A Mining Algorithm for Extracting Decision Process Data Models

Cristina-Claudia DOLEAN, Razvan PETRUŞEL
Business Information Systems Department,
Faculty of Economics and Business Administration
Babeş-Bolyai University, Cluj-Napoca, Romania
{cristina.dolean, razvan.petrusel}@econ.ubbcluj.ro

The paper introduces an algorithm that mines logs of user interaction with simulation software. It outputs a model that explicitly shows the data perspective of the decision process, namely the Decision Data Model (DDM). In the first part of the paper we focus on how the DDM is extracted by our mining algorithm. We introduce it as pseudo-code and, then, provide explanations and examples of how it actually works. In the second part of the paper, we use a series of small case studies to prove the robustness of the mining algorithm and how it deals with the most common patterns we found in real logs.
Keywords: *Decision Process Data Model, Decision Process Mining, Decision Mining Algorithm*

1 Introduction

Decision making is an activity performed on daily basis. There are a lot of different decision making strategies that are used by people without being aware of them. Therefore, most of daily decisions are the result of an empirical process, based rather on intuition then consciously planned and scientifically based. Even more than in regular life, making business decisions without a sound decision process is dangerous and potentially catastrophic. We look at business decision making as a process composed of a number of actions that can be performed in a sequence and/or in parallel. We argue that the quality of the overall decision is linked to the decision actions and their correct ordering. Therefore, in this paper, decision making is defined as the result of a workflow of mental actions performed by the decision maker. We propose a new way of researching business decisions that aims to: explicitly show the mental activities performed; their sequence; and the relationships between them.

In the well-established research literature of decision making, several researchers proposed decompositions of the decision process in various phases. The most common and well known are the decision making phases proposed by Simon in the 60s. The first phase is a) the research of the decisional context, then b) a number of alternatives are created, then c) the choice of one alternative is made and, finally, d) the chosen decision alternative is implemented [1]. For example, when going to a restaurant one has to decide what to order. The context of the decision is represented by the restaurant type, the companions, how hungry one is, etc. The decision alternatives are then created according to the context (usually, the restaurant menu is the starting point from which a short list of dishes is selected). The decision criterions are then used to choose one of the dishes. Criterions can be quantitative (e.g. how much a dish costs) or qualitative (previous experience with a certain taste or recommendations from the waiter, etc.). In the short example above, the decision phases are easily identified. But what about the actual decision actions and their sequence? For example, is it better to ask the waiter for recommendations and then read the menu or the other way around? And how can we compare the decision process of one person with the decision process of another person? These are some questions that can be answered by extracting a small-grained model of each individual decision process.

This paper aims to introduce an algorithm which automatically extracts a model from the decision activities performed by a decision maker. Specifically, the contribution presented in this paper relates to the Mining Tool (see Fig. 1) that converts a Decision Log into a Decision Data Model (DDM). We aim to prove the robustness and the quality of the results produced using the Mining Tool. Therefore, we focus on explaining how the DDM is extracted from the logs, and on proving that it produces the correct result, no matter what actions are stored in the logs.

The second section explains the notions related to the concepts of Product Data Models (PDM) and Decision Data Models which is essential for understanding the mining algorithm. The third section starts by describing our mining approach starting with the software which allows us to store the actions made by the user. In the second part of the third section we show details on how the records stored in the decision log are converted by the Mining Tool into o DDM-XML file. The evaluation of the results produced by the algorithm in several case studies is introduced in the fourth section of this paper.

2 Related work

It is more and more clear that an open project with a lot of contributors can achieve similar or better result than a closed project with a few contributors. One can think of open-source software compared to licensed software [e.g. Android and iOS operating systems] or to Wikipedia papers compared to other sources of knowledge (e.g. Encyclopedia Britannica [2]). We argue that, similarly, an aggregated decision model extracted from hundreds or thousands of decision makers can be as valuable as one created by just a couple of experts in a field. The advantages of a collective model are that it: can be easier to extract automatically and can be obtained cheaply.

The result of process mining is a model that reflects a real life process in an enterprise [3]. In the same way we argue that, through decision mining, we can extract a model of the real decision process performed by a decision maker.

We use the notions of log, trace and activity as in process mining. The starting point for process mining is the „event log". Each event log is composed of traces [4]. A trace is an iteration through the process. Each trace is composed of activities. An activity is an atomic action performed by one user and is always associated with the timestamp of its occurrence. Therefore, one trace is an ordered sequence of activities. We extract the traces by the interaction of the decision maker with the "decision-aware software". The decision maker needs to look at the data elements, compare them, calculate new data elements and, based on those activities, make a certain decision. All the actions performed by all users are stored as multiple traces of the process.

Since we look at the decision process like at a workflow, the control-flow discovery algorithms (e.g. Alpha, Heuristics, Fuzzy, Genetic) created for process mining [5], [6], [7], [8] were the starting point of our research. However, there are some unique features of decision making processes that render the process mining algorithms useless. One example of such a feature is the fact that, in business processes, some of the traces show up with an increased frequency (for example, the process of issuing an invoice is performed most of the time following the same sequence of actions). In the real life decision processes we researched so far, even for simple processes we did not find exactly the same trace twice. Therefore, we cannot efficiently use Alpha, Heuristics and Genetic mining algorithms because they rely at some point on calculating the frequency with which a certain activity (or set of activities) occurs in the traces and the sequence of activities.

Vanderfeesten [9] defined the notion of Product Data Model (PDM). This is an acyclic hyper-graph structure similar to a Bill of Materials (BoM). But, while a BoM represents a physical product, the PDM represents an informational product. The role of the PDM is to describe the structure of the process based on the input data provided by the user. An extension of the PDM (the

Decision Data Model - DDM) was depicted in [10] and will be elaborated on in Section 3 of this paper. We rely on this formalism to build the graphical depiction of the decision process. We found it better suited for the researched issues in terms of semantics and flexibility than other workflow models (e.g. Petri Nets, BPMN, UML Activity Diagram).
A PDM can be manually generated based on interviews, questionnaires or by using 'report while doing' approaches. Those knowledge acquisition methods are well established and need no further explaining. But one must note that they are quite expensive and creating a model this way requires a lot of time. Because of this and because so far we are unaware of an automated method to generate a decision model from certain mental activities performed by the user, we propose an algorithm which automatically extracts that kind of a model.

3 Theoretical approach

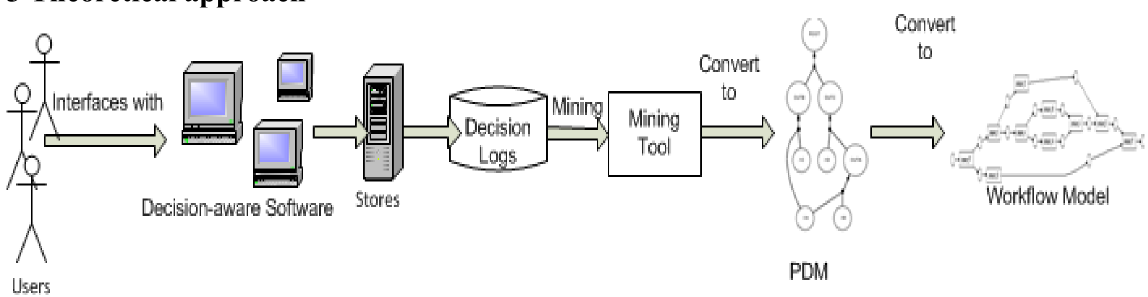
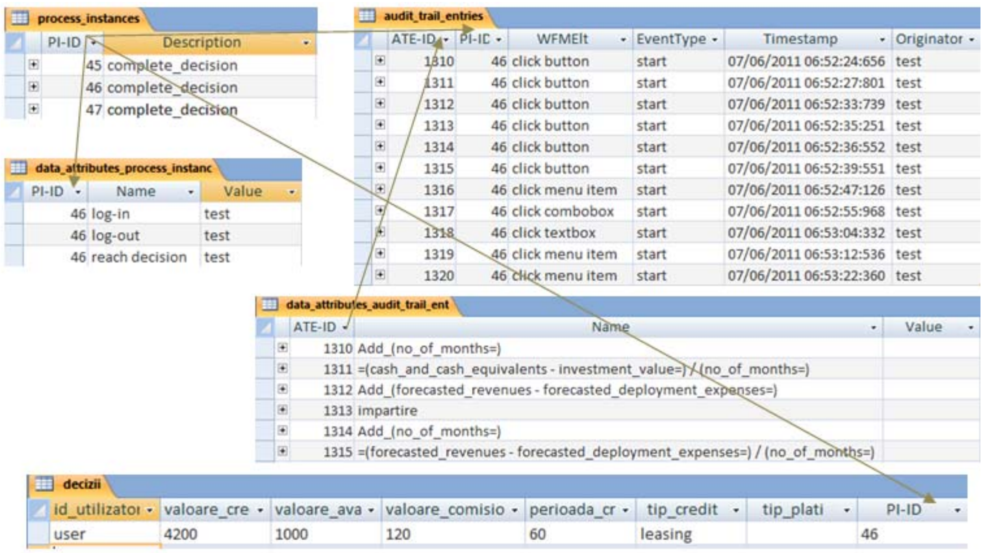


Fig. 1. Overall approach over decision process mining

First, we will briefly introduce the complete approach over decision process mining (see Fig. 1). All starts with a large number of users that interact with the decision-aware system. Such a system is software that provides: all necessary data regarding a certain decision scenario; and some tools for data derivation (e.g. a calculator for performing addition, multiplication, subtraction and division). The user needs to make a decision based on the available data and on the one that he derives. All the actions performed while making the decision are logged by the system. The exact mechanism employed for logging is presented in [11] and it is limited, so far, to recording the used data and the data derivations performed by the decision maker. As shown in Fig. 1, the log generated by the user interaction with the software is stored in a database. This database consists of five tables. In Fig. 2**Error! Reference source not found.** we introduce those tables along with a small data sample in order to provide the reader with a better understanding of the logged data. Then, by using the Mining Tool, described further in this paper, the logged data is converted into a Product (Decision) Data Model (some basic details are provided in the second part of Section 4) which can be easily converted further into a workflow model.
This section is continued with an introduction to the minimal knowledge necessary for understanding the Mining Tool. It presents the basics of user activity logging and the PDM/DDM formalism. This is essential knowledge for understanding how logged data is mined and transformed into the output data that can be converted into a PDM/DDM graphical representation.

3.1 Logging user interaction and storing it as decision logs

We refer to a decision process instance (one usage session of the software by one decision maker) as a trace. The actions performed by the user in the process of making a decision, given a certain scenario (actions belonging to a trace), are stored in the tables showed in Figure 2. The database contains five tables: Process_Instances, Audit_Trail_Entries, Data_Attributes_Audit_Trail_Entries, Data_Attributes_Process_Instances and Decisions. Each process has a unique id which is stored in Process_Instance table. At every change of the PI-ID we are dealing with another decision process instance. The user is aware of this unique id of his decision process because it is displayed at the end of the process, when he is inputting the chosen decision alternative.



process_instances

PI-ID	Description
45	complete_decision
46	complete_decision
47	complete_decision

audit_trail_entries

ATE-ID	PI-ID	WFMElt	EventType	Timestamp	Originator
1310	46	click button	start	07/06/2011 06:52:24:656	test
1311	46	click button	start	07/06/2011 06:52:27:801	test
1312	46	click button	start	07/06/2011 06:52:33:739	test
1313	46	click button	start	07/06/2011 06:52:35:251	test
1314	46	click button	start	07/06/2011 06:52:36:552	test
1315	46	click button	start	07/06/2011 06:52:39:551	test
1316	46	click menu item	start	07/06/2011 06:52:47:126	test
1317	46	click combobox	start	07/06/2011 06:52:55:968	test
1318	46	click textbox	start	07/06/2011 06:53:04:332	test
1319	46	click menu item	start	07/06/2011 06:53:12:536	test
1320	46	click menu item	start	07/06/2011 06:53:22:360	test

data_attributes_process_instanc

PI-ID	Name	Value
46	log-in	test
46	log-out	test
46	reach decision	test

data_attributes_audit_trail_ent

ATE-ID	Name	Value
1310	Add_(no_of_months=)	
1311	=(cash_and_cash_equivalents - investment_value=) / (no_of_months=)	
1312	Add_(forecasted_revenues - forecasted_deployment_expenses=)	
1313	impartire	
1314	Add_(no_of_months=)	
1315	=(forecasted_revenues - forecasted_deployment_expenses=) / (no_of_months=)	

decizii

id_utilizatoi	valoare_cre	valoare_ava	valoare_comisio	perioada_cr	tip_credit	tip_plati	PI-ID
user	4200	1000	120	60	leasing		46

Fig. 2. Logging tables

We argued that the mental decision process is exhibited to the outside as user interaction with the decision-aware software. This interaction is actually expressed by looking at certain data items and clicking some of the controls in the application's forms and additional tools (as the calculator). The clicks of the user are stored in Data_Attributes_Process_Instances table. The Audit_Trail_Entries table stores in the WFMElt (WorkFlow Model Element) field the action performed by the user (for example: click button, click menu item, click textbox etc.) and the timestamp of each action. The name of the textbox, buttons or menus used by the decision maker are stored in Data_Attributes_Audit_Trail_Entries (in Name field) along with the actual value displayed in the field. In case of a derived data item, Name field stores the names of all the data items that are aggregated in order to calculate the value of the derived item.

The tables in Fig. 2are then queried so that relevant data is extracted. For example, for the trace 46 the decision maker reached a final decision (in Data_Attributes_Process_Instances table we see that the user has logged-in, inputted and saved a decision and then logged-out). Some of the actions performed during his decision making process was to calculate a derived data item. For example, for this particular trace (PI-ID 46 in table Audit_Trail_Entries), the ATE-ID 1311 stores an action of clicking a button which is linked to the ATE-ID 1311 in Data_Attributes_Audit_Trail_Entries table where we can see, in the Name field, the data items used for deriving and, in Value field, the result of the calculation. For an extended insight into the user interface we refer the reader to Fig. 6 and for more details on how the logging is actually performed to [12]

For a condensed view over the logged data, a query extracting the most important fields is created. For a better understanding, we will use the data of trace 46 as a running example throughout this paper (see Fig. 3).

```
PI-ID | Description | Value | Timestamp | ATE-ID | WFMElt | Name | Value |
46 | complete_financing_decision | test | 07/06/2011 06:51:00:183 | 1294 | click menu item | evaluare_investitie | evaluare_investitie
46 | complete_financing_decision | test | 07/06/2011 06:51:04:794 | 1295 | click textbox | investment_value | 100000
46 | complete_financing_decision | test | 07/06/2011 06:51:11:26 | 1296 | click buton | minus | -
46 | complete_financing_decision | test | 07/06/2011 06:51:13:460 | 1297 | click buton | Add_investment_value | investment_value
46 | complete_financing_decision | test | 07/06/2011 06:51:16:617 | 1298 | click buton | =cash_and_cash_equivalents - investment_value | -41375
46 | complete_financing_decision | test | 07/06/2011 06:51:25:761 | 1299 | click textbox | forecasted_revenues | 310000
46 | complete_financing_decision | test | 07/06/2011 06:51:32:9 | 1300 | click textbox | forecasted_deployment_expenses | 4500
46 | complete_financing_decision | test | 07/06/2011 06:51:33:290 | 1301 | click buton | Add_forecasted_revenues | forecasted_revenues
46 | complete_financing_decision | test | 07/06/2011 06:51:36:988 | 1302 | click buton | minus | -
46 | complete_financing_decision | test | 07/06/2011 06:51:39:97 | 1303 | click buton | Add_forecasted_deployment_expenses | forecasted_deployment_expenses
46 | complete_financing_decision | test | 07/06/2011 06:51:40:895 | 1304 | click buton | =forecasted_revenues - forecasted_deployment_expenses | 305500
46 | complete_financing_decision | test | 07/06/2011 06:51:55:124 | 1305 | click textbox | no_of_months | 12
46 | complete_financing_decision | test | 07/06/2011 06:51:55:153 | 1306 | click buton | Add_no_of_months | no_of_months
46 | complete_financing_decision | test | 07/06/2011 06:52:13:22 | 1307 | click buton | =no_of_months | 12
46 | complete_financing_decision | test | 07/06/2011 06:52:20:264 | 1308 | click buton | Add_(cash_and_cash_equivalents - investment_value=) | (cash_and_cash_equivalents - investment_value=)
46 | complete_financing_decision | test | 07/06/2011 06:52:22:209 | 1309 | click buton | impartire | /
46 | complete_financing_decision | test | 07/06/2011 06:52:24:656 | 1310 | click buton | Add_(no_of_months=) | (no_of_months=)
46 | complete_financing_decision | test | 07/06/2011 06:52:27:801 | 1311 | click buton | =(cash_and_cash_equivalents - investment_value=) / (no_of_months=) | -3447.91666667
46 | complete_financing_decision | test | 07/06/2011 06:52:33:739 | 1312 | click buton | Add_(forecasted_revenues - forecasted_deployment_expenses=) | (forecasted_revenues - forecasted_deployment_expenses=)
```

Fig. 3. Partial example of a Decision Log

3.2 Theoretical approach of the Decision Data Model (DDM)

This section introduces the reader to the notion of DDM. The DDM needs to be understood as a representation of: the data items used in the decision process; and the dependencies between them.

A DDM [10] is a 3-tuple $(D,O;T)$ with:

- D: the set of data elements, $D = BD \cup DD \cup ID$, with
  - BD the set of leaf data elements
  - DD the set of derived data elements
  - ID the set of data elements inputted by the user
- $O \subseteq D \times P(D)$: the set of operations on the data elements.
  Each operation, $o = (d, ds)$:
  - has one output element $d \in DD$ and
  - has a set of zero or more input elements $ds \subseteq D$
- D and O form a hypergraph $H = (D, O)$ such that the structure of the graph is connected and acyclic.

For better understanding the definition, we will use the running example in Fig. 4. In this example, the user needs to make a decision about whether to make or not an investment. The sets of this particular DDM are:

BD = {available_cash (XA), investment_value (XB), forcasted_revenues (XD), forecasted_expenses (XE)}

ID = {no_of_months (XC)}

DD ={monthly_investment_payment (OUTA), monthly_investment_payback, decision (OUTB)}

O = {op1, op2, op3}

op1 = (monthly_investment_payment, {investment_value, available_cash, no_of_months})

op2 = (monthly_investment_payback, {forcasted_revenues, forecasted_expenses, no_of_months})

op3 = (decision, {monthly_investment_payment, monthly_investment_payback})

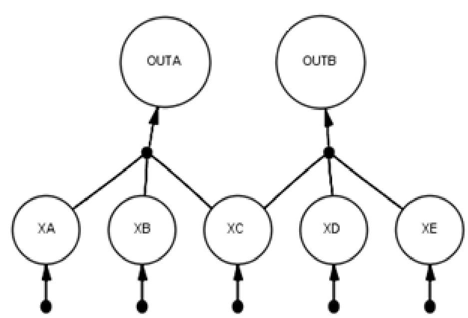


Fig. 4. Running example

3.3 The mining algorithm

This section introduces the main contribution of this paper, which is the mining algorithm implemented as a stand-alone software (Mining Tool in Fig. 1). We will demonstrate how a decision log (as introduced in section

3.1) can be converted into a DDM (as introduced in section 3.2).
The outline of the algorithm is:

```
Create date.xml
Create arrayWithDuplicates
Create arrayEgal
Create arrayX
Create arrayDerivedData
Create arrayDerivedDataName
Create Operations set
Create Data_Element set

Load XML Document
http://www.edirector.ro/processmining_v2
/export/pm.xml

   Do case for each record
            Case
      Find_click_textbox_in_WFMEIT_Field
      ()=True
               Add new item to
            arrayWithDuplicates

            Case
Find_"="_char_in_DAATE_Name_Field()=True
               Add new item to
arrayEgal
   Endcase

   For each element from
arrayWithDuplicates

existsInArray_function(arrayWithDuplicat
es,element)
            Add new item to arrayX
   endfor

   For each element from arrayX
            Add new item to
Data_Element set
   Endfor

   For each element from arrayEgal
            Subsir_function(element
arrayEgal)
            Add new item to
arrayDerivedDataName
   Endfor


   existsInArray_function(array,element)
        for each iterator in array
                  if (element is not
null && element = iterator)
                      return true
               endif
        endfor
               return false
   EndexistsInArray_function


   subsir_function ()
      if(element.Contains(„(’))

Reverse_of_element
                      if (lung > 0)
                           Extract
element_between_ parenthesis

      Identify_element_in_arrayDerivedDa
taName
                             Concatenate
„OUT”, the_next_letter_of_the_alphabet,
the_string_from_element1_starting_with_p
oz2+1 to element1
                              Concatenate
„OUT”, the_next_letter_of_the_alphabet
to ddElement
                             Add ddElement
to arrayDerivedData
                         endif
                                  return
subsir_function(element1)
                 else

                       Delete „=’ from
element
                       if
(!existsInArray_function(arrayDerivedDat
aName, element))
                              Add element
to arrayDerivedDataName
                              Add
                      element(„OUT” ,
                      (char)b++) to
                      arrayDerivedData
                      endif
      endif
              return ""
   endsubsir_function


   existsInArray_function(array,element)
        for each iterator in array
                   if (element is not
null && element = iterator)
                       return true
                   endif
        endfor
              return false
   endexistsInArray_function

   CreateDDM
      Output Operation set to XML
      Output Data_Element set to XML
   endCreateDDM
```

OBS: the algorithm outputs a PDM-XML file that needs to be imported in Prom 5.2 so that the graphical representation is created. Therefore, a 'fake' root node needs to be added for compliance with the PDM Plug-in in Prom 5.2. It is connected to all the data items that are not used in any data derivation.

To demonstrate how the algorithm works, we will introduce the decision log of the running example (Fig. 3). We will then go, step by step, through the algorithm to give a better understanding of how the DDM definition sets are extracted.

```
PI-ID | Description | Value | Timestamp | ATE-ID | WFMElt | Name | Value |
46 | complete_financing_decision | test | 07/06/2011 06:49:02:610 | 1291 | click menu item | bilant | bilant
46 | complete_financing_decision | test | 07/06/2011 06:50:51:547 | 1292 | click textbox | cash_and_cash_equivalents | 58625
46 | complete_financing_decision | test | 07/06/2011 06:50:53:381 | 1293 | click buton | Add_cash_and_cash_equivalents | cash_and_cash_equivalents
46 | complete_financing_decision | test | 07/06/2011 06:51:00:183 | 1294 | click menu item | evaluare_investitie | evaluare_investitie
46 | complete_financing_decision | test | 07/06/2011 06:51:04:794 | 1295 | click textbox | investment_value | 100000
46 | complete_financing_decision | test | 07/06/2011 06:51:11:26 | 1296 | click buton | minus | -
46 | complete_financing_decision | test | 07/06/2011 06:51:13:460 | 1297 | click buton | Add_investment_value | investment_value
46 | complete_financing_decision | test | 07/06/2011 06:51:16:617 | 1298 | click buton | =cash_and_cash_equivalents - investment_value | -41375
46 | complete_financing_decision | test | 07/06/2011 06:51:25:761 | 1299 | click textbox | forecasted_revenues | 310000
46 | complete_financing_decision | test | 07/06/2011 06:51:32:9 | 1300 | click textbox | forecasted_deployment_expenses | 4500
46 | complete_financing_decision | test | 07/06/2011 06:51:33:290 | 1301 | click buton | Add_forecasted_revenues | forecasted_revenues
46 | complete_financing_decision | test | 07/06/2011 06:51:36:988 | 1302 | click buton | minus | -
46 | complete_financing_decision | test | 07/06/2011 06:51:39:97 | 1303 | click buton | Add_forecasted_deployment_expenses | forecasted_deployment_expenses
46 | complete_financing_decision | test | 07/06/2011 06:51:40:895 | 1304 | click buton | =forecasted_revenues - forecasted_deployment_expenses | 305500
46 | complete_financing_decision | test | 07/06/2011 06:51:55:124 | 1305 | click textbox | no_of_months | 12
46 | complete_financing_decision | test | 07/06/2011 06:51:55:153 | 1306 | click buton | Add_no_of_months | no_of_months
46 | complete_financing_decision | test | 07/06/2011 06:52:13:22 | 1307 | click buton | =no_of_months | 12
```

Fig. 5. Important records for extracting the basic data elements

The most important fields of the decision log are WFMElt (Workflow Model Element) and DAATE_Name (Data Attributes Audit Trail Entries Name). WFMElt shows what kind of action the user performed while DAATE_Name stores the name of the control (it stores the name of the textboxes that contain the basic data items of the decision scenario or the formula used for calculating derived data items).

We will first focus on extracting the basic data elements (BD) and the inputted data elements (ID). Basically, we will explain how we create the set:

BD = {available_cash, investment_value, forcasted_revenues, forecasted_expenses} from the decision log of the running example (Fig. 3 and Fig. 5).

When the form loads all the textboxes storing the values of the data items are empty. In order to look at the actual figure, the user needs to explicitly click on the textbox. For example, the user sees there is a label (data item) called ‘Cash and cash equivalents’ but the textbox that shows the amount is blank (Fig. 6a).

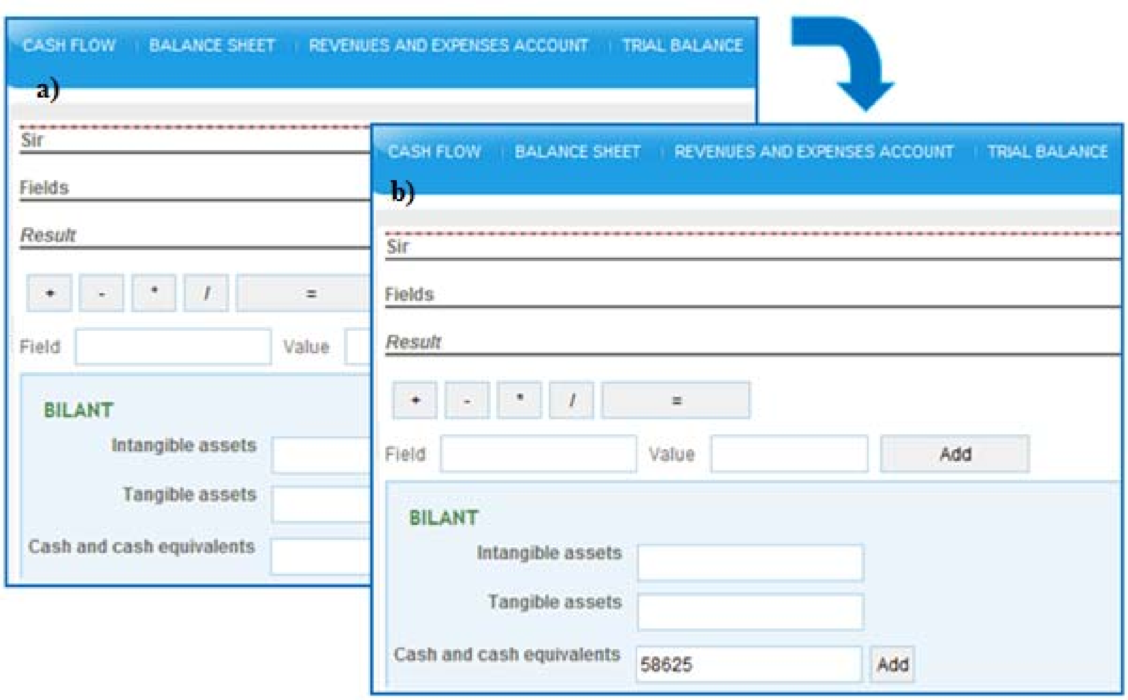


Fig. 6. a) the form before clicking Cash from other operations textbox and b) the form after the user clicked the Cash from other operations basic data element

When the textbox is clicked, the amount of 58625 is shown (Fig. 6b). The action of clicking the textbox and the value shown in the textbox is stored as a record in the decision log (Fig. 5 second record). Therefore, the value in the WFMEIT field we are looking for is „click textbox” because it shows that the decision maker used in the decision process one of the available basic data items.

The algorithm creates one DDM for each trace produced after each session. First of all, we proposed a list with all the elements of the column „DAATE_Name” and the value of „WFMEIT” being „click textbox”. We named the list „arrayWithDuplicates”. The users are allowed to click the same textbox many times. Therefore, the data items (textboxes) that were clicked multiple times must be identified. The rationale behind it is that once someone learns the value of a data item it can be considered as a known fact. Therefore, if the same textbox is clicked multiple times, the duplicate records in the decision log can be ignored. This is why there is a function which removes duplicate elements from arrayWithDuplicates.

Using the full name of the fields leads to huge graphical elements (circles that have a diagonal equal to the full name of the data element) in the model created with Prom 5.2. Therefore, we assigned a letter to each data item. So, we have two uni-dimensional arrays, one with the names of the basic data elements (from the input XML) and one with the short labels.

We will now focus on extracting the derived data elements (DD). We are looking to create the set:

DD ={monthly_investment_payment, monthly_investment_payback}

from the records of the decision log. We are looking at the records highlighted in Fig. 7.

```
PI-ID | Description | Value | Timestamp | ATE-ID | WFMElt | Name | Value |
47 | complete_financing_decision | test | 07/06/2011 06:57:47:91 | 1357 | click buton | Add_(forecasted_deployment_expenses=) | (forecasted_deployment_expenses=)
47 | complete_financing_decision | test | 07/06/2011 06:57:48:642 | 1358 | click buton | =(forecasted_revenues=) - (forecasted_deployment_expenses=) | 305500
47 | complete_financing_decision | test | 07/06/2011 06:57:50:368 | 1359 | click buton | Add_((investment_value=) - (cash_and_cash_equivalents=)=) | ((investment_value=) -
(cash_and_cash_equivalents=)=)
47 | complete_financing_decision | test | 07/06/2011 06:57:51:890 | 1360 | click buton | impartire | /
47 | complete_financing_decision | test | 07/06/2011 06:57:53:322 | 1361 | click buton | Add_(no_of_months=) | (no_of_months=)
47 | complete_financing_decision | test | 07/06/2011 06:57:54:779 | 1362 | click buton | =((investment_value=) - (cash_and_cash_equivalents=)=) / (no_of_months=) | 3447.91666667
47 | complete_financing_decision | test | 07/06/2011 06:57:56:409 | 1363 | click buton | Add_((forecasted_revenues=) - (forecasted_deployment_expenses=)=) | ((forecasted_revenues=) -
(forecasted_deployment_expenses=)=)
47 | complete_financing_decision | test | 07/06/2011 06:57:59:502 | 1364 | click buton | impartire | /
47 | complete_financing_decision | test | 07/06/2011 06:58:00:706 | 1365 | click buton | Add_(no_of_months=) | (no_of_months=)
47 | complete_financing_decision | test | 07/06/2011 06:58:01:834 | 1366 | click buton | =((forecasted_revenues=) - (forecasted_deployment_expenses=)=) / (no_of_months=) |
25458.3333333
47 | complete_financing_decision | test | 07/06/2011 06:58:06:932 | 1367 | click menu item | Decizii | Decizii
47 | complete_financing_decision | test | 07/06/2011 06:58:10:552 | 1368 | click combobox | monthly_payment_type | Rate
```

Fig. 7. Important records for extracting the derived data elements

The first step is to create an array with all the records in Name field that start with equal sign. We named it arrayEgal.

For that we use recursive function which applies to all elements. It is important that the element contains the character ”(” so that we can compare with the elements that already exist. We copy those elements to an array called derrivedDataName (e.g.: cash_from_suplliers, cash_to_customers, etc). Then, we use a second list which for each item in the first list assigns labels (like XA, XB, XC, etc) according to the index of the element. Then, a derived data element can be calculated based both on basic data and on previously derived data (e.g. one can first calculate A+B and then use the value to multiply it to C, thus calculating (A+B)*C). We created a complex function that looks for previously calculated derived data items and then replaces each data item with the correct short label (letter). We will explain how this function works using an example. In Fig. 8 we show how the third derived data element from the log in Fig. 7 is extracted.

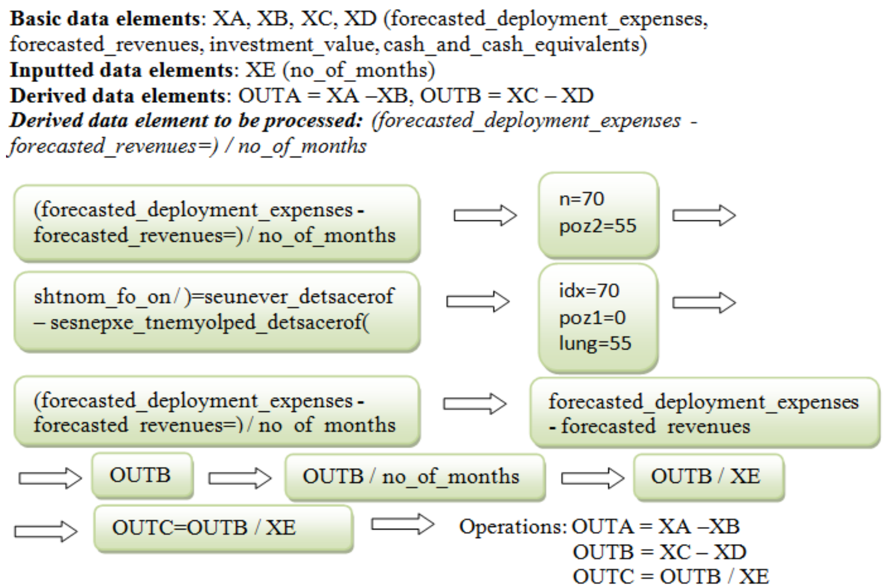


Fig. 8. Example of processing for a derived data element

First of all, we are interested in the length of the element (n), then we verify if the element contains „(". If it is true, we calculate the position of the next „)" (poz2) and we reverse the element (for example „ forecasted_deployment_expenses - forecasted_revenues" will become „seunever_detsacerof – sesnepxe_tnemyolped_detsacerof"). Another position (idx) that we calculate is the position of „(" starting from the length of the elements minus the position calculated previously (from position n-poz2). Position poz1 is calculated as n – idx and the value of lung is calculated as poz2 – poz1. Then we reversed again the element and we extracted the string between poz1 and lung. After that we compare that string with the elements from the array derivedDataName (at first this array is empty) and if the string is not found through the elements of the array we added it. We process similarly the array derivedData. The algorithm stops when the all records in Name field starting with "=" sign have been evaluated.

Based on BD, ID, DD, and O, the DDM/PDM structure can be created. For creating the PDM-XML file that can be imported in Prom 5.2, we use the items from the strings created by the algorithm. First, we need to define the data elements. They consist of the elements from basic data items (*arrayX*), from derived data (*arrayDerivedData*) and the "fake" root element. Next step is to define the operations. We are not interested, at the moment, in the arithmetical operations that were performed. Instead, we focus on the data items that are used to calculate a new derived data item. Extracting the input data items of an operation is done in two steps. First, we properly extract the operations depicted in the decision log (operation 1 to 3). For example, operation 1 of our running example (see Fig. 3) is depicted below (input and output), where the input is represented by XA which is the available cash, XB which is the investment value, and XC which is the number of months in a year. The outputted derived data element is OUTA, which refers to the monthly investment payment. The structure of the PDM-XML file is:

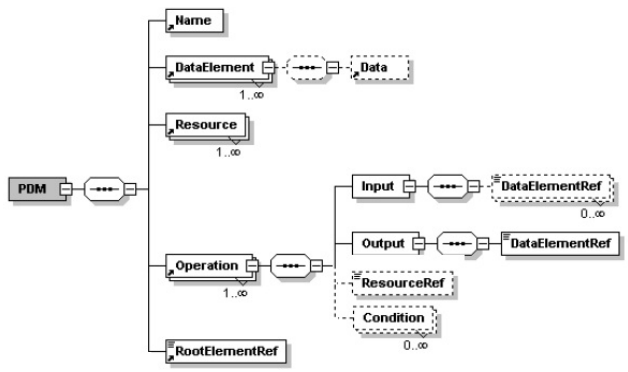


Fig. 9. PDM-XML structure mandatory for PDM Plugin in Prom 5.2

For example, the PDM-XML data outputted for operation 1 is:

```
<Operation OperationID="Op1">
   <Input>
<DataElementRef>XA</DataElementRef
>
<DataElementRef>XB</DataElementRef
>
<DataElementRef>XC</DataElementRef
>
   </Input>
   <Output>
<DataElementRef>OUTA</DataElementR
ef>
</Output>
   <ResourceRef>w1</ResourceRef>
   <Condition>true</Condition>
</Operation>
```

The second step is to create operations for the leaf nodes (XA, XB, XC, XD and XE). We are creating one operation for each leaf node (from 4 to 8). As can be seen below, for the operation number 4 the input set is empty and the output set is represented by the leaf node (XA).

```
<Operation OperationID="Op4">
  <Input>
   </Input>
  <Output>

<DataElementRef>XA</DataElementRef>
  </Output>
    <ResourceRef>w1</ResourceRef>
    <Condition>true</Condition>
  </Operation>
```

Strictly for compliance with the PDM-XML structure needed for properly importing the file in Prom 5.2, at the end of the DDM is defined the root element. All the elements that are not present as an input of any operation will be related directly to the root element (for example OUTA and OUTB).

4 Evaluation of the algorithm

One of our databases with decision logs contains a sample of 42 traces. These are recorded from the bachelor and master level students from Babeş-Bolyai University of Cluj-Napoca and from the West University of Timişoara. In other databases we store traces from expert users (some of them work in loan granting departments of different banks on various decision levels, other are expert accountants, are working in auditing or are managers of companies that have a long history of loan contracting) and other bachelor-level students. We used the mining tool introduced above to mine all those logs.

By studying the logs we recorded so far we found some common patterns for the DDMs. In this section we intend to analyze some use-cases that illustrate how the mining algorithm extracts the most common patterns of the DDMs. The goal is to prove that the algorithm is robust enough to correctly extract the data processing view of the decision processes performed by the users of the decision-aware software.

The first use-case is when the decision maker uses the decision-aware software only to look at the values in some textboxes. In this case, the decision process is clear: the user makes his decision without using any derived data and all the basic data items reviewed are the criterions used for choosing an alternative. From the Mining Tool's point of view, only the basic data set is built (*arrayX*) and the elements are related directly to the *Root Element*.

In order ease the understanding of our use-cases, we simplified the structure of the final query of the decision log file. We only kept the elements which are extracted by our algorithm (Timestamp, WFMEIT, DAATE_Name and Value).

Table 1. Log sample for the first use-case

Timestamp	WFMEIT	DAATE_Name	Value
Time1	Click menu item	Trial balance	Trial balance
Time2	Click textbox	Cash_from_customers	48943
Time3	Click textbox	Cash_paid_to_suppliers	73312
Time4	Click menu item	Balance sheet	Balance sheet
Time5	Click textbox	Cash_paid_to_suppliers	48943
Time6	Click textbox	Paid_vat	58625

As we showed before, the first step of the algorithm is to build the list/array with basic data element and make those elements unique. To build the basic data items sets, we consider the items of WFMEIT value equal to „click textbox" (records 2, 3, 5 and 6 in Table 1). Then, the algorithm assigns to each data item values as XA, XB, XC, etc. Therefore Cash_from_customers will be labeled XA, Cash_paid_to_suppliers will be labeled XB and Paid_vat will be labeled XC. There are cases when some elements appear more than once in the trace. For example, in Table 1, XB shows twice (for some reason the user clicked the Cash_paid_to_suppliers textbox twice). Even if XB appears more than once in the trace, in the DDM_XML it should show up only once because it indicates that this particular piece of knowledge (the total value of the invoices issued to customers) was used in the decision process.

Given that the trace doesn't involve derived data items (records in DAATE_Name field that start with "=" character), the elements from basic data array are directly related to the root node and so only one element is created in the operation set: op1 = (ROOT,{XA,XB,XC}). As we can see in Fig. 10, the set containing the elements inputted by the user and the derived data element set are empty because the user only considered the value of some textboxes and he didn't perform data derivations.

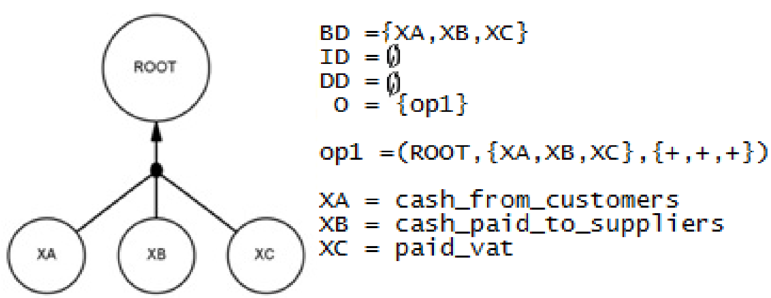


Fig. 10. PDM of the first use-case

The conclusion that can be drawn by looking at this type of model (where there are no derived data elements) is that the user is not sure of what he's doing or what he should do, and that there is no clear criterion for making the decision. Our overall conclusion would be that the user is making a rather intuitive decision.

The second use-case is the one where the decision maker calculates a derived data item and, then, uses it in a further calculation with a basic data item. The challenge is to identify when a derived data item calculated previously is used for further derivations in conjunction with a basic data item.

Table 3 represents the simplified version of a decision log with derived data elements.

Table 2. Log sample for the second use-case

Timestamp	WFMEIT	DAATE_Name	Value
Time1	click menu item	Trial balance	Trial balance
Time2	click textbox	customer_invoices_value	48943
Time3	click button	Minus	
Time4	click textbox	suppliers_invoices_value	73312
Time5	click button	Add_suppliers_invoices_value	suppliers_invoices_value
Time6	click button	=customer_invoices_value - suppliers_invoices_value	-24369
Time7	click button	Add_(customer_invoices_value - suppliers_invoices_value=)	(customer_invoices_value - suppliers_invoices_value=)
Time8	click button	Plus	+
Time9	click menu item	Balance sheet	Balance sheet
Time10	click textbox	cash_and_cash_equivalents	58625
Time11	click button	Add_cash_and_cash_equivalents	cash_and_cash_equivalents
Time12	click button	=(customer_invoices_value - suppliers_invoices_value=) + cash_and_cash_equivalents	34256

Table 3. Simplified view of the log sample of the second use-case

Timestamp	WFMEIT	DAATE_Name
Time1	click menu item	menu1
Time2	click textbox	XA
Time3	click buton	Minus
Time4	click textbox	XB
Time5	click buton	Add_XA
Time6	click buton	=XA – XB
Time7	click buton	Add_(XA - XB=)
Time8	click buton	Plus
Time9	click menu item	menu2
Time10	click textbox	XC
Time11	click buton	Add_XC
Time12	click buton	=(XA-XB=) + XC

As we explained before, first step is to create the basic data element set. In Table 2 and Table 3, customer_invoices_value (XA), suppliers_invoices_value (XB) and cash_and_cash_equivalents (XC) are the basic data items.

The second step is to find the derived data elements. The logging mechanism implemented in the decision-aware software places ‚=' (equal sign) in front of the record that is placed in DAATE_Name field when the user clicks the Equal Button in the calculator area. As for the basic data items, each derived data item is labeled by letters: OUTA, OUTB, OUTC, etc. In this use-case we have only two items in the derived data set (OUTA and OUTB). Like in the first use-case, when the user was looking only at certain textboxes, if one element (basic or derived) appears in the trace more than once in the final PDM-XML it will appear once. Once a derived data element is found, its components are analyzed. After we have replaced the names, we are looking for the elements between parenthesis, so we get OUTA= (XA – XB=) and OUTB=((XA – XB=) – XC=). For the first derived data element we are searching the string at the beginning of the first mathematical sign, so we get XA. Then we delete it and we are looking for the next data item in the same way (we get XB). Therefore we are recognized XA and XB as being part of the basic data element set. We conclude that OUTB consists of ((XA – XB=) – XC=) and it has more than one equal sign in his composition. This means that we have to look for ((XA – XB=) in the derived data elements array. We identify this as being OUTA and OUTB becomes OUTA - XC.

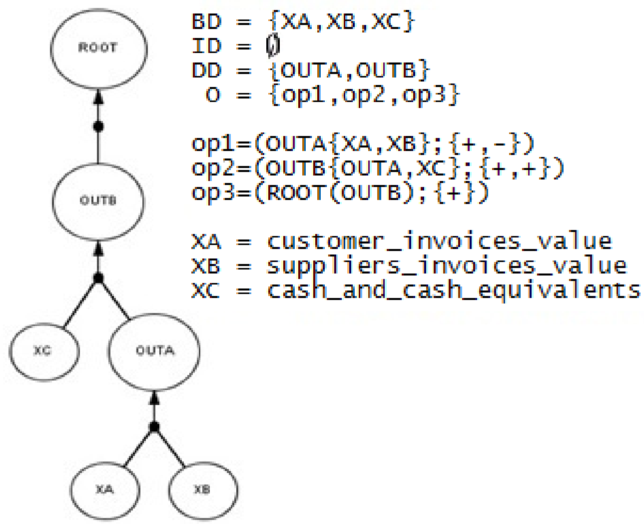


Fig. 11. PDM of the second use-case

By looking at the resulting model, we can state that the decision making process is well structured. This is a hint that the decision maker is well aware of the criteria used in choosing the decision alternative and exhibits clear knowledge of how the important criteria can be calculated.

The third use-case is the one where the decision maker calculates two derived data items and, then, uses both of them in a further calculation. The challenge is to identify when several derived data items calculated previously are used for further derivations.

Table 4. Log sample for the third use-case

Timestamp	WFMEIT	DAATE_Name	Value	Labels
Time1	click menu item	Balance sheet	Balance sheet	Menu1
Time2	click textbox	Cash_and_cash_equivalents	58625	XA
Time3	click button	Add_cash_and_cash_equivalents	cash_and_cash_ equivalents	Add_XA
Time4	click menu item	Investment	Investment	Menu2
Time5	click textbox	investment_value	100000	XB
Time6	click button	Minus		Minus
Time7	click button	Add_investment_value	investment_value	Add_XB
Time8	click button	=cash_and_cash_equivalents – investment_value	41375	= XA – XB
Time9	click menu item	Balance sheet	Balance sheet	Menu1
Time10	click textbox	Receivables	49096	XC
Time11	click button	Add_receivables	receivables	Add_XC
Time12	click textbox	short_term_debts	82761	XD
Time13	click button	Minus		Minus
Time14	click button	Add_ short_term_debts	short_term_debts	Add_XD
Time15	click button	= receivables - short_term_debts	-33665	= XC – XD
Time16	click button	Add_(cash_and_cash_ equivalents – investment_value=)	(cash_and_cash_ equivalents – investment_value=)	Add_(XA - XB=)
Time17	click button	Plus		Plus
Time18	click button	Add_(receivables - short_term_debts=)	(receivables - short_term_debts =)	Add_(XC - XD=)
Time19	click button	=(cash_and_cash_equivalents – investment_value=) + (receivables – short_term_debts=)	7710	=(XA - XB=) + (XC - XD=)

In this use-case, the basic data element set consists of XA, XB, XC and XD, the derived data element set consists of OUTA, OUTB and OUTC and the operation set contains four items (see Fig. 12). The most important part is how the algorithm recognizes the elements from derived data element set. In the decision logs in Table 4 and **Error! Reference source not found.** the 19th record is labeled OUTC. The formula used to derive it is *=(cash_and_cash_equivalents – investment_value=) + (receivables – short_term_debts=)*. If we replace the names of basic data elements with XA, XB, XC and XD we get the formula *=(XA - XB=) + (XC - XD=)*. Going through the derived data element set we recognize that OUTC consists of OUTA added to OUTB in the same way as explained in the previous use-case.

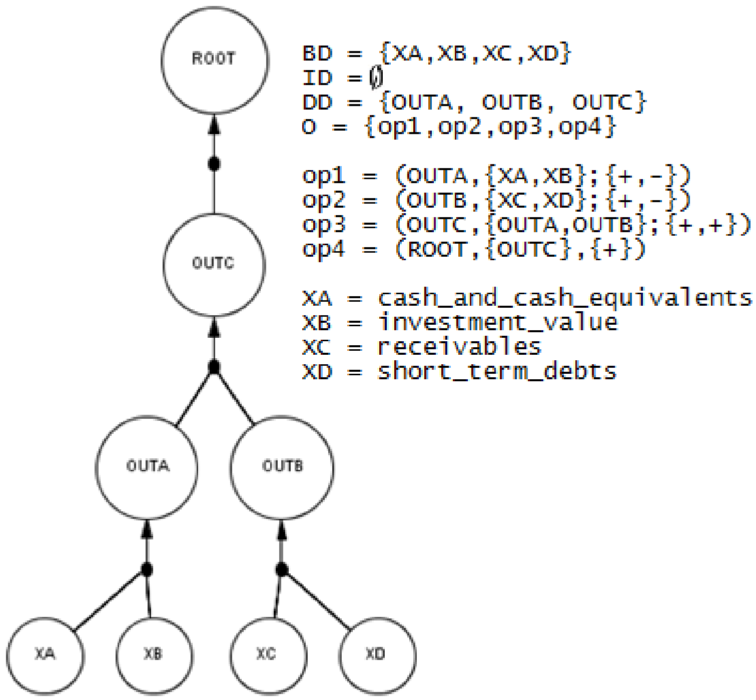


Fig. 12. Model of the third use case

Fig. 12 shows a more strucrtured decision process. One can see that it resembles a tree with intermediary data aggregations. The same result can be calculated without the use of the intermediary OUTA and OUTB. We are not interested in the result but on how the decision maker gets to it.

In the last use-case, as shown in Fig. 13, one element, whether basic or derived, may belong to one or more operations. In our case, the basic data elements XB (customers_invoices_value) and XC (cash_paid_to_employees_incl_taxes) are used in two different operations (XB is used once with XA when OUTA is calculated and once with OUTB when OUTD is calculated; while XC is used once with XD when OUTB is calculated and once with OUTA when OUTC is calculated).

Table 5. Logged data of the fourth use case

Timestamp	WFMEIT	DAATE_Name	Value	Labels
Time1	click menu item	Trail Balance	Trial Balance	Menu1
Time2	click textbox	Suppliers_invoices_value	73312	XA
Time3	click buton	Add_suppliers_invoices_ Value	suppliers_invoices_ value	Add_XA

Time4	click buton	Minus		Minus
Time5	click textbox	customers_invoices_value	48943	XB
Time6	click buton	Add_customers_invoices_ value	customers_invoices_ value	Add_XB
Time7	click buton	= suppliers_ invoices_value - customers_invoices_value	24369	= XA – XB
Time8	Click menu	Cash Flow	Cash Flow	Menu2
Time9	click textbox	cash_paid_to_employees_ incl_taxes	3005	XC
Time10	click buton	Add_cash_paid_to_ employees_incl_taxes	cash_paid_to_ employees_incl_taxes	Add_XC
Time11	click buton	Plus	+	Plus
Time12	click textbox	Paid_vat	538	XD
Time13	click buton	Add_paid_vat	Paid_vat	Add_XD
Time14	click buton	= cash_paid_to_ employees_incl_taxes + paid_vat	3543	= XC + XD
Time15	click buton	Add_(suppliers_ invoices_value - customers_invoices_value =)	(suppliers_ invoices_value- customers_invoices_ value=)	Add_(XA – XB=)
Time16	click buton	Minus	-	Minus
Time18	click buton	Add_cash_paid_to_ employees_incl_taxes	cash_paid_to_ employees_incl_taxes	Add_XC
Time19	click buton	=(suppliers_invoices_ value - customers_invoices _value =) -cash_paid_to_ employees_incl_taxes	21364	=(XA - XB=) – XC
Time20	click buton	Add_customers_ invoices_value	customers_invoices _value	Add_XB
Time21	click buton	Minus	-	
Time22	click buton	Add_(cash_paid_to_ employees_incl_taxes=) + paid_vat	cash_paid_to_ employees_incl_taxes + paid_vat=	
Time23	click buton	= (cash_paid_to_ employees_incl_taxes=) + paid_vat	45400	

The resulting DDM in Fig. 13 shows that the basic data element set consists of XA, XB, XC and XD and derived data element set consists of OUTA, OUTB, OUTC and OUTD. The components are obtained like we explained before by separating each derived data item and looking for its components in the basic or derived data element set.

If one takes a closer look at the model in Fig. 13 it is obvious that the decision maker is exhibiting a structured process. It can be easily observed that in-depth knowledge of the relationship between the data elements is made explicit by the model. It is also observable in the model, the criterions used in choosing the decision alternative.

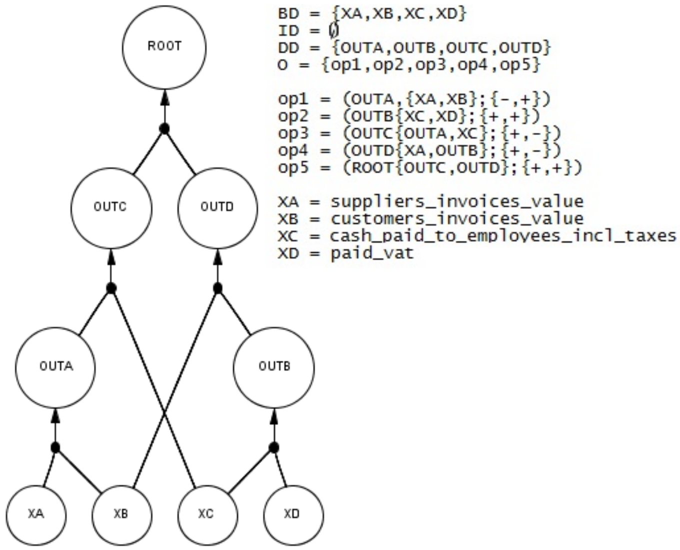


Fig. 13. Structured process

5 Conclusions

We argued that the decision maker is not always capable to consciously explain the decision process he is performing. Our approach gets large numbers of decision makers to reason on the same decision data and scenario in order to pick one of the provided decision alternatives. We log the behavior of the decision maker and extract a model from those logs. We argue that our approach is able to explicitly depict, in a model, the relationships between the data items used in making the decision. At the core of our approach is the algorithm that automatically mines some logs (of user interaction with simulation software) and extracts a model of the data perspective of a decision process. This paper demonstrated, in detail, how exactly is a decision log file converted into a Decision Data Model (DDM-XML file).

The first part of the paper was dedicated to a detailed presentation of the proposed mining algorithm. We started by introducing the outline of the algorithm (as pseudo-code) and provided detailed insights into the functions we used. In the second part of the paper, we set to prove the algorithm's robustness by analyzing different patterns that occur most frequently in the real decision logs (and subsequently in the mined models). We looked at increasingly complex patterns by showing how data is stored in the logs and explained how the decision data model is extracted by our algorithm.

Once a DDM is created for each decision maker, we aim to extend our research by aggregating individual DDMs into a common reference model. It is also possible to compare individual DDMs or to compare one individual DDM with an aggregated DDM by calculating a score of similarity between the models. Another direction for future research is aimed at increasing the quality of the logs by employing eye-tracking. This will require some future adaptations and extensions of the algorithm presented and evaluated in this paper.

Acknowledgement

This work was supported by CNCSIS-UEFISCSU, project number PN II – RU - TE 52/2010 code 292/2010.

References

[1] H.A. Simon, The New Science of Management Decision. *Harper and Row*, New York, 1960

[2] M. Hu, E.-P. Lim, A. Sun, H. W. Lauw and B.-Q. Vuong, „Measuring article quality in wikipedia: models and evaluation", *Proceeding CIKM '07 Proceedings of the sixteenth ACM*

conference on Conference on information and knowledge management, 2007

[3] W.M.P van der Aalst and K. van Hee, „Workflow Management: Models, Methods and Systems”, MIT Press, Cambridge, 2002

[4] W. M.P.van der Aalst, Maja Pesic and Minseok Song, „Beyond Process Mining: From the Past to Present and Future”, *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering* (CAiSE'10), volume 6051 of Lecture Notes in Computer Science, pp. 38-52, Springer-Verlag, Berlin, 2010

[5] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm and A.J.M.M. Weijters, Workflow Mining: A Survey of Issues and Approaches, *Data and Knowledge Engineering*. no. 47(2), pp. 237-267, 2003

[6] W.M.P. van der Aalst, A.J.M.M. Weijters, L. Maruster, Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering* 16(9), 1128–1142, 2004

[7] A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst, „Genetic Process Mining: An Experimental Evaluation”. *Data Mining and Knowledge Discovery*. 14(2), 245-304, 2007

[8] C.W. Gunther, W.M.P. van der Aalst, Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In: *Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007*. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg, 2007

[9] I. Vanderfeesten, “Product-Based Design and Support of Workflow Processes”. *Eindhoven University of Technology*, Eindhoven, 2009

[10] R. Petruşel, I. Vanderfeesten, C.C. Dolean and D. Mican, „Making Decision Process Knowledge Explicit Using the Decision Data Model”, *Ed. BIS 2011, LNBIP 87*, pp. 172-184, 2011, Springer-Verlag Berlin Heidelberg, 2011.

[11] R. Petruşel, D. Mican and C.C. Dolean, „Implementing a Decision-Aware System for Loan Contracting Decision Process”, *Informatica Economica Journal*, Vol. 15, no. 1/2011, pp. 167-182.

[12] R. Petruşel, I. Vanderfeesten, C. Dolean and D. Mican, „Making Decision Process Knowledge Explicit Using the Product Data Model”, *Beta working paper series*, WP 340, Technische Universiteit Eindhoven, 2011.

Cristina-Claudia DOLEAN has graduated the Faculty of Economics and Business Administration, Babeş Bolyai University, Cluj-Napoca in 2008. She holds a bachelor degree in Business Informatics and a master degree in E-Business. She is currently a PhD student in the field of Business Informatics. Her current research interest include Process mining and Workflow Management.

Răzvan PETRUŞEL holds a Ph.D. in Cybernetics, Statistics and Business Informatics starting 2008. He started in 2003 as a full-time Ph.D. student at the Business Information Systems Department, Economical Sciences and Business Administration Faculty, in Babeş-Bolyai University of Cluj-Napoca. In 2007 he became an assistant professor and since 2009 he holds the current position as lecturer. His research is focused on DSS Specification, Modeling and Analysis; Process Mining; Workflow Management; and Decision Mining and Analysis.