International Journal of Computer Science and Applications,

Vol. 12, No. 1, pp. 16 – 25, 2015

TOWARDS AN ACTIVE HELP ON DETECTING DATA FLOW ERRORS IN BUSINESS PROCESS MODELS

MOHAMMED ISSAM KABBAJ

Analyse, Modélisation et Intégration des Processus et Systèmes,
Ecole Mohammadia d'Ingénieurs, Mohammed V University, Avenue Ibnsina B.P. 765 Agdal,
Rabat, 10090, Morocco
kabbaj@emi.ac.ma

BETARI ABDELKADER

Mathématiques Appliquées, Traitement du Signal et Informatique, Mohammed Premier University,
Oujda, 60000, Morocco
betari.ump@gmail.com

ZOHRA BAKKOURY

Analyse, Modélisation et Intégration des Processus et Systèmes,
Ecole Mohammadia d'Ingénieurs, Mohammed V University, Avenue Ibnsina B.P. 765 Agdal,
Rabat, 10090, Morocco
bakkoury@emi.ac.ma

ASSIA RHARBI

Analyse, Modélisation et Intégration des Processus et Systèmes,
Ecole Mohammadia d'Ingénieurs, Mohammed V University, Avenue Ibnsina B.P. 765 Agdal,
Rabat, 10090, Morocco
rhassia@gmail.com

Successful business process management depends on effective Workflow design, modelling and also analysis [Sun *et al.* (2006)]. Many analysis techniques have been developed focusing on verification and discovery of design errors. Most of them check only the control flow perspective while ignoring data. However the role of data in the workflow is important since routing constraints in a workflow are typically based on data. In addition, data flow errors can arise even when the control flow structure is correct. A workflow model containing data-flow errors can cause unexpected process interruptions, resulting in high costs to debug and fix at the execution phase. The goal of this paper is to address an important issue in workflow data analysis by active help. Recently many business process data analysis models have been proposed, offering a passive help to the analyst. Which means that when analysing a workflow, it usually consists of creating the model, incorporating data and then ensuring its correctness. Still, the correction of errors discovery doesn't guarantee that the resulting model is correct. So, it is unavoidable to revalidate the entire model to ensure its correctness. In this paper we introduce an approach for ad hoc detecting data modelling errors in business process models. It consists on applying verification when modelling. At each time, we verify the fragment of the model and correct it before to continue modelling and so, we have a complete model fragment. Thus we don't need to revalidate the entire model at the end. We present a "DataRecord" concept that we have used to store the last set of each data in the model and the last activities that have read, update or destroy this data.

Keywords: Data flow; data validation; lost data; missing data; redundant data.

1. Introduction

A business process is considered as "the specific ordering of work activities across time and place, with a beginning, an end, and clearly identified input and output" [Davenport (1993)]. It is the core of most information systems since it reflects the business activities and their relationships in an organization. The automation of processes by an IT infrastructure, in whole or in part is commonly known as workflow. Designing workflow model is a complicated and error prone task. This activity will be more complicated if the goal is to complete the model with more information such as the integration of data risk management [Lhannaoui *et al.* (2013)] [Lhannaoui *et al.* (2014)]. This is why, several techniques was employed to fill this gap, such as process mining discovery [Boushaba *et al.* (2013)] [Boushaba *et al.* (2014)]. In all these cases, it is still very important to verify workflow models and detect errors during the design phase rather than the execution one or integrated more information. Verification needs to be applied on the conceptual level in order to detect errors that can cause unexpected process interruptions during execution and also to reduce the high costs to debug and fix these errors at runtime. Typically, workflow are looked from 3 perspectives [Trˇcka *et al.* (2008)]: (1) the control flow perspective that consists of a set of coordinated tasks describing the behaviour of the workflow, (2) the data flow perspective witch define what data are consumed and produced with respect to each activity in a business process and (3) the resource perspective describing the originators of tasks. The topic of workflow analysis has been studied since the mid-nineties. Unfortunately, existing analysis techniques typically check for errors in control flow perspective such as deadlocks, livelocks (infinite loop), etc. while ignoring data and resources. Resources are external and dynamic hence it can be ignored during verification however the role of data in the workflow is important since routing constraints in a workflow are typically based on data. In addition, data flow errors can arise even when the control flow structure is correct.

Traditionally, workflow designer create the control flow structure first, incorporate the flow of data second and then proceed to formal or informal verification to detect errors. This approach of verification offers a passive help to the designer, because before to check the correctness of the model, he have to achieve the entire model. Repairing detected errors doesn't ensure that the result is a correct model. It is obligatory to revalidate the model.

In this context, the contribution of our paper is to introduce an approach for an active help on detecting data modelling errors in business process model. Verification is applied on the modelling time. At each time, the designer has a complete fragment of model so he needn't to revalidate the entire model at the end. We used a "DataRecord" concept to store the last sate of each data in the model and the last activity that have read, update or destroy this data.

The paper is structured as follows. The next section will reports on related work. Different types of data anomalies are discussed in section 3. The approach used of detection of such anomalies is presented in section 4. Section 5 gives an illustrating example and in section 6 we conclude the paper and point to future work.

2. Related Work

Most existing research of workflow correctness criteria focus on the control flow perspective of workflows and ignore the also important data perspective. Only little work has actually been done to extend the existing results on control flow verification to the setting with data.

Reference [Sadiq *et al.* (2004)] have started investigating the different problems of dataflow validation and identified the essential requirements of dataflow modeling in workflow management. They showed the importance of dataflow verification in workflow processes and define seven types of data anomalies: redundant data, lost data, missing data, mismatched data, inconsistent data, misdirected data, and insufficient data. However, no concrete solutions to the dataflow validation problems have been reported nor detecting algorithms are proposed. Furthermore, operations on data are only classified into read and write type.

Later, [Sun *et al.* (2004)] and [Sun *et al.* (2006)] conceptualized these anomalies using UML diagrams, and gave supporting verification algorithms. They formulate the data-flow perspective by means of dependency analysis. The data-flow matrix and an extension of the UML activity diagram are proposed to specify the data flow in business process. Then, three basic types of data flow anomalies, missing data, redundant data, and conflicting data, were defined. Based on the dependency analysis, algorithms to data-flow analysis for discovering the dataflow anomalies are presented and execute in $0(n^3)$ time. However, as in [Sadiq *et al.* (2004)] work there is no explicit model proposed to characterize data behaviors. Also, the operation types are considered only read and initial write. This work was further extended and generalized in [Sundari *et al.* (2007)].

Reference [Trˇcka *et al.* (2009)] systematically presented anti-patters for workflows as temporal logic formulae including data as a first class citizen. The approach depends on the data anomalies discussed in [Sun *et al.* (2004)] to discover such anomalies in annotated workflow nets. Each anti-pattern is formally described using temporal logic CTL*. This approach takes into account both control and data flow perspectives.

Another work that uses also model checking techniques to verify business workflow from both control and data-flow perspective is [Sun *et al.* (2004)]. The underlying workflow language is UML diagrams as opposed to the petri net approach taken in [Trˇcka *et al.* (2008)].

Another approach to detect and resolve data anomalies in process models was introduced in [Awad *et al.* (2009)]. Based on the notion of data objects and their states, they extended the formalization of BPMN using Petri nets. They identify a set of modeling errors that appear at the interplay between data flow and control flow.A recent approach to detect dataflow errors in business process model were discusses in [Rgibi *et al.* (2011)]. This work provides a systematic technique of possible flaws related to the flow of data in business workflows. They formulated these errors as data-flow in RWD (Read, Write, Destroy) Boolean table that describe the data flow issue by split the workflow to sequential workflow and define the input and output to each task, create RWD Boolean table to all data element and compare current task data table with next task, by using

dataflow issue definition, they can monitoring and detect dataflow and dataflow errors, task by task until they get the end of flow. Yet, the approach is still limited to verification and does not provide remedy suggestions.

To sump, all these works provide a passive assistance to the user. The business process modeler first designs model and then check it to discover the control or data flow errors. However, repairing these errors doesn't mean that the model become correct. In this case, a revalidation of the entire model is necessary to ensure that there is no error. In our approach, we try to anticipate the error and provide a tool for real-time analysis. When modeling, the verification process is triggered whenever a model fragment is added. This gives us every time a free error fragment of the full model. The verification process is done as and when modeling. This is a modeling approach assisted by detecting data errors.

3. Data Anomalies in Business Process

In a workflow, each activity may consume input data and produce output data. If the dataflow are not specified correctly in a workflow system, errors and conflicts can occur referred to as dataflow anomalies [Sun (2007)]. Some of these anomalies are serious flaws (like, e.g., the missing data error), while other are not necessarily erroneous but only capture undesired behaviour (like, e.g., the redundant data error). Dataflow anomalies can be classified into three major types: missing data, conflict use of data and redundancy anomalies. A missing data anomaly: occurs when a data is accessed before it is created (initialized). A redundant data anomaly: occurs when an activity produces an intermediate data output but this data is not required by any succeeding activity. A conflicting data anomaly occurs: when different versions of the same data exist.

4. Our Approach

To specify dataflow in workflow, we introduce the concept of "DataRecord". It is an n x p matrix where n is the number of data items and p is the number of possible XOR branch's in the model+1. The matrix records the various data items in the workflow. For each data we specify its state witch is a triplet (x, y, z) where x, y and z refer to the operation each activity performs on data items. We categorize these operations into read, write/update and destroy. The value of x, y and z denote respectively the reading, writing and destroying state of data. If x, y or z is 0 it respectively means that the data is not read, write or destroy by any activity otherwise we put the activity that has respectively read, write or destroy. The state (R_{A1}, W_{A2}, D_{A3}) of a given data d means that d is read by activity A1, it's written by A2 and destroyed by A3. In case of data constraint like XOR split, we put on the DataRecord the two branches of the XOR containing the state of each data after each branch.

The DataRecord is initially empty, and as and when we draw the business process model, we insert data while respecting number of rules. These rules are deduced from the definition of each data anomalies given in section 3 and help us to detect whether an error exists before continuing our modeling. In case of error, modelling is locked until the error

is corrected otherwise the data is inserted into the DataRecord and hand is given for modelling. Next we expose these rules.

4.1. *Rule1: detecting uninitialized data error*

For an activity a given data d with the state (x,y,z). If d is inserted for the first time in the DataRecord and x <> 0 we have an error **==>** uninitialized data (missing data).
Explanation: The fact that this data is inserted for the first time in the DataRecord means that it has never been created before, while an activity tries to read this data (x <> 0). This leads to a missing data error.

4.2. *Rule2: detecting lost data error (conflict data error)*

For a given data d with the state (x,y,z). If d exists in the DataRecord (y <> 0) and not yet removed (z = 0), and if in the last copy of the DataRecord x = 0 (data not yet read). Update for this data leads to an error **==>** lost data or conflict data.
Explanation: This data exists in the dictionary with a given value and this value has not yet been read by any activity, updating this data means losing its initial value and so creates a conflict in reading the value of this data.

4.3. *Rule3: detecting missing data error*

Before to destroy data, verify that this data exists in the DataRecord. If yes: remove it from the DataRecord. Otherwise: declare an error **==>** missing data

4.4. *Rule4: detecting redundant data error*

In the latest copy of the DataRecord if there exist any data such as y <> 0 and x = 0, declare an error **==>** redundant data.
Explanation: the fact that in the latest copy of the DataRecord, some data have y <> 0 and x = 0 means that these data were created by some activities but after were not read by any activities. Therefore these data are redundant
Note that, even if this rule is applied only at the end, it does not require a revalidation of the entire model since redundant data will be removed from the model or reused later in other context.

5. Illustrating Examples

5.1. *Simple model: linear model*

As an illustrating example, we will use the same example in [Rgibi *et al.* (2011)] to show that with our approach we discover more errors. Consider the following example:

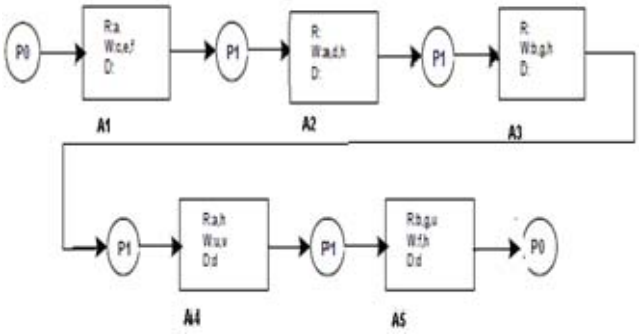

Fig. 1. Linear model.

In this model, three errors were discovered using the approach in [Rgibi *et al.* (2011)]; missing data on a and redundant data errors on d and v.

Using our approach, we begin modelling by drawing P0, A1 and incorporate data in A1. The DataRecord is initially empty and before any insertion we apply the rules presented above. After applying rule 1 we detect an uninitialized data error on a. In this case, modelling tool lock modelling and report the error to the user with a list of a possible correction. Proposed solutions are: create a on A1 or not to read a on A1. Suppose the modeller chose not to read a in A1. The DataRecord contains the following data (for readability, we do not display the other columns of the DataRecord as they are empty):

Table 1. The first DataRecord of the linear model.

Data	State
c	$(0,W_{A1},0)$
e	$(0,W_{A1},0)$
f	$(0,W_{A1},0)$

Then, we add P1 and A2 to the model and incorporate data into A2. The new state of DataRecord is:

Table 2. The second DataRecord of the linear model.

Data	State
c	$(0,W_{A1},0)$
e	$(0,W_{A1},0)$
f	$(0,W_{A1},0)$
a	$(0,W_{A2},0)$
d	$(0,W_{A2},0)$
h	$(0,W_{A2},0)$

We add A3 with its data. After applying Rule 1 and Rule 2 we detect a lost data error on data h. this data was created by the activity A1 and A2 try to update it however it was not read between these two activities. In this case, modelling tool lock modelling and report the error to the user with a list of a possible correction. Proposed solutions are: only A1 or A2 have to write h or h must be read in A2 before the update.

Suppose the modeler chose to read h in A3 before the update. The new state of DataRecord is:

Table 3. The third DataRecord of the linear model.

Data	State
c	$(0,W_{A1},0)$
e	$(0,W_{A1},0)$
f	$(0,W_{A1},0)$
a	$(0,W_{A2},0)$
d	$(0,W_{A2},0)$
h	$(R_{A3},W_{A3},0)$
b	$(0,W_{A3},0)$
g	$(0,W_{A3},0)$

We continue modelling and draw P1 and activity A4. The new state of DataRecord is:

Table 4. The fourth DataRecord of the linear model.

Data	State	
c	$(0,W_{A1},0)$	
e	$(0,W_{A1},0)$	
f	$(0,W_{A1},0)$	
a	$(R_{A4},W_{A2},0)$	
d	$(0,W_{A2},D_{A4})$	d is removed from the DataRecord
h	$(R_{A4},W_{A3},0)$	
b	$(0,W_{A3},0)$	
g	$(0,W_{A3},0)$	
u	$(0,W_{A4},0)$	
v	$(0,W_{A4},0)$	

Continue and draw activity A5 with its data and finish with P0. As data d was already removed so no longer exists in the dictionary. A5 tries to destroy inexistent data so we have a missing data error on d. The application of rule 2 detects a lost data error on f.
To correct this error, suppose that the designer choose to cancel destroying d in A5 and corrects the error associated of f by reading f into A5 before the update. The new state of DataRecord is:

Table 5. The Last State of The DataRecord of the linear model.

Data	State
c	$(0,W_{A1},0)$
e	$(0,W_{A1},0)$
f	$(R_{A5},W_{A5},0)$
a	$(R_{A4},W_{A2},0)$
d	$(0,W_{A2},D_{A4})$
h	$(R_{A4},W_{A5},0)$
b	$(R_{A5},W_{A3},0)$
g	$(R_{A5},W_{A3},0)$
u	$(R_{A5},W_{A4},0)$
v	$(0,W_{A4},0)$

At the end, rule 4 is applied to the last copy of the DataRecord, data v, c and e are redundant.
In conclusion, with our approach we have detected the following errors: “lost data” on h and f, “missing data” on a and d and “redundant data” on v, c and e. With our approach we detect more errors than in the approach of [Rgibi *et al.* (2011)].

5.2. *Model with an XOR split*

To address the XOR we will add a new column to the DataRecord that indicates which branch changes the state of the data and the state of data after the branch.
We take the same example in section A and modify only the first gateway to an XOR. The model became:

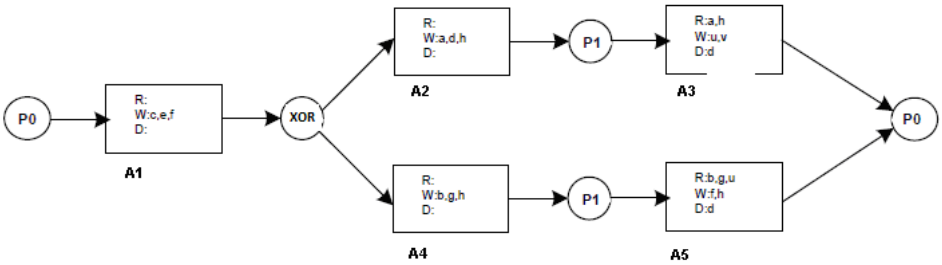

Fig. 2. Model with an XOR split.

We apply our approach to the current example as we have shown before, the last state of the DataRecord is:

Table 6. The Last State of The DataRecord of the model with an XOR split.

Data	State	State branch 1	State branch 2
c	$(0,W_{A1},0)$		
e	$(0,W_{A1},0)$		
f	$(0,W_{A1},0)$		$XOR_2\ (0,W_{A5},0)$
a	$(0,W_{A2},0)$	$XOR_1\ (R_{A3},W_{A2},0)$	
d	$(0,W_{A2},0)$	$XOR_1\ (0,W_{A2},D_{A4})$	$XOR_2\ (0,W_{A2},D_{A5})$
h		$XOR_1\ (R_{A3},W_{A2},0)$	$XOR_2\ (0,W_{A5},0)$
u		$XOR_1\ (0,W_{A3},0)$	
v		$XOR_1\ (0,W_{A3},0)$	
b			$XOR_2\ (R_{A5},W_{A4},0)$
g			$XOR_2\ (R_{A5},W_{A4},0)$

The column “state” of the DataRecord reflects the latest state of each data. In case of XOR this state is the one just before the last XOR.
Using rule 1, a missing data error is detected on u since this data was created only in the first branch of the XOR in the activity A3 and doesn’t exist in the second branch. So, this data is not available for reading in activity A5. With rule 2, lost data error is detected on f and h.

At the end, rule 4 is applied to the last copy of the DataRecord, data v, c and e are redundant.
Taking into account the treatment of XOR enabled us to detect the following errors: ”lost data” on a and h, “missing data” on v and “redundant data” on v c, e.

6. CONCLUSION

Dataflow analysis and workflow design are critical steps in business process management. In a workflow model, it’s very important to ensure consistent flow of data from one activity to another. In this paper, we propose a new approach for data flow anomalies detection based on an active help. To the best of our knowledge, this is the first paper that tries to offer an active help to the model designer in detecting and discovering data flow errors. The verification process is done as and when modeling. We used a concept of DataRecord to store data and their state each time an activity is added to the model. The verification process is based on four rules presented in this paper. An illustrating example is given to more explain the approach. Two cases are discussed; the simple model and model that contains a XOR split.
We currently continue our research in several directions. First we would like to enrich the present approach with new rules to take in count the loop processing. Second, we plan to implement this approach in a modelling tool. With this implementation, we can validate our theoretical results.

References

Awad, A.; Decker, G.; and Lohmann, N.; (2009). ‘Diagnosing and repairing data anomalies in process models’, Technical report, Potsdam,Germany.

Boushaba, S.; Kabbaj M.I.; and Bakkoury, Z.; (2013). “Towards Process mining: Matrix representation for bloc discovery”, Proceedings of the 8th International Conference on Intelligent Systems: Theories and Applications (SITA), Rabat, Morocco.

Boushaba, S.; Kabbaj M.I.; and Bakkoury, Z.; (2014). “Process discovery: Automated approach for block discovery”, Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2014), Lisbon, Portugal.

Eshuis, R.; (2006). “Symbolic Model Checking of UML Activity Diagrams”, ACM Transactions on Software Engineering Methodology 15, 2006.

Davenport, T.H. (1993). Process Innovation: reengineering work through information technology, Harvard Business School Press, Boston, MA, USA.

Lhannaoui, H.; Kabbaj M.I.; Bakkoury, Z. (2013). “Towards an approach to improve business process models using risk management techniques”, Proceedings of the 8th International Conference on Intelligent Systems: Theories and Applications (SITA), Rabat, Morocco, May.

Lhannaoui, H.; Kabbaj M.I.; Bakkoury, Z. (2014).“Analyzing risks in business process models using a deviational technique”, Proceedings of the 9th International Conference on Software Engineering and Applications (ICSOFT-EA 2014), Vienna, Austria.

Rgibi, A. E.; Yao, S.Z.; Xu, J.J.; (2011). “Dataflow Errors Detection in Business Process Model”, Applied Mechanics and Materials, 130-134, 1765.

Sadiq, S. W.; Orlowska, M. E.; Sadiq, W. ; Foulger, C.; (2004). “Data flow and validation in workflow modelling”, Australasian Database Conference (ADC 2004), Vol. 27 of CRPIT, Australian Computer Society.

Sun, S. X.; Zhao, J. L.; Sheng, O.R.; (2004). "Data flow modeling and verification in business process management", Proceedings of AMCIS New York,

Sun, S. X.; Zhao, J. L.; Nunamaker, J. F.; Liu Sheng, O. R.; (2006). "Formulating the data flow perspective for Business Process Management, Information Systems Research.

Sun, S.X.; (2007). "Dataflow analysis and workflow design in business process management", Dissertation Ph.D., The University of Arizona US.

Sundari, M. H.; Sen, A. K.; Bagchi, A.; (2007). "Detecting Data Flow Errors in Workflows: A Systematic Graph Traversal Approach", Workshop on Information Technology & Systems WITS-2007.

Trˇcka, N.; van der Aalst, W.M.P.; and Sidorova, N.; (2008). "Analyzing control-flow and data-flow in workfow processes in a unified way", Technical Report CS 08/31, Eindhoven University of Technology.

Trˇcka, N.; van der Aalst, W. M. P.; Sidorova, N.; (2009). "Data-Flow Anti-Patterns: Discovering Data-Flow Errors in Workflows", International Conference on Advanced Information Systems CAiSE 2009, Vol. 5565 of LNCS, Springer-Verlag.