Correction of Data-flow Errors in Workflows

Divya Sharma, Srujana Pinjala and Anup K Sen
Indian Institute of Management Calcutta
Joka, D.H. Road, Kolkata 700104, India
Email: {divyas12, pinjalas10, sen}@iimcal.ac.in

Abstract

When a workflow is designed, not only should it be free from control-flow errors like deadlocks and lack of synchronization, but should also be checked for data-flow correctness. Recent approaches have categorized data-flow errors and have suggested methods for detecting data-flow errors but only a few approaches have been reported yet on the automatic correction of data-flow errors. In this paper, we present methods and related issues for correcting data-flow errors in workflows. The methods can be incorporated in existing commercial Workflow Management Systems to make the software an intelligent system which will not only detect the data-flow errors, but also automatically suggest to the designer possible ways to correct the data-flow errors.

Keywords

Data-flow errors in workflows; Correction of data-flow errors; Information quality; Business process modeling

INTRODUCTION

A business process is a sequence of tasks that are performed in series or parallel to carry out a specific business function. The specific collection of tasks, resources and information elements involved in a particular circumstance comprise a workflow (Aalst and Van 2002; Dumas et al. 2013). An organization has several business processes, and associated workflows, varying in purpose, duration, individuals, teams or software that perform them. Workflow management systems help in modeling, verifying, automating, managing and optimizing business processes. Managing them efficiently is becoming more and more critical for the successful functioning of organizations.

When a workflow is designed, not only should it be free from control-flow errors like deadlocks and lack of synchronization (Liu and Kumar 2005; Wynn et al 2008), but it should also be checked for data-flow correctness. Every task in a workflow may require input data to execute and may produce data as output, which may again be used by subsequent tasks. Ensuring that every task and decision gateway in the workflow gets the data it needs leads to data-flow correctness. Data-flow errors can arise even when the control-flow structure is correct. As explained in (Sun et al . 2006; Trcka et al. 2009; Meda et al. 2010), basic types of data-flow errors, namely missing data, inconsistent data, lost data and redundant data, can be detected quite readily in workflows with simple loops. However, only a few approaches (Awad et al. 2010) have been reported yet on the correction of data-flow errors.

The aim of this research is to illustrate the issues in automatic correction of data-flow errors and to contribute methods for correction of some of the data-flow errors identified in literature so far. The methods can be incorporated in existing commercial Workflow Management Systems to make the software an intelligent system which will not only detect the data-flow errors but also automatically suggest to the designer possible ways to correct the data-flow errors. Thus the objectives of this paper can be summarized as follows:

- To describe methods for correction of various types of data-flow errors; for every data-flow error detected, the paper suggests appropriate actions to be taken to correct the data-flow error.
- To highlight issues that that managers may face while correcting the data-flow errors.

LITERATURE REVIEW

Workflows are represented graphically using formalisms such as BPMN (OMG 2006), Petri Nets (Aalst and Van 2002), YAWL (Aalst and Hofstede 2005). In addition to Exclusive (XOR) and Parallel (AND) gateways, Aalst et al. (2003) recommended the use of additional control-flow elements such as Inclusive (OR) gateways to enhance the expressive power of workflows. (Sun 2008) presents a workflow design methodology that is based on data-flow analysis.

Several methods are currently available for detecting control-flow errors that arise due to abnormal situations such as deadlock, lack of synchronisation, infinite looping during execution of a workflow. Some of these methods (Wynn et al. 2007; Aalst and Hofstede 2005) use Petri Nets to detect control-flow errors.

Verification of data-flow has not been studied as extensively as verification of control-flow. Sadiq (2004) identified seven basic data validation issues. Sun (2006) has regrouped these errors into three basic types, *viz.*, redundant data, missing data and conflicting data, and has described methods for their detection in nested workflows containing simple loops. This work has been extended and generalized in (Meda 2010); a novel feature is the application of the notion of corresponding pair (Liu and Kumar 2005) for the detection of lost data error. Trcka (2009) has proposed a petri net based approach for detecting data-flow errors. On the correction of data-flow errors, Awad et al. (2010) suggested a petri net based framework. Preserving data-flow correctness in process adaptation can be found in (Song et al. 2010).

DATA-FLOW ERRORS

We now briefly describe the data-flow errors termed as "anti-patterns" in (Trčka et al. 2009) with appropriate examples. Each of these errors can be weak or strong depending on whether it occurs in some of the instances or all of the instances of the workflow respectively. BPMN (OMG 2006) notation is used throughout the paper. We assume loopfree workflows.

Missing Data Error

If an activity requires a particular data item, but, the data item is not available at the time of execution of the task then a missing data error is said to occur. Such an error may occur when (a) the desired data item is accessed before it is initialized or is not initialized at all, (b) the desired data item is initialized by an activity subsequent to the activity desiring it (Sun et al. 2006).

Figure 1 shows an order shipping process. However, the data item 'Address' to which the order has to be shipped by Activity 'Ship Order' is neither produced by some activity in the process, nor is provided as a pure input. As a result, a missing data error is thrown.

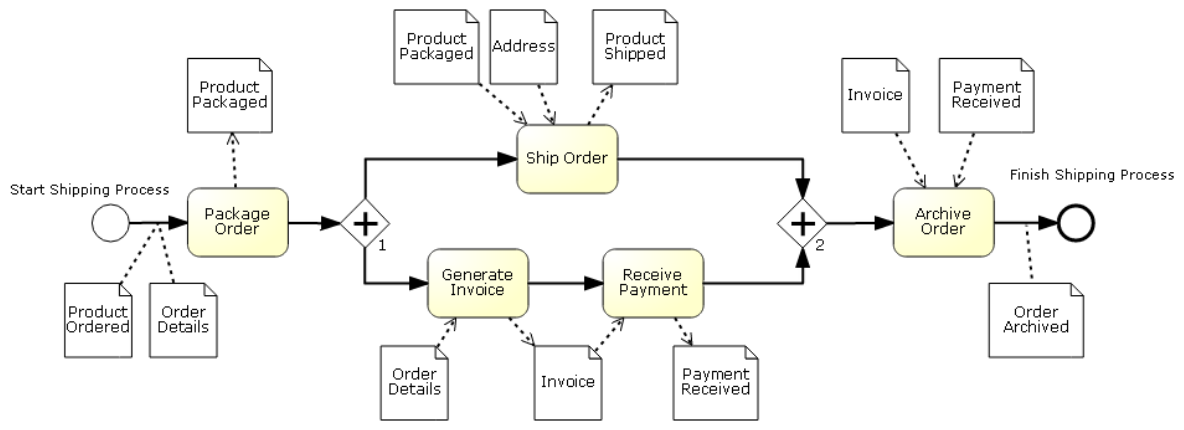

Figure 1. Order Shipment Process with Missing Data Error

Lost Data Error

A lost data error occurs when two activities which are executing in parallel produce the same data item or revise its value. As one activity overwrites the value of a data item produced by the other before the data item is read (Trčka et al. 2009), not only is the previously written value of the data item lost, but also it is difficult to establish which of the two activities last updated the data item (Sadiq et al. 2004). As a result, the outcome of subsequent activities is difficult to predict in case of this error. Sun et al. (2006) have referred to this error as conflicting data error. Further, Trčka et al. (2009) have discussed this error under the scope of lost data anti-pattern.

Figure 2 shows a loan assessment process where activities 'Check Income Source' and 'Check Credit History' both produce data item 'Loan Assessment'. As a result the activity which executes later overwrites the value of this data item which was produced by the activity which executed earlier.

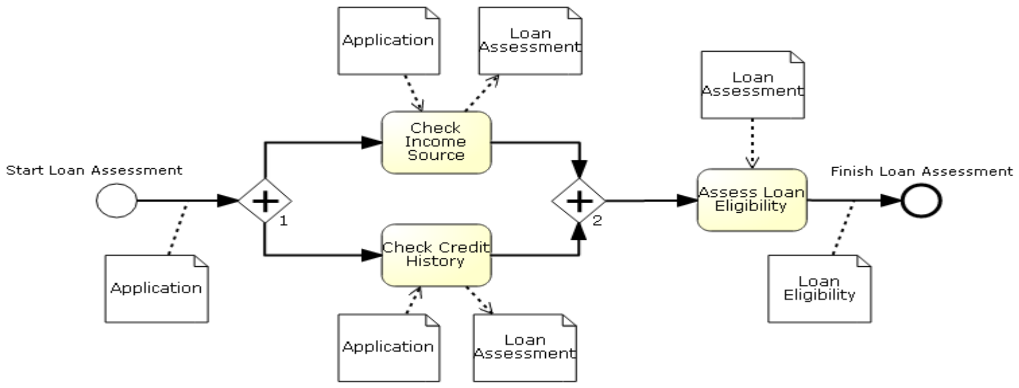

Figure 2. Loan Assessment Process with Lost Data Error

Inconsistent Data Error

When a data item required by a particular activity is produced or revised by another activity executing in parallel to the former, an inconsistent data error is said to occur (Trčka et al. 2009; Meda et al. 2010). The order of 'read' and 'write' determines the value of the data item read by the activity.

Figure 3 shows an order shipment process where activity 'Ship Order' reads data item 'Address' and activity 'Get Address' produces data item 'Address'. In this case, whether the product gets shipped to the address provided as a pure input or the one generated by the activity 'Get Address' depends on whether 'Ship Order' executes before or after 'Get Address'.

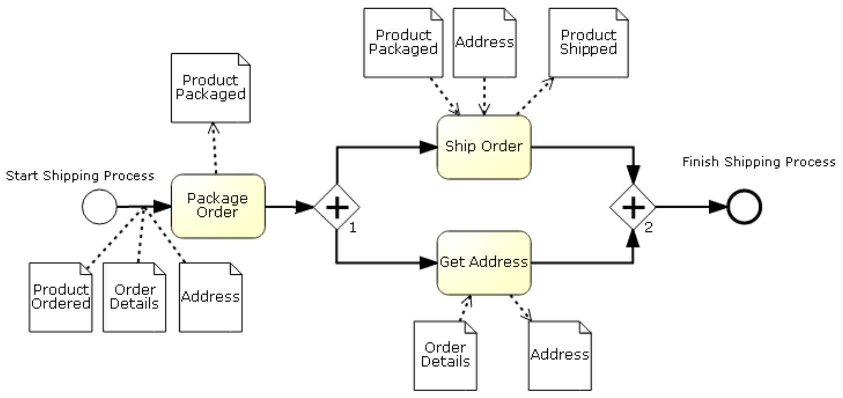

Figure 3. Order Process Shipping with Inconsistent Data Error

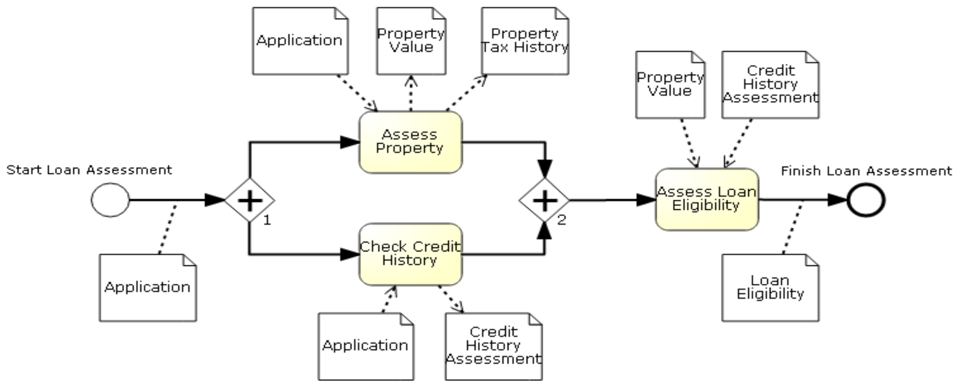

Figure 4.Loan Assessment Process with Redundant Data Error

Redundant Data Error

A data item is redundant in a process if it is produced by an activity, but, is of no subsequent use in the process. Such an error may occur when a data item produced by an activity is neither used by any subsequent activity, nor output as a result of the process execution (Meda et al. 2010). A redundant data error is said to be a weakly redundant anti-pattern if the data item is redundant only in some, but not all, instances of the workflow (Trčka et al. 2009). This type an error has also been termed as contingent redundancy (Sun et al. 2006). A redundant data error is said to be a strongly redundant anti-pattern if the data item is redundant in all instances of the workflow (Trčka et al. 2009). This type of error is also called inevitable redundancy (Sun et al. 2006).

Figure 4 shows an Loan Assessment Process where the activity 'Appraise Property' produces a data items 'Property Value' and 'Property Tax History', but the latter of these items is not used by any other activity in the process and is, hence, redundant.

AUTOMATIC DATA-FLOW ERROR CORRECTION AND RELATED ISSUES

In this section we suggest methods for correcting data-flow errors and discuss the resulting issues. In order to correct data-flow errors, the following actions may be taken:

(a) Introduction of a new activity
(b) Removal of an activity from the workflow
(c) Shifting an activity in the workflow
(d) Merging/Splitting of activities
(e) Modifying the nature of an activity in the workflow
(f) Inserting Exclusive or Parallel gateways

Missing Data Error:

Figure 5 below illustrates the case of a strongly missing data error where activity D requires data item *e* which is neither supplied as a pure input, nor is being produced by any activity preceding activity D.

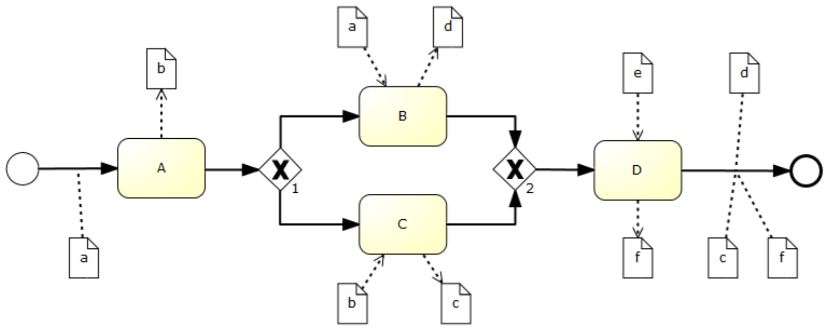

Figure 5. Strongly Missing Data Error

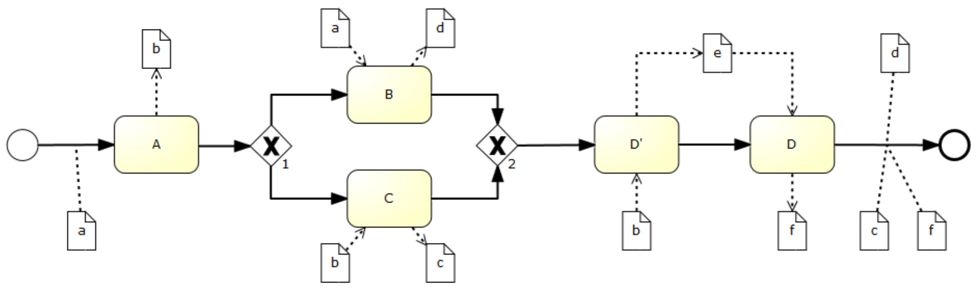

Figure 6. Corrected Workflow for Strongly Missing Data Error

The solution is to introduce a task that produces the required data item at some point in the workflow before the task requiring it. While the new task may be introduced in the very beginning, at a point mid-way or immediately before the task requiring the data item, it is suggested that this task be introduced immediately preceding the task requiring the data item. This is because, by delaying the introduction of this task to as late as possible, availability of as many data items as possible to produce this missing data item is ensured. Figure 6 illustrates this solution, where another activity D' is introduced to produce data item *e* using data item *b*.

The solution is similar in case of weakly missing data error when a task in one branch of a workflow is missing a data item. The solution can be generalized with an additional Exclusive-decision (XOR-split) gateway inserted between the new activity and the multiple branches of Exclusive-decision 1 where the data item is missing.

Issues: While introduction of a new activity in a workflow diagram is quite straightforward, doing so in a real life organizational setting is all the more difficult. Managers must keep the manpower and skill availability in mind before introducing new tasks to resolve missing data errors. Managers may also like to contemplate the possibility of modifying existing roles and responsibilities such that activities already present in the workflow may be able to produce the missing data items.

Special Case of Delayed Initialization

It might happen that a data item which is missing when an activity desires it, but, is being produced by a following activity in the workflow. Such an error is called delayed initialization (Sun et al. 2006). Figure 7 illustrates this scenario where activity B desires data item *c* which is produced by activity C that follows it.

This error may be resolved in either of the two following ways:

- Activity C may be shifted to a point immediately preceding activity B
- Activity B may be shifted to a point immediately following activity C

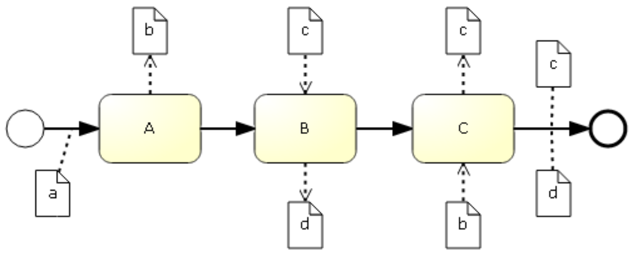

Figure 7. Special case of Delayed Initialization

Issues: A manager may choose either of these solutions to resolve this error, but, none of these methods is free from issues. If the latter of the two activities is moved before the former, another delayed initialization error may be introduced if it requires some data item which has not been produced so far. On the other hand, if the former of the two activities is moved after the latter, other data items which it may require may already have undergone a change by some other activity in the workflow**.**

Inconsistent Data Error

This error occurs when one activity reads a data item while another activity, executing in parallel along the two branches of a parallel gateway pair, writes the data item. An inconsistent data error may occur in two scenarios:

- *Read is not dependent on own branch:* Activity reading the data item does not use other data items produced by activities along the same branch.

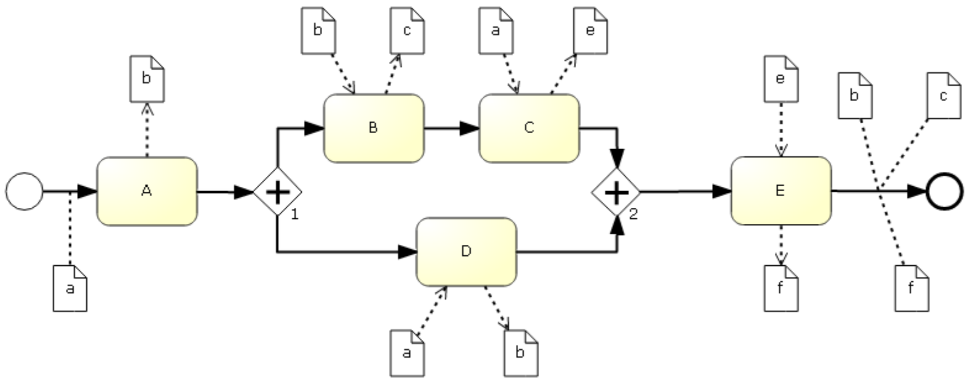

Figure 8. Inconsistent Data Error

Consider Figure 8 where activity B is reading data item *b* to produce data item *c* and, in parallel, activity D is reading data item *a* and producing data item *b*. Since the same data item is being read and written along different branches of the same parallel gateway pair (1, 2), an inconsistent data error for data item *b* is encountered. In this example activity B does not use any data item which is produced along the same (upper in this case) branches of parallel gateway pair (1, 2).

In order to resolve this error, the activity which is reading the data item is shifted to the other parallel branch, immediately following the activity which is writing the data item (Figure 9).

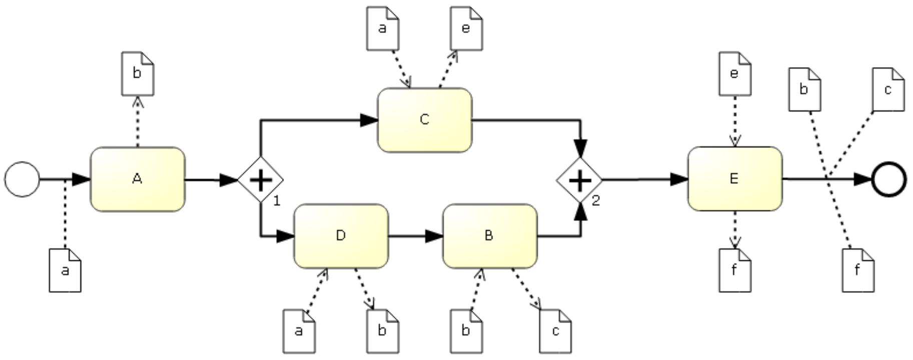

Figure 9. Corrected Workflow for Inconsistent Data Error

- *Read is dependent on own branch:* Activity reading the data item uses some data items produced by activities along the same branch

In more complex data-flows, resolution by shifting a task from one branch to another branch may lead to a cyclical data-flow error correction without reaching a possible resolution. Figure 10 illustrates this type of an issue. Activity C requires data item *c* and *d* as input; data item *c* is produced by activity D in the parallel branch and data item *d* is produced by activity B in the same branch.

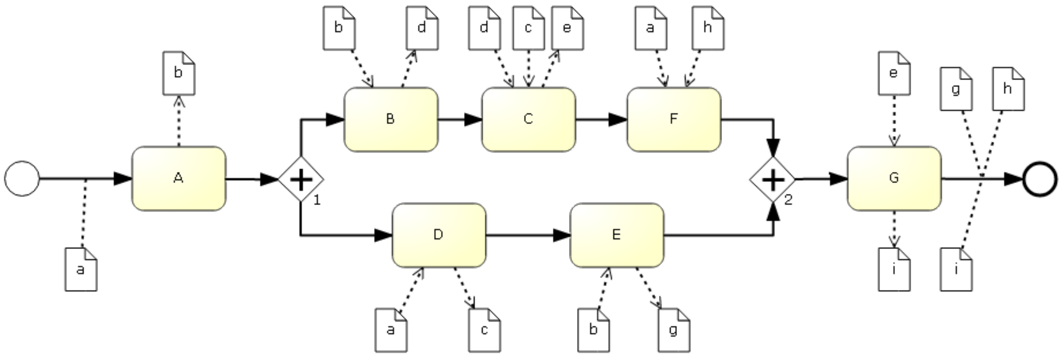

Figure 10. Inconsistent Data Error due to data item *c* (Activity C and D)

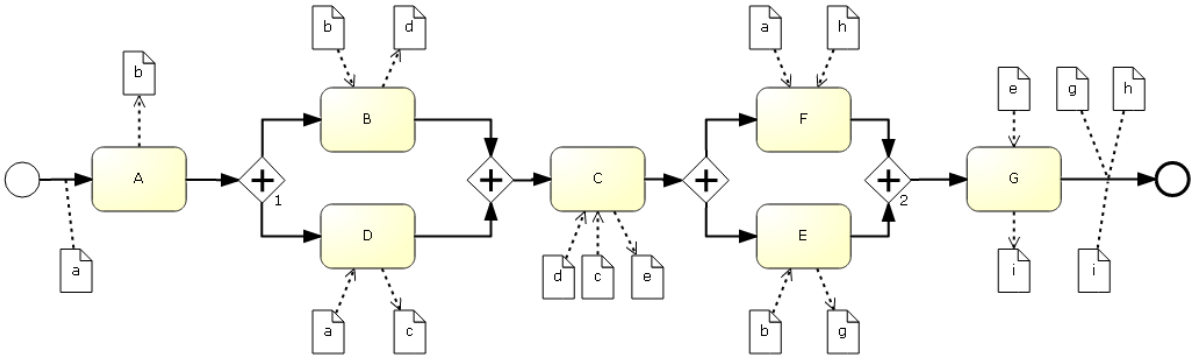

Figure 11. Solution of Inconsistent Data Error shown in Figure 10

It is, therefore, instructive to highlight that the solution of shifting the activity reading the data item to a point immediately following the activity writing the data item is not useful when one of these activities uses data items being produced by some other activity along their own branch. In such a scenario, the problem may be resolved by introducing a Parallel Synchronize (AND-join) gateway joining the two branches where the activities write the data items, followed by the activity that reads those data items, which is in turn followed by a Parallel Fork (AND-split) gateway as show in Figure 11.

Issues: (i) The outcome of the proposed method is serialization of the tasks which read and write the same data item. In the worst case scenario, all activities which were desired to be executed in parallel may get serialized. The manager must reconsider the data-flow and parallelization in such a scenario.

(ii) When two activities executing in parallel are reading and writing the same data item, it may be desirable in some cases that the activity which reads the data item should read the old value of the data item. In such a scenario, the activity which reads the data item may require to be shifted immediately prior to the activity which is writing the data item. The manager needs to make an informed decision when deciding on the resolution of this error.

It must be noted that the proposed solution is applicable to both, weak and strong forms of inconsistent data error. This is due to the structure of the workflow which causes such as error, i.e., a Parallel gateway pair.

Lost Data Error

Figure 12 shows a workflow in which activity D and E are both producing data item *e* in parallel. Such a scenario gives rise to lost data error as the data item written by the activity which executes earlier gets overwritten by the activity which executes later. The final value of data item *e* depends on the order in which the two activities D and E get executed; as a result this is a case of a race condition.

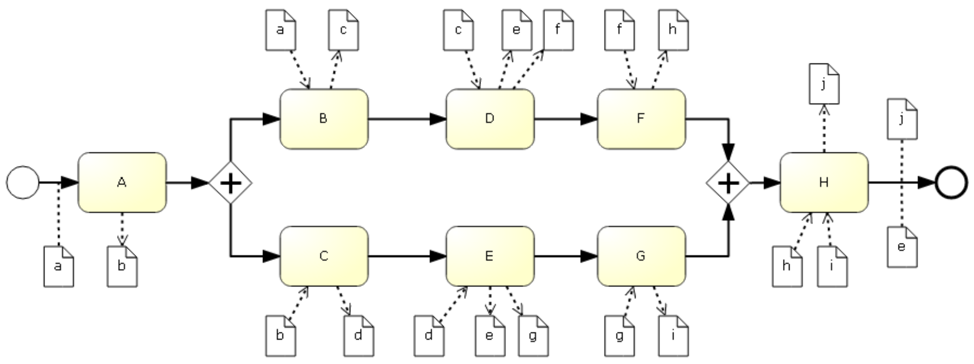

Figure 12. Example of Lost Data Error

In order to resolve this error, it is important to understand what might be happening physically in the process. When two activities simultaneously produce the same data item it means that the two activities can be performed in parallel without being resource constrained or violating the control-flow. However, the problem in the data-flow requires a rethinking about the two tasks producing the data item. In the Example of Figure 2, the data item 'Loan Assessment' is produced by both the activities 'Check Income Source' and 'Check Credit History' using different parts of the 'Application'. To resolve this issue, steps in these activities producing 'Loan Assessment' may be combined. This means that in Figure 12, activity D should not produce date item *e* and instead activity E should be modified to produce data item *e by* combining the steps from activity D or vice versa.

Issues: The solution to the lost data error involves change in organizational roles, responsibilities and work-loads. Such changes are often met with resistance in the organization because long since the activities have been performed in the organization. As a result, the manager needs to keep in mind the implications for the organizational hierarchy and employee satisfaction when resorting to this type of a resolution.

Redundant Data Error

In case of weakly redundant data error, the error occurs only in some, but not all instances of the workflow. In Figure 13a activity A produces data item *b* and *b'*. Now, when the lower branch of the Exclusive gateway pair (1, 2) is executed, this data item *b* is used by activity C. However, when the upper branch of the Exclusive gateway (1, 2) is executed, this data item is not used by any subsequent activity. It is important to understand that this type of error can occur only when the activity which is using the data item lies along some branch of an Exclusive gateway pair.

In this situation, the solution to the problem is slightly complex, where firstly the activity producing the redundant data item has to be suitably modified such that it does not produce the data item which is redundant in some cases, and secondly another activity which produces this data item needs to be introduced immediately prior to the activity using it (Figure 13b). If the activity A produces only the redundant data item b, only shifting of the activity would suffice. The solution can be generalized with an additional Exclusive-decision (XOR-split) inserted between the shifted activity and the multiple branches of Exclusive-decision 1 where the data item is used. The solution is similar for the strongly redundant data error.

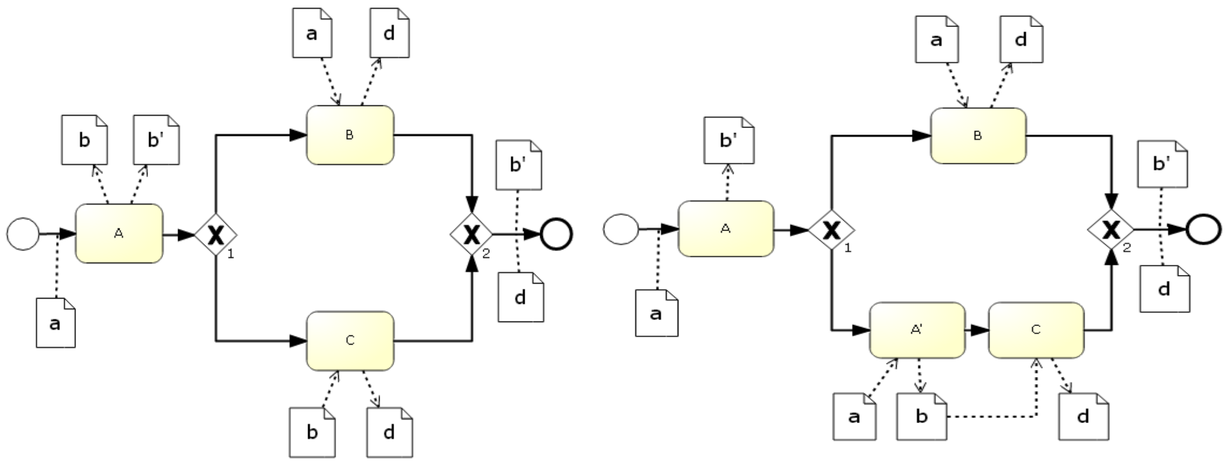

Figure 13a & b. Weakly Redundant Error –Activity producing multiple data items and its corrected version

Issues: It must be remembered that modifying the nature of an activity may require change in employee roles and retraining of staff. As a result, the manager must exercise caution in taking recourse to such a change when trying to resolve redundant data errors. Further, the same activity may be part of multiple processes in the organization due to which a data item which is redundant in one process may actually be used in some other process.

Based on the suggested ways to resolve data-flow errors occurring in workflows, the outline of an integrated algorithm is presented in Figure 14. Since resolution of one data-flow error may introduce another data-flow error, an iterative approach for resolution of data-flow errors is presented. It is assumed that one of the existing approaches for data-flow error detection may be suitably modified to identify one error at a time and resolve it using the methods proposed in this work.

MANAGERIAL IMPLICATIONS

The following issues which may occur as a result of correction of data-flow errors must be carefully addressed by the managers:

- *Data-flow error correction should not introduce control-flow errors in the process*

The correction of data-flow errors often involves change in the sequence of activities and introduction of new gateways. As a result, control-flow errors may creep into the workflow and prevent it from functioning properly. Managers must, therefore, carry out a control-flow error check after resolution of data-flow errors.

- *Repeated correction of data-flow errors should be done carefully*

Resolution of one data-flow error E_1 may in turn introduce another data-flow error E_2; resolution of data-flow error E_2 may introduce data-flow error E_3 and so on. However, the suggested approach to data-flow error correction is based on the assumption that repeated correction of data-flow errors will eventually lead to an error-free workflow. However, in some cases, this kind of error correction may lead to re-introduction of an error which was previously corrected in the workflow. That is to say, resolution of error E_3 may lead to introducing error E_1 again. If managers encounter this type of situation, it may point to poor workflow specification and managers may have to redesign the workflow.

- *Data-flow error correction should not lead to changed semantics of the process*

Changing the nature of the activities, sequence of activities and control-flow for resolving the data-flow errors may lead to a change in the output of the process itself. Such a resolution of data-flow errors is undesirable. The process owners must ensure that the workflow is still performing the task that it was supposed to perform.

```
Algorithm CorrDF
Input: A control-flow model of a workflow
Output: Suggested workflow without data-flow errors

while data-flow  errors exist do find the next data-flow error;
case data-flow error of
   Missing data-flow error:
        introduce an activity which can produce the missing data preferably just before the activity or  just
        before all the branches where the error occurs;
        if the error is weak then
            insert an Exclusive Decision (XOR-split) gateway between the new activity and the all the
            branches where the error occurs;
        endif
   Inconsistent data-flow error:
        for every set  of parallel paths where the error is generated do
            if the activity reading the data item  does not depend on other data items in its own branch then
               activity may be shifted to the parallel branch following the activity writing it
            else  introduce Parallel Synchronize (AND-join) gateway after the activities writing the relevant
                data in parallel paths;
                follow it by the activity reading the data items;
                introduce a Parallel Fork (AND-split) gateway for creating parallel flows as exists for the
                remaining tasks;
            elseif
        enddo
  Lost data data-flow error:
        for every set of parallel paths where the error is generated do
            parts in parallel activities that  produce the same data item must be combined into one of the
            activities;
        enddo
  Redundant data-flow errors:
        split the activity that produces redundant data into separate activities – one each for producing a
        redundant data item  and one for producing the other items  if any;
        shift each of the activity which is producing a redundant data item preferably just before all the
        branches where the data item is being used in case of weakly redundant case;
        insert an Exclusive Decision (XOR-split) gateway between the new activity and the branches;
 caseend
dowhile
```

Figure 14. Algorithm *CorrDF* for correcting Data-flow errors

- *Cyclical dependency of data items should be avoided*

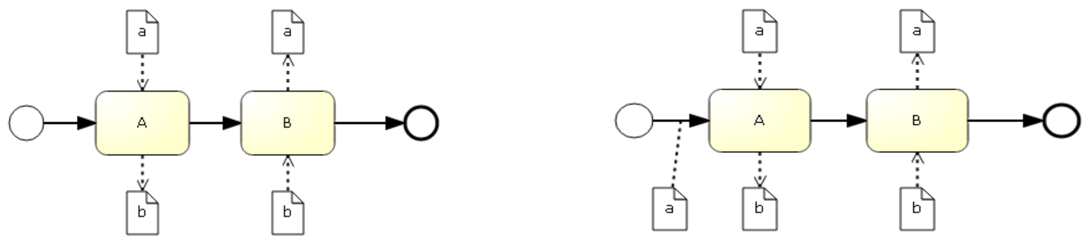

Figure 15a & b. Cyclical dependency of data items and its correction

Figure 15a shows a simple workflow with cyclical dependency of data items where activity A uses data item *a* which is produced by activity B and activity B uses data item *b* which is produced by activity A. This is an example of poor data-flow definition. Managers must ensure this type of data-flow is not present in a workflow.

This type of cyclic dependency of data can often be resolved by providing one of the data items as a pure input or introducing a task which produces the data items. Figure 15b shows a possible solution.

- *Data-flow error correction should not lead to disruption in the organization*

Managers must keep the manpower and skill availability in mind before introducing new activities or modifying old activities. Managers may also like to contemplate the possibility of modifying existing roles and responsibilities such that activities already working satisfactorily as part of multiple processes should not be disrupted. Managers must, therefore take a holistic view of their organization before modifying an activity.

CONCLUSION

The paper presents methods and related issues for correcting data-flow errors in workflows. There are two directions in which the current work is in progress. Firstly, we are in the process of implementing and testing the algorithm on a number of workflow models covering various types of unstructured (non-nested) workflows. The empirical performance of the algorithm is thus being judged. Secondly, we are extending the algorithm to workflows with loops.

REFERENCES

Aalst, W.M.P. And Hee, K. Van. 2002. *Workflow Management: Models, Methods and Systems*, The MIT Press.

Aalst, W.M.P., Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P. 2003. "Workflow Patterns", *Distributed and Parallel Databases* (14:1), pp 5-51.

Aalst, W.M.P. and Hofstede, A.H.M. 2005. "YAWL: Yet Another Workflow Language", *Information Systems* (30:4), pp 245-275.

Awad, A., Decker, G., & Lohmann, N. (2010). "Diagnosing and repairing data anomalies in process models" In *Business Process Management Workshops*, LNBIP 43, Springer Berlin Heidelberg, pp. 5-16.

Dumas**,** M., La Rosa, M., Mendling, J., and Reijers, H.A. 2013. *Fundamentals of Business Process Management*, Springer-Verlag.

Liu, R. and Kumar, A. 2005. "An analysis and taxonomy of unstructured workflows", In *Proceedings of the 3rd International Conference on Business Process Management*, Nancy, France, Sep. 2005, Lecture Notes in Computer Science 3649, Springer-Verlag, pp 268-284.

Meda, H. S., Sen, A. K. and Bagchi, A. (2010). "On detecting data-flow errors in workflows". *Journal of Data and Information Quality* (2:1), Article No.: 4.

OMG. 2006. "Business Process Modeling Notation Specification, BPMN 1.0", *OMG Final Adopted Specification dtc/06-02-01*, www.bpmn.org, February.

Sadiq, S., Orlowska, M., Sadiq, W. and Foulger, C. 2004. "Data-flow and validation in workflow modeling", In *Proceedings of the 15th Australasian Database Conference,* Dunedin, New Zealand, January, pp 207-214.

Song, W., Ma, X., Cheung, S. C., Hu, H., & Lǘ, J. (2010). "Preserving data flow correctness in process adaptation". In *Proceedings of Services Computing (SCC), 2010 IEEE International Conference,* pp. 9-16.

Sun, S.X., Zhao, J.L.,Nunamaker, J. and Sheng, O. 2006. "Formulating the data-flow perspective for business process management", *Information Systems Research,* (17:4), pp 374-391.

Sun, S.X. and Zhao, J.L. 2008. "Developing a workflow design framework based on data-flow analysis", In *Proceedings of the 41st Hawaii International Conference on System Sciences,* Big Island, Hawaii, January.

Trcka, N., Sidorova N. and Aalst, W.M.P. 2009. "Data-Flow Anti-Patterns: Discovering Data-flow Errors in Workflows", In Proceedings of the *21st International Conference on. Advanced Information Systems (CAISE'09),* volume 5565 of Lecture Notes in Computer Science, Springer, 2009, pp 425-439.

Wynn, M.T., Verbeck, H.M.W., Aalst, W.M.P., Hofstede, A.H.M. and Edmond, D. 2009. "Business process verification – finally a reality!", *Business Process Management Journal* (15:1), pp 74-92.

INTELLECTUAL PROPERTY