Supporting Cross-Organizational Process Control

Samuil Angelov[1], Jochem Vonk[1], Krishnamurthy Vidyasankar[2], 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. E-contracts express the rights and obligations of parties through a formal, digital representation of the contract provisions. In process intensive relationships, e-contracts contain business processes that a party promises to perform for the counter party, optionally allowing monitoring of the execution of the promised processes. In this paper, we describe an approach in which the counter party is allowed to control the process execution. This approach will lead to more flexible and efficient business relations which are essential in the context of modern, highly dynamic and complex collaborations among companies. We present a specification of the process controls available to the consumer and their support in the private process specification of the provider.

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 Research 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).

3 Specification and Support of Process Control

We call an activity at the external level a Visibility point (VP). A VP provides information about the activity and the states of the activity during the process execution. 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 [6]. 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 [6] 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
DELAY/PROCEED	The start of execution of an activity is delayed/continued.
START	The execution of an activity is started.
PAUSE/CONTINUE	The execution of a started activity is paused/resumed.
SKIP	The execution of a non-started activity is skipped.
CANCEL	The execution of a started activity is terminated. Partial results from the execution of the activity remain.
RESET	The execution of a started activity is stopped and the activity is put back in its pre-start state without undoing any of the work that has been performed.
UNDO	The execution of a started activity is stopped, what has been done is undone, and the activity is put back in its pre-start state.
RETRY	An activity that has ended is put back in its pre-start state. Results from previous executions are not undone.
CHOICE	From a set of activities, the consumer chooses an activity(s) that will be executed.
ORDER	From a set of activities, the consumer selects the order of execution of the activities.

We distinguish two types of I-options, i.e., *state* and *structural*. State I-options affect the execution state of an activity. Structural I-options are used to control the execution path of the process specification (shown in italic in Table 1).

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 possible parameters are time (e.g., for DELAY, PAUSE), sequence (for ORDER), etc. 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.

Table 2. Sample list of complex I-options

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

A state I-option leads to a change in the state of a VP. In Fig. 1, we present the state model of a VP and the state 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 where clear rules for the availability of I-options are required (see Section 3.3).

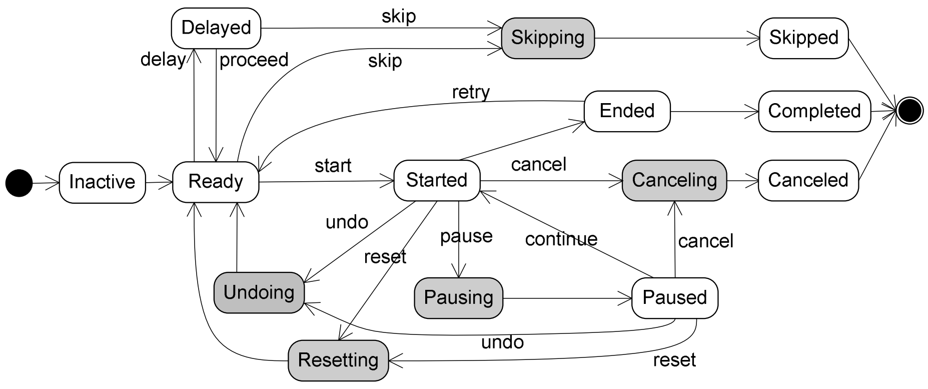

Fig. 1. The state model for the visibility points

A state change of a VP caused by a state 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. 1. The transitions that do not have an "ING" state take place synchronously at the external and conceptual levels (see Section 3.2).

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. 1. However, typically, the activity state model supported at the conceptual level would be more limited (see e.g., [7]), 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.

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 "state" 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 activities are non-interruptible).

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 block 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.

DELAY(x_i), PROCEED(x_i), START(x_i): The support of these I-options requires the introduction of a "wait" state before c_{i1}. The consumer is given some time to exert the DELAY control. If the control is exerted, the wait state is entered until a PROCEED is requested. Otherwise, the execution proceeds with the execution of c_{i1}.

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. The external activity is in state PAUSING until this state is reached.

SKIP(x_i): The invocation of this I-option requires 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. A "wait" state preceding the split must be introduced to represent the READY state of x_i during which the consumer has the time to apply the SKIP (see Fig. 2).

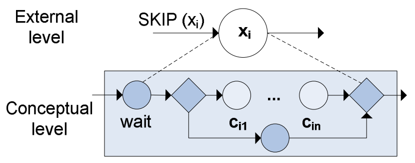

Fig. 2. Support of "SKIP"

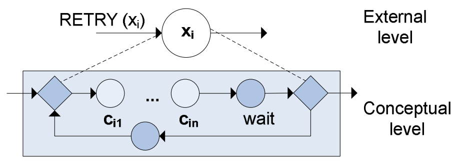

Fig. 3. Support of "RETRY"

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. During a cancellation, x_i is in state CANCELLING.

RESET(x_i): Similar to the cancel I-option, a "split" is necessary at the conceptual level to "implement" this I-option. 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.

UNDO(x_i): Two approaches can be used to support the 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 activities will undo the work done. The second approach makes use of transaction management support, e.g., atomicity and compensation techniques (see [9] for an overview).

RETRY(x_i): To support the invocation of the RETRY 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. 3). This activity represents the ENDED state of x_i.

[1] In [8], workflow control patterns are presented under the abbreviation WCP, followed by their number.

CHOICE(x_i,...,x_j), ORDER(x_i,...,x_j): The invocation of these I-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.

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.

3.3 Specification and System Support of I-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. The state model shown in Fig. 1 is a first step in the support of I-options availability.

4 An Example Case

In this section, we illustrate our approach using a real-life process from the healthcare domain: teleradiology. This process concerns 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.

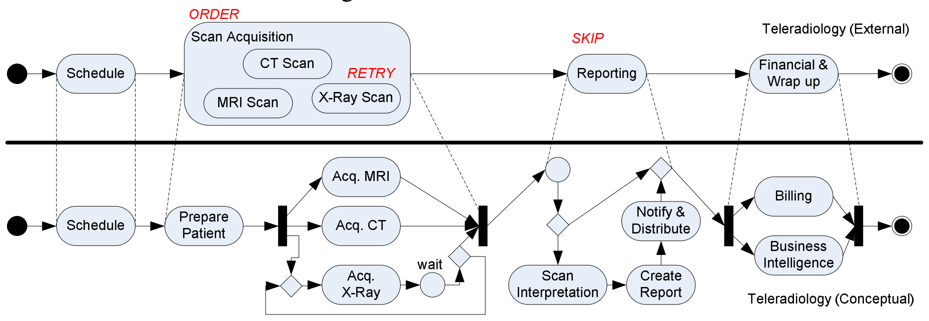

Fig. 4. Teleradiology example

Figure 4 shows a simplified teleradiology process (extensively described in [10]). 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. The dashed lines represent the mapping of external activities to conceptual activities, e.g., the 'reporting' activity actually consists of three conceptual activities.

As can be seen in the figure, three I-Options are specified for the teleradiology service: ORDER, RETRY, and SKIP. The scan acquisition activities are performed in sequence as the patient needs to be physically present. Using the I-option, the consumer can state the order of their execution. The conceptual process indicates that the three activities can be performed in any order defined with the I-request.

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.

If the service consumer determines that the scans he has received are providing enough information, the 'reporting' activity could be skipped and process execution would be continued with the 'Financial & wrap up' activity. Financial consequences of exerting an I-option should be included in the e-contract.

5 Related Work

The 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 were 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 [6]. 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], [11]. 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 [12], [13]. Web services, however, are black-box services that do not expose the process contained within. In [14], 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.

6 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.

We note that the I-options can be defined also on a process level. For example, a process may be delayed, canceled, restarted. A subset of the I-options can be applied also on parts of the process. We shall address the support of I-options on processes and process parts in our future work.

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. Angelov, S., Grefen, P.: The Business Case for B2B E-contracting. In: 6th international conference on Electronic commerce, pp. 31--40. ACM Press, New York (2004)
2. Xu, L.: Monitoring Multi-Party Contracts for E-business. PhD thesis, University of Tilburg (2004)
3. Grefen, P., Aberer, K., Hoffner, Y., Ludwig, H.: CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises. Int. J. of Computer Systems Science and Engineering, 15 (5), 277--290 (2000)
4. Grefen, P., Ludwig, H., Angelov, S.: A Three-Level Framework for Process and Data Management of Complex E-services. Int. J. of Coop. Inf. Systems, 12 (4), 487--531 (2003)
5. Eshuis, R., Grefen, P.: Constructing Customized Process Views. Data & Knowledge Engineering, 64 (2), 419--438 (2008)
6. Schonenberg, M., Mans, R., Russel, N., Mulyar, N., Aalst, W.: Process Flexibility: A Survey of Contemporary Approaches. In: Dietz, J., Albani, A., Barjis, J. (eds.) Advances in Enterprise Engineering. LNBIP, vol. 10, pp. 16--30 Springer, Berlin (2008)
7. Hollingsworth, D.: The Workflow Reference Model. TC00-1003, Workflow Management Coalition (1995)
8. Russel, N., Hofstede, A., Aalst, W., Mulyar, N.: Workflow Control-Flow Patterns: A Revised View. BPM Center Reports, BPM-06-22, BPM Center (2006)
9. Wang, T., Vonk, J., Kratz, B., Grefen, P.: A Survey on the History of Transaction Management: from Flat to Grid Transactions. Distributed and Parallel Databases, 23 (2) 235--270 (2008)
10. Vonk, J., Wang, T., Grefen, P., Swennenhuis, M.: An Analysis of Contractual and Transactional Aspects of a Teleradiology Process. Beta Working Papers, WP 263, Eindhoven Univ. of Technology (2008)
11. Milosevic, Z., Dromey, R.: On Expressing and Monitoring Behaviour in Contracts. In: 6th EDOC Conference, pp. 3-14, IEEE Computer Society, Washington (2002)
12. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Application. Springer (2004)
13. Papazoglou, M.: Web Services: Principles and Technology. Prentice Hall (2007)
14. Grefen, P., Ludwig, H., Dan, A., Angelov, S.: An Analysis of Web Services Support for Dynamic Business Process Outsourcing. IST, 48 (11), 1115--1134 (2006)