# Transactional Workflows or Workflow Transactions?

Paul Grefen

Computer Science Department, University of Twente
P.O. Box 217, 7500 AE Enschede, Netherlands
http://www.cs.utwente.nl/~grefen

**Abstract.** Workflows have generally been accepted as a means to model and support processes in complex organizations. The fact that these processes require robustness and clear semantics has generally been observed and has lead to the combination of workflow and transaction concepts. Many variations on this combination exist, leading to many approaches to transactional workflow support. No clear classification of these approaches has been developed, however, resulting in a badly understood field. To deal with this problem, we describe a clear taxonomy of transactional workflow models, based on the relation between workflow and transaction concepts. We show that the classes in the taxonomy can directly be related to specification language and architecture types for workflow and transaction management systems. We compare the classes with respect to their characteristics and place existing approaches in the taxonomy – thus offering a basis for analysis of transactional workflow support.

## 1 Introduction

Workflows have generally been accepted as a paradigm for modeling and supporting processes in complex organizations. Often workflow processes have a business character, but workflow concepts have also been used for other processes types, e.g., scientific processes or software production processes. The use of workflows for core processes of organizations has lead to the requirements of clear process semantics and robustness in process execution, both in regular process execution and under exception or error conditions. The notions of transaction management, already used for several decades in the database world, have been combined with workflow notions to satisfy these requirements. Resulting from this, the notion of transactional workflow or workflow transaction has emerged. Many variations on the notion of transactional workflow or workflow transaction have been developed, however, by merging the worlds of workflow and transaction management in different ways – the two more or less synonymous terms are an omen of this. No clear classification has been developed yet that provides a framework for the analysis of transactional workflow models and systems supporting these models. Matching models and systems with application requirements and comparing approaches is therefore not easy.

In this paper, we present a classification framework that provides two main classes for the combination of workflows and transactions, based on the relation between workflow and transaction concepts. The main classes are further refined into subclasses with specific properties, resulting in six basic classes. We show that the conceptual classes can directly be mapped to specification language and architecture classes for workflow and transaction management support. We analyze the classes with respect to their goal, means to achieve this goal, and advantages and disadvantages. The

R. Cicchetti et al. (Eds.): DEXA 2002, LNCS 2453, pp. 60–69, 2002.

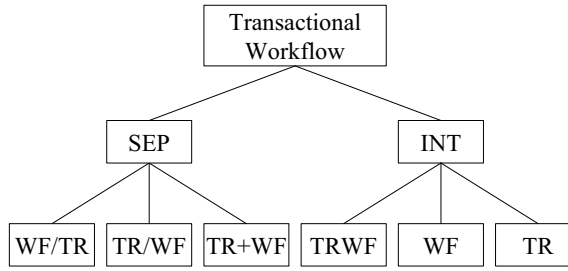


**Figure 1.** Transactional workflow taxonomy

framework and analysis together provide a clear basis for comparing approaches and selecting specific approaches for specific application classes.
The structure of this paper is as follows. In Section 2, we describe our basic taxonomy that underlies the classification presented in this paper. In Section 3, we present the conceptual point of view of our classification, focused around language aspects. Section 4 presents the system point of view, centered on architecture aspects. In Section 5, we apply the framework by comparing the various classes and classifying existing approaches to transactional workflows. Section 6 contains conclusions.

## 2 The Taxonomy

To support transactional workflows, there are two basic approaches: either transactional aspects and workflow aspects are treated as separate issues, or they are seen as one integrated issue. In the former case, separate transaction and workflow models exist that are combined to obtain transactional workflows. In the latter case, one single transactional workflow model is used. These two main classes are refined below.
In the situation where we have separate workflow and transaction models, we need to relate these two models. We have three possible basic relations, based on the abstraction relation between the models:

Workflows over transactions (WF/TR): workflows are more abstract than transactions – transaction models are used to provide semantics to workflow models.

Transactions over workflows (TR/WF): transactions are more abstract than workflows – workflow models are used to provide process structure to transaction models.

Transactions and workflows as peers (TR+WF): workflow and transaction models exist at the same abstraction level – workflow and transaction models can be seen as two submodels of an implicit, loosely coupled process model.

In the case of one single model for both workflow and transaction aspects, obviously there is no relation between models. There are, however, three main variants with respect to the nature of the single model:

Hybrid transactional workflow model (TRWF): a single hybrid model is used that contains both transaction and workflow concepts.

Transactions in workflows (WF): a single workflow model is used, in which transactional aspects are mapped to workflow primitives.

Workflows in transactions (TR): a single transaction model is used, in which workflow aspects are mapped to transaction primitives.

The resulting taxonomy is depicted in Figure 1, in which the 'SEP' main class contains basic classes with separate models for workflows and transactions, the 'INT' main class contains basic classes with a single integrated model. We use this taxonomy to discuss conceptual and architectural characteristics of each of the classes.

## 3 The Conceptual Point of View

In this section, we discuss the conceptual point of view of our framework. To do so, we will take the specification language perspective, which we explain below. Then, the various classes of our taxonomy are discussed from this language perspective.
In the conceptual point of view, we are interested in the conceptual specification of transactional workflows formulated in one or more specification languages. Given the two main classes in the taxonomy of Figure 1, we can have two situations. In the first situation, there is a separate language for specifying workflow aspects, the workflow definition language (WFDL), and a separate language for specifying transaction aspects, the transaction definition language (TRDL). In the second situation, there is an integrated language for specifying both workflow and transaction aspects, the transactional workflow definition language (TRWFDL).
If we have two languages, the languages can have two relations: either one language is a refinement of the other, or the two languages are orthogonal with respect to each other. If the two languages have a refinement relation, we have the following. A language offers primitives to specify transitions in a state space. A language $L_2$ is a refinement of a language $L_1$ if there is a notion of correspondence (a relation in the mathematical sense) between its state space and that of $L_1$, and between its primitives and those of $L_1$, such that the transitions specified by the primitives maintain the correspondence between states (see [11] for a further explanation). If the TRDL is a refinement of the WFDL, the WFDL level contains workflow attributes and the intermediate states at the TRDL level are related to transaction states. If the WFDL is a refinement of the TRDL, the TRDL level contains transactional attributes and the intermediate states at the WFDL level are related to control flow states.
In the integrated approach, all aspects are merged into a single language, covering a state space that is the cross product of the two state spaces discussed above.

### 3.1 Separate Languages

The main reason for using two separate languages is separation of concerns in dealing with control flow and transaction aspects in complex applications. Below, we discuss the three basic classes of the separate models approach.
In the WF/TR case, the control flow aspect is leading in the specification of transactional workflows. Low-level workflow semantics are based on transactional semantics of individual workflow tasks or groups of workflow tasks. Hence, the TRDL is a refinement of the WFDL. Primitives of the WFDL are mapped to primitives of the TRDL. Transaction semantics are often imported from the data management level – the TRDL is a sublanguage of a data manipulation language (DML) in this case. The WF/TR approach is taken in most commercial workflow management systems that

support (usually limited) transactional behavior of workflows. Below, we show a simple example in which individual workflow tasks can be parameterized to behave as business transactions (atomic and isolated units of execution). On the left, we see the specification of a workflow task. The second and third lines of this specification are expanded on a lower abstraction level to the transaction specification shown on the right. When executed, the TRDL specification will induce intermediate states with respect to the WFDL specification.

**WFDL**

```
TASK task1
 BUSINESS TRANSACTION
 USES FORM form1
END TASK
```

**TRDL**

```
BEGIN TRANSACTION
 READ form1.field1
 READ form1.field2
 USE form1
 WRITE form1.field1
 WRITE form1.field2
 IF status_ok
 THEN COMMIT TRANSACTION
 ELSE ABORT TRANSACTION
END TRANSACTION
```

In the TR/WF class, transactional behavior is the leading aspect in the specification of transactional workflows. High-level transactional semantics are specified with a workflow as elaboration of the underlying process structure. Hence, the WFDL is a refinement of the TRDL. The TR/WF approach is applied for example in workflow management for e-commerce applications. Here, the transaction between two business partners is the starting point and the elaboration of the control flow a refinement of the transaction. We show a simplified example below. On the left, we see a TRDL specification of a transaction that states transactional properties. The control flow is seen as an implementation detail to be specified at a lower level of abstraction. This is elaborated in the WFDL specification on the right. Note that the WFDL specification concerns a non-linear process, which is not easy to specify in traditional TRDLs. The execution of the WFDL specification will introduce intermediate states with respect to the execution of the TRDL specification.

**TRDL**

```
TRANSACTION tr1
 EXECUTE ATOMIC
 IMPLEMENTATION wf1
END TRANSACTION
```

**WFDL**

```
WORKFLOW wf1
 TASK task1 task2 task3 task4
 SEQUENCE task1 task2
 SEQUENCE task1 task3
 SEQUENCE task2 task4
 SEQUENCE task3 task4
END WORKFLOW
```

In the TR+WF approach, there is a balance between control flow and transactional behavior. High-level transactional semantics are defined on the same conceptual level as workflow processes. Hence, workflow and transaction specifications refer to each other on the same level of abstraction. Below, we show a stylized example. On the left, we see a WFDL specification that specifies a control flow and refers to the TRDL specification for the transactional properties. The TRDL specification shown on the right imports the task list from the WFDL specification and specifies transactional properties over this. The TRDL specification specifies compensating tasks [5] and a safepoint [10] to allow flexible rollback by compensation. Control flow and compensation functionality can be changed independently of each other, thus creating a separation of concerns between workflow and transaction specification.

**WFDL**

```
WORKFLOW wf1
 REFERS TRANSACTION tr1
 TASK task1 task2 task3
 SEQUENCE task1 task2
 SEQUENCE task2 task3
END WORKFLOW
```

**TRDL**

```
BEGIN TRANSACTION tr1
 REFERS WORKFLOW wf1
 COMP ctask1 task1
 COMP ctask2 task2
 SAFEPOINT task1
END TRANSACTION
```

### 3.2 Integrated Models

In the integrated model class of the taxonomy, workflow and transaction semantics are combined into one single model.

In the TRWF class of our taxonomy, we find hybrid workflow and transaction models. These models are reflected in hybrid transactional workflow specification languages. These languages contain typical workflow-related primitives – e.g., to express control flows – and transaction-related primitives – e.g., to express atomicity or isolation requirements. An obvious way to create a TRWF language is to 'merge' a pair or languages of the TR+WF class. Following this approach, we can obtain the example below from the TR+WF example shown above. Clearly, the TRWF and TR+WF approaches are exchangeable to some extent.

**TRWFDL**

```
WORKFLOW wf1
 TASK task1 COMP ctask1 SAFEPOINT
 TASK task2 COMP ctask2
 TASK Task3 COMP none
 SEQUENCE task1 task2
 SEQUENCE task2 task3
END WORKFLOW
```

In the WF class, transactional semantics are expressed in workflow processes. Specific process patterns are used to express transaction behavior of workflow processes. An example is the specification of compensation patterns in workflow definitions to achieve relaxed atomicity characteristics for a workflow. We show an example below. In the WF specification, we see the definition of regular tasks and a regular control flow (three consecutive tasks) and the definition of compensating tasks and compensating control flow (two consecutive tasks). The compensating control flow is linked to the regular control flow through or-splits (alternative paths). At an or-split, a condition is evaluated to check whether rollback of the workflow is required – if not, the regular control flow is followed – if so, the compensating control flow is followed. Note that the example is in fact a static specification of all possible cases of the dynamic compensation behavior of the TRWF example above.

**WFDL**

```
WORKFLOW wf1
 TASK task1 task2 task3  # regular tasks
 TASK ctask1 ctask2      # compensating tasks
 SPLIT or1 or2
 SEQUENCE task1 or1      # start regular control flow
 SEQUENCE or1 task2
 SEQUENCE task2 or2
 SEQUENCE or2 task3
 SEQUENCE or1 ctask1     # start compensation control flow
 SEQUENCE or2 ctask2
 SEQUENCE ctask2 ctask1
END WORKFLOW
```

In the TR class of the taxonomy, workflow semantics are expressed in transaction specifications. In this approach, transactions have structured processes as their action

specification. An example is shown below. Here we see a transaction consisting of two subtransactions that can be executed in parallel – thus constituting a rudimentary form of control flow.

```
TRDL
TRANSACTION tr1
 SUBTRANSACTION s1
  action1; action2
 END SUBTRANSACTION
 SUBTRANSACTION s2
  action3; action4
 END SUBTRANSACTION
 PARALLEL s1 s2
END TRANSACTION
```

## 4 The System Point of View

After having discussed the conceptual point of view in the previous section, we turn to the system point of view in this section. Where the conceptual point of view explains the 'what', the system point of view explains the 'how' – i.e., the support of workflow and transaction models.

We base the system point of view on the architecture aspect, focusing on the high-level structure of transactional workflow support systems. We use abstract architectures to identify the elementary system characteristics of the classes of the taxonomy. We relate these abstract architectures to concrete architectures in Section 5. In the description of the architectures, we place workflow management and transaction management modules on top of a function and data support (FDS) layer. The details of this layer are not relevant in the context of this paper.

Below, we turn to the various classes of transactional workflows, again organized as depicted in Figure 1.

### 4.1 Separate Models

In the separate models category of our taxonomy, we have separate workflow and transaction management modules in the architecture (WFM respectively TRM). These modules can have three architectural relations (as depicted in Figure 2): WFM as client of a TRM server, TRM as client of a WFM server, and WFM and TRM as peer-to-peer modules. These three architectures coincide with the three classes WFM/TRM, TRM/WFM and TRM+WFM. In discussing the characteristics of the three classes, the focus is on the WFM-TRM interface, as indicated by triangles in Figure 2. This interface is used in all three architectures to synchronize the control flow state in the WFM and transaction state in the TRM.

The WFM/TRM architecture is depicted in the left hand side of Figure 2. The interface between WFM and TRM is both a control and a data channel. The WFM uses the TRM interface to open a transaction context and perform data manipulation operations in this context. In this class, TRM and FDS are often integrated into one database application environment based on a DBMS with built-in transaction management functionality. The WFM/TRM architecture is 'standard' for commercial systems.

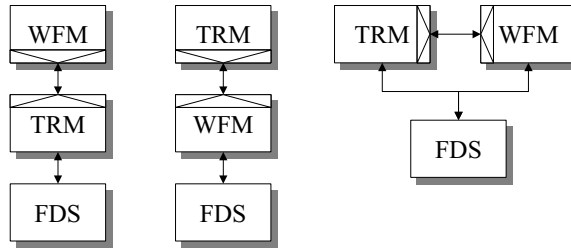


**Figure 2.** WFM/TRM, TRM/WFM and TRM+WFM architectures

The TRM/WFM architecture is depicted in the center of Figure 2. The interface between TRM and WFM is both a control and a data channel. The TRM uses the WFM interface to open a workflow context and next to invoke control flow primitives. We observe this architecture class in e-commerce environments where a high-level transaction engine invokes processes supported by workflow management technology.
The TRM+WFM architecture is depicted in the right hand side of Figure 2. In this architecture, we have a TRM as transaction server and a WFM as process server in a peer-to-peer relation. The interface between WFM and TRM is strictly a control channel: the WFM communicates process states to the TRM, the TRM communicates transaction contexts and workflow commands to effectuate transactional effects on process states to the WFM. Note that this interface is not a standard interface as defined by the WfMC – its Interface 4 standard describes communication between two workflow servers [17]. The TRM+WFM architecture is – trivially – fit for TR+WF language support. TRWF language support is possible by filtering a TRWF specification into the right parts for TRM and WFM.

### 4.2 Integrated Models

In the integrated models class of our taxonomy, we have a single transactional workflow management module in the architecture. This can either be a transactional workflow manager (TRWFM), a traditional workflow manager (WFM), or an advanced transaction manager (TRM). The three cases are shown in Figure 3.
The TRWFM offers integrated support for transaction management and workflow management. A hybrid transaction and workflow state is maintained within the TRWFM. It supports languages in the TRWF and TR+WF classes. To handle TR+WF specifications, the two subspecifications are merged into one specification by a preprocessor.
In the WFM architecture class, the state of a transactional workflow is completely maintained by WFM. Transactional attributes of this state are mapped to workflow attributes. The WFM can only interpret specifications in the WF class. Specifications in other classes (typically TRWF) have to be translated to WF format. This architecture class is fit for support by commercial workflow management systems.
In the TRM class, the state of a transactional workflow is completely maintained by the TRM. Control flow attributes of this state are implemented by transaction attributes. The TRM can only interpret specifications in the TR class. Specifications in other classes (typically TRWF) have to be translated to TR format. Nested process

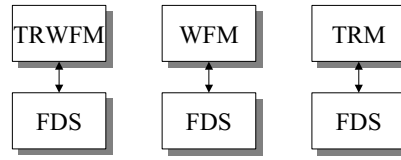


**Figure 3.** TRWFM, WFM and TRM architectures

structures can be supported by standard transaction management technology. More advanced structures are typically only supported by research prototypes.

## 5 Application of the Framework

In this section, we apply the taxonomy we have developed in two ways: we present a comparison of the various classes with respect to their characteristics and we place existing work in our taxonomy.

### 5.1 Comparing the Classes

In Table 1, we show a comparison of the classes in our taxonomy. For each class, we list the main goal, the means used to achieve this goal, and a brief list of advantages and disadvantages – which we explain in the sequel.
Flexibility in coupling models and separation of concerns between workflow and transaction aspects are main advantages of the classes with separate models. Problems with integration of models (and support based on these models) form the downside of this aspect. Consistency of specifications is a main advantage of the single-model approach: there are no separate models to be kept consistent. Consistency can certainly be a problem in the TR+WF class, as two specifications of a transactional workflow exist without a 'leading' specification. Limited expressiveness is a clear disadvantage of the WF and TR classes, as possible semantics of transaction aspects are limited by available workflow primitives and vice versa. The TRWF class does generally not have this problem, but a complex formalism with equally complex semantics usually is the basis for models in this class. Finally, system support is (cur-

**Table 1.** Comparison of classes

| class | goal | means | advantages | disadvantages |
|---|---|---|---|---|
| **WF/TR** | WF with robust character | data management in WFs | sep. of concerns, flexibility, support | integration |
| **TR/WF** | TR with complex control flow | process management in TRs | separation of concerns, flexibility | integration |
| **TR+WF** | integrated WF and TR | coupled process and data mngmnt | separation of concerns, flexibility | integration, consistency |
| **TRWF** | integrated WF and TR | hybrid process and data management | integration, consistency | complex formalism, inflexibility |
| **WF** | WF with robust character | advanced process management | simple formalism, consist., support | limited expressiveness |
| **TR** | TR with complex control flow | advanced TR management | simple formalism, consistency | limited expressiveness |

rently) best for the WF/TR class (supported by combinations of existing WFM and TRM systems) and the WF class (implementable on commercial WFM systems).

### 5.2 Positioning Existing Approaches

In this subsection, we place a selection of existing approaches in our taxonomy. For reasons of brevity, the overview is far from complete. We first discuss approaches in the separate models category, then turn to the integrated model category, and finally pay attention to an approach that combines aspects of both categories. A more elaborate analysis is presented in [11].

In the Mercurius initiative [8], a WF/TR architecture has been proposed: workflow management functionality is placed on top of transaction management functionality. The CrossFlow approach [9, 16] is an example of the TR/WF class. In the CrossFlow language perspective, cross-organizational transactions are specified in an electronic contract that is mapped to workflow definitions. In the CrossFlow architecture perspective, inter-organizational transaction management functionality is placed on top of workflow management systems. In [4], an approach to a language in TR+WF class is proposed in which independence between specification of control flow on the one hand and transactional characteristics on the other hand is a starting point.

In the context of the FlowMark WFMS, an approach for the support of business transactions has been developed [14]. It uses specification of transactional spheres in a workflow process that is interpreted by extended workflow technology – the approach can hence be placed in the TRWF class. ObjectFlow [12] can also be placed in the TRWF class. In the Exotica approach [1], a language is proposed in TRWF class. This language is preprocessed to be interpretable by an architecture in the WF class. In the FlowBack project [13], a similar approach has been developed. In the WF language class, we find work in which transactional characteristics are coded into Petri Nets [3]. The ConTracts model [15] is an approach in the TR class providing an environment for reliable execution of long-lived computations. The control flow primitives of the ConTracts language have been used for the realization of transactional workflows. TSME [6] is an approach in the TR class with comparable goals.

In the WIDE project, an approach has been developed that combines aspects of both main classes of our taxonomy. The WIDE workflow specification language belongs in the TRWF class, as it is a ‘classical’ workflow definition language extended with (among other things) transactional primitives. The WIDE architecture contains three levels [7]. The two high-level transaction management levels GTS [10] and LTS [2] belong in the TR+WF class, the low-level LTI level [2] belongs in the WF/TR class.

## 6 Conclusions and Outlook

In this paper, we have described a taxonomy for transactional workflow support, paying attention to both the conceptual and system point of view. Characteristics of the conceptual point of view have been described in terms of specification language classes. System characteristics have been discussed in terms of architecture topologies and interfaces. The result is a taxonomy that provides a background for selecting or analyzing transactional workflow support. Choosing appropriate classes from our taxonomy for the conceptual and system points of view is a basis for the configuration

of transactional workflow support in complex applications, where functional requirements and architectural context both play an important role. The choice can be different in both points of view to some extent, but a mapping must be possible.
Completing the analysis of existing approaches to obtain an overview of the state of the art is an obvious follow-up of the presented work. Extending the framework to better cover multi-level transaction support is an important research direction.

**Acknowledgments**. Maarten Fokkinga is acknowledged for his assistance with respect to language concepts and his feedback on the draft version of this paper.

## References

1. G. Alonso et al.; *Advanced Transaction Models in Workflow Contexts;* Procs. Int. Conf. on Data Engineering, 1996; pp. 574-581.
2. E. Boertjes et al.; *An Architecture for Nested Transaction Support on Standard Database Systems*; Procs. 9th Int. Conf. on Database and Expert System Appls., 1998; pp. 448-459.
3. J. Dehnert; *Four Systematic Steps towards Sound Business Process Models*; Procs. 2nd Int. Colloq. on Petri Net Techn. for Modeling Comm. Based Systems, 2001; pp. 55-63.
4. W. Derks et al.; *Customized Atomicity Specification for Transactional Workflows*; Procs. 3rd Int. Symp. on Cooperative Database Systems for Adv. Appls., 2001; pp.155-164.
5. H. Garcia-Molina, K. Salem; *Sagas*; Procs. 1987 ACM SIGMOD Int. Conf. on Management of Data, 1987; pp. 249-259.
6. D. Georgakopoulos, M. Hornick, F. Manola; *Customizing Transaction Models and Mechanisms in a Programmable Environment Supporting Reliable Workflow Automation*; IEEE Trans. on Knowledge and Data Engineering, (8)4, 1996; pp. 630-649.
7. P. Grefen et al.; *Two-Layer Transaction Management for Workflow Management Applications*; Procs. 8th Int. Conf. on Database and Expert System Appls., 1997; pp. 430-439.
8. P. Grefen, R. Remmerts de Vries; *A Reference Architecture for Workflow Management Systems*; Journ. of Data & Knowledge Engineering, (27)1, 1998; pp. 31-57.
9. P. Grefen et al.; *CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises*; Int. Journ. of Computer Systems Science & Engineering, (15)5, 2000; pp. 277-290.
10. P. Grefen, J. Vonk, P. Apers; *Global Transaction Support for Workflow Management Systems: from Formal Spec.to Practical Impl.*; VLDB Journal, (10)4, 2001; pp. 316-333.
11. P. Grefen; *A Taxonomy for Transactional Workflows*; CTIT Technical Report 02-11; University of Twente, 2002.
12. M. Hsu, C. Kleissner; *ObjectFlow and Recovery in Workflow Systems*; in: V. Kumar, M. Hsu; *Recovery Mechanisms in Database Systems*; Prentice Hall, 1998; pp. 505-527.
13. B. Kiepuszewski, R. Muhlberger, M. Orlowska; *FlowBack: Providing Backward Recovery for WFMs*; Procs. ACM SIGMOD Int. Conf. on Management of Data, 1998; pp. 555-557.
14. F. Leymann; *Supporting Business Transactions via Partial Backward Recovery in WFMs*; Procs. Datenbanksysteme in Büro, Technik und Wissenschaft, 1995; pp. 51-70.
15. A. Reuter, K. Schneider, F. Schwenkreis; *Contracts Revisited*; in: S. Jajodia, L. Kerschberg; *Adv.Transaction Models and Architectures*; Kluwer Academic, 1997; pp. 127-151.
16. J. Vonk et al.; *Cross-Organizational Transaction Support for Virtual Enterprises*; Procs. 5th Int. Conf. on Cooperative Information Systems, 2000; pp. 323-334.
17. *Workflow Management Coalition Workflow Standard – Interoperability Abstract Specification*; Doc. No. WfMC TC-1012; Workflow Management Coalition, 1996.