A workflow is a long-duration multi-step activity. In this paper we are interested in workflows that execute under the control of various processing stations that may be located at different nodes of a distributed system. The stations may be autonomous and only partially automated. We present the design and a proposed implementation of a new model for workflow management that is based on a concept of an Information Carriers (INCA). Workflow computations are carried out as interactions between INCAs and the processing stations with the locus of control of a computation migrating with the workflow. The model presented is modular in the sense that modification of a sub-activities of the workflow does not necessarily require changes to the workflow specification. Furthermore, the model preserves the autonomy of the processing stations and does not require them to change the means they use to process the individual steps of the workflow activity.

Workflow management has recently received a lot of attention within the database research community. (For a survey of recent work on workflow management readers are referred to in Hsu (1993)). A workflow is a long-duration multi-step activity with some data and control flow between its constituent steps. A typical example of a workflow is travel expense reimbursement in an office. Other examples of workflows include admitting a patient into a hospital (Dayal et al., 1990) or service order provisioning in the telecommunications industry (Ansari et al., 1992). In this paper, we develop a computation model for workflows in dynamically evolving autonomous office environments. These environments exhibit the following features:

**Autonomy**: Each processing station is autonomous in the way it executes the task on behalf of the workflow. That is, a processing station must be treated as a “black-box” whose mode of operation can be neither changed nor fully known to other processing stations, or to the workflow designer. For example, the exact sequence of operations that execute at the claims office to process a travel expense reimbursement is known only within the claims office. As a result no single person (or system) has complete knowledge of all the steps that comprise a workflow. A workflow is executed as a result of interactions among such partially cognizable processing stations.

**Dynamic Control Flow**: The processing stations may evolve with time, resulting in changes to the control and data flow of the workflow. For example, the claims office may change its policy to reimburse travel expenses such that if the total amount claimed by the employee of a department exceeds a certain limit, then the reimbursement must be authorized by the vice-president in charge of the department. Thus, the course of a workflow may alter in an unpredictable manner during its execution on account of modifications to the processing rules and policies. This means that no workflow can be fully specified *a priori* either by a fixed script or by a
complete set of processing rules.

**Partial Automation:** The environment in which the workflow executes may be only partially automated. It is possible that certain processing stations are manual while others are fully automated units. Even though systems are automated, they may only have limited communication support. For example, it is possible that only mode of communication to file travel claims is to send a fax containing the relevant information to the claims office.

**Partial Connectivity:** Some of the processing stations on which the workflow executes may be portable computers and/or mobile units that are disconnected most of the time and may connect to the network only occasionally, for short periods, and only at certain (possibly unknown a priori) points in time. In such environments, the underlying computation model cannot assume that all nodes containing data and processing elements are constantly connected to each other and that “network partitioning” is a transient problem. In such environments, network partitioning may be the normal mode of operation and the network may never be fully connected. For example, the processing station at a hospital may never be connected to the processing station at a pharmacy, except when a patient carrying a smart card (McCredle, 1990) (a card that contains a limited amount of memory and can be read and written by a computer) containing his medical records visits a pharmacy to fill a prescription.

Many workflow models proposed in the literature do not address all these issues. In this paper we present a new workflow computation model that is well suited to dynamically evolving environments in which the constituent processing stations may be autonomous and only partially automated. A fundamental concept in the proposed model is the notion of an Information Carrier (INCA). Workflow computations are carried out by submitting INCAs to the processing stations. An INCA contains information about the service requested and the relevant context for the execution of the service. It also contains rules that are used to code the control and data flow, as well as the failure atomicity requirements of the computation. Each processing station, on receipt of the INCA, performs the service requested and, as part of the service, may add, delete, or modify the INCA data and rules, as well as its own data and rules. After performing the requested service, the processing station determines the next set of destinations (using the INCA’s rules and data as well as its own data and rules) and routes the computation (that is, forwards the INCA) to the new destination using the appropriate communication medium. The rules associated with the INCA computation (encoding the control and data flow and the failure atomicity requirements) migrate from one processing station to another along with the INCA. In some sense, INCAs formalize and automate the concept of office forms that are commonplace in many work environments. By doing this, the INCAs provide a suitable paradigm to deal with truly distributed and autonomous environments.

Since rules are used to encode the control and data flow of the INCA computation and furthermore, since the processing stations may add, delete, or modify the INCA rules and data as well as their own rules, the path of the INCA computation and the actual actions performed at each processing station may not be known in advance, but are determined by the interactions between the activity and the local processing station. In particular, the path of the computation may dynamically change due to the execution of a local rule, or due to the modification of the INCA rules. This makes our model suitable for dynamically evolving environments.

Using rules to model control and data flow of workflows in dynamic environments has been previously proposed in Dayal et al (1990; 1991). What makes our computation model unique is that, unlike Dayal et al (1990; 1991), in which a workflow executes under centralized control, in our model the locus of control of a computation migrates between the processing stations. This makes our computation model suitable for decentralized environments, since it permits the local procedures and local rules of the processing stations to execute completely under their control. Furthermore, supporting our model requires minimum cooperation from the processing stations. All we require of the processing stations is that they understand the INCA format and participate in the protocol of receiving an INCA, performing the requested service, and forwarding the INCA to the appropriate destinations using the correct communication media. As will be shown, if a processing station is unable to participate in the above protocol, it can be encapsulated with a software module that provides the above functionality.

Our computation model, based on submitting INCAs for the execution of the steps of a workflow, also provides a natural framework for modeling workflows in environments in which the constituent systems are only partially automated. In particular, the specifications of a workflow in an INCA do not depend on how the local services are provided and do not need to change when manual operations are replaced by an automated system. Also, we do not assume that the processing stations are connected at all times by a network as certain extended transaction models in distributed environments. An extensive example illustrating how workflows execute under our computation model appears in Barbara et al (1991).

**Related Work**

A number of workflow models have been described in the past (Zism, 1978; Chang, 1982) in the context of office automation. All these models view a workflow as a set of steps (also referred to as procedures, or tasks) with control and data flow between the steps explicitly specified using a suitably augmented Petri net or triggers. The focus is to develop a model that allows the designer to analyze the correctness of the workflows. (For example, one can reason about the absence of deadlocks in an office procedure specification.) Most