A Constraint Specification Approach to Building Flexible Workflows

Peter J. Mangan and Shazia Sadiq

School of Information Technology and Electrical Engineering
The University of Queensland
QLD 4072 Australia
Email: pmangan@optushome.com.au, shazia@csee.uq.edu.au

Process support systems, such as workflows, are being used in a variety of domains. However, most areas of application have focused on traditional production-style processes, which are characterised by predictability and repetitiveness. Application in non-traditional domains with highly flexible process is still largely unexplored. Such flexible processes are characterised by lack of ability to completely predefine and/or an explosive number of alternatives. Accordingly we define flexibility as the ability of the process to execute on the basis of a partially defined model where the full specification is made at runtime and may be unique to each instance. In this paper, we will present an approach to building workflow models for such processes. We will present our approach in the context of a non-traditional domain for workflow deployment, which is, degree programs in tertiary institutes. The primary motivation behind our approach is to provide the ability to model flexible processes without introducing non-standard modelling constructs. This ensures that the correctness and verification of the language is preserved. We propose to build workflow schemas from a standard set of modelling constructs and given process constraints. We identify the fundamental requirements for constraint specification and classify them into selection, termination and build constraints. We will detail the specification of these constraints in a relational model. Finally, we will demonstrate the dynamic building of instance specific workflow models on the basis of these constraints.

Key Words: workflow, flexible, tertiary domain, constraint specification
Classification: I.6.5 Model Development

1. INTRODUCTION

Effective business organisations develop processes to ensure that their activities are performed in a consistent and reliable manner. These processes consist of four primary components, namely objects, tasks, performers and constraints (Sadiq and Orlowska, 1999a). The object is a business entity, such as a home loan application, which is required to be worked on during its life cycle. The business activities defining this work form the tasks to be performed, such as checking the applicant's credit rating or current financial commitments. Performers carry out these tasks on the

Manuscript received: 1 May 2002
Communicating Editor: Professor Maria Orlowska

objects, as the task cannot perform its associated work on the object directly. A performer may be a person, typically someone filling an organisational position, such as a manager or a supervisor, or it may be another form of resource such as a database or computer application. Constraints are required to ensure that the work contained in the tasks which is carried out by the performer on the entity, is done correctly to ensure the integrity, reliability and consistency of the business process. Information systems in the business community are currently shifting from a data centric approach towards a process centric approach that controls and coordinates these processes.

Workflow systems are currently the leading technology for supporting business processes. This technology manages the execution of the tasks involved in a business activity, the scheduling of resources and the control of the flow of the associated information required by performers to execute the tasks. Typically the tasks involved in the business process are interdependent in that the execution of one task is conditional upon the execution of one or a number of other tasks. Workflow management systems employ a process model to capture this flow of execution between tasks. This model is used by a workflow management system to schedule and coordinate the execution of these tasks. Production workflows based on this framework have been developed to automate the coordination of the activities for processes that are typically characterised by well-defined procedures and are highly repetitive in nature. The processes may be complicated in nature, involving a large number of tasks, performers and coordination constraints. However many new categories of workflow technology are emerging to address the diverse range of processes now looking to engage the technology due to the automated coordination benefits that it provides.

The ability to predefine a business process completely cannot be relied upon. Furthermore even highly predictable processes are subject to change as organisations adjust their activities in response to influences such as new legislations, innovations and competitive pressures. Another pressure on the rigid process definition comes from the need for support of exceptional cases, in particular unforeseen scenarios that cannot be suitably addressed by the existing process definition. At the other extreme of a continuum of support for process change is the complete relaxation of coordination constraints, which has led to developments in ad-hoc workflows to cater for complete flexibility in the execution. Positioned between the extremes of this continuum is the support for processes that can be partially defined but not completely specified until runtime. Several applications exist where the complete definition of the process cannot be eloquently predefined:

- A typical example is healthcare, where patient admission procedures are predictable and repetitive, however, in-patient treatments are prescribed uniquely for each case, but none-the-less have to be coordinated and controlled.
- Another application is higher education, where students with diverse learning needs and styles are working towards a common goal (degree). Study paths taken by each student need to remain flexible to a large extent, at the same time providing study guidelines and enforcing course level constraints is necessary to ensure a certain quality of learning.
- Web content management is also characterised by flexible processes, where especially in large projects, every development suggests the need for an overall plan to provide the objectives, approvals, and strategy, as well as a flexible means of coordinating the combined efforts of the theme designers, graphic experts, programmers, and project planners.
- Effective Customer Relationship Management (CRM), a critical component in enterprise solutions, also signifies the need to provide a flexible means of composing call center activities according to the available resources and data, but within certain constraints, thus allowing for an integration of CRM systems with core organisational workflow processes and underlying applications.

We will introduce in the following section, some basic terminology and a simple workflow modelling language. In the subsequent sections we will present an analysis of flexible processes in the context of tertiary degree processes, and identify the fundamental requirements for constraint specification for such processes. We will also demonstrate the unsuitability of "prescriptive" process modelling languages in this area. We will introduce our proposed approach and show how flexibility of specification can be achieved without compromising the simplicity and consequently verifiability of the modelling language. Finally we will describe how a relational model can support the required constraint specification.

2. BASIC TERMINOLOGY

We will use basic workflow modelling concepts to demonstrate our approach. Below we introduce basic terminology for the sake of clarity:

We define a workflow *schema* as a representation of a set of activities, the ordering and inter-dependencies between these activities, the resources available to perform them and their information or data requirements that are required to complete a business process.

A workflow *instance*, like its database namesake, denotes a particular occurrence of the business process as defined by the schema.

An *instance type* is the set of instances that follow the same execution path through the conditions that exist in the workflow schema.

2.1 Semantics of the Workflow Language

We also introduce a graphical workflow definition language (Sadiq and Orlowska, 1997) which will be used for demonstrating various concepts. This language conforms closely to the workflow management coalition standards (Workflow Management Coalition, 1995). Note that we will utilise only the basic set of modelling constructs, consisting of Sequence, Exclusive Or Split (Choice) Exclusive Or Join (Merge), And Split (Fork), And Join (Synchronizer). The graphical representation of these constructs is shown in Figure 1.

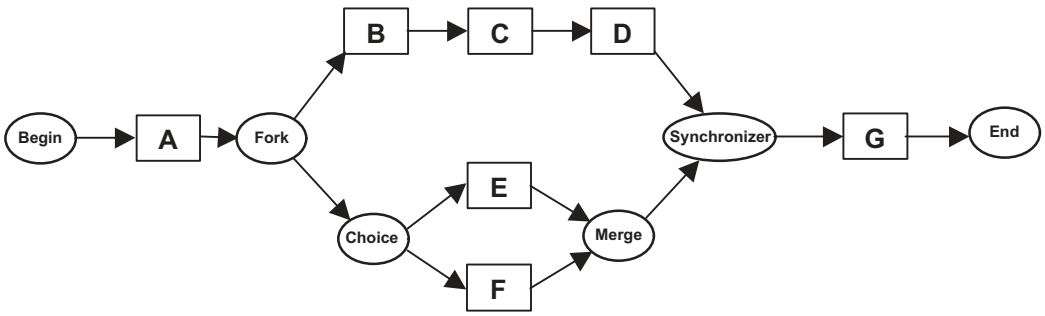


Figure 1: Basic Workflow Constructs

The semantic context of the basic constructs is presented with reference to the process depicted in Figure 1.

- Sequence enables an activity in the workflow after the completion of another activity in the same process as shown by activity C's dependency on Activity B.
- And Split (Fork) supports simultaneous execution in the process by allowing a single thread of control to split into multiple threads that can be executed in parallel. The branch containing

Activities B, C and D can execute simultaneously with the branch containing Activities E and F.

- And Join (Synchronizer) brings together multiple parallel threads into a single thread of control. Activity G will only trigger after the incoming transitions from the branch containing Activity B,C and D and the branch containing Activity E and F are triggered.
- Exclusive Or Split (Choice) represents a point in the workflow process where one of several branches is chosen based upon the results of a condition. That is, Activity E or Activity F will follow Activity A in this branch of the process.
- Exclusive Or Join (Merge) allow alternate branches to come together without synchronization. Completion of either Activity E or Activity F will trigger the merge.

The above defines the *structural* layer of the workflow that captures the flow of execution from one task to another. In addition a *temporal* layer is required to capture scheduling constraints additional to those imposed by the structure, a *data* layer to capture the data requirements of workflow tasks, and data flow between tasks, and an *execution* layer to schedule and control the actual tasks invocations, resources and data. Therefore, each task in the workflow is described by a set of properties. These properties relate to the data, time, underlying applications, resources, clients, compensation and much more. We do not elaborate on task properties in this paper. However, the task is a complex object with rich semantics, and cannot be considered as a mere node in the workflow graph.

3. RELATED WORK

The work presented here is closely related to developments in a number of areas of workflow technology. Greater support for flexibility using the descriptive model may be possible through extensions to the workflow definition language. Possibly as a response to commercial concerns, work has been undertaken to extend this language to support more complex routing and control primitives not possible with the existing basic constructs. For example the basic Or-Join (Merge) construct could be extended to allow the subsequent activity to activate once *N* paths of the *M* paths converging into the merge have completed. Completion of the remaining paths would be ignored. (van der Aalst, Barros, Hofstede and Kiepuszewski, 2000). The various contemporary workflow management systems support different levels of expressive power in the workflow definition languages developed for them to capture the more complex requirements that recur frequently in business processes. Given the fundamental differences this introduces to a workflow management system, use of these language extensions will have implementation consequences resulting in the loss of genericity and limited verification support.

Within well defined fixed processes the need to support dynamic change as the business process evolves in response to competitive and regulatory influence, still exists (Agostina, de Michelis, 2000). The biggest problem is the handling of the active instances that were initiated in the old model. Defining a migration strategy that preserves correctness of active workflow instances is a complex problem and has been the target of extensive research (Casati, Ceri, Pernici and Pozzi, 1996; Kradolfer and Geppert, 1999; Liu, Orlowska and Li, 1998; Sadiq, 2000a; Joeris and Herzog, 1999).

Furthermore the need to handle exceptions to cater for instances that cannot be anticipated at design must also be supported. As the support for this level of change is required in the traditional workflow implementations considerable work has been devoted to it also (Reichert and Dadam, 1997; Sadiq, 2000b). Approaches to the problem vary and include less restrictive model definitions (Joeris, 1999) and using generic process models which define a family of variants of the same workflow process (van der Aalst, 1998). Another aspect of work in this area is based on the concept

that an exception may result in a change in the level of specificity of the process. For example a highly specified process of processing customer orders may move to a highly unspecified collaborative process involving emails and consultations as a result of strike conditions at a particular manufacturing plant (Bernstein, 2000). The developments supporting these dimensions of change are based on the underlying assumption that the change is exceptional. In the processes that our work aims to support the change is inherent in the process. Consequently the overheads of these developments are too large to implement on a process that is to be dominated by change that cannot be predefined.

The concept of using a rule based method for modelling workflows is described in (Knolmayer, Endl and Pfahrer, 2000). The approach takes a rule based description of a business process and transforms it, by applying several refinement steps, to a set of structured rules which represent the business process at different levels of abstraction, that is a rule-based workflow specification. The underlying concept of developing a workflow specification from a set of rules describing the business processes is similar in principle to the work presented here. However the approach is primarily directed towards the development of coordinated processes that span enterprise boundaries by providing a layered approach that separates the transformation of business (sub-) processes and the derivation of workflow specifications and does not address the issue of catering for processes that cannot be eloquently predefined.

To accommodate change in organisational procedures, tighter integration between the workflow specification and the enactment modules is recommended (Ellis and Keddara, 2000). Under such a framework the change itself can be treated as a process that can be modelled and enacted. The modelling associated with this work represents the elements of a workflow such as the process and its activities and rules as object classes. It also supports a variety of pre-defined change schemes, which replace sections of the old version of the procedure affected by the change. In particular an Ad-hoc scheme to support changes where the change specification is precisely completed after the change process is enacted. Although the tighter integration between specification and enactment, and the enactment on a partial specification seen in this work are reflected in the framework proposed in the work presented here, it relies on a prescriptive definition of the processes rather than a rule based generation of valid models.

The Object orientated paradigm also appears in the enterprise modelling where companies are restructured around a number of business operations and processes to accommodate process design change and facilitate the synthesise of business processes out of fragments (Papazoglou and van den Heuvel, 2000).

Moving to the other end of our continuum for organisational processes that spans from highly specified and routine processes to highly unspecified and dynamic processes we acknowledge the significant work that has been performed in the coordination of collaboration intensive processes (Bogia and Kaplan, 1995). The complete relaxation of coordination, to support ad-hoc processes is not conducive to the processes targeted by our work. However, structured ad-hoc workflows, which are identified as ad-hoc workflows where patterns can be derived form the activities in the process as a result of underlying rules to achieve certain goals, are the foundation for a related workflow paradigm shift. The paradigm shift allows the workflow system to derive the workflow incrementally from workflow fragments and avoids the need to predefine the process prior to enactment (Han and Shim, 2000). The completion of a structured ad-hoc workflow instance allows flows to be derived for other instances of workflows that share the same process rules. Although this work is concerned with a Connector facility to interconnect workflows from manual and automated departments the concept of defining parts of a process incrementally rather than enacting

on a predefining process is similar to the underlying assumption in our work. However the development of this concept to address the modeling of processes that cannot be eloquently predefined contains significant differences as a result of the rules being more explicit and consistent across instances and the fragments having a much smaller granularity.

More closely aligned to the traditional workflow paradigm are developments in the areas of evolving workflows (Herrmann, 2000; Sadiq, Sadiq and Orlowska, 2001). This work focuses on processes where only part of the procedure can be adequately predefined and the balance is non-procedural content. The specification of the selection and sequencing of the activities of the unchoreographed content, which involves resources and collaboration, is supplied by the performers at the time the tasks are invoked. This work relates closely to the work presented in this paper but is distinguished from it in that the unchoreographed content is contained to specific sections of the predefined process.

4. ANALYSING FLEXIBLE PROCESSES

In this section we will present an analysis of flexible processes through the tertiary education domain. Today's student communities are constantly changing, with more and more part time, mature age and international students with a wide variety of educational, professional and cultural backgrounds. These students have diverse learning needs and styles. Where as degree programs are generally well defined in terms of overall process constraints, it is difficult to judge the validity of specific choices made by students.

Tertiary programs often offer a diverse collection of courses that allow specialisation on various aspects of a program. The wide variety of valid combinations of courses that satisfy a particular program's requirement indicates a high degree of flexibility. A simple program structure consisting of nine courses of compulsory material and a further three courses of elective material which are selected from a schedule of 14 available electives was considered and found to yield a total of some 364 instance types. A further illustration considers a less structured program, such as Arts or Science, where some 20 to 30 courses are required from a schedule that can contain thousands of courses. A number of factors impact on the choices made by the students including changing areas of interest, changing workload requirements and changing program rules. The multiplicity of valid combinations of courses that can be undertaken, and ensuring that these satisfy the requirements of programs, particularly where these requirements have changed during the duration of the student's enrolment constitute a complex problem.

Although academic courses are not currently deployed as workflow tasks in the typical sense, the appropriateness of workflow modelling concepts can be easily seen. Academic courses equate to process tasks, these courses are interdependent and the academic program represents a long duration business process. At the same time, there is an inherent flexibility in these processes, required for diverse student requirements, which makes their modelling in traditional process definition languages very difficult.

In Figure 2 we give an example of a degree program with nine compulsory and three electives out of 14 offered courses. This Figure partially captures one of the eleven sub-processes required to model the program and is intended to demonstrate how a prescriptive language generates an explosive number of instant types.

Due to the above factors, we found this to be an appropriate and challenging domain for investigating the modelling of flexible processes.

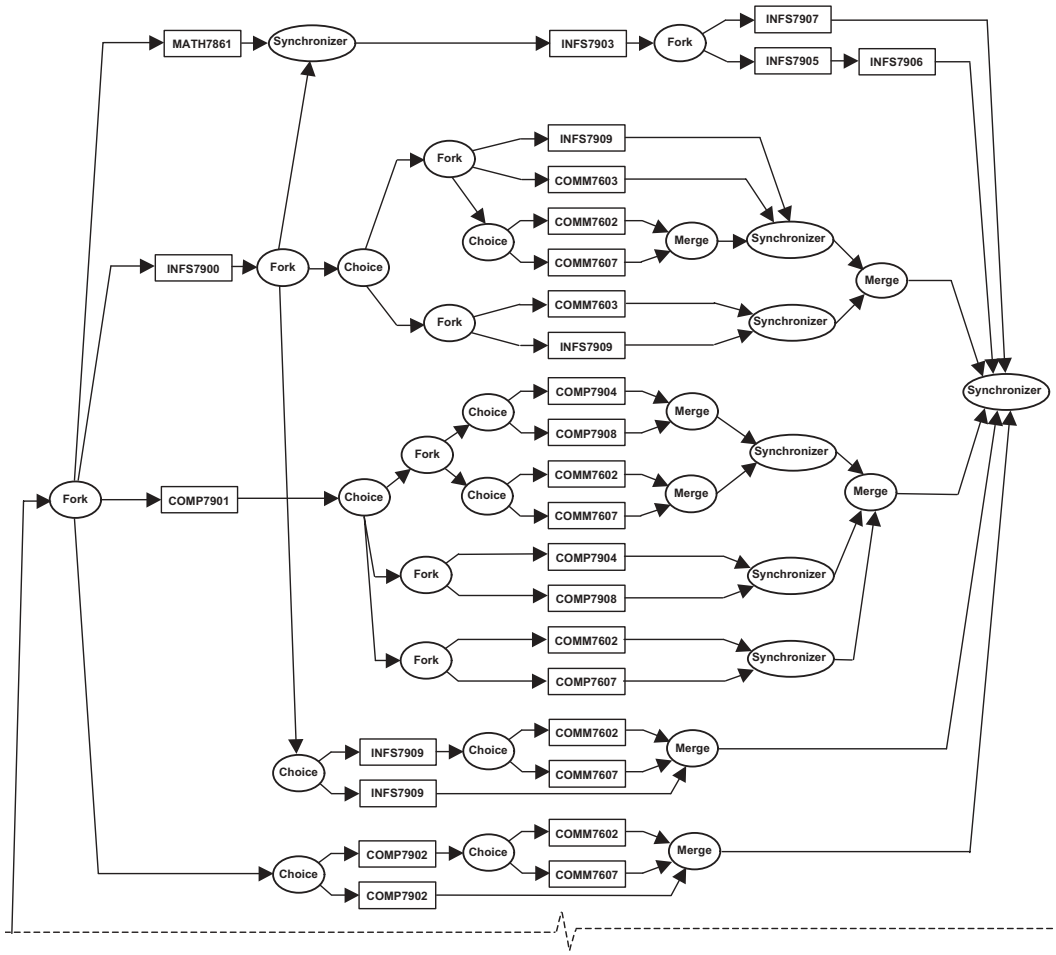


Figure 2: Modelling Flexible Processes in a Prescriptive Language

5. DEFINING THE FLEXIBLE MODEL

Basically we define flexibility as the ability of the workflow process to execute on a partially defined model which is only fully specified at runtime and may be unique to the instances (Sadiq *et al*, 2001). Although the fundamental question we are addressing is the modelling of the partial (rather flexible) model, it is important to point out that the flexible model is to be used to build valid instances. We identify the following as the essential components of the flexible model:

- A set of fragments from which an instance can be built.
  - a workflow task
  - a complex sub-process
- A set of modelling constructs that can be used to compose the fragments for a given instance.
  - Sequence
  - Fork/Synchronize
  - Multiple executions

In this paper we will utilise two constructs, namely sequence and fork/synchronize in the context of the chosen domain. However, at a more generic level, we see the use of structures that allow multiple executions in sequence and/or in parallel. While choice and merge constructs may be present within workflow fragments, we propose that these constructs not be used to build the instances. Since an instance represents a particular occurrence of the workflow process, the choices should be made during the building of the instance. The elimination of the choice-merge construct from the instance template, further has the advantage of simplifying the model, and removing the chance of deadlocks or lack of synchronization (Sadiq and Orlowska, 1999 b).

- A set of constraints that will define the rules under which valid instances can be built. We identify three levels of constraint specification.
  - Selection Constraints
  - Termination Constraints
  - Build Constraints

5.1 Selection Constraints

Selection constraints dictate the inclusion of a fragment in the pool of fragments available for building. In degree programs, fragments are simply courses. A course is available for selection (included in the pool of fragments available for building) when certain constraints have been met. In this domain we identified three types of constraints which are basically interdependencies that exist between courses. These are prerequisite, companion and incompatible.

- Prerequisite requirements for a course describe its dependency on the completion of a set of courses. These prerequisite requirements may take the form of satisfactory completion of
  - a specific course
  - a number of specific courses
  - a number of subjects in an area of study or at a study level or in an area of study at a level
  - advanced standing
  - one of a number of alternatives, where each alternative may be in any one of the above forms
- Companion requirements describe courses that are dependent upon other courses being undertaken in concurrence with them.
- Finally a course may be incompatible with another course. Successful completion of an incompatible for a course will prevent the credit for that course.

These dependencies can be depicted as a control flow structure in the given workflow language, as shown in Figure 3.

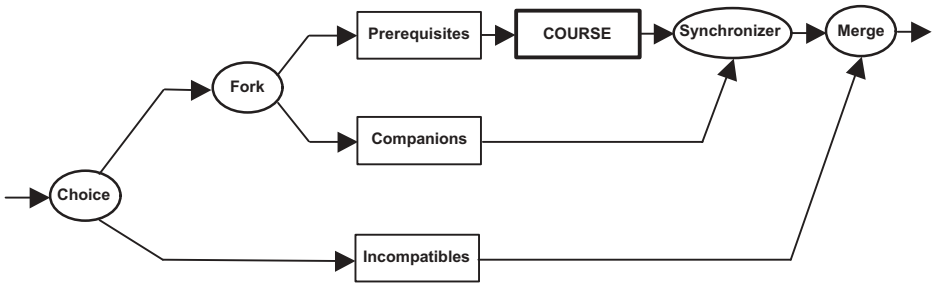


Figure 3: Higher Education Domain Task Interdependencies

5.2 Termination Constraints

In the absence of an explicit termination task, it is essential to identify the termination constraints, which define to a large extent the process goal. Furthermore, these constraints must be defined in terms of the fragments from which the instances are built. In the higher education domain, we can identify the following termination constraint:

- The total number of courses (units) to be completed is always specified but some programs may require that a specified number of units be completed in a particular area or at a particular level or a combination of both.

5.3 Build Constraints

In addition to the rules captured by the selection and the termination constraints, further restrictions exist to control the building of the process. The build constraints specific to this domain are typically:

- rules specifying the availability of course (semester 2, alternate years).
- rules specifying the minimum and maximum student workloads (units taken per semester).
- rules specifying performance requirements. For example, entry into honours programs may require an overall grade average or a grade average taken over a number of courses.

6. BUILDING VALID INSTANCES

The dynamic build approach allows for the continuous build of an instance from the set of available fragments and modelling constructs. The absence of a constraint violation permits the build to continue. Initially the instance specification will be undefined and the pool of fragments available to the build will contain only those fragments without selection constraints. Specification of the instance can begin with a selection of the available fragments connected through the constructs to form what we call the *open instance*. The instance will remain open until such times as the fragments it is comprised of satisfy the termination constraints of the process. The closed instance specification therefore defines a valid instance type that may be adopted by future instances of the process. Building of the open instance is progressive and dynamic. The set of fragments available to the build activity, at a point of time in the build, is dependent upon the fragments that exist in the open instance at that point of time. As fragments are added to the open instance more fragments have their selection constraints satisfied by it and are added to the pool of available fragments. This is demonstrated in Figure 4 by an example of a set of five fragments of which two are available through the satisfaction of their selection constraints by the fragments contained in the open instance. In the Figure it should be noted that the selection constraints for each fragment have themselves been represented in the workflow modelling language for clarity of the example.

The validity of the generated instance, that is its structural correctness, is ensured by the restriction of the constructs to sequence and fork/synchronize. In the absence of choice/merge constructs it is not possible to introduce deadlock and loss of synchronism into the structure. The semantic correctness of the composition of the fragments is ensured by the selection constraints governing the availability of fragments, as well as the build and termination constraints. For example, a fragment is only available if the selection constraints identify it. In Figure 5, Fragment E cannot be included in the process as the completion of Activities A and B is required by the process before activity E can be executed. This is captured by the selection constraint for Fragment E that hides this fragment from the process until such time as the open instance supports it.

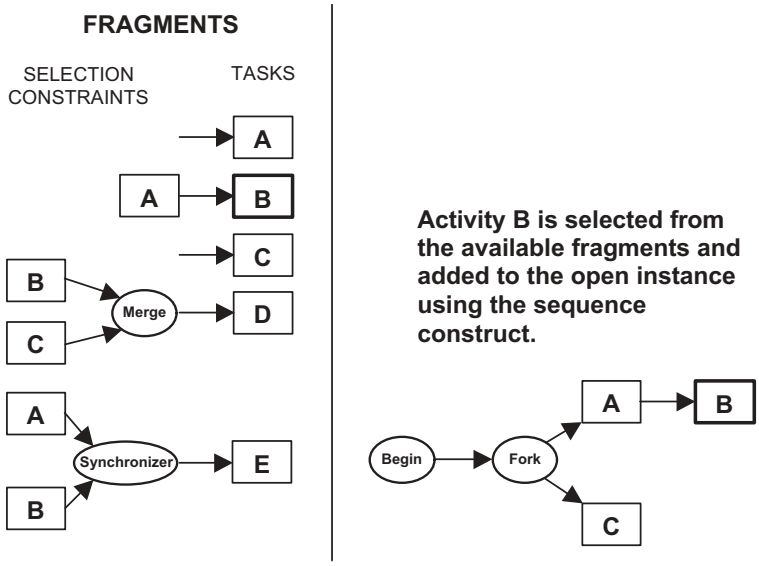


Figure 4: Dynamic build using fragments with satisfied selection constraints

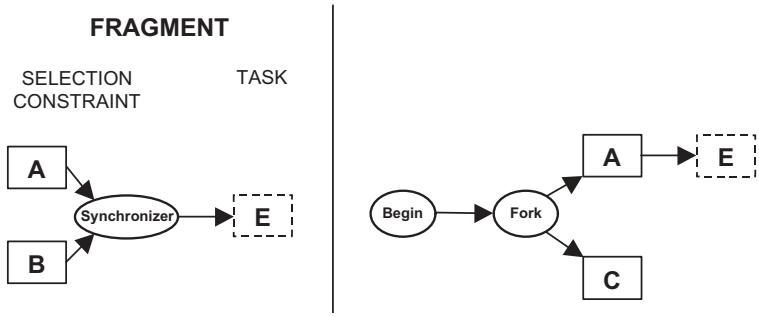


Figure 5: Enforcement of semantic correctness in the open instance

The build constraints capture the restrictions on the extent of building that are not enforced by the selection constraints. Checks to ensure adherence to these rules will be required after each step in the build process. For example a student may not be permitted to undertake more than three courses concurrently. The application supporting the build will prevent structures being generated in the open instance which violate this constraint. The termination constraints that define the process goals are enforced by checks on the instance data of the open instance under development. Once the instance data check identifies that a termination constraint has been satisfied, the instance is allowed to complete. A simple example is presented in Figure 6. In the example the termination constraint is satisfied if any four fragments from the set Fragments (A,B,C,D,E) are successfully completed.

It is important to note that the approach has very little impact on the functionality of the underlying workflow engine. The open instance generated by the build acts as a process model. Execution takes place with full enforcement of all coordination and temporal constraints. A constraint specification language that can be integrated with the workflow definition language, thus extending the functionality of the process definition tool, to allow the building of valid instances as demonstrated above is detailed

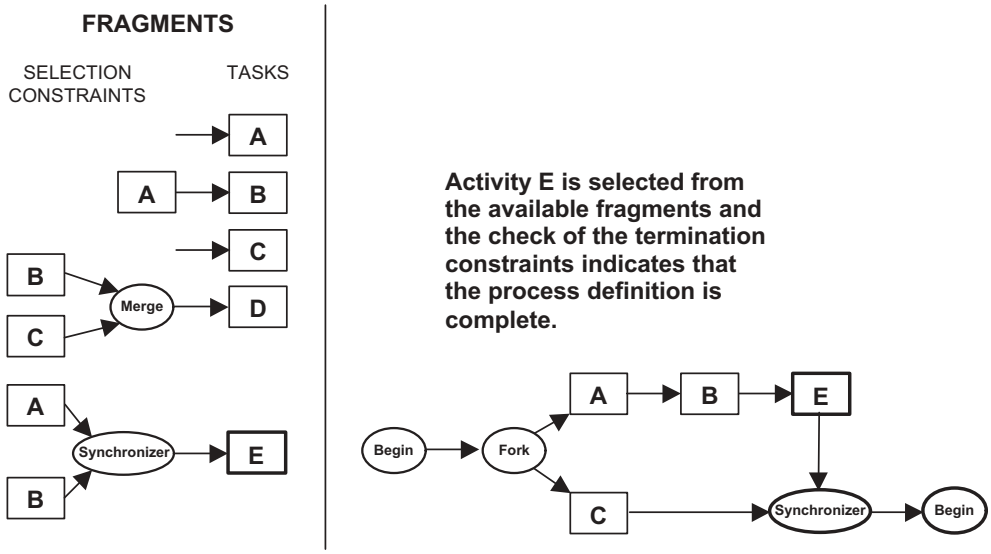


Figure 6: Enforcement of Termination constraints

in the following section. We do anticipate the need for an interface between the process definition tool and the workflow scheduler, so that the open instance can be changed (built) and deployed interactively during the execution of the flexible workflow process. However the facility to access an instance copy of the process model is already being provided by several workflow management systems.

7. CONSTRAINT SPECIFICATION IN A RELATIONAL MODEL

7.1 Constraint Specification

The extension of process definition tools to support flexible processes is dependent upon the existence of a constraint specification language that integrates with the process definition language to capture the constraints defining the rules under which valid instances can be built.

The three levels of constraint specification that have been identified, namely selection, termination and build, have been implemented using a language based upon a relational database structure. A relational data structure may not be the most efficient storage for all types of data structures (e.g. graphs). However they are appropriate in this application as the powerful query optimisers associated with Relational Database Management Systems (RDMS) enables them to efficiently evaluate the queries associated with constraint enforcement. Furthermore, the low cost of storage negates the need to use alternative storage structures.

The specification of constraints using the relational data structure is based on the assumption that each constraint is a composition of one or more *elementary conditions.* An elementary condition identifies a single atomic requirement of a constraint. A constraint is formed by composing these elementary conditions using logical AND and OR constructs to form a *clause.* Each elementary condition is one of two identified types, namely

- Task conditions
- Instance Data conditions

A *Task condition* is satisfied by the completion of specific tasks, for example

Constraint X → Task B

states that Task X can be undertaken upon completion of Task B.

An *Instance Data condition* is satisfied by the completion of non-specific tasks which have characteristics, described by attributes, that meet the requirements of the constraint, for example,

> Constraint X → 2 Tasks of LEVEL = x
>
> states that Task X can be undertaken upon completion of any two tasks which have a value of x for their "level" attribute.

Instance Data conditions are more complex than the simple existence requirements of the Task conditions. They are comprised of a *description* such as "Tasks of LEVEL = x" and a *requirement* such as "count >= 2" from the above example. Both components contain an *operator* such as ">" or "=" and a *value* such as "2" and "x" in the above example. The requirement consists of a further element known as the *function* such as "count". A set of functions containing the following elements is considered complete for the purpose of constraint specification; {Count, Sum, Average, Maximum, Minimum}

A set of operators containing the following elements is considered to minimal. From this base set more advanced operators such as "between" could be composed; {Equal to, Not equal to, Greater than, Less than}

Two special keywords exist for the value field: ALL, EACH

The keywords are used to compose constraint enforcement queries that test a condition against ALL values of a parameter and against EACH value of a parameter.

A constraint in its simplest form is a single elementary condition as in the previous examples. Alternatively, it may be satisfied by a clause, that is the concatenation of a number of elementary conditions, for example

> Constraint X → Task B AND 2 Tasks of LEVEL = x
>
> states that the Constraint X is satisfied upon completion of Task B and any 2 tasks which have a value of "x" for their "level" attribute (assuming that Task B does not have a value of "x" for its "level" attribute).

An elementary condition or a clause is termed an *exclusive condition* if it satisfies the entire constraint in its own right. In the above example the clause is exclusive as it satisfies the constraint, however the elementary conditions that compose the clause are not exclusive as each separately only partially satisfies the constraint. In the complete specification of a constraint, several alternate exclusive conditions may exist, consider the example;

> Constraint X → Task A OR Task B
>
> both Task A and Task B are separate alternate exclusive conditions as each satisfy the constraint

In our case study of the higher education domain, constraints with this single level structure were found adequate to capture the task conditions. However, this constraint structure could not adequately capture some Instance Data conditions as they involved the use of several attributes to describe them, for example

> e.g. Constraint X → 2 Tasks of (LEVEL > x and AREA = "y")
>
> where (LEVEL > x and AREA = "y") represents the concatenation of two Instance Data requirements to form the condition

Support for a structure containing *Instance Data* clauses within an elementary condition is required to complete the constraint specification.

7.2 Generic Attributes

In order to capture the instance data necessary to describe the constraints in a process it is necessary to define a set of attributes that can be used to describe the components of a process.

A process is comprised of a number of tasks, and a number of instances are possible for any one process. Consequently, it is necessary to facilitate the creation of generic attributes that describe the process, each instance of a process and each task that may be involved in a process.

In its basic form a generic attribute will have an identifier and a description as well as a value field and a type. The type identifies the component of the process that is described by the attribute.

The definition of a process is performed in two stages, namely;

- Specifying the process components
- Describing the process components

Specifying the process components involves firstly defining the tasks of the process, that is selecting the set of tasks that can be undertaken in an instance of the process.

Secondly, it is necessary to define the attributes that describe;

- the process
- the task used in the process
- the instances of the process
- the outgoing data for a completed task in the process

Finally, for each attribute performing one of the above roles it is possible to further define a set of valid values that apply to it when it is used in the process.

The second stage of the process definition is to describe the process components. This involves assigning a value to each description attribute in the process, from the set of valid values where applicable. For example the task "COMP7901" may be specified as available for use by a process, and the attribute "SubjectArea" may be specified to describe tasks in that process. Assigning a value of "Computer Science" to the "SubjectArea" attribute for the "COMP7901" task would be necessary to complete this segment of the process definition.

7.3 Mapping Constraints to a Relational Structure

To map constraints suitably for capture in a relational model requires they are first rewritten as a set of exclusive conditions, and these exclusive conditions decomposed into their elementary conditions. Each elementary condition is assigned a unique *identifier* and allocated a *constraint name* and a *clause number* to form a tuple in a table structure. Exclusive elementary conditions of a constraint are assigned a separate clause number in the relation. The elementary conditions composing an exclusive clause are assigned a common clause number. Instance data conditions require additional fields in the relational table in order to capture their description and requirement terms. These two terms of the condition are mapped to separate tuples which are assigned the same clause number. Finally a constraint that includes an Instance Data clauses within an elementary condition is simply treated as a compound application of the above mapping rules.

Constraint Z → 2 Tasks of (LEVEL > x and AREA = "y") OR [Task E]
where (LEVEL > x and AREA = "y") represents the Instance Data clause

Id	Cnst	Cls.	Variable	Attribute	Op	Value	Function Value
10	Z	5		TaskLevel	>	"x"	
11	Z	5		TaskArea	=	"y"	
12	Z	5			=	3	COUNT
13	Z	6	Task E				

The relational model is required to capture both the meta-data relevant to describing a process, its instances and the tasks associated with it, as well as the three levels of constraints that apply to the process. The relevant meta-data is captured through the defined set of generic attributes prepared for the process, and the constraints are expressed in terms of the associated values of these attributes. In appendix A, we provide a data model for our constraint specification approach. This model is given in the format of an Object Relational Model (ORM) and was developed in Microsoft Viso Modeler™ version 3.1c.

7.4 Enforcement of the Constraints

Specifying the constraints in a relational structure allows the Structured Query Language SQL to be used to check for constraint violations. Each constraint may be composed of Task conditions and or Instance Data conditions. The enforcement of each of these conditions is achieved using predefined SQL statement structures. Further detail of constraints enforcement can be found in Mangan (2001).

Confirming that the required tasks have been completed validates task conditions. The enforcement of a constraint containing task conditions requires that for each task condition in the constraint the associated task requirement exists as a completed tasks for the instance of the process that the constraint is being applied.

Confirming that the instance data requirements are satisfied by the instance data of the completed tasks validates instance data conditions. Enforcing a constraint containing instance data conditions requires that for each instance data condition in the constraint, the associated requirements are satisfied by the instance data of the completed tasks for the instance of the process that the constraint is being applied.

8. FUTURE WORK

The work reported in this paper is part of an ongoing project [www.dstc.edu.au/praxis]. This work is now being extended and further developed. Below we present a summary of ongoing and future work in this area.

Generic constraint specification and enforcement: The functionality required to facilitate building instance specifications under the flexible model includes at least a constraint specification platform; the ability to analyse these constraints and identify conflicts and/or redundancy; and the validation of instances built against the given constraints.

Extending existing workflow modelling tools with this functionality and developing a tighter integration between the workflow specification and the enactment modules facilitates the design of the workflow instance as a part of the enactment state, where a change to the model is treated as a valid enactment transaction. These concepts are being further developed and implemented in advanced research prototypes for a process modelling tool FlowMake and a flexible workflow engine Chameleon. More details of these prototypes can be found at www.dstc.edu.au/praxis.

Planning tool for academic advising – a domain specific application of the concept: The iterative nature of the specification process supported by the framework is suited to planning applications such as a tool to assist students develop valid study programs that ensure their selected courses meet the requirements of their degree. An interface layer, specific to the process domain which consists of multi-choice nodes representing the aggregation of all possible tasks that can form the variants of the model, would relieve these users of the need to understand the underlying workflow technology. Through interaction with this interface layer, the task of designing a workflow instance specification reduces to one of selecting and scheduling a number of desired tasks from the set of those available. Custodians, responsible for the creation and maintenance of

the process tasks and constraints used in the builds, would provide the necessary support for these applications.

Based on this design model (Mangan, 2001) the supported functionality for the planning tool would typically include query support, that would allow users to experiment with different task selections to develop processes that were for example, completed in the minimum time or specialised in a particular area. View generation is another important aspect, that is seen as the ability to specify desired tasks to be included in the process and have the application map the path of required tasks to be undertaken to support the inclusion of the desired task in the process.

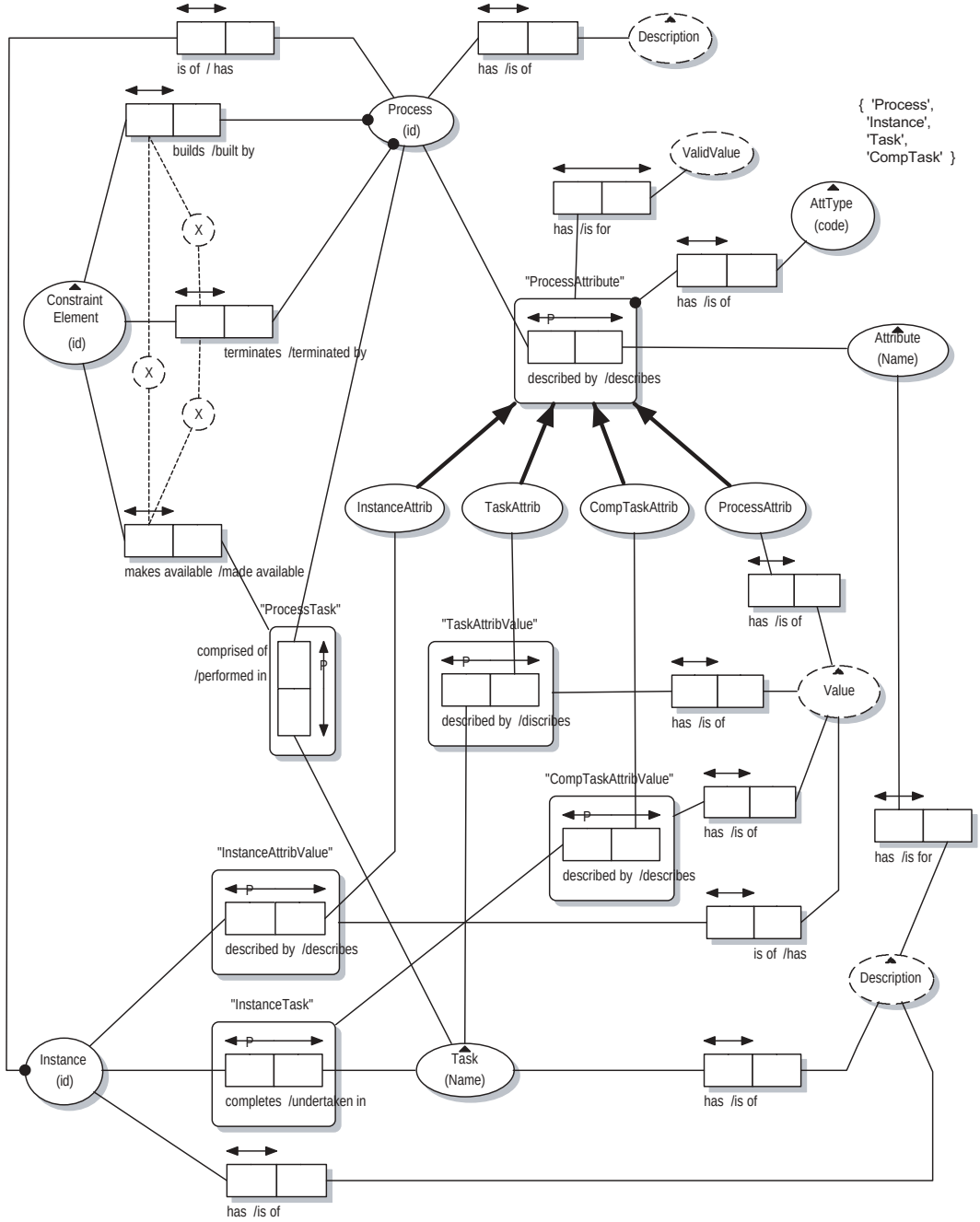
CONCLUSIONS

The inability of workflow technology to eloquently model processes containing a high degree of flexibility has limited its deployment in some domains. This paper provides a comprehensive analysis of flexible workflow processes derived from an analysis of the higher education domain. It attempts to address this flexibility by developing a framework for specifying the process model that can be tailored to individual instance requirements. Our approach provides a means of capturing the logic of highly flexible processes without compromising the simplicity and genericity of the workflow specification language or seriously impacting on the functionality of the underlying workflow engine.

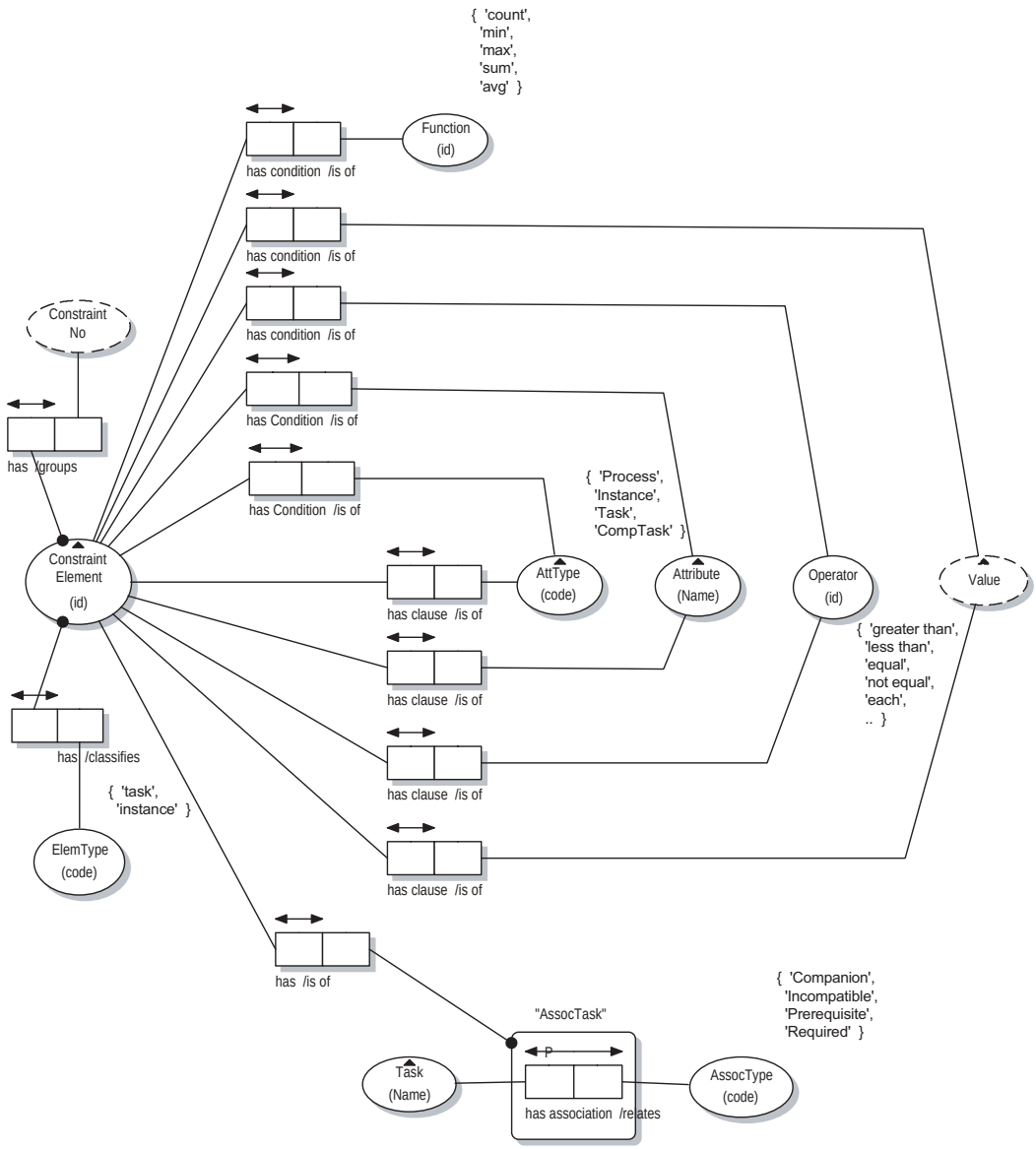
ACKNOWLEDGEMENTS

The authors would like to acknowledge the comments and feedback provided by Maria Orlowska at the University of Queensland.

APPENDIX A: Data Model (ORM) for Constraint Specification



each ProcessAttrib is a ProcessAttribute with AttType 'Process'
each InstanceAttrib is a ProcessAttribute with AttType 'Instance'
eachTaskAttrib is a ProcessAttribute with AttType 'Task'
each CompTaskAttrib is a ProcessAttribute with AttType 'CompTssk'


{ 'count',
'min',
'max',
'sum',
'avg' }
Function
(id)
has condition /is of
has condition /is of
has condition /is of
has Condition /is of
has Condition /is of
Constraint
No
has /groups
Constraint
Element
(id)
{ 'Process',
'Instance',
'Task',
'CompTask' }
AttType
(code)
Attribute
(Name)
Operator
(id)
Value
{ 'greater than',
'less than',
'equal',
'not equal',
'each',
.. }
has clause /is of
has clause /is of
has clause /is of
has clause /is of
has /classifies
{ 'task',
'instance' }
ElemType
(code)
has /is of
{ 'Companion',
'Incompatible',
'Prerequisite',
'Required' }
"AssocTask"
P
Task
(Name)
has association /relates
AssocType
(code)

REFERENCES

AGOSTINA, A. and DE MICHELIS, G. (2000): Improving flexibility of workflow management systems. van der Aalst W. *et al.* (Eds.) *Business Process Management, LNCS* 1806: 218–234.

BERNSTEIN, A., (2000): How can cooperative work tools support dynamic group processes? Bridging the specificity frontier. *CSCW'00,* December 2-6, 2000, Philadelphia.

BOGIA, D. P. and KAPLAN, S. M. (1995): Flexibility and control for dynamic workflows in the worlds environment. *Proceedings of ACM Conference on Organizational Computing Systems* (COOCS 95), Milpitas, CA. USA, November.

CASATI, F., CERI, S., PERNICI, B. and POZZI, G. (1996): Workflow evolution. In *Proceedings of the 15th International Conference on Conceptual Modeling,* ER'96, Cottbus, Germany. Springer Verlag, Lecture Notes in Computer Science.

ELLIS, C.A. and KEDDARA, K. (2000): A workflow change is a workflow, van der Aalst *et al.* (Eds.) *Business Process Management, LNCS* 1806: 201–217, 2000.

HERRMANN, T. (2000): Evolving workflows by user-driven coordination, *Proceedings of DCSCW,* Munich, Germany, 102–114, September.

HAN, D. and SHIM, J. (2000): Connector-oriented workflow system for the support of structured ad hoc workflow, *Proceedings of the 33rd Hawaii International Conference on System Sciences.*

JOERIS, G. and HERZOG, O. (1999): Managing evolving workflow specifications with schema versioning and migration rules, *TZI Technical Report* 15–1999, Center for Computing Technologies (TZI), University of Bremen.

JOERIS, G. (1999): Defining flexible workflow execution behaviors, submitted for publication 1999.

KRADOLFER, M. and GEPPERT, A. (1999): Dynamic workflow schema evolution based on workflow type versioning and workflow migration. *Proccedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS99).* Edinburgh, Scotland. Septmber 2–4.

KNOLMAYER, G., ENDL, R. and PFAHRER, M. (2000): Modeling processes and workflows by business rules, van der Aalst W. *et al.* (Eds.) *Business Process Management, LNCS* 1806: 16–29.

LIU, C., ORLOWSKA, M.E. and LI, H. (1998): Automating handover in dynamic workflow environments. *Proceedings of 10th International Conference on Advances in Information System Engineering (CAiSE 98),* Pisa, Italy, June.

MANGAN, P. (2001): Building workflow models for flexible processes, *Masters thesis,* School of Information Technology and Electrical Engineering, University of Queensland.

PAPAZOGLOU, M. P. and VAN DEN HEUVEL W.J. (2000): Configurable business objects for building evolving enterprise models and applications, van der Aalst W. *et al.* (Eds.) *Business Process Management, LNCS* 1806: 328–344.

SADIQ, S. (2000): Handling dynamic schema change in process models. *Australian Database Conference,* Canberra, Australia. January 27–February 02.

SADIQ, S. (2000): On capturing exceptions in workflow process models. *Proceedings of the 4th International Conference on Business Information Systems.* Poznan, Poland. April 12–13.

SADIQ, W. and ORLOWSKA, M.E. (1997): On correctness issues in conceptual modeling of workflows. In *Proceedings of the 5th European Conference on Information Systems (ECIS '97),* Cork, Ireland, June 19–21.

SADIQ, W. and ORLOWSKA, M.E. (1999): On capturing process requirements of workflow based information systems. In *Proceedings of the 3rd International Conference on Busi-ness Information Systems (BIS '99),* Poznan, Poland, April 14–16, 195–209. Springer-Verlag.

SADIQ, W. and ORLOWSKA, M.E. (1999): Applying graph reduction techniques for identifying structural conflicts in process models, *11th Conf on Advanced Information Systems Engineering, CAISE'99,* Germany, 195–209.

SADIQ, S., SADIQ, W. and ORLOWSKA, M.E. (2001): Pockets of flexibility in workflow specifications. (To Appear) *Proceedings of the 20th International Conference on Conceptual Modelling (ER2001).* Yokohama, Japan. November 27–30.

REICHERT, M. and DADAM, P. (1997): ADEPTflex – Supporting dynamic changes of workflow without loosing control. *Journal of Intelligent Information Systems (JIIS),* Special Issue on Workflow and Process Management.

VAN DER AALST, W.M.P. (1998): How to handle dynamic change and capture management information. *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems.*

VAN DER AALST, W.M.P., BARROS, A.P., HOFSTEDE TER A.H.M., and KIEPUSZEWSKI B. (2000): Advanced workflow patterns. Etzion O. and Scheuremann P., editors, *Proceedings Seventh IFCIS International Conference on Cooperative Information Systems,* CoopIS 2000, Lecture Notes in Computer Science, 1901: 18–29, Eilat, Israel. September. Springer-Verlag.

WORKFLOW MANAGEMENT COALITION (1995): Workflow management coalition – The workflow reference model. Document No. TC00-1003, 1(1): 19 January.

BIOGRAPHICAL NOTES

Peter Mangan

Peter Mangan is a consulting Electrical Engineer specialising in the design of protection systems for electrical power transmission and distribution networks throughout Australia and Asia. Peter holds a Bachelor of Engineering from the School of Engineering of the Queensland Institute of Technology, Brisbane, Australia and a Masters degree in Information Technology Studies from the School of Computer Science and Electrical Engineering of the University of Queensland, Brisbane, Australia. His Masters studies were specialised in the field of Data Management and his research focused on Dynamically Changing Workflow Processes. Peter has worked on large scale information system design and development projects associated with the capture and management of data for the modelling and analysis of electrical power systems through Queensland and for the design and management of power system protection data.

Shazia Sadiq

Shazia Sadiq is currently a Lecturer in Information Systems at The University of Queensland. Her responsibilities included teaching large undergraduate as well as post graduate courses in the School of Information Technology and Electrical Engineering. She also supervises honours and masters students in a variety of research topics. She is contributing a portion of her time as a seconded researcher at the Distributed Systems Technology Centre, Brisbane, Australia. Shazia holds a Masters degree in Computer Science from the Asian Institute of Technology, Bangkok, Thailand. She has recently submitted her PhD at the School of Computer Science and Electrical Engineering of the University of Queensland, Brisbane, Australia. Her PhD research is in the field of Dynamically Changing Workflow Processes. She has made a significant research contribution in the field of dynamic and flexible workflows and has published several papers on this topic. Her research ideas are now being implemented as part of DSTC's flexible workflow engine known as Chameleon. Prior to joining The University of Queensland, Shazia worked as a lecturer at the Regional Computer Centre, Asian Institute of Technology, Bangkok, Thailand. At RCC/AIT she was responsible for conducting training courses in the field of Information Systems and Database Management. She has also successfully completed several information systems design and development projects as an independent consultant in the areas of legal systems, hospital management, and library cataloguing.