Workflow Transparency

Peter Bichler, Günter Preuner, Michael Schrefl

Department of Business Information Systems, University of Linz, Austria
E-Mail: {bichler | preuner | schrefl}@dke.uni-linz.ac.at

Abstract. Workflow transparency refers to the ability of considering a business process independently of the workflow implementing it. Unlike current workflow systems, which do not differentiate between business processes and workflows, workflow transparency requires to model business processes and workflows separately.
Workflow systems supporting workflow transparency offer advantages comparable with database systems supporting data independence. Similarly as data independence allows to change the actual organization of data without effecting applications accessing the database at the conceptual level, workflow transparency allows for dynamic re-organisation of the actual flow of work without effecting the overall results of business processes.

1 Introduction

Modern organizations must continuously adapt their workflows to be competitive. The support for adaption of workflows is thus a critical issue since poor support may result in error-prone and time-consuming adaption processes. Hence workflow management systems (WFMS) face high demands and must offer several features: mechanisms to change workflow descriptions while they are enacted, mechanisms to test the correctness of changed workflows wrt. to the underlying business process, and mechanisms to deal efficiently with business cases that existed at the time of change (known as the dynamic change problem [6]). Although these problems have been identified (cf. [1, 4, 6, 17]), contemporary WFMSs do not provide adequate support for the adaption of workflows.

Contemporary WFMSs suffer from two serious shortcomings: First, their modeling components rely on a flat, single-schema architecture which provides no means to explicitly capture the business process a workflow implements. Second, they do not provide efficient support for the dynamic change of workflows. Therefore, expensive change procedures can cause the day-to-day business of organizations to be interrupted in order to avoid unpredictable side effects, e.g., wrong re-assignments of business cases.

How do related disciplines solve the problem of evolving systems? In object-oriented programming, the concept of encapsulation abstracts the interface from the implementation of objects, which allows to adapt the implementation of objects without effecting their clients (cf. [14]). In data engineering, the three schema ANSI SPARC architecture describes the contents of a database at three levels. Application programs that access the database at higher levels are widely

independent from changes at lower levels (*physical data independence* and *logical data independence*. To provide a similar flexibility for dynamic objects, Saake et al. [15] have introduced an analogous three-schema architecture.

In this paper, we introduce a *two-schema architecture* for the modeling components of WFMSs. The proposed architecture naturally extends existing single-schema architectures by *abstracting the essential business process logic out of a workflow schema and modeling it separately in a business process schema.* The resulting two-schema architecture copes very efficiently with a number of adaptions that are reflected in the workflow leaving the supported business process the same. Radical changes in the business processes, however, require additional effort.

Section 2 describes the basic concepts of a two-schema architecture for workflow systems. Section 3 presents $OBD_{bp/wf}$, a notation for the conceptual design of business process schemas and workflow schemas, which relies on the proposed two-schema architecture. Section 4 discusses the advantages of the presented schema architecture wrt. dynamic change of workflows and compares it to related previous work. Section 5 summarizes the main achievements of this work.

2 A Two-Schema Architecture for Workflow Systems

The *business policy* of an organization determines what kind of work the organization performs and in which way it performs that work. The market forces organizations more or less often to change their business policy to be competitive. To implement such changes, information systems must be frequently adapted to changed requirements. Loucopoulos [12] pointed out that maintenance and evolution of information systems could be improved significantly if the knowledge expressed in business policies were represented explicitly. Therefore, recent approaches to the modeling of information systems, e.g., [5, 8, 12, 13], express the business policy of an organization in form of business rules.

Business rules may be classified into *external* and *internal business rules* [7]. External business rules—e.g., business rules based on natural facts or law—determine a frame within which an organization may operate. Internal business rules—i.e., business rules based on organizational commitments—restrict this frame and determine the way an organization actually performs work. To react to changes in the business environment, an organization may change its internal business rules but, usually, it may not influence external business rules. *This paper focuses on business rules that concern the flow of work.* These business rules, although expressed by *event-condition-action* rules by some authors, are *better represented by appropriate high-level concepts*, such as extended Petri Nets, at the conceptual level (cf. [10]).

The two-schema architecture for the conceptual modeling of workflows, which we present below, offers high-level concepts for the modeling of business policies and allows to separate external from internal business rules. This separation allows to *verify* the change of internal business rules wrt. external business rules. In conventional workflow systems, which rely on single-schema architectures, changes cannot be verified within the system, but can only be *validated* wrt. the environment.

The two schemas are the *business process schema* and the *workflow schema.* The business process schema specifies a comprehensive set of alternative ways to meet the objectives of a business process. The business process is specified by a set of business process activities and constraints on performing these activities, where only constraints implied by external business rules are considered. For example, external rules might constrain the execution order of activities (e.g., it is a natural fact that input-output dependencies among activities require their serialization) and the invocation of activities in general (e.g., the cancellation fee a hotel might charge is limited by law depending on the time of cancellation).

A workflow schema models the realization of a business process in a certain organization at a certain time. To adapt a business process schema to a context of internal business rules, i.e., to design a workflow schema, means (a) to determine the actual flow of work, possibly selecting only a subset of the alternative ways provided by the business process to meet its objectives and (b) to specify how input data for business process activities are collected.

Notice, ARIS [16] provides also a multi-level architecture for modeling workflows where each level consists of several views. The aims of these levels and views, however, are orthogonal to our aims. The levels of ARIS reflect "different proximity to information technology" and the views represent different aspects of an enterprise (e.g., function view, organization view). Our two schemas separate external and internal business rules and could be included in ARIS as part of the *function view* at the *design specification* level.

We conclude this section with a definition of workflow transparency: *Workflow transparency* denotes the quality of a workflow system to consider the specification of a business process independently from the workflow(s) implementing it.

3 Modeling Business Processes and Workflows

In this section we present $OBD_{bp/wf}$ (Object/Behavior diagrams for the conceptual modeling of business processes and workflows). They extend Object/Behavior diagrams (*OBD*), which have been originally introduced for the conceptual design of object-oriented database schemas [9], and build on the proposed two-schema architecture: The *business process schema* is modeled using pure *OBD*. The *workflow schema* is modeled by the extension of *OBD* presented in Subsect. 3.2. As a running example we use a simple *hotel reservation system* [9, 11].

3.1 The Business Process Schema

The business process schema specifies what data are collected about business cases and how business cases are processed irrespective of internal business rules currently effective. A business process schema consists of a set of business object types, each defining the structure and behavior of its instances (which represent business cases) by an *object diagram* and a *behavior diagram*, respectively. The behavior diagram of an business object type is split into a *life cycle diagram*, several *activity specification diagrams*, and several *activity realization diagrams.* Due to space limitations, we consider only life cycle diagrams in this paper. For the other diagrams, the reader is referred to [9, 11].

A *life cycle diagram* provides an overall view of a business process by depicting the states in which a business case may reside and the activities which may be performed on a business case. An activity may be invoked on a business case—or on an object in general—only if the object resides in every pre state of the activity. During the execution of an activity on an object, the object resides in an implicit state named after the activity, the *activity state*. After the successful execution of an activity, the object resides in every post state of the activity.

Example: Figure 1 shows the life cycle diagram of object type RESERVATION. Once a reservation has been issued, a hotel may, at any time, bill the customer and the customer may either cancel the reservation or check-in at the hotel. A customer that cancels after having paid for a reservation, will get a refund. A customer that cancels before being billed may be charged with a cancellation fee.

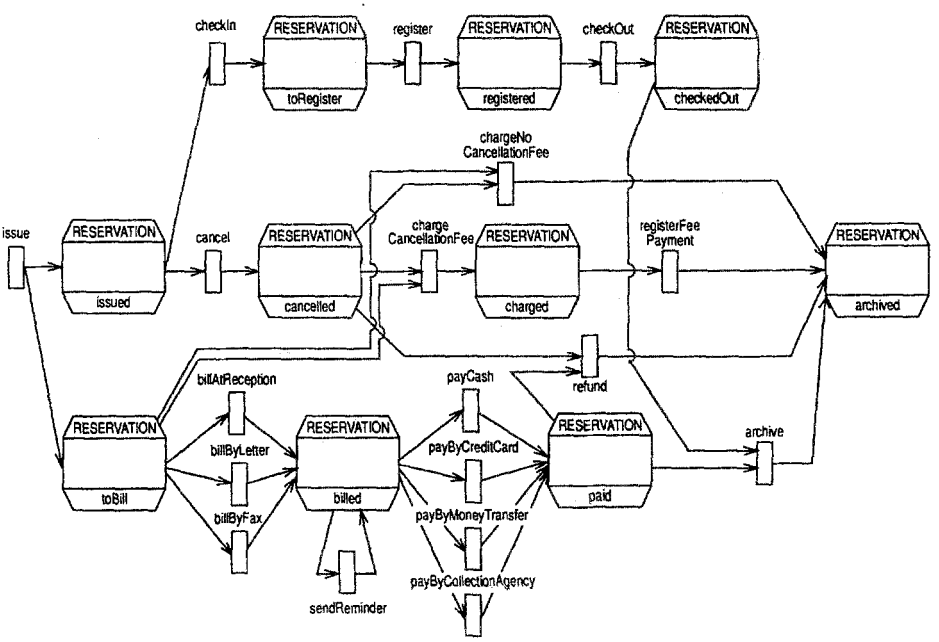

Fig. 1. Life cycle diagram of RESERVATION

3.2 The workflow schema

The workflow schema specifies how business cases are *currently* handled in a *specific* organization. A *workflow schema* consists of a set of workflow types, each defining the structure and behavior of its instances, which are called workflows. A business case is treated according to a certain workflow if it has been assigned to it as member.

A *workflow* consists of a set of boxes that serve as containers for members which fulfill certain conditions and a set of worksteps that manipulate its members by invoking business process activities and performing logically related

external operations (e.g., collecting cash). Similar as life cycle diagrams provide an overall view of business processes, workflow diagrams provide an overall view of workflows. A *workflow diagram* shows how the members of a workflow of some workflow type flow between boxes and worksteps.

A box is specified by (1) the business object type whose business cases the box may hold, (2) a life cycle state in which business cases held by the box must reside, and (3) a boolean condition which instances held by the box must fulfill. A business case belongs to a box of a workflow if it is a member of the workflow and if it qualifies for the box according to the box specification. Notice that business cases are neither inserted explicitly into boxes nor are they removed explicitly from boxes. Box membership solely depends on the current life cycle state, the current property values, and, possibly, the history of a business case. If a business case is assigned to a workflow as a new member, the business case becomes a member of all boxes for which it currently qualifies. Therefore, as we will see later, business cases can be easily re-assigned to another workflow.

Boxes are depicted by rectangles with double-line borders. A unique name for the box is shown above, the business object type and the life cycle state of the business cases which are held by the box are shown inside, and the condition that business cases held by the box must fulfill is shown below.

Example: Figure 2 shows the workflow diagram of workflow type RESERVATION-WF. Members of a workflow of this type are instances of business object type RESERVATION. One of the boxes defined by workflow type RESERVATION-WF is box R-4, which holds reservations in state toBill, where the guest's credit rating is greater than three. Note: the path expression guest.creditRating refers to properties that are defined in the object diagrams of object types RESERVATION and CUSTOMER, which are not shown.

A workflow may contain boxes that always hold a subset of the members of some other box. To indicate subset relationships between boxes, boxes are organized in specialization hierarchies. A specialization of a box into subboxes is depicted by a triangle symbol. Subboxes of the same specialization are always disjoint. A box may, however, have different specializations.

Example: In our workflow box R-3 contains all reservations in state toBill, subbox R-4 holds all reservations which reside in box R-3 and whose guests have a credit rating greater than three.

A workstep is specified by (1) one or several input boxes on whose members the workstep may be applied, (2) additional internal or external data needed by the workstep, (3) pre- and postconditions, and (4) the business process activities that are possibly invoked by the workstep. Workstep specifications are depicted by *workstep specification diagrams*, which are introduced later. For better readability, a workflow diagram usually visualizes workstep specifications only partly.

Worksteps are depicted by rectangles with a unique name at the top. Each workstep has exactly one primary input port (depicted in black) with connecting arcs to boxes on whose members the workstep may be performed. A workstep may be performed on any business case that belongs to each box connected to its primary input port.

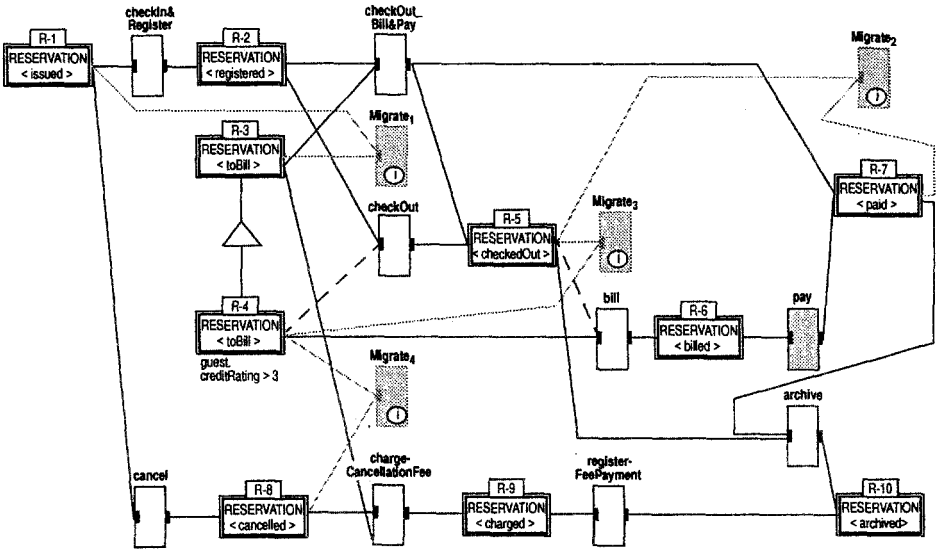

Fig. 2. Workflow diagram of RESERVATION-WF

Boxes may be connected to input ports of worksteps by *read-arcs* (depicted by dashed lines) or *transfer-arcs* (depicted by solid lines). If a box is connected to a workstep by a transfer-arc and the workstep is performed on some business case, the workstep must change the business case (by invoking some business process activity) such that it no longer qualifies for this box. In the case of a read-arc, the workstep may not modify the business case in a way such that it no longer qualifies for the box sometimes during or immediately after the execution of the workstep.

Arcs connecting the primary output port of a workstep with boxes indicate into which boxes a business case handled by a workstep flows. Once a workstep is completed, the business case assigned to its primary output port must qualify for all boxes connected to the port.

Example: Workstep checkIn&Register of RESERVATION-WF can be performed on any reservation in box R-1 as its primary input port is connected solely to that box. Workstep checkOut_Bill&Pay has a primary input port connected with two boxes. It can be performed on any reservation which resides in boxes R-2 and R-3. The primary input port of workstep checkOut is connected by a transfer-arc to box R-2 and by a read-arc to box R-4. After executing checkOut, the handled business case no longer resides in box R-2, but it still resides in box R-4 and qualifies for box R-5. The worksteps named Migrate_i will be explained in the next section.

A business case may be handled by a workflow in a more restrictive way than its business process would require. Choices between alternative business process activities may be restricted (or omitted at all by supporting only one alternative) and business process activities that could be invoked in any order may be serialized. Furthermore, different worksteps may invoke the same business process activity. Usually, such worksteps will have different realizations or are connected to different boxes.

Example: According to the business process of RESERVATION a customer may check-out from a hotel irrespective of whether he has already paid or not (cf. Fig. 1). Workflow RESERVATION-WF requires that customers with a low credit rating are billed and pay immediately when they check-out from the hotel (cf. Fig. 2); thus workstep checkOut_Bill&Pay (which invokes business process activities checkOut, billAtReception and thereafter payCash or payByCreditCard) must be used. Workstep checkOut (which invokes business process activity checkOut) may be performed only on reservations of customers with a credit rating greater than three (cf. the read-arc to box R-4). Notice that workflow diagram does not show which business process activities a workstep invokes. This information is provided only in the workstep realization diagram.

According to the business process of RESERVATION, one can choose between charging a cancellation fee for cancelled and yet unbilled reservations or not charging a cancellation fee (cf. Fig. 1). Workflow RESERVATION-WF omits the alternative of waiving the cancellation fee. Only workstep chargeCancellationFee (which invokes business process activity chargeCancellationFee) may be performed on cancelled and yet unbilled reservations.

To provide for modularity, a workflow diagram may contain abstract worksteps that are expanded later into a subdiagram. Abstract worksteps are depicted by shaded rectangles.

Example: Figure 3 shows the subdiagram of abstract workstep pay in the workflow diagram RESERVATION-WF.

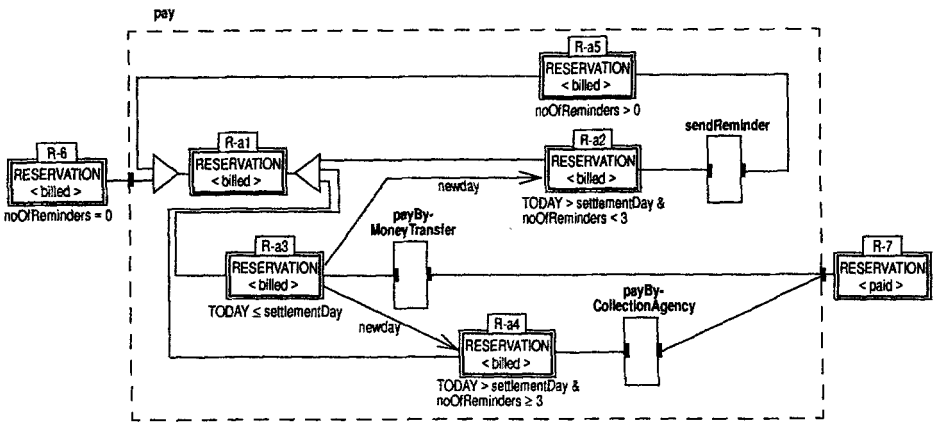

Fig. 3. Subdiagram of abstract workstep pay

Business cases may move between boxes of a workflow diagram although no workstep depicted in the workflow diagram is executed. Transition-arcs, denoted by arrows, are used to visualize such transitions. The transition-arc is annotated by the event or workstep that might cause (but need not always cause) the transition. The condition that causes a transition to occur can be inferred from the specifications of the boxes at the tail and the head of the transition-arc.

Example: Subdiagram pay of workflow RESERVATION-WF (cf. Fig. 3) handles payments of bills and sending of reminders. At the day after the settlement day, billed

reservations which have not been paid so far flow from box R-a3 into box R-a2 or box R-a4, which is indicated by transition-arcs. If a customer has not paid in time and less than three reminders have been sent so far, a reminder with a new settlement day is sent (cf. box R-a2 and workstep sendReminder); if three reminders have already been sent, a collection agency is engaged (cf. box R-a4 and workstep payByCollectionAgency).

The entire workstep specification is depicted in a *workstep specification diagram*. Next to the primary ports, a workstep specification diagram may show some secondary ports (depicted in white) indicating additional data that are needed or are produced by the workstep. These additional data may be internal (e.g., another business case) or external (e.g., a credit card). Secondary ports referencing internal data are connected to boxes the same way as primary ports. Secondary ports referencing external data are connected to ovals that describe the data informally. A workstep specification diagram also describes which business process activities may be invoked by the workstep realization. The activities that may be invoked are depicted next to the workstep together with their pre and post states. Further, additional pre- and postconditions for the workstep are specified textually.

Example: Figure 4 shows the workstep specification diagram of checkOut_Bill&Pay. The secondary input ports indicate that, besides a reservation object, the workstep requires a key card of the customer and a means of payment. The secondary output ports indicate that the workstep returns a means of payment together with a receipt. The business process activities depicted in the right half indicate that not all forms of payment supported by the business process RESERVATION can be used in this workstep but only payments by cash or by credit card. The workstep may only be invoked at the last day of a guest's stay.

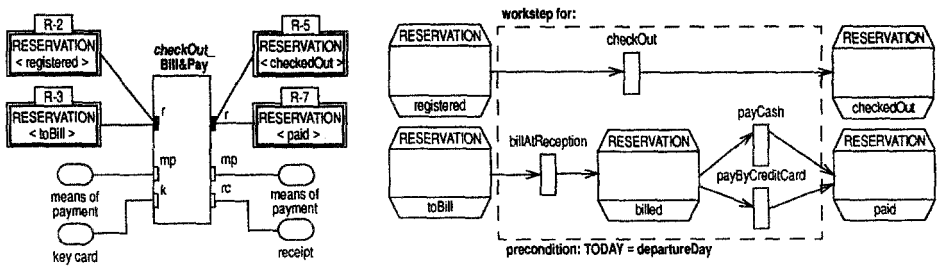

Fig. 4. Workflow specification diagram of checkOut_Bill&Pay

Workstep realization diagrams, which show how a workstep is carried out by a human user or some machine agent, are described in [3].

4 Dynamic Change of Workflows

Dynamic change from an old workflow to a new workflow (or possibly, to several new workflows) comprises three steps: *migration* (some existing business cases are migrated to a new workflow, others are not), *classification* (migrated business cases are placed within the new workflow), and *transition* (some migrated business cases are handled according to transitional regulations in order to be properly included in the new workflow).

Dynamic change of workflows has been investigated in the work of Ellis and Keddara [6] and the work of Casati et al. [4]. We will compare our approach in detail to their work. Related work exists also in the area of software engineering that investigates the problem of evolving software process models. From the numerous software process environments, we will consider in our comparison the SPADE environment [2], which is—to our knowledge—the most flexible one.

4.1 Migration

Migration regulations specify which business cases of a workflow may migrate to what target workflow. A business case may be selected for migration on the basis of (a) its progress in the business process (life cycle), (b) its property values, and (c) its history. Every workflow type offers a generic workstep Migrate that provides the behavior to re-assign a business case contained in some boxes to another workflow. Migrate may be instantiated several times in one workflow. Each instantiation (denoted by Migrate_i) specifies the boxes that must hold the business case to be migrated, a target workflow, and a migration mode, which specifies whether a business case is migrated immediately if it qualifies for migration (i-mode) or not (u-mode). In the latter case, the decision whether to migrate or to continue in the old workflow is left to the user.

Three *migration strategies* can be distinguished for workflows: (1) *No migration:* Existing business cases continue according to the old workflow. New business cases are started with the new workflow. (2) *Static migration:* Static migration requires to decide once in time for all business cases whether they shall be migrated to a new workflow or stay with the old workflow. There is no opportunity to migrate business cases later if they did not migrate immediately. (3) *Dynamic migration:* Dynamic migration allows existing business cases not yet ready for migration to continue according to the old workflow until they qualify for migration. Further, the decision whether to migrate or not to migrate a business case that qualifies for migration may be left to the user.

Ellis and Keddara [6] have investigated the dynamic change problem for the first time in detail and presented a solution to it by static migration. Casati et al. [4] provide a very flexible support for static migration through "progressive case evolution policies" that determine which business cases are migrated and how they are migrated. $OBD_{bp/wf}$ fully supports dynamic migration which is the most powerful strategy. It allows business cases to take advantage of the old and the new workflow during a migration period.

The SPADE environment offers a flexible approach to static migration. The migration policy can be represented as a meta-process (cf. [2]).

Example: Our hotel has considerably changed its workflow for managing reservations (cf. Fig. 5(a)). In future, all customers will be billed immediately after their reservation has been issued. Moreover, they must pay for their bill before they check in. Customers who cancel will get a refund. Ignore the lower half of the workflow diagram, which shows Transition Regulations, for the moment.

Our hotel decided to immediately migrate (1) all reservations of customers which have neither checked in nor cancelled (cf. workstep Migrate_1 in Fig. 2) and (2) all reservations of customers who have already checked out and have paid for their reservation

(cf. workstep Migrate_2 in Fig. 2). A reservation of a current guest, i.e., a reservation in boxes R-2 and R-3 (cf. Fig. 2), will be treated according to the old workflow until the guest has checked out and paid. Then the reservation qualifies for boxes R-5 and R-7 and is migrated. Until later, assume that no other reservations are migrated (i.e., assume that worksteps Migrate_3 and Migrate_4 have not been defined). Note: Figure 2 does not show the target workflow for worksteps Migrate_i. It is assumed that all selected business cases migrate to the same workflow of type RESERVATION-WF-2 (cf. Fig. 5).

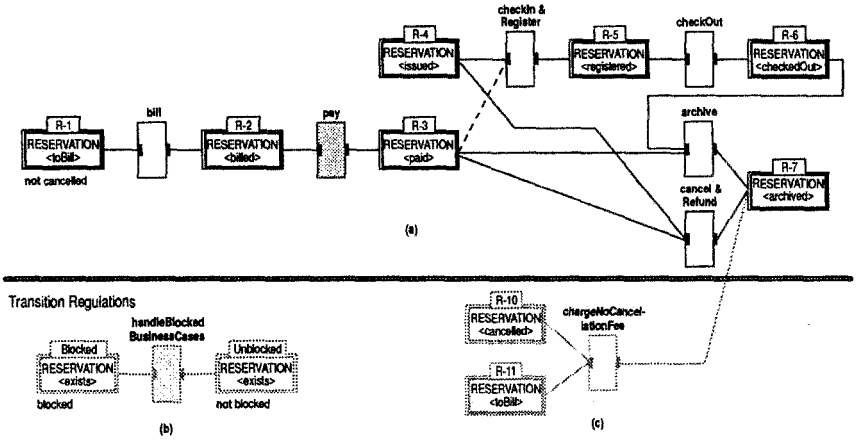

Fig. 5. Workflow diagram for RESERVATION-WF-2

4.2 Classification

Classification regulations specify how migrated business cases are placed within the new workflow. For this placement, two *classification strategies* can be applied: (1) *Static classification:* Each migrated business case is explicitly mapped to a specific "state" of the new workflow. (2) *Dynamic classification:* A migrated business case is classified automatically to that "workflow state" for which it qualifies due to its business process state.

$OBD_{bp/wf}$ applies dynamic classification. To ensure that every migrated business case can be classified, every workflow type offers a box that contains all business cases that are blocked. A business case is *blocked* if it resides in some non-final state (i.e., a state that is pre state of some business process activity) and no workstep is currently performed or can be started for the business case. How blocking can be avoided and how blocked business cases are handled is discussed later.

Example: Business cases migrate from a workflow of type RESERVATION-WF to a workflow of type RESERVATION-WF-2 by worksteps Migrate_1 and Migrate_2 (cf. Figs. 2 and 5). Once migrated by these worksteps, business cases are classified dynamically into boxes R-1 and R-4 or boxes R-3 and R-6 of the new workflow.

Ellis and Keddara [6] offer static classification in the form of a so-called "token mapping function". Casati et al. [4] use also static classification and apply the

same change primitives that modified the old workflow schema to old business cases.

The SPADE environment [2] prompts the user to classify existing "business cases" (instances of software processes) on a case-by-case basis.

To demonstrate the flexibility of dynamic migration and dynamic classification versus static migration and static classification, we compare $OBD_{bp/wf}$ to the approach of Ellis and Keddara [6] by an example. Ellis and Keddara use *information control nets* (ICN) to model workflows. An ICN is a single entry, single exit, directed graph whose nodes are worksteps or control nodes. The latter specify branching, parallelism, or synchronization. The specification of a workflow consists of an ICN and a correctness criterion that defines valid execution sequences for the worksteps in the ICN.

Example: Figure 6(a) shows an ICN that presents a simplified version of the workflow diagram of type RESERVATION-WF (cf. Fig. 2). Its correctness criterion (not shown) defines that workstep checkIn&Register must precede workstep checkOut and that workstep bill must precede workstep pay. Figure 6(b) shows the ICN that corresponds (but is not equivalent to) the workflow of type RESERVATION-WF-2 (cf. Fig. 5). As in the setting of Ellis and Keddara a new workflow must comprise exactly all worksteps of the old workflow, workstep chargeCancellationFee is included in the ICN although it is no longer present in the new $OBD_{bp/wf}$ workflow.

In the approach of Ellis and Keddara, all business cases are migrated at once to the new workflow, and in the absence of dynamic classification, each business case must be assigned explicitly to a specific position in the new workflow. But, if the new workflow has re-ordered some worksteps, old business cases cannot always be mapped into some position of the new workflow without sacrificing correctness. E.g., if reservations of current guests that have not yet checked out are mapped to the position between worksteps checkIn&Register and checkOut in the new workflow (cf. Fig. 6(c)), no bills are issued for them. To take care of such cases, Ellis and Keddara have introduced a concept called *synthetic cut-over rewriting* by which some part of the old workflow (cf. the shaded nodes in Fig. 6(b)) is included in the new workflow. Now, old reservations of current guests that have not yet checked out can be mapped to this "copy" of the old workflow which they follow until they have been billed and paid (cf. Fig. 6(d)).

As $OBD_{bp/wf}$ supports dynamic migration, no part of the old workflow needs to be included in the new workflow if the migration of business cases is delayed until they "fit" into the new workflow. In our example, $OBD_{bp/wf}$ (cf. Fig. 2) achieves the same effect as the ICN solution above by migrating reservations of current guests that have not yet paid only after they have paid (assume that worksteps Migrate_3 and Migrate_4 have not been defined). Further, migrated reservations are classified automatically into all boxes for which they qualify such that the old workflow need not be linked to internal details (i.e., specific boxes) of the new workflow.

Casati et al. [4] also use static migration but combine it with selective migration, i.e., some business cases may migrate and others may continue forever in the old workflow. To cope with business cases that are not "compliant" with the new workflow, a temporary, hybrid workflow schema specifies the worksteps that must be performed such that the migrated business cases can be treated according to the new workflow. Temporary, hybrid workflow schemas are more powerful than the synthetic cut-over rewriting offered by Ellis and Keddara in that they may not only be used to redo worksteps of the old workflow, but also to specify transitional regulations in general (see below).

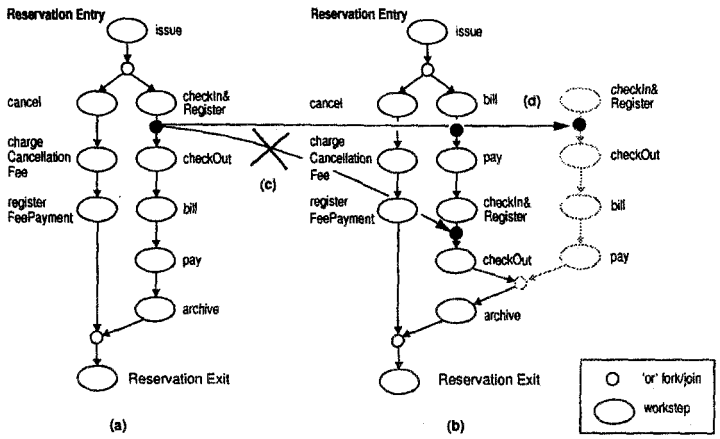

Fig. 6. Dynamic change in the setting of Ellis and Keddara

4.3 Transition

Transitional regulations specify how migrated business cases that are not "compliant" with the new workflow are handled. Three *transition strategies* can be distinguished: (1) *Implicit transition:* Migrated business cases with a history of performed activities that could never be generated in the new workflow may still find a way through the new workflow by applying dynamic classification (see example below). (2) *Ad-hoc transition:* Migrated business cases that are blocked in the new workflow (see definition in Subsect. 4.2) are handled on a case-by-case basis. (3) *Regulated transition:* Transition rules regulate how migrated business cases are taken over to the new workflow. They are especially used to avoid blocking of migrated business cases.

$OBD_{bp/wf}$ supports all three transition strategies. Migrated business cases may float through the new workflow in an "irregular way" without the need of specifying transitional regulations.

Example: Our hotel changed its migration strategy. In addition to its previous strategy, it immediately migrates also reservations of credit-worthy customers who have already checked out but have not yet been billed (cf. workstep Migrate_3 in Fig. 2). These reservations are classified dynamically into boxes R-1 and R-6 of the new workflow (cf. Fig. 5). Notice that a reservation handled solely according to the new workflow could never be in boxes R-1 and R-6. Yet, due to dynamic classification the migrated reservations can be correctly billed, paid for, and archived.

Blocked business cases are collected in box Blocked and are treated by a generic workstep handleBlockedBusinessCases, which allows a human agent to invoke any business process activity that is applicable to the blocked business case.

Example: Our hotel decided to migrate immediately all reservations of credit-worthy customers that have cancelled (cf. workstep Migrate_4 in Fig. 2). Now, assume condition not cancelled has not been defined with box R-1 of the new workflow (cf. Fig. 5). Then, the migrated reservations qualify for this box. After they have been billed and paid for, they will be blocked (workstep cancel&Register cannot be executed as box R-4 will

not contain such reservations). The blocked reservation is handled by a human agent who will refund the payment (cf. Figs. 5(b) and 1).

If condition not cancelled is defined with box R-1, the migrated reservations will qualify for no box of the new workflow and will be blocked. Again, a human agent will take care of these reservations, possibly on a an individual basis. E.g., he may waive the cancellation fee for regular customers and charge a cancellation fee for others by invoking the appropriate business process activities (cf. Fig. 1).

Transitional regulations are specified by a set of boxes and worksteps that transfer migrated business cases to a state in which they may switch to the regular workflow. They may also keep migrated business cases off the regular workflow if their history is not acceptable. Transitional regulations typically involve a set of compensating (or rollback) activities or invoke alternative business process activities not supported by the new workflow. The boxes specified with transitional regulations hold only migrated business cases.

Example: Our hotel decided to waive the cancellations for migrated reservations that have cancelled but have not yet been billed. To assure that such reservations are not billed, box R-1 is extended by condition not cancelled and workstep chargeNoCancellationFee is introduced in the transitional regulations (cf. Fig. 5(c)).

Ellis and Keddara [6] do not discuss transitional regulations. Casati et al. [4] offer a very flexible approach for the definition of transitional regulations. But, they do not support implicit transition and ad-hoc transition. Transitional regulations are specified by temporary, hybrid workflow schemas and are subject to a very strict correctness criterion: A business case may only be treated by the new workflow if its history of performed worksteps contains only worksteps that are also supported by the new workflow. $OBD_{bp/wf}$ is less restrictive. Business cases can migrate to a new workflow that supports different (but alternative) activities since its correctness criterion for dynamic change is based on the common business process which the old and the new workflow implement.

The presentation of the SPADE environment in [2] does not discuss transitional regulations.

5 Conclusion

In this paper we have introduced a two-schema architecture for workflow systems. The business process schema models the logics of *business processes* abstracting from the workflows implementing them. The workflow schema models the actual *flows of work* currently effective in a particular organization to implement the business processes. The two-schema architecture provides for *workflow transparency* which allows to modify a workflow dynamically without effecting the correctness of the business process it implements. In that a workflow may change business cases only by business process activities, any modification of the workflow obeys to the constraints defined in the business process schema. Whereas modifications of workflows in single-schema architectures must be *validated* with respect to their environment, modifications of workflows in the two-schema architecture are implicitly *verified* against the business process they implement.

The two-schema architecture supports workflow evolution in a very flexible way. Business cases handled by the old workflow need not be migrated at once

to the new workflow (*static migration*) but can be migrated from time to time to the new workflow (*dynamic migration*). Business cases which have been migrated to a new workflow need not be assigned explicitly to specific boxes (*static classification*) but qualify automatically for appropriate boxes (*dynamic classification*). Dynamic classification reduces the need to specify transition regulations explicitly as in many cases migrated business cases find their own way through the new workflow.

We believe that other workflow models (e.g., [4, 16, 18]) could be extended to a two-schema architecture in a similar way.

References

1. K.R. Abbot and S.K. Sarin. Experiences with Workflow Management: Issues for the Next Generation. In *Proc. CSCW'94*, 1994.
2. S.C. Bandinelli, A. Fugetta, and C. Ghezzi. Software Process Model Evolution in the SPADE Environment. *IEEE Trans. on Software Engineering*, 19(12), 1993.
3. P. Bichler, G. Preuner, and M. Schrefl. Workflow Transparency. Technical Report 97.02, University of Linz, Department of Business Information Systems, 1997.
4. F. Casati, et al. Workflow Evolution. In *Proc. ER'96*, LNCS 1157, Springer, 1996.
5. I. D'Haenens, et al. Experiences with rule-based dynamic modelling. In *Dynamic Modeling of Information Systems*, North-Holland, 1991.
6. C.A. Ellis and K. Keddara. Dynamic Change within Workflow Systems. In *Proc. COOCS'95*, 1995.
7. H. Herbst. A Meta-Model for Business Rules in Systems Analysis. In *Proc. CAiSE'95*, LNCS 932, Springer, 1995.
8. H. Herbst, et al. The specification of business rules: a comparison of selected methodologies. In *Information Systems Life Cycle*, North Holland, 1994.
9. G. Kappel and M. Schrefl. Object/Behavior Diagrams. In *Proc. ICDE'91*, 1991.
10. G. Kappel and M. Schrefl. Modeling Object Behavior: To Use Methods or Rules or Both? In *Proc. DEXA '96*, LNCS 1134, Springer, 1996.
11. G. Kappel and M. Schrefl. *Objektorientierte Informationssysteme: Konzepte, Darstellungsmittel, Methoden.* Springer, 1996. (in German).
12. P. Loucopoulos, B. Theodoulidis, and D. Pantazis. Business Rules Modelling: Conceptual Modelling and Object-Oriented Specifications. In *Object Oriented Approach in Information Systems*, North-Holland, 1991.
13. P. McBrien, et al. A Rule Language to Capture and Model Business Policy Specifications. *Proc. CAiSE'91*, LNCS 498, Springer, 1991.
14. B. Meyer. *Object-oriented Software Construction.* Prentice Hall, 1988.
15. G. Saake and R. Jungclaus. Views and Formal Implementation in a Three-Level Schema Architecture for Dynamic Objects. In *Advanced Database Systems, BNCOD 10*, LCNS 618, Springer, 1992.
16. A.-W. Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises.* Springer, 2nd edition, 1994.
17. K.D. Swenson. Visual Support for Reengineering Work Processes. In *Proc. COOCS'93*, ACM Press, 1993.
18. D. Wodtke, et al. The Mentor Project: Steps Towards Enterprise-Wide Workflow Management. In *Proc. ICDE'96*, 1996.