Making Recommendations for Decision Processes Based on Aggregated Decision Data Models

Razvan Petrusel and Paula Ligia Stanciu

Faculty of Economical Sciences and Business Administration, Babes-Bolyai University,
Teodor Mihali str. 58-60, 400591 Cluj-Napoca, Romania
{razvan.petrusel,paula.stanciu}@econ.ubbcluj.ro

Abstract. The decision making process is a sequence of (mostly mental) actions. But individual decision making is a fuzzy process that lacks a clear workflow structure. This issue may decrease the quality of data-centric business decisions where information must be processed in the right order and used at the right time. We argue that, when faced with such a decision, step-by-step recommendation provides help in steering the process and valuable guidance in improving it. Our Data Decision Model (DDM) is an acyclic graph that suits the fuzzy nature of decision processes. In our approach, the recommendation is based on an aggregated DDM extracted from a large number of individuals. This paper introduces two algorithms that, given a certain state of the process, provide suggestions for the next action the decision maker should perform.

Keywords: Decision Making Recommendation, Decision Data Model, Decision Process Mining.

1 Introduction

Decision making is a daily activity for every individual. Most of the decisions we make are based on experience and heuristics rather than on a scientific methodology. But what happens when a decision maker is in a decision situation in which he doesn't know where to start from or what to do next? Wouldn't it be nice if, when stuck, there was some recommendation that would indicate what others have done in a similar situation?

Our approach aims to provide the framework and tools for researching the business decision making behavior of many individuals. The decision making process is looked at as a workflow of actions directed towards choosing a decision alternative. In our previous papers we argued that it is feasible to use management simulation software that logs what users are doing while making a particular decision. We also showed how a model that aggregates the behavior of all those individuals can be mined from those logs. This paper shows how the model is used in order to provide recommendations (what to do next), given a certain state in the decision process.

We are not aiming at automating the decision process or at providing a way for choosing the "best" decision alternative out of a set of choices. Our approach is intended to provide support by steering the flow of decision maker's activities aimed at

W. Abramowicz et al. (Eds.): BIS 2012, LNBIP 117, pp. 272–283, 2012.

generating and evaluating the choices. We do not aim to recommend which of the choices is the best one. Our recommended next action can either be performed or ignored by the decision maker. On the next step of the process, the recommendation is updated according to the action performed by the decision maker. This kind of support could be valuable since a decision maker may either overlook some essential aspects of a decision, or may lack the knowledge required to make a particular decision.

Our research is placed in the context of business decisions. The recommendations are based on a pre-requisite aggregate Decision Data Model (DDM). Our approach is looking at data because most business decisions are data-centric. The actions performed during a business decision process (e.g. find out the trading price of some stock; look at the trend for a stock over the last week) usually overlap with data manipulations (e.g. retrieve a figure from some source; compute the daily changes of stock prices).

The next section of the paper introduces the reader to an overview of the approach and to its formal fundaments. The third section introduces two algorithms that can be used for providing recommendations based on a mined decision data model (DDM). In the fourth section we validate and compare the proposed algorithms using an aggregated model extracted from 50 decision makers. The last two sections deal with the related work and the conclusions.

2 The Approach

This first sub-section introduces an overview of our previous work in order to provide the context of the work presented in this paper. The second sub-section introduces the formal fundaments that will be used throughout the paper.

2.1 The Framework

This sub-section aims to introduce the reader to the framework of decision process mining. This paper focuses on a small part of this framework (i.e. providing recommendations based on a previously mined decision model).

Everything starts (Fig. 1) with a large number of decision makers that interact with software in order to make some decision. It can be any software (even a simple spreadsheet) which simply provides the users with a set of values that are relevant to the decision to be made. The goal of the process should be to choose one of the given decision alternatives, based on the actual values shown in each scenario. Since we are interested in business decision making, we can use as examples of such software the on-line stock trading platforms, management simulation games, on-line shops, etc. One of the features of such software is that the decision maker should not be influenced in any way during the process. The software should be able to log the activity of the user by various means (e.g. following the mouse moves and/or clicks, eye-tracking, etc.). We call this kind of software ‘decision-aware’. More details on the decision-aware software and on about user activity logging are available in [1].

The logs are processed by a mining application that outputs individual or aggregated Decision Data Models. More details, on how the data is mined and the individual DDM is created, are available also in [1]. The aggregation can be done for all the traces in the log or just for a selection. Once an aggregated model is created,

the process of one individual cannot be distinguished. Also, in an aggregated model, one cannot look at the most frequent path in the sense of a workflow model. But it is annotated with the frequency of the operations, so the most frequent activities of the subjects are easily visible.

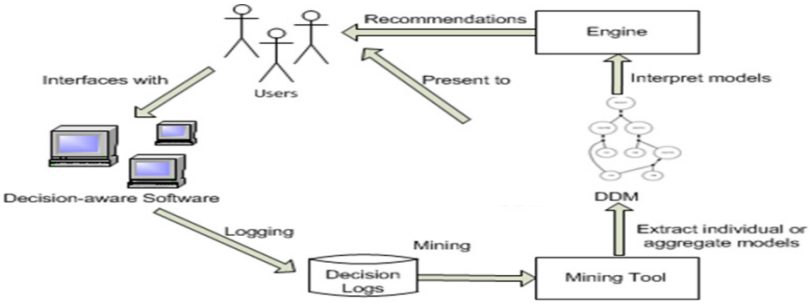

Fig. 1. The framework of decision making process mining

Once the DDM (individual or aggregated) is created, it can be introduced as it is to the users. It can be used to either to gain a better understanding of the actions of a particular decision maker or to look at the aggregated behavior of many individuals. It is also possible to measure the distance between two DDMs (individual and/or aggregate) using different metrics. Calculating a similarity score of two users is one of the interesting outputs of our research. Another interesting output is the possibility to cluster the decision makers (based on the distance matrix of all the models).

The upper right area of Fig. 1 is where the goal of this paper lays. We show how recommendations for "best" next action to be performed, by the decision maker, can be produced. Algorithms are based on a previously extracted aggregated DDM.

The essential assumptions related to providing recommendations are:

- the decision scenario provided to the user contains all the data needed for the decision at hand. There is no other essential information relevant for the decision;
- the data provided in the scenario allows the user to explore and evaluate all the possible decision alternatives;
- a particular user might overlook certain aspects of a decision. But, all the aspects of a particular decision should be discoverable, if a large number of users are observed;
- the more frequently a certain derived data item is observed in decision making processes, the more essential it is for performing the choice of a decision alternative.

For example, our approach can be used to gain insights into buying stocks from the stock market. We could add the logging feature to a site providing stock information (e.g. http://www.reuters.com/finance/stocks). Then, we could ask a lot of users to start deciding whether to buy or sell stock and log their actions while making the decision (data that is looked at, data that is compared or added, etc.). Based on the aggregated DDM we could provide recommendations about which data is important. We can suggest the order of finding out and using it based on frequent elements of the DDM.

In order to improve the understanding of the aggregate DDM and the algorithms based on it, we will use a running example. Let's suppose that we observed 100

decision makers while making the same decision, based on the same data. The sequences of operations and their frequency (F) are: F 10 (opA, opB, opC, opD, opB, op1, op2, op3), F 10 (opA, opB, opC, opD, opB, op1, op2, op3, op4), F15 (opA, opB, opC, opD, op1, op2, op5), F15 (opA, opB, opC, opD, opE, op1, op2, op5, op6), F20 (opA, opB, opC, opD, opF, op1, op2, op5, op7), F30 (opC, opD, opA, opB, opF, op2, op8). Starting from those traces, we can mine the aggregated DDM (in Fig. 2). Each operation is annotated with the frequency e.g. opA shows up 100 times.

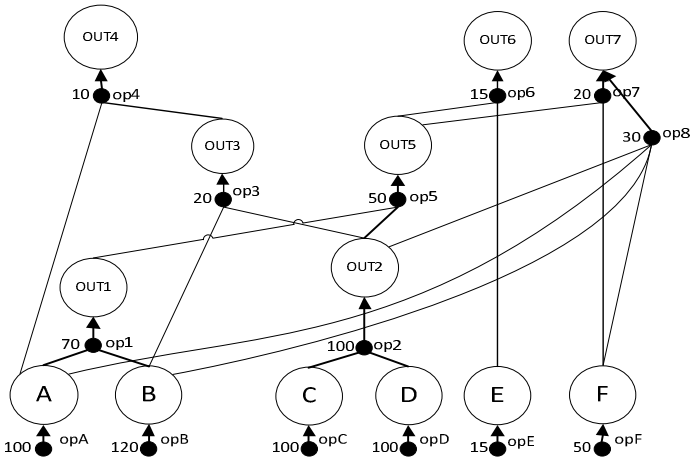

Fig. 2. A Decision Data Model used as the running example

The operations are labeled according to the type of output (i.e. if the input is the empty set then it produces a basic data items and is labeled with its name; or else it produces a derived data items and is labeled artificially). The semantics of the model implies that an operation can be executed only if its input data items are known (e.g. op1 can be executed only if the values of A and B are both known). If we use the example with the stock market, the model states that the trend of some stock (out1) can be calculated only if the previous price (A) and the current price (B) are known.

2.2 The Formal Approach

This sub-section introduces the essential notions used further in the paper. Basically, we are looking at a search problem in an acyclic, not rooted graph (DDM). The problem of providing recommendations is to determine the "best" path through the DDM. Our problem can be mapped to a general search problem with five components: S, S_0, S_g, *successors* and *cost* (where S is a finite set of states; $S_0 \subseteq S$ is the non-empty set of start states; $S_g \subseteq S$ is the non-empty set of goal states; *successors* is a function $S \rightarrow P(S)$ which takes a state as input and returns another state as output (probabilities may be used in connection with it); and *cost* is a value associated to moving from state $s \in S$ to $s' \in S$). The total cost is the sum of the *costs* incurred by a sequence of movements from state $s \in S_0$ to a state $s' \in S_g$. A recommended strategy is a sequence of actions such as the total cost is minimized (or

maximized under some circumstances). A particular feature of our problem is that, because we use simulation software, there are no costs for moving from a state to the next (as used in classical search problems). Instead, the notion of cost is derived from the notion of frequency.

Definition 1: A Decision Data Model (DDM) is a tuple (D, O) with:

– D: the set of data elements d, D = BD ∪ DD where BD is the set of basic data elements and DD is the set of derived data elements;
– O: the set of operations on the data elements. Each operation, o is a tuple (d, v, DS, t), where:
  - $d \in$ DD, d is the name of the output element of the operation;
  - v is the value outputted by the operation. Can be numeric or Boolean;
  - DS, a set of $d \in$ D, is the set of input data elements of the operation.
  - $t \in$ T, where T is the set of timestamps at which an operation from O occurs (i.e. the time when the element d is created using o).
– D and O form a hyper-graph H = (D, O), connected and acyclic.

An example of a DDM is depicted in Fig. 2.; where BD = {A, B, C, D, E, F}, DD = {Out1, Out2, Out3, Out4, Out5, Out6, Out7} and O = {opA, opB, opC, opD, opE, opF, op1, op2, op3, op4, op5, op6, op7, op8}. For example, the operation opA = (A, 100, ø, time_10) and op3 = (Out3, 1000, {B, Out2}, time_100).

Definition 2: $o_i = (d_i, v_i, DS_i, t_i)$ and $o_j = (d_j, v_j, DS_j, t_j)$ are *alternate (or mutually exclusive) operations* if $d_i = d_j$ and $v_i = v_j$. One can note that, in Fig. 2, op 7 and op8 output the same derived data item (Out7). However, they are not the same operation because their inputs are different. Mutually exclusive operations are important because if, for example, op7 is performed there is no need to perform op8 and the other way around.

Definition 3: An *aggregated DDM* is a DDM annotated with the *frequency* of the operations. *Frequency* indicates how many times an operation shows up in the log. In Fig. 2 the number near an operation indicates its frequency.

Property 1: An aggregated DDM allows the user to 'zoom-in" or "zoom-out'. Zooming-in shows less frequent behavior while zooming-out shows only operations with a frequency above a certain threshold. This is extremely useful when the aggregated DDM shows a lot of outlying behavior. For example, the outliers can be removed by setting the threshold value to 2 (i.e. the same operation should be performed by at least two different decision makers).

Definition 4: An operation o = (d, v, DS, t) is *enabled* when the input data elements (DS) are available (known to the decision maker). If an operation is enabled it may be executed so that the output element d is produced and the output value v is known. When any operation is executed, the process moves from one state to another.

Property 2: All the basic data elements have as input the empty set. Therefore, at the start of the decision process, the only enabled operations are the ones producing the basic data elements. Executing such operations may be seen as the stage in any decision process in which the decision maker finds out the specific data.

Definition 5: A *final derived data element* is a data item that is not used as input in any operation. We assume that it is a criterion for performing the choice and therefore, given a rational decision maker, it is a sub-goal of the decision process.

Definition 6: A *state of a DDM* is a particular distribution of operations over the sets of Enabled, Not-Enabled and Executed operations. For example, the initial state is the one in which all the operations that produce the basic data elements are enabled, all the operations producing derived data elements are not-enabled, and there is no executed operation (e.g. for the DDM in Fig. 2, Enabled = {opA, opB, opC, opD, opE, opF}, Not-Enabled = {op1, op2, op3, op4, op5, op6, op7, op8} and Executed = ø). As the process progresses, the different states are represented by different placements of the operations in the three sets. The end state is reached when all operations are placed in Executed set, while Enabled and Not-Enabled are empty. There is no need to reach the end state in order to make the decision.

3 The Recommendation Algorithms

This section introduces the algorithms that produce recommendations for a decision process, based on a DDM. We aim to provide the answer to one question: "Which is the action that the decision maker should do next, given his previous actions?". Therefore, the problem we are faced with can be stated such as: "How to select the best operation to be performed next (from a set of enabled operations), in any given state of the process?". As explained before, the state of the process is a certain distribution of the DDM operations over the Enabled, Not-Enabled and Executed sets. The decision maker can choose to execute any operation from the Enabled set, or may perform any new operation that is not in the model (as long as the data needed as input for that operation is available). If an Enabled operation in the DDM is executed, it is moved to the set of Executed operations. After the decision maker performs an operation (the suggested one or any other), the process moves to a new state and the system provides another recommendation.

We assume that:

- each operation can be executed only once (since it is useless to calculate something once you know its value);
- each operation can only be executed successfully;
- the more frequent an operation is performed, the more important it is. Therefore the primary goal of any algorithm should be to identify the most frequent path in the model.

There are several possible approaches over DDM based recommendations, according to the underlying assumptions. For example, we can either assume the decision process has memory or is memory less (e.g. moving from one state to the next one depends on all previous states or only on the current state). Or, we can assume the process can take infinite time or is restricted to a given time horizon. Therefore, we developed several algorithms. Given the space limits we can introduce only two in this paper, each looking at the problem from two perspectives. The naïve one suggests the next operation by considering the absolute frequency. It has no clear target, and only aims to guide the user through the most frequent operations. The second algorithm assigns priorities to operations producing a final derived data element (which are actually the decision

criterions). Then, it guides the user along a path so that the operation producing a certain final data element is reached at a minimal cost.

Algorithm 1: We first introduce a naive algorithm which uses a Greedy approach, recommending the most frequent operations that is enabled.

1. Let $DDM_{agg} = (D_{agg}, O_{agg})$;
2. Let *op* be the list with the operations in O_{agg};
3. Let *no_of_occurences* be the list with the number of occurences for each *op*;
4. Select *op* with max(*no_of_occurences*) and place it in *Max_Occ* set;
5. Compute *Enabled* and *Executed* sets;
6. For each in *Executed* set, search for mutually exclusive operation. If found, move them from *Enabled* set to *Executed* set;
7. Compute *Recommendation* = *Max_Occ* ∩ *Enabled*

Table 1. Example of recommendation for the running example using Algorithm 1

State	Enabled	Not enabled	Max Occ (frequency)	Recomm	Executed
1	opA, opB, opC, opD, opE, opF	op1, op2, op3, op4, op5, op6, op7, op8	opB (120)	opB	Ø
8	opE, op3, op5, op8	op4, op6, op7	op5 (50)	op5	opB, opC, opD, op2, opA, op1, opF
9	opE, op4, op5, op8	op6, op7	op5 (50)	op5	opB, opC, opD, op2, opA, op1, opF, op3
10	opE, op4	op6	op5 (50)	op5	opB, opC, opD, op2, opA, op1, opF, op3, op8, op7
12	op6, op4	ø	op6 (15)	op6	opB, opC, opD, op2, opA, op1, opF, op3, op8, op7, op5, opE,

In the initial state none of the data items is known and the Enabled set contains the operations that produce the basic data items, while all the operations producing derived data items are Not-Enabled. In this initial state, the most frequent enabled operation (opB) is recommended. Later in the decision process (State 8), op5 is recommended but user decides to perform op3 (see last item in Executed set in State 9). The system again recommends op5 since it is still the most frequent one enabled. In State 9, again the user ignores the recommendation and performs op8. Performing op7 becomes pointless, so it is moved to Executed set (see last two items in Executed set in State 10), even if it wasn't enabled. The reason is that the user already found out the value of the derived data item out7, therefore there is no need to calculate it again. The user decides to make the decision without performing all the operations (see State 12 where there are still two Enabled operations, while the performed ones are logged in Executed set). The DDM will be updated according to the new trace, so the frequency for some operations increases by 1, while for others remains the same.

Algorithm 2: This approach is inspired from the A* path finding algorithm [2].

1. Create array *Final* with the operations that output final data elements (*fo*) and their frequency (f_{fo});
2. Use depth-first search to calculate the direct paths to each element in *Final* and place them in *Paths*;

3. Evaluate each in *Paths* using formula $F_i = (G + H)$ where F is the score of each path, G is the total individual cost of the operations executed in the prior states of the process and H is the total cost of the remaining operations along the selected path. The cost of an operation is calculated as the sum of the frequencies of all operations divided to the frequency of that operation;
4. *Current path* = the element from *Paths* where F_i is minimal;
5. *Recommendation* = max frequency (*Enabled* ∩ *Current Path)*;
6. If *Recommendation* = ø
   End
   Else Compute New State and go to step 3.

In Fig. 2, the ranked list of pairs comprising operations leading to final derived data and their frequencies is <(Op8, 30), (Op7, 20), (Op6, 15), (Op4, 10)>. There is only one path that leads to enabling op8 and can be determined using depth-search as {op2, opC, opD, opA, opB, opF}. There is also only one path to op6 which is {op5, op1, opA, opB, op2, opC, opD, opE}. In the initial state (there are no operations in Executed) the first objective would be to perform op8 because F_{op8} = (0 + 81.34) while F_{op6} = (0 + 152.75). In Table 3 we show State 6 in which F_{op8} = (50.09 + 42.66) = 92.75. The total score increased because the user decided to perform op1 which is outside the path to op8. However, F_{op6} is still 152.75 because the executed op1 was in its path. In State 8 the objective has changed because F_{op6} = (99.41 + 53.33) and the user only needs to perform op6. Meanwhile, to perform op8 the user needs to perform first opF and then op8 at a higher total cost.

Table 2. Example of recommendation for the running example using Algorithm 2

State	Enabled	Not enabled	Objective (score)	Reco mm	Executed
6	opE, opF, op3, op5	op4, op6, op7, op8	op8 (92.75)	opF	opA, opB, opC, opD, op1, op2
8	opF, op3, op6	op4, op7, op8	op6 (152.75)	op6	opA, opB, opC, opD, op1, op2, opE, op5

One can notice that there is a potential problem with the Greedy approach (Algorithm 1). It can get stuck in providing the same recommendation over and over if there is a high frequency operation that is repeatedly ignored by the user. The second algorithm adapts itself to the decision process by changing the objective if a path with a lower cost is available to a final derived data element (decision criterion) when the user repeatedly ignores the recommendation.

4 Case Study

This section aims to provide a comparative evaluation of the algorithms introduced in the previous section. The evaluation uses the decision-aware implementation we created (available at www.edirector.ro/v3_1 and accessible with username and password "test"). The decision problem to be solved by the subjects is to decide if they buy or rent a house. Some of the data elements available are the price of the house, the savings, the monthly income, etc.

The entire log we use in this section can be downloaded from http://www.edirector.ro/v3_1/export/pm.xml. From a selection of 50 user traces we created an aggregated DDM that was used as input to the recommendation algorithms. The aggregated DDM was 'zoomed out' (i.e. we selected only elements with a frequency at least 3) to get rid of the exceptional behavior.

We set up an experiment involving a focus group of 9 decision makers (users). Since the number of subjects is small for establishing a proper scientific claim, we will replicate this experiment. The objective of the experiment is two fold:

- to find out if there is any difference between the unsupervised decision making process and the one in which some recommendations are provided;
- to find out which of the algorithms produces recommendations used more frequently by the decision makers.

The experiment is divided into two parts. First, each subject is required to make a decision given the scenario data, without any recommendation. Then, the subject is provided at each step with recommendations, as generated by each algorithm. Each usage of the software produces a trace which is actually a sequence of actions. We also logged the sequence of actions recommended by each algorithm. At the end of the experiment we conduct an interview with the subjects for a qualitative assessment of the experience with the software and the recommendations.

The measured variable relevant for the first sub-objective is the distance between traces. The metric we used for trace comparison is Jaccard index. The similarity score of two traces, A and B, score(A,B) is calculated as the number of identical operations ($A \cap B$) divided to the union of operations in the two traces ($A \cup B$). This score abstracts from the sequence of operations.

The measured variable for the second sub-objective is the recommendation number of 'hits' for each algorithm (i.e. how many times the user followed the recommendation). We cannot use precision and recall as used in information retrieval since there is no notion of 'good' or 'bad' recommendation. The 'hit' can be substituted for the notion of 'good' recommendation but we argue that the user is biased since the recommendation is disclosed before he makes his choice. Even more, the direct observations during the experiment revealed that, if the first few actions were hits, the user begins to trust the recommendation and follows it more often.

There are several risks that threaten the validity of the experiment. One concern is the quality of the aggregated DDM. It was mined mostly from the traces of master students at our business faculty. Therefore, we see it as halfway between expert and beginner. A second concern is the subjects of the experiment. We used 3 expert (i.e. advanced knowledge of the decision and lots of experience with the software), 3 intermediate (regular knowledge of the decision and some previous experience with the software) and 3 beginner subjects (regular knowledge of the decision and first time users of the software). A third concern relates to the degree of the software's interface influence over the subjects. It can be easily observed that most of the traces start by evaluating the purchasing data simply because this is the first tab. Even more, the layout of the textboxes is important (e.g. the majority of traces start with op1 and then follows op2 because these are the first textboxes of the first tab). To mitigate this risk, for the next experiments we will change the order of the tabs and also reorder the textboxes in each tab. And finally, since the number of subjects in not statistically

relevant, we used the simple average. As the experiment will be repeated and more data gathered, we will employ more advanced metrics.

Because of the limited space, in Table 4 we show just one trace for each category of subjects. From the log introduced at the beginning of the section, the unsupervised (supervised) traces for experts are 181 (195), 183 (194), 217 (218), for intermediates are 197 (198), 201 (207), 213 (215), and for beginners are 185 (187), 190 (192), 211 (214).

Table 3. Traces for an expert (user 1), intermediate (user 2) and beginner (user 3)

User	Unsupervised trace	Supervised trace	Recommendation A1	Recommendation. A2
1	op1, op2, op29, op3, op4, op5, op50, op14, op6, op10, op16, op51	op1, op2, op14, op29, op5, op3, op4, op52, op6, op10, op16, op15, op53, op54, op55	op1, op2, op6, op6, op6, op6, op6, op6, op6,op11,op11, op11, op11, op11, op11	op14, op14, op14, op29, op5, op7, op27, op27, op27, op10, op16, op15, op7, op7, op7
2	op1, op2, op26, op28, op63,op7, op24, op64	op1, op2, op29, op5, op3, op4, op6	op1, op2, op6, op6, op6, op6, op6	op14, op14, op29, op5, op7, op7, op27
3	op2, op65, op66, op67, op68, op14	op1, op2, op14, op29, op5, op7, op24, op13, op3, op4, op27, op26, op6, op11, op5, op20	op1, op2, op6, op6, op6, op6, op6, op6, op6, op6, op6, op11, op5, op20	op14, op14, op14, op29, op5, op7, op24, op13, op3, op4, op27, op26, op28, op28, op10, op28

Examining the data in Table 4 one can notice the main limitation of Algorithm 1 concerning the re-occurrence of the same recommendation. On the other hand, Algorithm 2 adapts more to the actual process being performed.

The Jaccard distance calculated between the unsupervised and supervised traces for user 1 is 0.69. This reveals that the user changed his process because of the recommendations. The number of hits for Algorithm 1 is 6 while the number of hits for Algorithm 2 is 12, therefore we can argue that the user found the recommendations of Algorithm 2 more useful.

From the first chart in Fig. 3 we can answer the first research question and safely argue that the recommendation had a large impact on the decision process for all types of users. The average change is close to 40% for the experts and it is dramatic for the intermediates and the beginners (more than 80% of the actions were different).

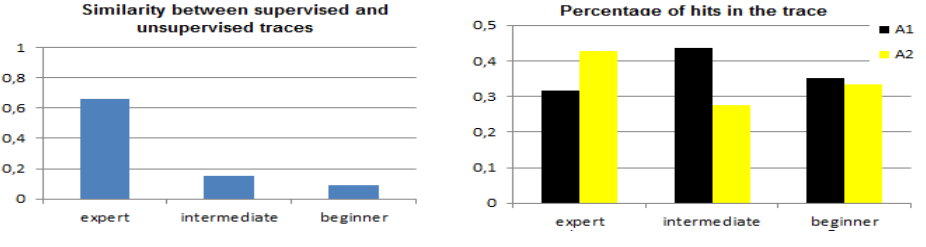

Fig. 3. Experiment results

Considering the second chart in Fig. 3 we see that the experts preferred the recommendations of Algorithm 2 (more than 40% of the operations performed were the ones recommended by A2). Interestingly, the situation is inversed for the intermediates. The beginners seem to prefer both recommendation algorithms. But,

the main point of this chart is that none of the processes follows closely one recommendation. We conclude that the subjects did use the recommendations, but didn't rely only on it. This chart doesn't provide a definitive answer to the second research question. It seems that the adaptive Algorithm 2 follows more closely the more structured process of experts. Meanwhile, the 'dumber' Algorithm 1 seems to work better with individuals that have some insight into the decision process but not a clearly structured process in mind. We will further investigate this issue.

5 Related Work

The work introduced in this paper may seem somewhat similar to providing recommendations in web-based systems. There, too, is a large log with user actions (e.g. what items were purchased) based on which some recommendations are provided. But the fundamental approach focuses on the associations between properties of items. We do not focus on the choice itself but on the reasoning process that leads to that choice. Therefore, the work done in content-based or collaborative filtering poorly fits our needs.

From artificial intelligence field there are many algorithms that search graphs [2]. We found that A* algorithm fits our problem and adapted it for use in Algorithm 2.

The decision processes can be represented, and recommendations can be produced, using some executable workflow formalism (e.g. Petri Nets, BPEL) [3]. But, in those approaches, adding behavior to a model is not a trivial problem. Our entire approach has at its core the idea of extracting a model from a large number of individuals. Therefore, as new individuals use the software, new behavior is logged all the time. A DDM is easily and fast updated [1], therefore the recommendation algorithms performance is not reduced. Even more, the workflow approaches focus mostly on the control flow perspective [3]. Instead, we focus on the data flow perspective aimed at producing a choice. Considering this focus, we found our inspiration in the Product Based Workflow Modeling approach [4]. The approach has at its core a model called Product Data Model (PDM). The DDM is derived from the PDM, having some specific features and properties.

Given a PDM, providing recommendations of the next action to be taken can be done using Markov Decision Process (MDP) approach [5]. It is suggested that the MDP-based recommendation is difficult to apply for real situations because of the State Explosion problem. It is argued that it can be partly avoided by applying heuristics (i.e. by replacing the MDP global choice with local choices). A DDM-specific property is that an operation can only be executed successfully, and the order of operations for basic data is irrelevant. Therefore, the state space stays small.

Applying decision trees to our approach is unfit since we look at decision making as at a workflow and we don't aim to classify the actions of the user. Even more, applying decision trees to such an approach would lead to a state explosion problem with respect to the size of the tree.

Our evaluation of the algorithms was inspired by [6]. The offline analysis is suggested to be used mainly in evaluating predictive accuracy. Our goal is not to predict what a certain decision maker would do next. On the other hand, live experiments fit reactive recommendation systems. Since all our recommender algorithms are reactive, we used a focus-group based experiment.

6 Conclusions

The research presented in this paper is placed in a data-centric business decision making context. The basic aim of this paper is to introduce several algorithms that allow an individual to perform a guided walk through a particular type of acyclic graph. The graph is extracted from a log containing the activities of large numbers of persons performed while making a specific decision.

We created two algorithms: one that uses a Greedy approach and one that looks at the previous actions in the process and determines the best path towards a sub-objective of the process (a decision criterion).

Our evaluation shows that providing a recommendation changes the decision process for all classes of users (experts, intermediates or beginners). However, our experiment didn't allow us to state, at this point, which of the two algorithms is better.

We are aware that we didn't cover all possible types of algorithms. For example, we are developing a mapping of our problem to a Markov Decision Process since it may produce better results than the algorithms presented in this paper.

The future work will also aim to create clusters of similar DDM and dynamically place the user in such a cluster. The recommendation algorithms will be applied on the clustered aggregated DDM rather than on the entire aggregated DDM. Therefore, we will be able to provide better context-aware recommendations.

Acknowledgments. This research was supported by Human Resources Development Operational Program through the project Transnational Network of Integrated Postdoctoral Research in the Field of Science Comunication, Capacity Building (Post-doctoral School) and Scholarship Program (CommScie) POSDRU/89/1.5/S/63663.

References

1. Petrusel, R., Vanderfeesten, I., Dolean, C.C., Mican, D.: Making Decision Process Knowledge Explicit Using the Decision Data Model. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 172–184. Springer, Heidelberg (2011)
2. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (2003)
3. van der Aalst, W.M.P.: Process Mining. Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
4. Reijers, H.A., Limam, S., van der Aalst, W.M.P.: Product-based Workflow Design. J. of Management Information Systems 20, 229–262 (2003)
5. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product-based Workflow Support. J. Information Systems 36, 517–535 (2011)
6. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22, 5–53 (2004)