Data Flow and Validation in Workflow Modelling

Shazia Sadiq[†], Maria Orlowska[†], Wasim Sadiq[§], Cameron Foulger[†]

[†] School of Information Technology and Electrical Engineering
University of Queensland, St Lucia QLD 4072 Australia
[§] SAP Corporate Research Centre Brisbane
Level 12, 133 Mary Street, Brisbane QLD 4000 Australia

shazia@itee.uq.edu.au; maria@itee.uq.edu.au;
wasim.sadiq@sap.com; s367197@student.uq.edu.au

Abstract

A complete workflow specification requires careful integration of many different process characteristics. Decisions must be made as to the definitions of individual activities, their scope, the order of execution that maintains the overall business process logic, the rules governing the discipline of work list scheduling to performers, identification of time constraints and more. The goal of this paper is to address an important issue in workflows modelling and specification, which is data flow, its modelling, specification and validation. Researchers have neglected this dimension of process analysis for some time, mainly focussing on structural considerations with limited verification checks. In this paper, we identify and justify the importance of data modelling in overall workflows specification and verification. We illustrate and define several potential data flow problems that, if not detected prior to workflow deployment may prevent the process from correct execution, execute process on inconsistent data or even lead to process suspension. A discussion on essential requirements of the workflow data model in order to support data validation is also given.

Keywords: Business Process Modelling, Data Validation, Workflow Data

1 Introduction

Workflow or business process modelling can be divided into a number of aspects including structural, informational, temporal, transactional, and functional (Jablonski and Bussler 1996), (Sadiq and Orlowska 1999). Traditionally, workflow modelling has focused on structural aspects of processes, mainly indicating the order of execution of the component activities in the process, using specification languages. There exist a large number of proposals for process specification, each with its own flavour of semantic richness, expressability levels, and particular grammar.

Time management for process-oriented systems is another important aspect. (Marjanovic 2000) (Eder et al 1999) identified critical temporal properties for process activities. Temporal aspect includes duration and deadline constraints, which may be absolute or relative, as well as the issue of temporal consistency, where a temporal constraint is consistent within a given process model if and only if it can be satisfied based on syntax of the process model and expected minimum and maximum duration of process activities.

Processes will also have associated resources, in terms of participants and invoked applications and in turn these resources may have extended descriptions in terms of the organizational model, and application environment respectively.

A necessary prerequisite to the deployment of a process model is the verification of the model in term of the syntactic and structural correctness criteria of the underlying language. In other words, process verification is required to at least ensure that the resulting process model is executable in the given process management system (syntactic verification). Depending upon the constructs supported by a given process specification language, and the associated correctness criteria, a number of structural errors may be identified during verification, for example deadlock causing structures, or structures that inadvertently trigger multiple executions. Where as syntax checking may not be complex for typical languages, structural verification is a challenging issue requiring complex algorithms (Sadiq and Orlowska 2000b) (Van der Aalst 1997). Verification however, remains generic across all models built in a particular language, that is, it is business context independent.

However, the syntactic or structural correctness of a process model does not necessarily indicate that that process model has satisfactorily captured the underlying business requirements. Validation of the process usually requires the knowledge of a domain expert on the underlying business rules, business process logic and many types of local constraints. Clearly, manual validation for any reasonably complex process models is complex and hard to achieve with an adequate dose of accuracy. Furthermore, it is unrealistic to assume that domain experts would also carry expert knowledge on process modelling. One can expect that workflow designers and domain experts would typically be disjoint but collaborating users of the process technology.

More precisely, process validation is a means by which the process design can be checked against expected execution behaviour, that is, determine if a model will execute as intended by the designer and also by the end users. Due to the overall complexity of workflows specification, deployment of a process without validation may lead to undesirable execution behaviour that compromises process goals.

Although process validation applies to all aspects of process specification, one of the most critical aspects of process design and analysis is specification of data requirements and data flow between process activities.

In this paper we target restricted process model validation: the validation of the data aspect of workflow processes. We envisage this being a step in the process design life cycle after it has been specified in a given language and verified for structural correctness (see Figure 1).

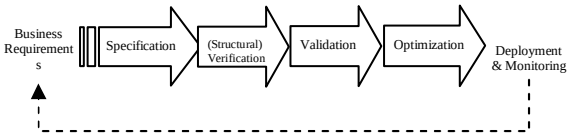


Figure 1. Process Life Cycle

Even a brief examination of data aspect in an extensive business process reveals that complex data analysis is required. It is worth to mention that it is unavoidable to deal with small-scale data integration issues here. Workflows technology often is promoted as an integration platform/technology for selective combination of pre-existing, often legacy applications. Practical experience demonstrates that re-use of existing applications as components of a business process is not easy and rather poorly supported by tools and methods. Compositions that may be suitable for a local sub-process may not necessary fit well to a larger scale consideration. Potentially any additional component (an activity) incorporated to an existing structure may trigger a change to the data transfer, mapping or even existence. We believe that several design shortcomings may be identified during data validation, such identification of as missing data, data generated by an activity and never used by the process, data availability from multiple not always consistent sources, and data syntactic and semantic mismatch.

In general, the correspondence between the system functional and data aspect is well established and understood (Batini, Ceri and Navathe 1992). Although, the emphasis is on specification and grammar of the language(s), very little is done on the verification of a design. Verification problems require careful semantic mappings and many data constraints satisfy-ability is computationally intractable.

Similarly, data has been recognized as a fundamental aspect of workflow specification (Leymann and Roller 2000), (Jablonski and Bussler 1996). However, literature reports very little work on the validation of process data. We will present a brief summary of the literature review in section 2.

The main focus of this paper is to present the data modelling and validation issues. Accordingly, in section 3, we will first present a discussion into the requirement for activity and process level properties that will be necessary for data validation. We will then identify a collection of generic data validation problems for typical workflows. A brief discussion on the implementation of data validation procedures is also presented.

2 Related Work

In general, process validation and optimization potentially triggers a series of refinements to the process structure as well as to the applications invoked by the local activities, in order to achieve desired effects of the overall process implementation. This identifies a classic lifecycle for the process (Figure 1), where business requirements are mapped to process models through a modelling methodology e.g. ARIS (Scheer 2000); the process model is then analysed through a series of verification, validation and optimization procedures; and eventually deployed, which in turn may impact on the business process as is typical of post implementation monitoring. This lifecycle is in tune with well established techniques in information systems and software engineering.

It is important to note that validation, like structural verification is not intended to provide guarantees on the correctness and/or completeness of the process model against business requirements. It is rather intended to support the process modelling activity, by providing smart tools for cross checking process and activity properties, and identifying potential conflicting and redundant elements in the specification, thereby boosting user confidence in the deployment of the process model.

A closely related area is simulation (Robinson 1997), (Hlupic and Robinson 1998), (Casati, et al 2001, 2002).The simulation of a business process provides designers with valuable information prior to the process' deployment such as execution times of tasks, loads on task resources and the load on the process management system itself.. As such, it provides a means of evaluating the execution of the business process to determine inefficient behaviours and subsequent re-engineering (Ferscha 1998).

In terms of data validation in workflows, there is very little reported in literature. As is commonly known, any tool can only validate what it knows. As such, it is obvious that validation will dependent on the meta-model supported by the process engine. There have been some proposals for capturing the data aspect of workflow specification, notably (Dadam and Reichert 1998), (DSTC 1999), (Jablonski and Bussler 1996), (Leymann and Roller 2000), (Sadiq and Orlowska 1999). All of the models stress the importance of supporting the data flow aspect.

Typically, there is a distinction between application specific data for individual activities and process control

related data. There is also a distinction between input and output data for an activity. This approach is evident in many commercial workflow products as well (e.g. IBM MQWorkflow concept of input and output containers).

However, the aforementioned models mainly focus on the specification of data and data flow in terms of name-value pairs. In terms of data validation, there is some evidence of the identification of the problem of varied representations, and subsequently a specification of data transformation that inevitably strives for data consistency. Others, notably (Dadam and Reichert 1998), have identified some of the long-standing data flow problems associated with workflow such as missing data (see section 3). However, their consideration is also limited to dynamically changing workflow models.

3 Validation of Workflow Data

We target validation of a process model, after it has been specified in a given language and verified for structural correctness. Thus the structure of the business process and the requirements of its constituent activities must be known before the process model can be validated. Requirements include anything that an activity requires to *initiate*, *continue* or to be *completed*. This is an essential step that must be considered before hand.

It is difficult if not impossible to capture the flow of data in large complex processes. Thus, process designers and associated domain experts would generally be competent to provide activity specific data requirements, and to some extent the flow of data in an activity's immediate neighbourhood. However, building a process wide model of data flow can prove to be difficult.

At the same time, it is unrealistic to assume that the workflow specification can be extended to include all relevant information on underlying databases of invoked applications. Workflow Management Systems (WFMS) primarily focus on the coordination of activities, and the data processing functionality of invoked applications is to a large extent outside the control of the WFMS. This distinction between application data and workflow relevant data is well known (WFMC, 1999).

In light of the above two observations, we propose a simple activity-based data model. Basically, data requirements are captured in terms of individual activities. This model is intended to provide the minimal requirement for process models, in order to address the selected validation problems. However, it is important to reiterate that the richness of the model has a direct impact on the level of support that can be provided for validation. The more extensive the model is at capturing the data requirements, the better we can support data validation, but the more difficult it will be in practice to record the extended data properties. Hence, as always, there is a trade off between pragmatic and conceptual considerations.

Typically, the data requirements are captured on the basis of the underlying workflow model. Thus, in order to establish terminology, we first briefly introduce the structure of the workflow modelling language (Sadiq 2000a). Note however, that this language provides standard, widely accepted workflow modelling constructs.

The workflow model (W) is defined through a directed graph consisting of Nodes (N) and Flows (F). Flows show the control flow of the workflow. Thus W = <N, F> is a directed graph where N is a finite set of nodes, F is a flow relation $F \subseteq N \times N$. Nodes are classified into tasks (T) and coordinators (C), where $C \cup T$, $C \cap T = \phi$

Task nodes represent atomic manual / automated activities or sub processes that must be performed to satisfy the underlying business process objectives. Coordinator nodes allow us to build control flow structures to manage the coordination requirements of business processes. Basic modelling structures supported through these coordinators include Sequence, Exclusive Or-Split (Choice), Exclusive Or-Join (Merge), And-Split (Fork), And-Join (Synchronizer). An *instance* within the workflow graph represents a particular case of the process. The graphical representation of these constructs is shown in Figure 2.

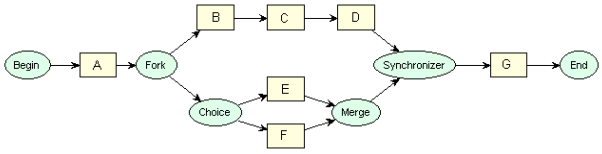


Figure 2. Basic Workflow Constructs

As we discussed earlier in section 1, data flow is an essential part of workflow execution in conjunction with control flow. Activities in a workflow model provide necessary data to their underlying application components and human performers to correctly identify the context of the work they are supposed to carry out. At the same time, the underlying application components and human performers may return some data to the workflow model that may be needed by some other activities in the workflow model. The way an activity communicates with its underlying associated implementation can be compared with a function call in a procedural language. Similar to the way we have input and output parameters to a function call, an activity definition provides input and output data values for its underlying implementation. The key requirements of any data flow mechanism in a workflow model are to have the ability to:

- Manage the data that is needed as input to and provided as output from activities in the workflow model.
- Ensure that the data provided by an activity in the workflow model is available to other activities in the workflow model that need it.
- Provide mechanisms to ensure consistent *flow* of data from one activity to another.

As such (and rather simplistically at this stage), data requirements for the workflow model can be defined as below:

Let WF-Data be a process data store[1], consisting of a set of data items $\{v_1, v_2, \ldots v_k\}$ for W, where

- $\forall$ n $\in$ **N**, Input(n) is a set of data items V_i^n that n consumes, where $V_i \subseteq$ WF-Data
- $\forall$ n $\in$ **T**, Output(n) is a set of data items V_o^n that n produces, where $V_o \subseteq$ WF-Data

where only nodes of type T (task) have an associated output set of variables, this is because nodes of type C (coordinator) require only to read data and make control flow decision based on that data.

In figure 3, we give an example of a cheque issue workflow, and a simple activity based data model, consisting of the input and output data. In this example, "Prepare Cheque for ANZ Bank" is a node n in this workflow graph. "Prepare Cheque for ANZ Bank" will have associated Input(n) and Output(n) container.

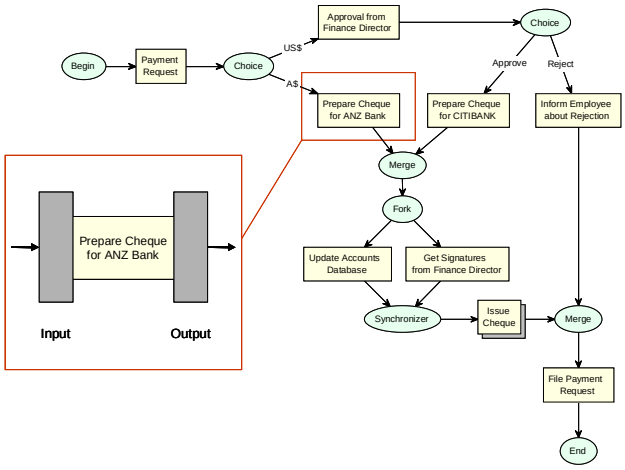


Figure 3. Basic Activity Based Data Model

3.1 Data Model Requirements

It is apparent that in order to provide any reasonable analysis of the process data, further information about the data must be captured. We propose below a number of properties of data that should constitute an essential part of a data model. These are presented below under three main categories of type of data, source of data and structure of data

3.1.1 Type of data

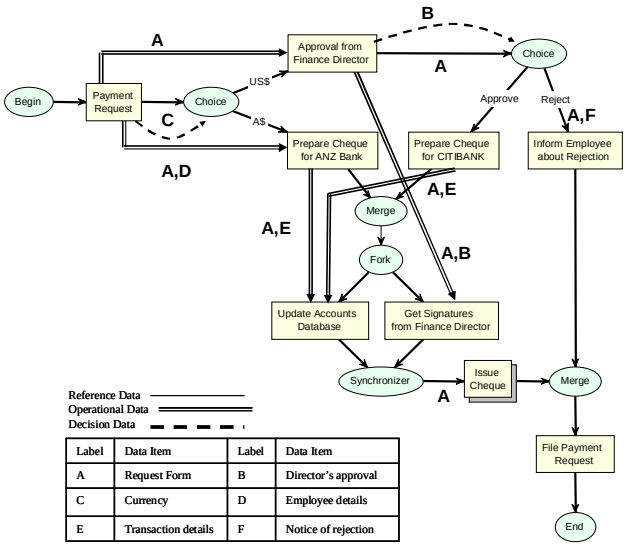
The nature of the data with which an activity may deal with, can be quite diverse, ranging from structured, predefined data needed for identification, to complex, unanticipated or exceptional data. Note that this is distinct from internal process control data where attributes such as instance number, start time of an activity, owner of the process etc. are maintained for scheduling and logging purposes. Below we present a brief description of the various typical types of data used by workflow activities.

R_i:Reference. Reference data refers to the data that is used for the identification of the instance or case, e.g. order number, insurance policy number etc. Typically, the route of reference data items would overlap with the control flow of the given process, although during execution, reference data would only flow through a subset of activities that relate to the particular instance type. Furthermore, one can also expect that reference data will mostly be established by an initiating activity, although this cannot be generalized. Figure 4 shows the map of reference data (thick line) within the example of Figure 3.

O_i:Operational. Operational data refers to all data items that are needed by a particular activity. Where this data comes from and in what form is another question, which we will discuss in the sections below on sources and structure. Example of operational data is customer address, student visa status, size of dwelling etc. Figure 4 shows the map of operational data (double line) as it flows through the given workflow example.

D_i:Decision. Decision data is a subset of operational data ($D_i \subseteq O_i$), which is needed by choice coordinators to make control flow decisions. For example, the currency type (A$ or US$), Age of a person (≥55 or <55), etc. Figure 4 shows the map of the decision data (broken line) for the same example.

C_i:Contextual. Contextual data is typically of a complex structure, and may or may not be pre-defined. Basically contextual data may be required as input, as well as generated as output, for the benefit of proceeding activities. Example of contextual data is quotation for a fax machine, student transcript, building covenants etc.



Label	Data Item	Label	Data Item
A	Request Form	B	Director's approval
C	Currency	D	Employee details
E	Transaction details	F	Notice of rejection

Figure 4. Types of Data

In summary, Input(n) will be a union of the above types of data, i.e. $V_i^n = \{ R_i^n \cup O_i^n \cup C_i^n \}$ where $D_i^n \subseteq O_i^n$. Similarly, data items that constitute Output(n) can also be distinguished according to the above classification.

[1] Process data store is a complex data structure which contains values assigned to process data items as well as a history of read/write transactions performed on the data items. Use of process data store is a particular implementation approach, and consequently its design is also implementation dependent. We will discuss this and other implementation approaches later in the paper.

3.1.2 Sources of data

In any activity based data model, there will be flow of data between activities of the workflow. However, not all data is available through internal workflow activities. There is a need to at least distinguish between the internal and external sources of data.

Internal Sources: We define data items which can be found in WF-Data to be internal. Read/write on this data item can be considered internal. However, every data item will need to be established (given a value) by some activity within the process. For example, the bulk of the reference data may be established by the initiating activity. Thus even though these data items are part of the WF-Data, values for them may be assigned by some external sources. In other words, an internal data item, may not always be internal.

External Sources: Data which is available from any source other than the process data store, is termed external. External sources include workflow clients creating new data, e.g. a desk clerk filling out a form, retrieve operations from underlying application databases, downloading from an ftp site, etc. Data items established by external sources, may or may not be internalized. Thus, not every data item that an activity reads/writes will be present in the process data store WF-Data.

The above distinction identifies an important extension to the activity data model given previously. That is, every data item must identify its source/destination:

$\forall\ v_k \in V_i^n$ Source: $V_i^n \rightarrow$ {Internal, External}, Source(v_k) : Source of the input data item v_k for node n

$\forall\ v_m \in V_o^n$ Destination $V_o^n \rightarrow$ {Internal, External}, Destination(v_m) : Destination of the output data item v_m for node n

An *Internal* source or destination implies a read/write from the process data store. An *external* source/destination must provide further information. Capturing additional information about data sources and destinations provides the means to further reason about the process data requirements. Thus, questions such as "should an external data item be made part of the process data store?", "will a value for the data item be available to this activity?", and so on, can be addressed.. Although specifying the items an activity needs, and from where it gets them is realistic. However, even a preliminary consideration of the above, indicates that specifying at design time the destination (all activities which will need this data item) is a rather unrealistic expectation.

3.1.3 Structure of the data

As defined in the beginning of section 3, WF-Data can be considered a set of data items. However, as is obvious from the discussion above, the data that an activity may deal with can be of a complex structure, including complex data types (such as images) as well as unstructured data (such as text documents).

It is essential to provide a data model that is capable of dealing with extended structures as is typical in XML based data exchange, i.e. having a data model over and above simple parameter passing. This is especially true for more advanced types of data such as contextual data, although any restrictions of the structure of other types of data (reference, operational, decision) would also be unrealistic.

Data especially in the form of semi-structured documents may in addition require mapping or extraction rules in order to provide flow of data to other activities. That is, data items may be required by multiple activities, but the structure in which the data item is available and the structure in which an activity utilizes the data item may not always be compatible.

A complete design of a process data model based on the requirements set forth above is beyond the scope of this paper. However, in summary, and based on the requirements given above, we propose that each activity has an input *schema* and an output *schema*, thus providing a means of dealing with complex and semi-structured data, over and above simple name value pairs. In addition, we propose that each external input data schema has associated with it source information, and where applicable, functions to extract/transform the data to be received from the external source.

It will become apparent in the next section, that the additional data requirements identified above provide the necessary information on which data validation algorithms may be developed, and in some cases, may even implicitly eliminate some of the data validation problems.

3.2 Data Validation Problems

We have identified seven basic data validation problems:

Redundant Data

Occurs when a designer specifies data item/s v_k in the output schema Output(n) of an activity, but this data is not required in the input schema Input(n) of any succeeding activity. The assumption is that if a data item has been specified in the output schema of an activity and is being maintained by the process data store, then it is there because it is required by one or more of the activities that succeed it in the workflow.

For example, consider that X is generated by the activity A as output, but is not required by any of the following activities (say B) in order to execute. X generated by A can be considered redundant.

Analysis of the process model, can identify such cases, although the designer may still choose to maintain X as an output of B especially if the output destination is external.

Lost Data

Occurs when the same data item v_k has been specified as the output of two (or more) activities that may be executed in parallel as part of a fork structure in the process model. Thus the activity responsible for the last update cannot be established, which may potentially cause a problem for activities that require the data item later in the process.

In the workflow model fragment in the figure below, activities A and B are a part of a fork/synchronise structure. X is generated by both A and B separately. However, since there is no control dependency between A and B, it cannot be established which activity will complete first. Thus it cannot be established whether subsequent activities e.g. C, will get the value of X from A or B in other words, one update of X will be lost.

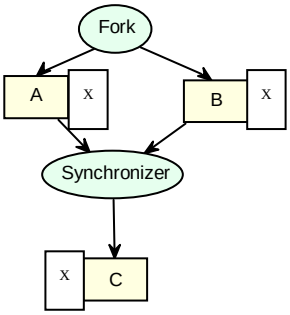


Figure 5. Lost Data

Such discrepancies may be identified during data validation. However, only a process designer can identify the resolution, which in this case maybe identifying one of A or B as the source for C; or ensuring that C receives the value of X with the most recent timestamp; or changing the data items altogether.

Missing Data

Occurs when a data item v_k has been specified in the input schema Input(n) for an activity but the source of the data item is unknown/unspecified That is, the source of the data item is not from either a performer of activities (human or application) or from (the output schema of) any previous activity.

For example, X is required as input for the activity B, but there is no source for X, and neither has X been established in the process data store as an internal data item by a preceding activity. X can thus be considered missing data.

Mismatched data

The case of mismatched data may arise when the structure of the data produced by the source is incompatible with the structure required by the activity, or more precisely by the application(s) invoked by the activity. An example of this would be if an activity writes (outputs) the postal address of a person and this information is required by a later activity, however the format of the address for the later task is different. The following figure illustrates the different representations of address by two different activities in the same workflow.

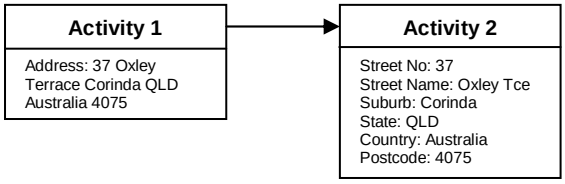


Figure 6. Mismatched Data

However, this mapping of data item structures is a well recognized problem, and many technology products address this quite effectively, (see e.g. Biztalk mapper) by providing GUI interfaces that facilitate the transformation. Transformation rules however need to be defined, and are commonly undertaken through mechanisms such as XSLT (Stylesheet language for XML transformations).

Inconsistent data

An activity retrieves a data item v_k from an application or human participant and stores that item and its value in the process data store. This data item and its value may be utilised later in the workflow by another activity and is thus read from the store. The problem is if during that idle time (after the item has been retrieved and before being read) the data item maybe updated externally to the workflow and this may cause an inconsistent view of the data. This problem indicates the need for a refresh strategy to be established for the data items in the process data store.

Data inconsistency can also be introduced when different activities deal with multiple sources for the same data item v_k at various times in the process. It is essential to ensure that the map of the data item can be established to identify any broken links.

For example, X is an input data received by an external source by activity A. X is processed and stored as internal data in the process store by A. A subsequent activity B in turn is receiving the same data from another external source, and processing it further before producing a different version of X, perhaps for an external entity. The inconsistencies between the value of X produced by the two activities must be identified as part of data validation.

Misdirected data

Occurs when data flow direction conflicts with control flow in the same workflow model. The control flow determines the order and selection of activities that need to be performed to satisfy underlying business objectives of the workflow model. The data flow determines the flow of data from *provider* activities to *receiver* activities. Thus a provider activity must always precede a receiver activity in terms of control flow.

Following figure shows some of the cases where conflicting control flow and data flow would compromise the correctness of workflow model:

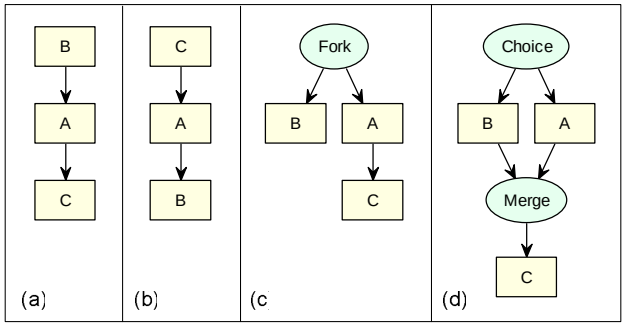


Figure 7. Conflicting Control Flow and Data Flow

Let us suppose that an activity C needs data provided by B to carry out its operations. If the activity B precedes activity C in the control flow model, as shown in 7(a), through a sequential structure, we can ensure that activity B would always be able to provide required data to activity C. However, if activity B is followed by C in the control flow model, as shown in 7(b), it would be executed after B and hence would not be able to provide required data to C. Similarly, activity B could also be in a parallel path to C, as shown in 7(c), and would only be able to provide data to C if it was completed before C was started. Activity B could also be in a Choice path preceding the activity C, as shown in 7(d), and may or may not be able to provide necessary data to C depending on the path selected by relevant Choice structure. Such conflicting control flow and data flow modelling aspects will only be applicable to specific instance types of a process model.

Insufficient data

Perhaps a critical, yet difficult question to ask is, if the data specified is sufficient to successfully complete an activity. There is a range of contextual and semantic issues in this regard which will depend on availability of process and exception histories, information regarding underlying applications, and expertise of designers. The discussion of this aspect of data validation is beyond the scope of this paper.

3.3 Data Flow Implementation Models

As identified in the preceding section, a number of generic data validation problems exist. How these problems are actually tackled, in terms of analysis and reporting, will be dependent on the particular implementation approach for a given conceptual data flow model. The underlying implementation model for data flow impacts on the development and effectiveness of data validation algorithms and even on the ability to execute them.

We identify at least the following implementation models for process data requirements and data flow:

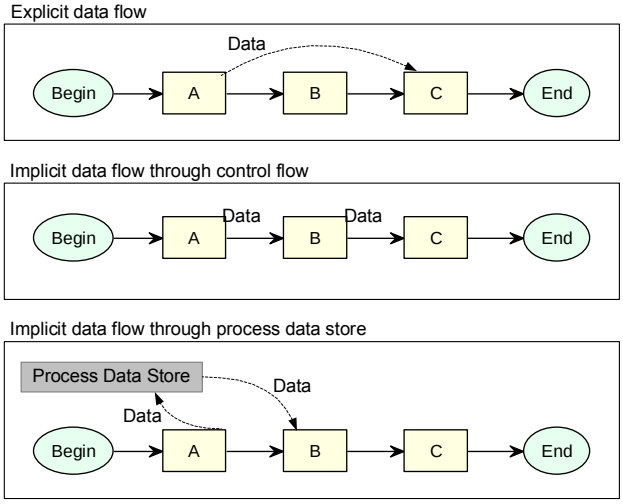


Figure 8. Data Flow Implementation Models

Explicit data flow

In this approach, similar to control flow, the data flow transitions are explicitly defined as part of the workflow model. These data flow transitions model the flow of data from one activity to another. For example, in above example an explicit data flow from activity A to C shows the data dependency of C on A. When activity A completes, it makes the input data required C through the explicit data flow. In this case, activity B is not aware of any data flow dependency between A and C.

Implicit data flow through control flow

This approach makes use of control flow to pass data from one activity to another. However, some data may be passed through activities that may not need it. For example, in above example, even though data C needs data provided by A, it has to pass through two control flows that connect B to A and C.

Implicit data flow through process data store

This is an approach which we advocate and use as a basis for our data flow validation research. The process data store is basically a repository of process maintained data (i.e. all input and output schemas which are identified as having an internal source/destination). The activities pass on their output data to the common process data store. Other activities may access the process data store to retrieve the data for their needs.

4 Conclusions

The main focus of this paper is to present data flow validation issues in workflow modelling. As such we have identified essential requirements of data flow modelling in workflow specifications, namely the type, source and structure of data. We have also identified seven generic data validation problems. Clearly effective data validation algorithms can be developed to address these problems. Although, presentation of the algorithms is beyond the scope of this paper, we have presented the foundation for implementing them.

We envisage the implementation of data validation algorithms through a validation tool with smart visualization and reporting capabilities. Such a tool can chart out data flow maps for process designers and provide a critical analysis of the process model prior to deployment, thereby boosting user confidence.

Acknowledgements

The authors would like to acknowledge the comments given by Tony O' Hagen of the Distributed Systems Technology Centre, Brisbane. This work is partly supported by the SAP Corporate Research Centre Brisbane.

References

Van der Aalst, W.M.P. (1997): Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pg. 407-426. Springer-Verlag, Berlin.

Casati, F., Jin. L. and Shan, M. (2001): Business Process Simulation with HP Process Manager. *Software Technology Laboratory, HP Laboratories.*

Casati, F., Jin. L. and Shan, M. (2002): Design of A Business Process Analyzer. *Software Technology Laboratory, HP Laboratories.*

Cooperative Research Centre for Enterprise Distributed Systems Technology (DSTC) (1999): *UML Profile for Enterprise Distributed Object Computing.*

Edger. J., Panagos, E., Pezewaunig, H. (1999): Time Management in Workflow Systems. *3rd Int. Conf. on Business Information Systems*.

Ferscha, A. (1998): Optimistic Distributed Execution of Business Process Models. *Proc. 31st Annual Hawaii International Conference on System Sciences*, Kohala Coast, HI, **7**:723-732, Software Technology Track.

Hlupic, V. and Robinson, S. (1998): Business process modeling and analysis using discrete-event simulation. *Proceedings of the Winter Simulation Conference*.

Jablonski, S. and Bussler, C. (1996): *Workflow Management: Modeling Concepts, Architectures and Implementation*. London; Sydney, International Thompson Computer Press.

Leymann, F. and Roller, D. (2000): *Production Workflow: Concepts and Techniques*. Sydney, Prentice-Hall of Australia Pty. Limited.

Marjanovic, O. (2000) Dynamic Verification of Temporal Constraints in Production Workflows *Proceedings of the Australian Database Conference* ADC'2000, IEEE Press, pp.74 – 81

Rakitin, Steven R. (1997) *Software verification and validation : a practitioner's guide*. Boston : Artech House, 1997.

Reichert, M. and Dadam, P. (1998): ADEPTflex – Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management*.

Robinson, S. (1997): Simulation Model Verification and Validation: Increasing the User's Confidence. *Proc. Winter Simulation Conference.*

Sadiq, W. and Orlowska, M. (1999) On Capturing Process Requirements of Workflow Based Information Systems. In *Proceedings of the 3rd International Conference on Business Information Systems* (BIS '99), Poznan, Poland, April 14-16, 1999. pp. 195-209. Springer-Verlag, 1999.

Sadiq, W. (2000a) On Verification Issues on Conceptual Modeling of Workflow Processes. PhD Thesis. The University of Queensland. 2000

Sadiq, W. and Orlowska, M.E. (2000b): Analyzing Process Models using Graph Reduction Techniques. *Information Systems* **25**(2):117-134.

Scheer, August-Wilhelm (2000) *ARIS – Business Process Modeling*, Third Edition, Springer-Verlag 2000.

Workflow Management Coalition (1999): *Workflow Management Coalition Terminology & Glossary / The Workflow Management Coalition Specification*. Winchester, Workflow Management Coalition.