Routledge
Taylor & Francis Group

TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
ISSN 2029-4913 print/ISSN 2029-4921 online

2012 Volume 18(4): 623–635
doi:10.3846/20294913.2012.740517

VERIFICATION OF BUSINESS PROCESS WORKFLOWS

Henrikas Pranevičius[1], Regina Misevičienė[2]

Department of Business Informatics, Kaunas University of Technology,
Studentų g. 56, LT-51424 Kaunas, Lithuania
E-mails: [1]henrikas.pranevicius@ktu.lt; [2]regina.miseviciene@ktu.lt (corresponding author)

Received 10 January 2012; accepted 14 April 2012

Abstract. Modeling of Business processes is essential in many areas. Workflows represent the Business processes. It is possible to identify potential problems while performing verification of workflows. One of the objectives of the verification is to assure reachability. This includes analysis of the deadlock and tempo blocking freeness properties. The paper presents verification approach based on using an adjacency matrix. Spreadsheets are used as a verification tool. The approach is illustrated by the examples which justify the importance of verification in workflow processes.

Keywords: business processes, workflows, validation, verification, graph, adjacency matrix.

Reference to this paper should be made as follows: Pranevičius, H.; Misevičienė, R. 2012. Verification of business process workflows, *Technological and Economic Development of Economy* 18(4): 623–635.

JEL Classification: C51, C52, D83, D85, H25.

1. Introduction

Business processes modeling is essential in many areas. The business processes include dozens of tasks, representing the work of company (Bisztray, Heckel 2007). A workflow is a model to represent the business process. Workflows, also known as process models, express compositions of individual tasks that assembled together account for various aspects of an overall business process (Kim *et al.* 2010).

Because these workflows can be very complex in an enterprise business processes, it is important to model the processes flows (Basu, Kumar 2002; Dzemydienė, Dzindzalieta 2010). Business processes modeling has triggered great interest in methods to define, analyze, and manage the flows. Workflow management system is often used to model and to analyze these flows. The workflow management system is a computer system that provides automated support for defining and controlling various tasks associated with business processes. The system

facilitates the everyday operation of business processes. The workflow management systems are becoming increasingly important because they are enablers of successful e-business solutions (Basu, Kumar 2002).

However, most of commercial workflow management systems do not yet provide workflow designers with formal workflow verification tools (Barkaoui *et al.* 2007; Karamanolis *et al.* 2000) or at the best these systems do some basic syntactic checks, but allow for the modeling of processes with deadlocks and other anomalies (Wynn *et al.* 2009). Workflow verification remains an open and challenging research area. There is clearly a need for analysis of the tools that take care of verification (Karamanolis *et al*. 2000).

The main goal of our paper is to extend the existing approaches. The object of the paper is verification of workflows. The described verification approach is based on using graphs and adjacency matrix. Spreadsheets are used as a verification tool. The presented approach provides a simple verification technique, which does not require sophisticated instruments, enabling the end users, who do not have workflows formalization backgrounds (such as business managers, analysts), to create valid business process models or workflows.

New approach combines benefit of graph notation for presentation of business workflows and algebraic techniques for their verification. The proposed approach is very simple, easily understandable from their visual presentation. No fancy tools beyond regular spreadsheets are required.

The approach is illustrated by examples which justify the importance of verification in workflows processes.

These statements are more fully explained and put into context in the remaining part of this paper.

2. Related Works

Verification of workflows is not a new idea. Traditionally, workflow modeling has focused on structural aspects of processes, mainly indicating the order of execution of the component activities in the process (Sadiq *et al.* 2004). The structural modeling includes methods analyzing the structure of processes and workflows during the design (or redesign) phase. Often these types of structural analyses such as validation, verification and performance analysis are mentioned (Aalst *et al.* 2002). **Validation** is known as testing of semantic completeness to ensure that the workflow behaves predictably in all scenarios. **Verification** establishes the syntactic correctness of a workflow and eliminates redundancies and deadlocks (Basu, Kumar 2002). That is why, the research of our paper focuses only on those errors (called reachability errors) like deadlocks and endless loop leading to unreachability.

While validation can be done by interactive simulation, more advanced analysis techniques are needed for verification. Fortunately, many powerful verification and validation techniques have been developed.

Many researchers have been working on workflow verification techniques (Aalst *et al.* 2002; Sadiq *et al.* 2004; Karamanolis *et al*. 2000; Basu, Kumar 2002; Barkaoui *et al.* 2007; Kim *et al.* 2010; Wynn *et al.* 2009; Dreiling *et al.* 2008; Pranevičius, Misevičienė 2008; Vasilecas

et al. 2011; Tick 2006). It is impossible to give a complete overview here. Moreover, most of the papers on business workflow verification focus on such representation like the Petri nets, Business Process Modeling Notation (BPMN), UML Activity Diagrams (UML-AD), Event-driven Process Chains (EPCs), and Business Process Execution Language (BPEL), Piece-Linear Aggregate (PLA) and Algebraic Notation.

Petri-Net is one of the most popular graphical representation possibilities of workflows. Van der Aalst (Aalst 2000; Aalst *et al.* 2002; Aalst 2007) with his more than three hundred publications worked out the whole theory and methodology of the Petri-Net based workflow management. A lot of problems are already solved with Petri-Net modeling. Other widespread modeling languages, e.g. Unified Modeling Language (UML) and Process-Graph (or P-Graph) are presented as a possible alternative solution to the already existing modeling techniques (Tick 2007; Hruby 1998; Eshuis, Wieringa 2002). However, the mentioned approaches, as a disadvantage, have the necessity of simulating the execution (Clemente *et al.* 2005).

In the papers (Vasilecas *et al.* 2011; Smaizys, Vasilecas 2009) business rules are used for presentation and verification of workflows. Linear algebraic techniques are also used to verify many properties of business workflows (Aalst *et al.* 2002). Researches in the papers (Miseviciene, Pranevicius 2008; Pranevičius, Budnikas 2008) successfully used PLA-based formalization of business rules for formal specification and verification of business processes. In (Lavbič *et al.* 2010; Gil *et al.* 2011) semantic framework and multi-agent system models are used to support business flows management. However, the techniques using interface requires considerable effort to learn, and sometimes requires programming or scripting experience from its users.

Models based on Algebraic notation are presented in the papers (Jakstonyte, Boguslauskas 2010; Misevičienė, Nikonov 2011). However, the researchers use the models for the verification of knowledge based systems or modeling in econometrics.

Researchers in the papers (Sroka *et al.* 2011; Rygg *et al.* 2008; Hihn *et al.* 2009) present the spreadsheet as the tool of workflows visualization and data analysis. They define workflows using the spreadsheet interface and analyze the results using the spreadsheet toolset. However, the researchers do not use the toolset for verification.

Our paper presents a simple verification technique. This paper highlights the following techniques:

- Graphs notation to represent the workflows;
- Algebraic models to verify properties such as accessibility;
- The spreadsheet as a verification tool.

The work presented in this paper differs from other works and introduces approach as a possible alternative solution additionally to the already existing modeling techniques. New approach combines the benefit of graph notation for presentation of business workflows and algebraic techniques for their verification. The proposed method is very simple, easily understandable from visual presentation. We do not need any fancy tools beyond the regular spreadsheet.

3. Key definitions of workflow and the elements representation

This section introduces the basic workflows terminology and notations.

3.1. Workflow graphs

A workflows graph $G=(X,\ W)$ is a simple directed graph where:

- A set $X=\{x_1,x_2,...,x_n\}$ is a finite set of nodes or vertices (further in the text "vertex"). The vertices present a finite set of tasks;
- A set $W\subseteq X\times X$ is a finite set of flows representing directed edges between two vertices. The edges show the flow of the workflows;
- A set $X^s\subseteq X$ is a finite set of start tasks, $X^g\subseteq X$ is a finite set of goal tasks.

The graph presents the following parameters (Sadiq *et al.* 2004; Aalst *et al.* 2002):

- **Environment** of the vertex $x\in X$ is a set of vertices adjacent to it and denoted by $E(x)=\{u\in X:\{x,u\}\in W\}$. Two vertices $u,v\subset X$ are adjacent if they are connected by an edge. **Inputs** $E^+(x)$ and **outputs** $E^-(x)$ for the vertex are denoted by $E(x)=E^+(x)\cup E^-(x)$.
- **Degree** of the vertex is defined as $\deg(x)=|E(x)|$. The **indegree** is denoted $\deg^-(x)$ and the **outdegree** is denoted as $\deg^+(x)$**.** The degree defines $\deg(x)=\deg^-(x)+\deg^+(x)$ ingoing and outgoing edges to / from the vertex.
- A vertex $x\in X$ with $\deg^-(x)=0,\deg^+(x)>0$ is called a **source or start**.
- Similarly, a vertex with $\deg^+(x)=0,\deg^-(x)>0$ is called a **sink or goal**.
- Other vertices $x\in X$ with degree $\deg(x)=\deg^-(x)+\deg^+(x)$ are called **internal** vertices of the graph.

Errors checked in the graph:

- **Deadlock** vertex is an internal vertex $x\in X$ when $\deg^+(x)=0,\deg^-(x)>0$ which does not belong to the goal vertices.
- **Endless loop** contains a circular sequence of vertices leading to unreachable.

Figure 1a illustrates an example of workflow graph. The example graph is defined by the sets presented below:

- The set of the vertices of the given graph is:

$$X=\{x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10},x_{11},x_{12}\}.$$

- The set of the edges of the graph is:

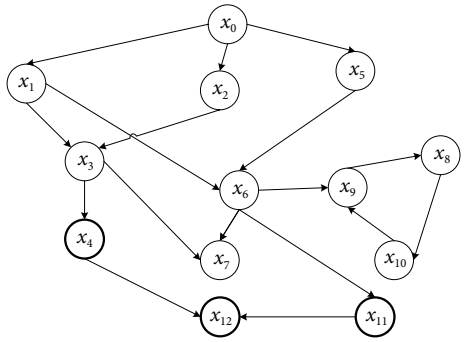
$$W=\{(x_0,x_1),(x_0,x_2),(x_0,x_5),(x_1,x_3),(x_1,x_6),(x_2,x_3),(x_3,x_4),(x_3,x_7),(x_4,x_{12}),(x_5,x_6),$$
$$(x_6,x_7),(x_6,x_9),(x_6,x_{11}),(x_8,x_{10}),(x_9,x_8),(x_{10},x_9),(x_{11},x_{12})\}$$

- **Start** vertex is: $\{x_0\}\subset X$ where $\deg^-(x_0)=0,\deg^+(x_0)>0$.
- **Goal** vertex is: $\{x_{12}\}\subset X$, where $\deg^+(x_i)=0,\deg^-(x_i)>0,\ i=12$.

Errors checked in the graph:

- **Deadlock** vertex is an internal vertex $\{x_7\}\subset X$, where $\deg^+(x_7)=0,\deg^-(x_7)>0$ and it doesn't belong to the goal vertices.
- Vertices in **endless loop** are $\{x_9,x_8,x_{10}\}\subset X$.

a) b)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
2	x_0		1	1			1							
3	x_1				1			1						
4	x_2				1									
5	x_3					1			1					
6	x_4													1
7	x_5							1						
8	x_6								1		1		1	
9	x_7													
10	x_8											1		
11	x_9									1				
12	x_{10}										1			
13	x_{11}													1
14	x_{12}													

Fig. 1. An example of the workflows graph and its Adjacency matrix

3.2. Adjacency matrix

Adjacency matrix denoted by $A = \|a_{ij}\|$, is an n by n matrix A, where

- n is the number of vertices in the graph.
- $a_{ij} = \begin{cases} 1, (x_i, x_j) \in X, \\ 0, (x_i, x_j) \notin X. \end{cases}$. The 'ones' in the $i-th$ row (and 'ones' in the column) meet the outgoing/ingoing edges from/to the vertex, correspondingly.
- **Degree** of the vertex is defined as: $\deg^+(x_i) = \sum_{j=1}^{n} a_{i,j}$, $\deg^-(x_i) = \sum_{j=1}^{n} a_{j,i}$.

For example, for the workflow graph above (Fig. 1a), Adjacency matrix $A^{(1)}$ (Fig. 1b) is made.

3.3. Verification algorithm

The presented verification algorithm solves a reachability problem. Figure 2 explains the problem. In the reachability verification all the paths must be analyzed from the start vertex (s) to the goal vertex (g). Every path p_j from the source vertex s to the goal vertex g can be decomposed into $s \xrightarrow{p_{sj}} x_j \xrightarrow{p_{jg}} g$, j = 1, 2, 3. The vertex x_3 is an internal vertex and it cannot reach a goal vertex g from the start vertex s. This leads to a deadlock.

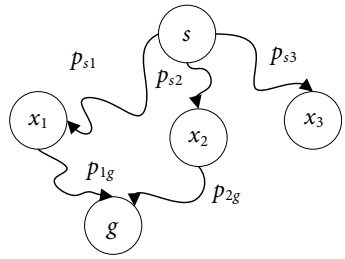


Fig. 2. The reachability illustration

The reachability problem can be solved using graph-paths finding algorithms (Cormen *et. al.* 2001). Single-source-shortest- path finding algorithms (for example, Dijkstra's and Bellman-Ford's algorithms) use weighted graphs and find the only one shortest path from the source vertex to each reachable vertex. To find all paths the algorithms can be modified running a single- source-shortest path algorithms $|X|$ times (there X is a set of the graph vertices), once for each vertex as the source (for example, Floyd-Warshall's algorithm). The modified algorithms use recursive solution to find all paths. The algorithms compute all-pairs-shortest-paths in bottom-up style. All the mentioned algorithms solve the problem on the weighted graphs. Another algorithm (for example, breadth-first search) is used for the graphs without weights. It also computes the only one shortest distance (smallest number of edges) from s (source vertex) to each reachable vertex.

In this paper presented modified algorithm uses a breadth-first search for the graphs without weights. The algorithm uses recursive solution of breadth-first search to each vertex and computes the shortest-distance in bottom-up style. Unlike the above-mentioned algorithms, the modified algorithm not only finds the shortest distance from the source vertex to each reachable vertex but also compounds hierarchical structure where all vertices are presented in different levels. Only spreadsheet is used as an analysis tool. Analysis of literature also showed that the articles did not apply the graph search algorithms for the verification of business process flows.

The verification algorithm is based on the analysis of the reachable vertices from **goal** vertex. Verification begins from **goal** vertex $x_g \in X$ in a graph. Reachable vertices from **goal** vertex are assigned to **verification set**. Initially the **verification set** is $V = \{\ \}$.

Steps of the verification algorithm:

1. Set $k = 0$. The Adjacency matrix $A^{(k)}$ of the workflow graph is drawn.
2. Assign **goal** vertex to $V^k = \{x_g\}$. Delete rows from Adjacency matrix of vertices from V^k set.
3. In the Adjacency matrix $A^{(k)}$ mark the columns of vertices from V^k set. In the marked columns find in "ones" (1) and mark the rows. Assign to verification set V^{k+1} the vertices of the marked rows. This will be vertices reachable from **goal** vertex. The shortest path is equal to $k+1$. Delete the marked columns from the matrix of set V^k. Delete rows from the matrix of set V^{k+1}.
4. Set $k = k+1$. A new matrix is established without the deleted columns and rows.
5. Repeat steps 3–4 until the entire matrix will be reviewed.

Analysis steps of Adjacency matrix are presented in accordance with the given verification algorithm in Figures 3–4.

Verification starts from **goal** vertex x_{12} (Fig. 3a). Assign $V^0 = \{x_{12}\}$. Row x_{12} is deleted. Column x_{12} is marked. There are "ones" (1) in the rows x_4, x_{11} of the column x_{12}. So, vertices x_4, x_{11} can reach **goal** vertex at the shortest path (the shortest distance from the source vertex) of length 1. Assign $V^{(1)} = \{x_4, x_{11}\}$. These rows x_4, x_{11} must be removed from the matrix. The column from the matrix of set $V^{(0)} = \{x_{12}\}$ is also deleted.

A new matrix $A^{(1)}$ is formed after eliminating the vertices (Fig. 3b). There are "ones" (1) in the rows x_3, x_6 of columns x_4, x_{11}. So, these vertices can reach **goal** vertex at shortest path of length 2. Assign $V^{(2)} = \{x_3, x_6\}$. These rows must be removed from the matrix. The columns from the matrix of set $V^{(1)} = \{x_4, x_{11}\}$ are eliminated as well.

a)

$A^{(0)} =$

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
x_0		1	1			1							
x_1				1			1						
x_2				1									
x_3					1			1					
x_4													1
x_5						1							
x_6							1		1			1	
x_7													
x_8											1		
x_9									1				
x_{10}										1			
x_{11}													1

$k = 0;\ V^0 = \{x_{12}\}$

b)

$A^{(1)} =$

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_0		1	1			1						
x_1				1			1					
x_2				1								
x_3					1			1				
x_5							1					
x_6								1		1		1
x_7												
x_8											1	
x_9									1			
x_{10}										1		

$k = 1;\ V^1 = \{x_4, x_{11}\}$

Fig. 3. Verification steps of the Adjacency matrix

A new matrix $A^{(2)}$ is formed (Fig. 4a) after eliminating the vertices. There are "ones" (1) in the rows x_1, x_2, x_5 of the columns x_3, x_6. So these vertices can reach **goal** vertex at the shortest path of length 3. $V^{(3)} = \{x_1, x_2, x_5\}$. These rows are removed from the matrix. The columns from the matrix of the set $V^{(2)} = \{x_3, x_6\}$ are also eliminated.

After eliminating the vertices a new matrix $A^{(3)}$ is formed (Fig. 4b). There are "ones" (1) in the row x_0. So, these vertices can reach **goal** vertex at shortest path of length 4. $V^{(4)} = \{x_0\}$. This row x_0 is removed from the matrix. The columns from the matrix of set $V^{(3)} = \{x_1, x_2, x_5\}$ are eliminated as well.

After eliminating the vertices a new matrix $A^{(4)}$ is formed (Fig. 4c). There are no "ones" (1) in the column x_0. The verification shows that vertices x_7, x_8, x_9, x_{10} cannot reach the **goal** vertex.

a)

$A^{(2)} =$

	x_0	x_1	x_2	x_3	x_5	x_6	x_7	x_8	x_9	x_{10}
x_0		1	1		1					
x_1				1		1				
x_2				1						
x_5						1				
x_7										
x_8										1
x_9								1		
x_{10}									1	

$k = 2;\ V^2 = \{x_3, x_6\}$

b)

$A^{(3)} =$

	x_0	x_1	x_2	x_5	x_7	x_8	x_9	x_{10}
x_0		1	1	1				
x_7								
x_8								1
x_9						1		
x_{10}							1	

$k = 3;\ V^3 = \{x_1, x_2, x_5\}$

c)

$A^{(4)} =$

	x_0	x_7	x_8	x_9	x_{10}
x_7					
x_8					1
x_9				1	
x_{10}				1	

$k = 4;\ V^4 = \{x_0\}$

Fig. 4. Verification steps of the Adjacency matrix (continuation)

After verification the graph is constructed (Fig. 5).

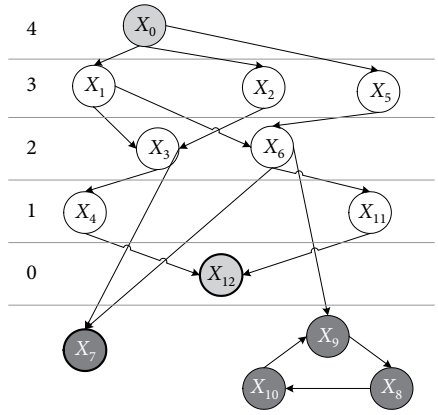


Fig. 5. Reconstructed graph after verification

Explanation of the graph verification:

- Only vertex $x_0 \in X$ of the graph represents **source** data.
- The only vertex $x_{12} \in X$ represents the **goal** data.
- Internal vertex $x_7 \in X$ presents **deadlock**, which doesn't belong to the goal vertices.
- A set of vertices $x_8; x_9; x_{10}$ present the **endless loop**, which contains a circular sequence of vertices leading to unreachable.

4. Workflows verification of tax inspection calculations

To illustrate the given approach an example of tax inspection workflows in Lithuania is employed in the paper (Misevičienė, Nikonov 2011). **Tax inspection** means an inspection conducted by the tax administrator in respect of the taxpayer to control the taxpayer's compliance with the requirements prescribed by tax laws in the fields of calculation, declaration and payment of taxes and, in the cases prescribed by law, in other fields as well. The creation and the evaluation of efficient tax inspection workflows is one of the most problems for today.

The purpose of this investigation is to finding out, if the used data and performed calculations meet the logical sequence. Figure 6 presents a form for tax inspection workflows that are conducted by the tax administrator to control the taxpayer. Table 1 clarifies the meaning of the fields of calculations in the tax inspection form.

The inspection of the Personal income tax

Tax period	Taxpayer data		Tax administrator data (Tax inspection)		Additionally calculated tax for period (5)–(3)	Additionally calculated tax total Σ(6)	Personal income tax overpayment (+) anears (–)		Calculation of late payment interest			
	Tax base (Person income)	Tax sum	Tax base (Person income)	Tax sum			Date	Sum	Anears for calculation of late payment interest	Number of days	Late payment interest rate, per cent	Late payment interest sum
Equivalent to graph vertex	–	X_1	–	X_2	X_3	X_4	X_5	X_6	X_7	$X_{9(i+1)}$–$X_{9(i)}$	X_8	X_{10}, X_{11}
1	2	3	4	5	6	7	8	9	10	11	12	13

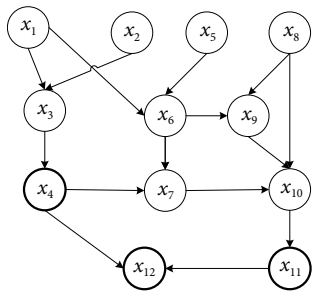
Fig. 6. The tax inspection calculations

Table 1. Explanations of the tax inspection calculations

Vertex	Meaning (equivalent)	Formula
x_1	Sum of tax calculated and declared by a taxpayer	Initial data
x_2	Sum of tax calculated by a tax administrator during tax inspection	Initial data
x_3	Additionally calculated sum of tax during tax period (for example, month) – established as the difference between sum of tax calculated by a taxpayer and sum of tax determined by a tax administrator	$x_3 = x_2 - x_1$
x_4	Additionally calculated sum of tax total (during the whole inspection period) – established as a cumulative sum of tax	$x_4 = \sum x_3$
x_5	The taxpayer tax payments data (dates, sums) – when the taxpayer pays certain tax and sum of payment	Initial data
x_6	The taxpayer tax overpayment (+) / arrears (–) (by periods) – established by comparing tax declaration and tax payment data	Being calculated by the Tax accounting system of the State tax inspectorate (Tax administrator)
x_7	Arrears for calculation of late payment interest. The late payment interest of certain tax is calculated taking into account overpayments of other taxes in accordance with the Law on Tax administration.	$x_7 = \begin{cases} x_4, \text{ if } x_6 < 0 \\ x_4 - x_6, \text{ if } x_6 > 0 \text{ and } x_6 < x_4 \\ 0, \text{ if } x_6 > 0 \text{ and } x_6 > x_4 \end{cases}$
x_8	Late payment interest rate (per cent per day) – established by the Minister of Finance of the Republic of Lithuania for each quarter	Initial data
x_9	Late payment interest calculation periods – when the total additionally calculated sum of tax (X_4) and taxpayer tax overpayment / arrears (X_6) are constant	Periods, when x_4 and x_6 are constant
x_{10}	Calculated late payment interest during period mentioned above (X_9)	$x_{10} = x_7 \times x_8 \times (x_{9(i+1)} - x_{9(i)})$
x_{11}	Calculated late payment interest total (during the whole inspection period) – cumulative sum of the late payment interest	$x_{11} = \sum x_{10}$
x_{12}	Total result of the inspection – additionally calculated sum of taxes and late payment interest	$x_{12} = x_4 + x_{11}$

The graph and its Adjacency matrix are shown in Figure 7. Verification steps of tax inspection workflows by the given verification algorithm are presented in Figures 8–9.

a)



b)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12
2	x1			1			1						
3	x2			1									
4	x3				1								
5	x4							1					1
6	x5						1						
7	x6							1		1			
8	x7										1		
9	x8									1	1		
10	x9										1		
11	x10											1	
12	x11												1
13	x12												

Fig. 7. Graph model and Adjacency matrix of the tax inspection workflows

a)

$A^{(0)}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
x_1			1			1						
x_2			1									
x_3				1								
x_4							1					1
x_5						1						
x_6							1		1			
x_7										1		
x_8									1	1		
x_9										1		
x_{10}											1	
x_{11}												1

$k = 0;\ V^0 = \{x_{12}\}$

b)

$A^{(1)}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_1			1			1					
x_2			1								
x_3				1							
x_5						1					
x_6							1		1		
x_7										1	
x_8									1	1	
x_9										1	
x_{10}											1

$k = 1;\ V^1 = \{x_4, x_{11}\}$

Fig. 8. Verification of tax inspection workflows

c)

$A^{(2)}$	x_1	x_2	x_3	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1			1		1				
x_2			1						
x_5					1				
x_6						1		1	
x_7									1
x_8								1	1
x_9									1

$k = 2;\ V^2 = \{x_3, x_{10}\}$

d)

$A^{(3)}$	x_1	x_2	x_5	x_6	x_7	x_8	x_9
x_5				1			
x_6					1		1

$k = 3;\ V^3 = \{x_1, x_2, x_7, x_8, x_9\}$

e)

$A^{(4)}$	x_5	x_6
x_6		1

$k = 4;\ V^4 = \{x_6\}$

f)

$A^{(5)}$	x_5
x_5	

$k = 5;\ V^5 = \{x_5\}$

Fig. 9. Verification of tax inspection workflows (continuation)

After verification the arranged graph is constructed (Fig. 10).

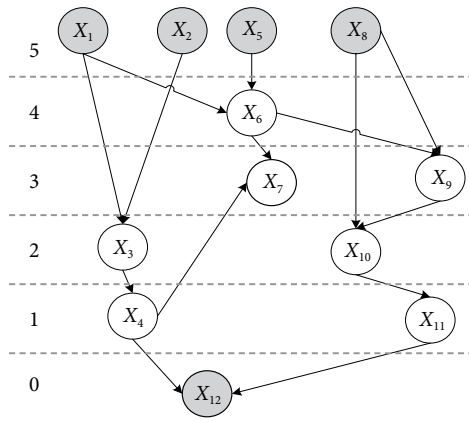


Fig. 10. Reconstructed graph of the tax inspection workflows

Explanation of the tax inspection workflows analysis:

- Vertices $x_1, x_2, x_5, x_8 \in X$ of the graph represent **source** data.
- The only vertex $x_{12} \in X$ represents the **goal** data.
- There are no loops in the graph. Its final vertex $x_{12} \in X$ is accessible from every **initial** vertex $x_1, x_2, x_5, x_8 \in X$.
- It is possible to make a conclusion, that calculations meet the logical sequence.

5. Conclusions and Future Work

The paper presents approach as a possible alternative solution next to the already existing workflows verification approach. This approach includes analysis of the deadlock and tempo blocking freeness properties. The proposed verification approach is based on using graphs and adjacency matrix. The spreadsheets are used as a verification tool. The approach is very simple, easily understandable from visual presentation.

The presented approach provides a simple verification technique, which does not require sophisticated instruments, enabling the end users, who do not have workflows formalization backgrounds (such as business managers, analysts), to create valid business process models or workflows.

The approach is illustrated by examples which justify the importance of verification in workflows processes. One of the examples is the Lithuanian tax inspection model. The created model of tax inspection system in Lithuania shows the possibilities of given approach application during the verification of tax inspection system. The results received with the help of the proposed approach showed that the calculations meet the logical sequence.

The future of the research is extending the approach. We intend to automate the process for fixing detected errors.

References

Aalst, W. M. P. 2000. Workflow verification: finding control-flow errors using petri-net based techniques, *Business Process Management* 1806: 161–183. http://dx.doi.org/10.1007/3-540-45594-9_11

Aalst, W. M. P.; Hirnschall, A.; Verbeek, H. M. W. 2002. An alternative way to analyze workflow graphs, in *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, 27–31 May, 2002, Toronto, Ontario, Canada, Berlin: Springer-Verlag, 2348: 535–552.

Aalst, W. M. P. 2007. Trends in business process analysis: from verification to process mining, in *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS 2007)*, 12–16 June, 2007, Funchal, Madeira, Portugal, 12–22.

Barkaoui, K.; Ayed, R. B.; Sbai, Z. 2007. Workflow soundness verification based on structure theory of petri nets, *International Journal of Computing & Information Sciences* 5(1): 51–61.

Basu, A.; Kumar, A. 2002. Research commentary: workflow, *Management Issues in e-Business Information Systems Research* 13(1): 1–14. http://dx.doi.org/10.1287/isre.13.1.1.94

Bisztray, D.; Heckel, R. 2007. Rule-level verification of business process transformations using CSP, in *Proceedings of the Sixth International Workshop on Graph Transformation and Visual Modeling Techniques* 6: 1–14 [online], [cited 10 January 2012]. Available from Internet: http://www.easst.org/eceasst/

Clemente, J.; Antonio, A.; Ramirez, J. 2005. Crib: a method for integrity constraint checking on knowledge bases, *Computation Systems* 8(4): 265–280.

Cormen, T.; Leiserson, Ch.; Rivest, R.; Stein, C. 2001. *Introduction to algorithms*. Massachusetts, USA: The Massachussetts Institute of technology. 1180 p.

Dzemydienė, D.; Dzindzalieta, R. 2010. Development of architecture of embedded decision support systems for risk evaluation of transportation of dangerous goods, *Technological and Economic Development of Economy* 16(4): 654–671. http://dx.doi.org/10.3846/tede.2010.40

Dreiling, A.; Rosemann, M.; Aalst, W. M. P.; Sadiq, W. 2008. From conceptual process models to running systems: a holistic approach for the configuration of enterprise system processes, *Decision Support Systems* 45(2): 189–207. http://dx.doi.org/10.1016/j.dss.2007.02.007

Eshuis, R.; Wieringa, R. 2002.Verification support for workflow design with uml activity graphs, in *Proc. International Conference on Software Engineering (ICSE 2002)*, May 19-25, Orlando, Florida, USA: ACM Press, 166-176. http://dx.doi.org/10.1145/581360.581362

Jakstonyte, G.; Boguslauskas, V. 2010. Graphic model regulating the application of land site taxation deductions, *Inzinerine Ekonomika – Engineering Economics* 21(3): 238–243.

Gil, Y.; González-Calero, P.; Kim, J.; Moody, J.; Ratnakar, V. 2011. A semantic framework for automatic generation of computational workflows using distributed data and component catalogues, *Journal of Experimental & Theoretical Artificial Intelligence* 23(4): 389–467. http://dx.doi.org/10.1080/0952813X.2010.490962

Hihn, J.; Lewicki, S.; Wilkinson, B. 2009. How spreadsheets get us to mars and beyond. System sciences, in *42nd Hawaii International International Conference on Systems Science (HICSS-42 2009), Proceedings (CD-ROM and online),* 5-8 January 2009, Waikoloa, Big Island, HI, USA: IEEE Computer Society, 1–9.

Hruby, P. 1998. Structuring specification of business systems with UML (withan emphasis on workflow management systems). OOPSLA 98 Business Object Workshop IV, [online], [cited 10 January 2012]. Available from Internet: http://jeffsutherland.org/oopsla98/pavel.html.

Karamanolis, C.; Giannakopoulou, D.; Magee, J.; Wheater, S. M. 2000. Model checking of workflow schemas, in *Enterprise distributed object computing conference* [online], [cited 10 January 2012]. Available from Internet: http://0-ti.arc.nasa.gov.iii-server.ualr.edu/m/profile/dimitra/publications/edoc00.pdf

Kim, J.; Gil, Y.; Spraragen, M. 2010. Principles for interactive acquisition and validation of workflows, *Journal of Experimental and Theoretical Artificial Intelligence* 22(2): 103–134. http://dx.doi.org/10.1080/09528130902823698

Lavbič, D.; Vasilecas, O.; Rupnikc, R. 2010. Ontology-based multi-agent system to support business users and management, *Technological and Economic Development of Economy* 16(2): 327–347. http://dx.doi.org/10.3846/tede.2010.21

Miseviciene, R.; Pranevicius, H. 2008. Rule based business process modeling using PLA, in *Information Technologies' 2008 : research communications of the 14th International Conference on Information and Software Technologies, IT 2008,* Kaunas, Lithuania, 24–25 April, 2008, Kaunas: Technologija, 108–115.

Misevičienė, R.; Nikonov, J. 2011. Validation of tax inspection model, in Butleris, R.; Butkiene, R. (Eds.). *Information Technologies' 2011: proceedings of the 17th international conference on Information and Software Technologies, IT 2011,* 27–29 April, 2011, Kaunas, Lithuania, Kaunas University of Technology. Kaunas: Technologija, 61–68.

Pranevičius, H.; Budnikas, G. 2008. PLA-based formalization of business rules and their analysis by means of knowledge-based techniques, *Technological and Economic Development of Economy* 14(3): 328–343.

Pranevičius, H.; Misevičienė, R. 2008. Verification of business rules using logic programming means, in *Proceedings of the International Conference Modelling of Business Industrial and Transport Systems*, 7–10 May, 2008, Riga, Latvia, Riga: Transport and Telecommunication Institute, 99–106.

Rygg, A.; Roe, P.; Sumitomo, J. 2008. GPFlow: an intuitive environment for web-based scientific workflow, in *First International Workshop on Workflow Systems in Grid Environments (WSGE2006)*, 23 October, 2006, Changsha, China, Concurrency and Computation: Practice and Experience, *WILEY* 20(4): 393–408.

Sadiq, S.; Orlowska, M.; Sadiq, W.; Foulger, C. 2004. Data flow and validation in workflow modelling, in *ADC'2004 Dunedin, New Zealand,* 18–22 January 2004, *Conferences in Research and Practice in Information Technology* (27): 1–8.

Sroka, J.; Krupa, Ł.; Kierzek, A.; Tyszkiewicz, J. 2011. CalcTav – integration of a spreadsheet and taverna workbench, *Bioinformatics* [online], [cited 10 January 2012]. Available from Internet: http://bioinformatics.oxfordjournals.org/content/early/2011/07/19/

Tick, J. 2006. Workflow model representation concepts, in *Proceedings of 7th International Symposium of Hungarian Researchers on Computational Intelligence, HUCI,* 329–337.

Tick, J. 2007. Workflow Modelling Based on Process Graph, in *5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics*, 25–26.

Vasilecas, O.; Dubauskaitė, R.; Rupnik, R. 2011. Consistency checking of UML business model, *Technological and Economic Development of Economy* 17(1): 133–150. http://dx.doi.org/10.3846/13928619.2011.554029

Wynn, M. T.; Verbeek, H. M. W.; Aalst, W. M. P.; Hofstede, A. H. M.; Edmond, D. 2009. Business process verification: finally a reality!, *Business Process Management Journal* 15(1): 74–92. http://dx.doi.org/10.1108/14637150910931479

Smaizys, A.; Vasilecas, O. 2009. Business rules based agile ERP systems development, *Informatica* 20(3): 439–460.

Henrikas PRANEVIČIUS. Prof., Dr Habil, is full time Professor in the Business Informatics Department at the Kaunas University of Technology. He is a scientific leader of the research group "Formal Specification, Verification and Simulation of Distributed Systems". The research group has been working in the field of creation of formal description methods for systems with distributed information processing and for modelling complex systems for a long time. The results of investigations have been successfully applied creating the computerised system for specification, validation, and simulation/modelling of computer network protocols. H. Pranevičius has published over 200 scientific papers and 4 monographs.

Regina MISEVIČIENĖ. Dr, Associate Professor in the Business Informatics Department at the Kaunas University of Technology. She is a member of Prof. H. Pranevičius' research group. Her interests include creation of formal specifications using logic programming techniques, validation, and verification of business processes and knowledge bases. Regina Misevičienė is the author of about 40 scientific papers and 1 monograph on the topic of her research.