] W. M. P. v. d. Aalst, "The Application of Petri Nets to Workflow Management," *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21-66, 1998.

[19] Z. Milosevic and R. Dromey, "On expressing and monitoring behaviour in contracts," *Proceedings of the 6th International Enterprise Distributed Object Computing Conference* IEEE Computer Society Washington, DC, USA, 2002, pp. 3-14.

[20] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Application* Springer, 2004.

[21] M. Papazoglou, *Web Services: Principles and Technology* Prentice Hall, 2007.

[22] P. Grefen, H. Ludwig, A. Dan, and S. Angelov, "An analysis of web services support for dynamic business process outsourcing," *Information and Software Technology*, vol. 48, no. 11, pp. 1115-1134, 2006.

Appendix A

The complex I-options are formed as combinations of two basic I-options. In Section 3.1 a number of example complex I-options are given. The complete set of available complex I-options is listed in the table below. They are identified through a thorough analysis of the possible combinations as shown in Fig. 20. Basic I-options on the horizontal axis are followed by I-options on the vertical axis to form a complex I-option.

As we have identified 11 basic I-options, a total number of 121 combinations can be formed. However, most of these combinations are impossible according to the state transition diagram shown in Fig. 2. For example, a start I-option leads to the 'started' state at which point another start I-option is not possible. Analyzing each possible combination resulted in a total number of 92 impossible combinations, indicated with an 'x' in Fig. 20. Thus, 29 combinations are possible according to the state transition diagram.

Analyzing the 29 possible combinations revealed that a number of them do not make sense, i.e. they are not useable. These are indicated as '*senseless*' in the figure. For example, it does not make sense to issue a pause immediately followed by a cancel as the cancel could have been issued directly (the pause immediately preceding the cancel is therefore useless). As there are 14 combinations that are considered senseless, a total number of 15 complex I-options remain and these are listed in the table below.

Note that, analogous to Section 4, an equivalent reasoning, as given here for the complex I-options on activities, is also applicable for the process level resulting in complex PI-options. Only the last four complex I-options listed in the table have an equivalent SI-option, i.e., ABOLISH, TERMINATE, NULL, and NULLIFY. As can be seen in the table, the other complex I-options involve basic I-options that are not available as SI-options (DELAY, PROCEED, START, PAUSE, and CONTINUE).

A number of observations can be made from the complex I-options matrix shown in Fig. 20 (recall that group 1 I-options are applicable before execution starts, group 2 I-options are applicable during execution, and group 3 I-options are applicable after execution has finished):

- A complex I-option cannot have a group 3 I-option (RESET and UNDO) as the second basic I-option. The reason is that no basic I-option exists that can bring the activity into the 'Ended' state, and thus it is not possible to combine a group 3 I-option after any other I-option.
- Only a group 1 I-option can serve as the second I-option in case the first I-option is part of group 3. Because a group 3 I-option returns the activity to the 'Ready' state (thus before execution has started), it is only possible to combine it with a group 1 I-option.

- The SKIP and CANCEL I-options cannot be followed by any other I-option. As the SKIP and CANCEL both lead to dedicated ‘end-states’ in the state transition diagram from which no other state can be reached, it is not possible to combine them with any other I-option.
- Although, according to the state transition diagram, the START I-option can be followed by a subset of the group 2 I-options, none of them make sense.

Table 5. Complete list of complex I-options

Complex I-options	Constituent I-options	Comments
POSTPONE	DELAY+PROCEED	The execution of an activity is postponed. The time taken between the delay and proceed should be explicitly given in the I-option.
LAUNCH	PROCEED + START	The execution of a delayed activity is started.
DEFER	PAUSE + CONTINUE	The execution of an activity is stopped for a specific amount of time (which needs to be included in the I-option), after which execution is continued.
PART-RESET-RETHINK	PART-RESET + DELAY	A started activity is stopped and without undoing the performed work the task is put in the delayed state (to rethink what should happen).
PART-UNDO-RETHINK	PART-UNDO + DELAY	A started activity is stopped and the performed work is undone, after which the task is put in the delayed state (to rethink what should happen).
RESET-RETHINK	RESET + DELAY	An ended activity is stopped and without undoing the performed work the task is put in the delayed state (to rethink what should happen).
UNDO-RETHINK	UNDO + DELAY	An ended activity is stopped and the performed work is undone, after which the task is put in the delayed state (to rethink what should happen).
RESTART	PART-RESET+START	A started activity is stopped and is started from the beginning.
PART-REDO	PART-UNDO+START	A started activity is stopped, undone, and started again.
RETRY	RESET+START	An ended activity is started from the beginning.
REDO	UNDO+START	An ended activity is undone and started again.
ABOLISH	PART-RESET + SKIP	A started activity is stopped and the work performed is kept. The control flow is passed to the next activity.
TERMINATE	PART-UNDO+SKIP	A started activity is stopped and undone. The control flow is passed to the next activity.
NULL	RESET + SKIP	An ended activity is put back in the ready state (keeping the work that has been done) and the control flow is passed to the next activity. In effect, the activity is placed in the skipped state instead of the completed state.
NULLIFY	UNDO + SKIP	An ended activity is undone and the control flow is passed to the next activity.

First \ Second		Delay	Proceed	Start	Skip	Pause	Continue	Cancel	Part-Reset	Part-Undo	Reset	Undo
Group 1	Delay	x	Postpone (Delay->Proceed) needs explicit condition	x	*senseless*	x	x	x	x	x	x	x
	Proceed	*senseless*	x	Launch (Proceed->Start)	*senseless*	x	x	x	x	x	x	x
	Start	x	x	x	x	*senseless*	x	*senseless*	*senseless*	*senseless*	x	x
	Skip	x	x	x	x	x	x	x	x	x	x	x
Group 2	Pause	x	x	x	x	x	Defer (Pause->Continue) needs explicit condition	*senseless*	*senseless*	*senseless*	x	x
	Continue	x	x	x	x	*senseless*	x	*senseless*	*senseless*	*senseless*	x	x
	Cancel	x	x	x	x	x	x	x	x	x	x	x
	Part-Reset	Part-Reset-Rethink (Part-Reset->Delay)	x	Restart (Part-Reset->Start)	Abolish (Part-Reset->Skip)	x	x	x	x	x	x	x
	Part-Undo	Part-Undo-Rethink (Part-Undo->Delay)	x	Part-Redo (Part-Undo->Start)	Terminate (Undo->Skip)	x	x	x	x	x	x	x
Group 3	Reset	Reset-Rethink (Reset->Delay)	x	Retry (Reset->Start)	Null (Reset->Skip)	x	x	x	x	x	x	x
	Undo	Undo-Rethink (Undo->Delay)	x	Redo (Undo->Start)	Nullify (Undo->Skip)	x	x	x	x	x	x	x

Fig. 20. Complex I-Options