On Building Workflow Models for Flexible Processes

Peter Mangan, Shazia Sadiq
School of Computer Science and Electrical Engineering
The University of Queensland
QLD 4072 Australia

pmangan@optushome.com.au, shazia@csee.uq.edu.au

Abstract

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 characterized by predictability and repetitiveness. Application in non-traditional domains with highly flexible process is still largely unexplored. Such flexible processes are characterized 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 modeling constructs. This ensures that the correctness and verification of the language is preserved. We propose to build workflow schemas from a standard set of modeling constructs and given process constraints. We identify the fundamental requirements for constraint specification and classify them into selection, termination and build constraints. Finally, we will demonstrate the dynamic building of instance specific workflow models on the basis of these constraints.

Keywords: flexible workflow, tertiary education, constraint specification

1 Introduction

Effective business organizations 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 [SO99a]. 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 objects, as the task cannot perform its associated work on the object directly. A performer may be a person, typically someone filling an organizational 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 characterized 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 organizations adjust their activities in response to influences such as new legislation, 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 lead 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 process cannot be eloquently predefined:

This paper appeared at the Thirteenth Australasian Database Conference (ADC2002), Melbourne, Australia. Conferences in Research and Practice in Information Technology, Vol. 5. Xiaofang Zhou, Ed.

- 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 characterized 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 organizational workflow processes and underlying applications.

We will introduce in the following section, some basic terminology and a simple workflow modeling language. In the next section 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 modeling languages in this area. Finally, we will introduce our proposed approach. We will show how flexibility of specification can be achieved without compromising the simplicity and consequently verifiability of the modeling language.

2 Basic Terminology

We will use basic workflow modeling 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 interdependencies 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 [SO97], which will be used for demonstrating various concepts. This language conforms closely to the workflow management coalition standards [WfC95]. Note that we will utilize only the basic set of modeling 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.

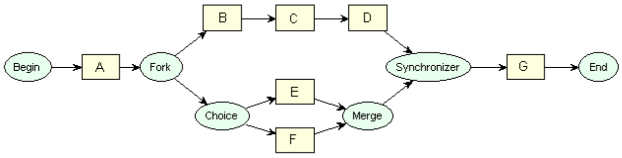


Figure 1. Basic Workflow Constructs

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 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. We see two primary areas of application in this domain:

- Flexible Learning: Mixed mode delivery of education is receiving increasing attention as academic institutes look to technology for a solution to the constraints that time, location and recourses place on the availability of their programs to potential students in an international marketplace. Although a number of collaboration tools exist, these tools basically enable building of intelligent websites. A novel approach has been presented in [LHS+01], which makes use of workflow-driven e-learning. The Flex-eL system introduced in the above provides students with a flexible, yet controlled means of progressing through the study process by making use of workflow functionality to define and execute study and assessment tasks. This approach not only enables effective collaboration between the teaching and learning groups, but also provides coordination, control and monitoring of individual study plans.
- Student Planning and Academic Advising: Tertiary programs offer a diverse collection of courses that allow specialization 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 9 courses of compulsory material and a further 3 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 modeling 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 modeling in traditional process definition languages very difficult.

In Figure 2 we give an example of a degree program with 9 compulsory and 3 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.

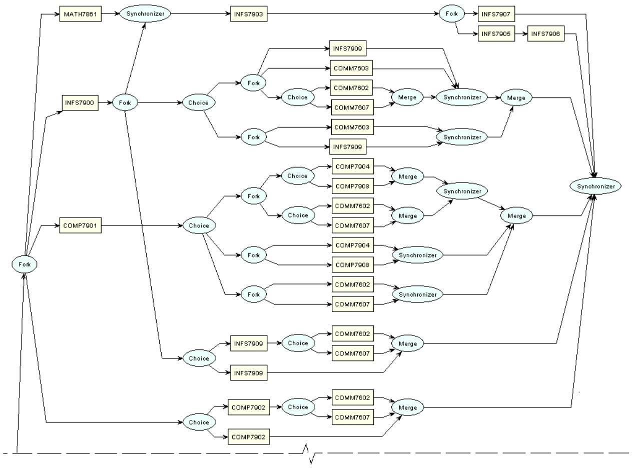


Figure 2. Modeling Flexible Processes in a Prescriptive Language

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

4 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 [SSO01]. Although the fundamental question we are addressing is the modeling 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 modeling constructs that can be used to compose the fragments for a given instance.
    - Sequence
    - Fork/Synchronize
    - Multiple executions

In this paper we will utilize 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 [SO99b].

- 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

4.1 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

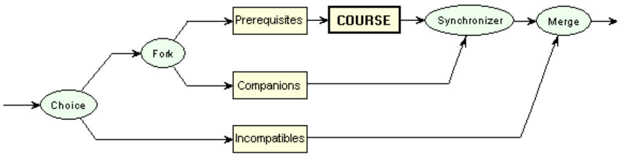


Figure 3. Higher Education Domain Task Interdependencies

4.2 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.

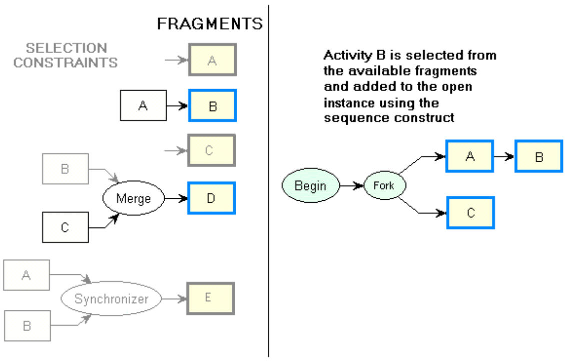
4.3 Build Constraints

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 honors programs may require an overall grade average or a grade average taken over a number of courses.

5 Building Valid Instances

The dynamic build approach allows for the continuous build of an instance from the set of available fragments and modeling 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 modeling language for clarity of the example.



Activity B is selected from the available fragments and added to the open instance using the sequence construct

Figure 4. Dynamic build using fragments with satisfied selection constraints

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.

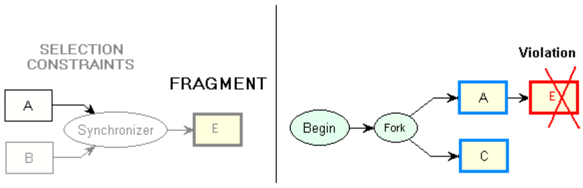
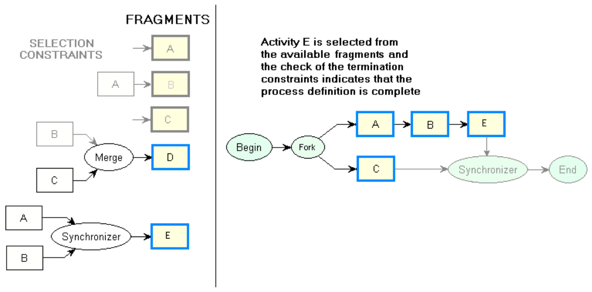


Figure 5. Enforcement of semantic correctness in the open instance

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 a fork structure being generated in the open instance with greater than three concurrent paths each containing course fragments. 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.



Activity E is selected from the available fragments and the check of the termination constraints indicates that the process definition is complete

Figure 6. Enforcement of Termination constraints

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. We are currently developing a constraint specification language that can be integrated with the workflow definition language, thus extending the functionality of the process definition tool. This will allow the building of valid instances as demonstrated above. 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.

6 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. [ABH+00]. 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. The biggest problem is the handling of the active instances that were initiated in the old model. Defining a migration strategy is a complex problem and has been the target of extensive research [CCP+96], [KG99], [LOL98], [Sad00a]. 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 [RD97], [Sad00b]. 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 [Ber00]. 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.

Moving to the other end of our continuum for organizational 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 [BK95]. The complete relaxation of coordination, to support ad-hoc processes is not conducive to the processes targeted by our work.

Positioned between the two extremes of our continuum are developments in the areas of evolving workflows [Her00], [SSO01]. 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.

7 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 basic idea is to provide a means of capturing the logic of highly flexible processes without compromising the simplicity and genericity of the workflow specification language or seriously impact the functionality of the underlying workflow engine. The framework proposed will provide an approach to modeling dynamic organizational processes that avoids the overheads incurred in their specification, and modification in response to change, that arise when a prescriptive approach is taken.

8 Acknowledgements

The authors would like to acknowledge the comments and feedback provided by Maria Orlowska at the University of Queensland, currently a visiting professor at the University of Science and Technology, Hong Kong.

9 References

References should be in the standard JRPIT format, examples of which are shown below (shown is a conference paper, a thesis, a book section, a book and a journal paper in that order).

[ABH+00] W.M.P. VAN DER AALST, A.P. BARROS, A.H.M. TER HOFSTEDE, and B. KIEPUSZEWSKI. (2000) Advanced Workflow Patterns. O. Etzion and P. Scheuremann, editors, *Proceedings Seventh IFCIS International Conference on Cooperative Information Systems*, CoopIS 2000, volume 1901 of Lecture Notes in Computer Science, pages 18-29, Eilat, Israel. September 2000. Springer-Verlag.

[Ber00] ABRAHAM BERNSTEIN, How can Cooperative Work Tools Support Dynamic Group Processes? Bridging the Specificity Frontier, *CSCW'00*, December 2-6, 2000, Philadelphia.

[BK95] DOUGLAS P. BOGIA, SIMON M. KAPLAN. (1995) Flexibility and Control for Dynamic Workflows in the wOrlds Environment. *Proceedings of ACM Conference on Organizational Computing Systems COOCS 95*, Milpitas, CA. USA, Nov 1995.

[CCP+96] FABIO CASATI, S. CERI, B. PERNICI, G. POZZI. (1996) Workflow Evolution. *Proceedings of the 15th International Conference on Conceptual Modeling, ER'96*, Cottbus, Germany. Springer Verlag, Lecture Notes in Computer Science.

[Her00] THOMAS HERRMANN, Evolving Workflows by User-driven Coordination, *Proceedings of DCSCW*, Munich, Germany, Sept. 2000, pp. 102-114.

[LOL+98] CHENGFEI LIU, MARIA E ORLOWSKA, HUI LI, Automating Handover in Dynamic Workflow Environments. *Proceedings of 10th International Conference on Advances in Information System Engineering (CAiSE 98)*, Pisa, Italy, June 1998

[KG99] MARKUS KRADOLFER, ANDREAS GEPPERT. (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. Sep 2-4, 1999.

[Sad00a] SHAZIA SADIQ (2000) Handling Dynamic Schema Change in Process Models. *Australian Database Conference*, Canberra, Australia. Jan 27 - Feb 02, 2000.

[Sad00b] SHAZIA SADIQ (2000) On Capturing Exceptions in Workflow Process Models. *Proceedings of the 4th International Conference on Business Information Systems*. Poznan, Poland. April 12 -13 2000.

[SO97] WASIM SADIQ, MARIA E. ORLOWSKA (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, 1997.

[SO99a] WASIM SADIQ, MARIA ORLOWSKA (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, 1999. pp. 195-209. Springer-Verlag, 1999.

[SO99b] SADIQ, W & ORLOWSKA, Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models, *11th Conf on Advanced Information Systems Engineering, CAISE'99,* Germany, pp. 195-209, 1999.

[SO00] WASIM SADIQ, MARIA E. ORLOWSKA (2000) Analyzing Process Models using Graph Reduction Techniques. *Information Systems, Vol. 25, No. 2, pp. 117-134, 2000.* Elsevier Science. June 2000.

[SSO01] SHAZIA SADIQ, WASIM SADIQ, MARIA ORLOWSKA. (2001) Pockets of Flexibility in Workflow Specifications. (To Appear) *Proceedings of the 20th International Conference on Conceptual Modeling ER2001*. Yokohama, Japan. November 27-30, 2001.

[RD97] REICHERT MANFRED, DADAM PETER (1997) ADEPTflex - Supporting Dynamic Changes of Workflow without loosing control. *Journal of Intelligent Information Systems JIIS*, Special Issue on Workflow and Process Management

[WFMC95] WORKFLOW MANAGEMENT COALITION. (1995) Workflow Management Coalition - The Workflow Reference Model. Document No. TC00-1003, Issue 1.1, 19-Jan-95.

[LHS+01] JOE LIN, CHARLEY HO, WASIM SADIQ, MARIA E. ORLOWSKA (2001) On Workflow Enabled e-Learning Services. *(To Appear)* ICALT 2001, Madison, Wisconsin, Aug 2001.