A social norms based approach for enhancing decision support in business process execution

Thomas Keller
Institute of Business Information Systems
Zurich University of Applied Sciences
Winterthur, Switzerland

Bastin Tony Roy Savarimuthu
Department of Information Science
University of Otago
Dunedin, New Zealand

Abstract

Decision making in complex and volatile business environments is still the realm of human decision makers since their knowledge and experience are paramount in making good decisions. The standard run-of-the-mill automation approaches for making decisions do not work well in these contexts. This paper investigates the use of a socially-inspired norm inference mechanism proposed in the area of multi-agent systems to assist human users to make decisions. Using simulation results the paper demonstrates that using the norm-based mechanism results in reduced failure rates in the decision making of a knowledge worker while still providing maximum flexibility for the user to choose from a range of actions to execute.

Keywords: operational decision support, norm inference, process mining

1 Introduction

The introduction of Business Process Management (BPM) in a medium to large enterprise is often triggered by the objectives to minimize time to market and to reduce the cost. Today's BPM suites support these objectives well. They even provide functionality for business activity monitoring [2] and enable subsequent application of statistical means to analyze optimization possibilities, for example through data min-ing techniques. However, there are two main limitations. First, the integration of optimized solutions into the system based on insights gained is often carried out offline (i.e. the integration isn't carried out on the fly at run-time). Second, optimizing busi-ness processes often consider purely quantifiable and economical facts like cost and time without taking a holistic perspective of how decisions are being made in an organization. This is particularly so for the case of human-centric business processes with human-based decisions that rely on experience and very often informal knowhow (e.g. norms). This paper aims to extract informal know-how (i.e. norms) employed during execution of business processes involving human-oriented tasks and integrating that knowledge into the system at run-time, thus addressing the two limitations.

Norms as studied in the fields of sociology and psychology are used to aid human decision making. Norms are expectations of behavior observed in human societies [16]. Norms provide guidance about how one has to behave in a certain social situa-tion (e.g. tipping in restaurants). However, there is the possibility of violation. The violation is the result of the freedom enjoyed by the individuals through their autono-my. There are different types of norms followed by societies such as prohibitions, obligations and permissions and these are often called as deontic norms [9].

In this paper, we are interested in prohibition norms because these norms specify which actions are prohibited in a society (e.g. not littering a park and not using the public transport without a ticket). The concept of prohibition norms can be applied to decision making of individuals when enacting a business process through inferring the actions that are prohibited and proposing choices that are not prohibited. It should be noted that, through this mechanism, individuals are not confined to one choice but a range of choices (each with an associated probability for the success) so that an in-formed decision can be made. For example, the user might be told that executing action Y after X produces faults 90% of the time and that choices that other users have made previously are A, B or C which have a lower error rates associated with them. Additionally, the user is allowed to make an informed choice (i.e. they can still choose action Y if they think it is the right action to execute under prevailing condi-tions) which retains their autonomy and respects their decision making capabilities. Thus, norms-based decision support systems allow flexibility to human decision mak-ers and we believe this is crucial in domains involving rapid changes in the function-ing environment.

In order to motivate how our approach might be beneficial, let us consider the emergency prioritization system in hospitals. Patients arriving in the emergency de-partment (ED) are assessed by a knowledge worker (e.g. a triage nurse) and are as-signed different priorities based on the assessment. Assessments of individual cases can be complex and often decision support systems are used for prioritization. Our approach will enhance the existing system through a) learning from history of cases available in the system by looking particularly at failed instances and b) recommend-ing the most suitable option (path) for treating a patient in question.

Assigning priorities to individual cases is a complex task. Different knowledge workers might assign slightly different priorities to different cases (e.g. case A waits 2 hours and case B waits 4 hours). Feedback on whether the knowledge worker made a good decision is usually available from the ED doctor who treats the patient. So, for each case a set of attributes for a particular case (e.g. arrival time, type of care provided while waiting, total waiting time, complexity of the case) and the feedback (e.g. positive or negative feedback from ED doctor about the appropriateness of assigned priority, eventual outcome of the treatment) are known. Using attributes of cases and feedback information from a large number of cases, we can infer what the norms are. In particular, it would be possible to infer paths that should be prohibited (e.g. waiting times for complex cases of type T must be reduced to less than 1 hour because longer waiting times have resulted in poor outcomes in 75% of the cases). We harness such information and integrate this into the system proposed in this work.

The paper is organized as follows. Sections 2 presents the background on norm inference and an overview of the related work. Section 3 presents the experimental model that is used for the simulation of a system that infers norms and uses the identified norms to reduce failures. Section 4 presents the results obtained. Section 5 presents the discussion of the contributions of the work and avenues for future work.

2 Background and related work

This section presents a background on the norm inference mechanism used to identify prohibition norms and also presents an overview of the related work.

2.1 Background on norm inference

In multi-agent systems, researchers have investigated mechanisms for norm inference based on observing actions of agents (i.e. software entities) and their interactions. A sample norm could be that a game playing agent that was helped by another agent to slay a dragon is obliged to reciprocate help when the other agent is in need. These identified norms can then form the basis of decision making (which actions are forbidden and which actions are obligated). Re-searchers have developed two norm identification algorithms to identify prohibition and obligation norms respectively [15].

The norm inference mechanism used in this paper is adapted from the prohibition norm identification algorithm [9] and has been previously employed in another busi-ness context [17]. The algorithm infers actions that are prohibited based on a sequence of actions that have been observed. Let us assume that the following are the four sequences of actions that have been executed in the context of a business process: ‘abcd’, ‘ac$’, ‘abc’, ‘abd$’. The sequences ending with a ‘$’ indicate that there has been a failure in the execution. There are two sequences with some form of failure and two sequences that do not contain failures. Based on identifying supersequences of ‘ac’ and ‘abd’ (that lead to the failures), the algorithm identifies that ‘a’ followed by ‘c’ is prohibited and ‘ab’ followed by ‘d’ is prohibited. Also, it can identify that ‘b’ is obliged after ‘a’ and also that ‘c’ is obliged after ‘ab’. The algorithm contains parameters to govern the length of the action sequences observed, the window size to infer the prohibitions (i.e. to model whether a sanction appears immediately after a wrong action is performed or does it happen after some delay). In this work, we assume that the sanction happen immediately after a wrong action because business processes often result in failure if a wrong sequence is followed.

2.2 Related work

Our work in this paper is related to works in several areas. Related work can be found in the areas of a) decision support systems, b) process mining (a specific application of data mining), and c) business process management.

Decision Support Systems (DSS) integrated in a business process management context can be found mainly in engineering design and manufacturing [8]. In this domain it has been demonstrated that, decision support and operational processes need to be properly integrated [19] to realize benefits. DSS write process related data to a knowledge base. Other applications (e.g. agents and robots), use this data to discover insights. New rules are formulated which are then fed into DSS. Sometimes, rules can be fed into a simulated system to investigate the impacts of the rule change before deploying into a real system. Our work here employs a simulated system. However, the work presented in this paper differs from a traditional DSS in that the scope comprises decisions that are not formally describable by rules and conditions and are therefore intrinsically human-based (i.e. decisions made by two humans for a similar case could be slightly different).

In less structured situations good decisions might not achieved just by applying rules. In those situations, decision makers will have to adapt old solutions which yielded successful results, to fit the new problem situations. This approach is well known as case based reasoning (CBR) [7] and has been applied in many domains including the health industry. For example in [2], chronic disease prognosis and diag-nosis is done by integrating CBR with a data mining approach. CBR in this case is used for knowledge inference, refining and sharing. An example for business process analysis in the Health Care industry is described in [12]. A general overview of clinical decision support systems in the US can be found in [10], and a specific application for a surgery theatre planning is described by [2]. The work presented in this paper is suitable to extend the solutions presented above from a human-oriented perspective.

Our work in this paper is related to the area of process mining. In this field, typical-ly, the logs produced by business information systems are provided as the input to a process mining engine. There are three goals of process mining [18]: a) discovery of processes, b) checking conformance of existing processes and c) enhancement of processes.

Business process automation as proposed in [11] aims at substituting human knowledge by artificial knowledge. Our approach here aims at supporting human decision, not to substitute it. The introduction of decision support in business process-es is also described in [20]. Domain specific rules are encoded in the system a priori. Our approach differs in that prohibition norms are inferred at run-time (i.e. they do not exist ahead of time) and are then applied. Successful business process automation relies on well-defined processes that are stable and have ideally no variations [4]. If business processes comprise ill-structured business cases, then the decisions need to be human-based. Our approach aims at supporting these decisions through inferring norms. Process simulation as described in [13] is closer to our work. In this work knowledge is extracted from log files produced by the process engine which is complemented by actual data of the respective process instance. Our work goes beyond this line of work and shows how knowledge can be inferred from failure cases to improve the process and the impact of environmental variables on the results obtained.

3 Experimental set-up

3.1 Process Model

The process model shown in Figure 1 is an abstract representation of a two stage work arrangement that is managed through a software system. Let us assume that the inputs of the system comes from three abstract input domains A, B and C and that there are three possible ways of handling each of the inputs (alternatives D, E and F). Only one of the choices will be the right choice for a given input. Typically, a knowledge worker will make choices between alternatives. Once a choice is made, another knowledge worker will evaluate the choice made. The evaluation points in the process are represented as control gates . The feedback from evaluation is then used as a basis for decision making in the future to choose between alternatives.

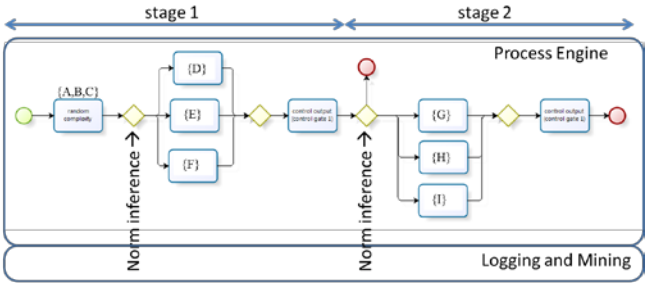

Figure. 1. Important aspects of norm inference in the context of an abstract business process

Relating to the Emergency Department example, the first stage consists of input assessment (condition of a patient brought into ED) and the selection of the best alternative by a triage nurse. A, B and C represent inputs of different complexity (i.e. three types of patients requiring prioritization). For each input, there is a choice of three alternatives in our simulation[1] (D, E or F). Consider these options to be the waiting times assigned by the triage nurse. There would be a feedback provided whether the assigned waiting time was appropriate given the condition of the patient by another knowledge worker (e.g. the ED consultant) who would assign positive or negative scores to a decision made.

For simulation purposes we replace a human decision-maker by a random action selector, which selects actions at random, initially. The choice made is then evaluated by our system using a simple look-up table that mimics the feedback from the second knowledge worker (i.e. the ED consultant). The appropriate value in the look-up table forms the feedback provided to the decision-maker[2]. The decision-maker learns about which actions not to select based on feedback obtained. Table 1 shows the assignment between input and the choice of alternative selected. Pass indicates the right choice and fail indicates a poor choice. So, combinations AD, BE and BF are considered successful (i.e. choosing D if the input is from domain A will result in a satisfactory (good) outcome).

Table 1. Success criteria for different input domains and alternatives available

		Alternatives		
		D	E	F
Input domain	A	Pass	Fail	Fail
	B	Fail	Pass	Fail
	C	Fail	Fail	Pass

To allow for a realistic simulation the assignments are enhanced with a variable for error probability. This variable allows for erroneous assignments, e.g. AD is taken as 'Fail' instead of 'Pass'. For example, if the probability of AD to be wrongly assigned to a failure is set to 0.05 then it gets wrongly assigned 5% of the time. This is to allow for errors (or noise) like social feedback systems typically exhibit.

The subsequent stage (stage 2) of the work arrangement in Figure 1 is only executed for successful cases. The second stage consists again of three alternatives, G, H and I. One alternative is considered the most adequate (as defined by the setup in Table 1 where the output columns are changed from D, E and F to G, H and I while the inputs remain the same). The second stage models what might happen once the patient is moved from the waiting area and is now being treated by the ED consultant. We note that there could be many levels of this process and we limit ourselves to the two-level process in this paper. Similar to stage 1, the human assessor who chooses amongst the alternatives is again replaced by a pure random action selector in our simulation which learns

[1] However, in theory, there are no limitations in the number of alternatives.

[2] The feedback is based on human judgement. Hence, there may be inconsistencies across different consultants.

from mistakes (i.e. learns to avoid prohibited actions). Also, a second control gate is used for evaluation purposes.

The description above details how a particular case is instantiated (i.e. pathway for a case is worked out such as paths ADG being taken in say, case 1). Let us now imagine that information about several process instances has been recorded and this information is available in the form of a log (typically stored in a database). This log forms the basis of the process mining phase, i.e. prohibition norm inference. Figure 1 illustrates the setup graphically where the second phase is represented as Logging and Mining phase. Norm inference happens at two points in the abstract process given in Figure 1 (indicated in the figure by diamonds). When a new instance arrives (say a task of type A arrives), the next task should be assigned. Before assigning the next task, the list of prohibitions are checked to see which out of actions D, E and F are prohibited. If an action is prohibited, then it would not be considered for execution based on certain probability. Thus, the prohibition information is considered during the evaluation of which action to pursue next during run-time. This results in minimizing errors in the system.

3.2 Hypotheses

From Table 1 it can be inferred that the probability of success for stages 1 and 2 are 1/3 and 1/3 respectively and the overall probability of success of stages 1 and 2 combined is 1/9 (i.e. if all possible pathways (all 9) in Figure 2 are generated exactly once, only one of them will result in success). The probability of failure will be 8/9 and this is referred to as baseline failure. Below, we present three hypotheses pertaining to different approaches to the reduction of failure rates using prohibition norm inference mechanism.

We argue now that it should be possible to minimize the failure rate to a lower value than the baseline by analyzing the history of failure cases with the norm inference algorithm **[Hypothesis 1]**. The inferred prohibitions are then applied as a constraint to both the gateways as shown in Figure 2 (i.e. prohibited actions will be discouraged). Hence, the failure rate will decrease. The failure will decrease to a value of zero as soon as all possible failure cases have been identified.

There are a couple of limitations to the simple case presented above. First, it considers the entire history of failure cases which can be quite large. This is a drawback because any wrongly assigned failures/successes (i.e. successes labeled as failures and vice versa) will be considered indefinitely (i.e. not eliminated). Second, changes in the business environment are marginalized since any historic data is weighed the same[3]. To overcome these problems, we introduce a controller with the length of historic data as the parameter. There are two variables of interest that get captured, the current failure rate in the system (as-is failure rate) and the desired failure rate (to-be failure rate). The desired failure rate is the rate that the system administrator would lie to achieve in the context of business process execution. The difference of the as-is and to-be (i.e. desired) failure rates is the input to the controller. Note that the failure rate is defined as the ratio of number of failure cases and the total number of cases considered. We argue that the length of historic data is inversely proportional to the failure rate in the system **[Hypothesis 2]**. That is to say that larger history length will result in fewer errors in the system.

There are still a couple of assumptions in the system that needs to be relaxed to make the system more realistic: (i) it is assumed that all prohibition norms are respected by the knowledge worker (i.e. there is no violation of norms) and (ii) the business environment is stable (i.e. business environment and the norms are static). The first assumption can be relaxed by assuming that the system can set a threshold which governs when the norm inference is triggered. For example, if the threshold is set to 5, this means that the system will avoid those prohibited actions that were observed at least five times. The second assumption can be relaxed by introducing an uncertainty parameter which models the uncertainty in the system (i.e. noise in the system) which takes into account wrongly assigned labels for failures/success cases. We hypothesize that relaxing the assumptions will still result in the reduction of overall failure rates in the system when compared to the baseline **[Hypothesis 3]**.

3.3 Simulation setting and parameters

The process model shown above is directly coded using Java for the simulation purposes. The controller for the time window considered, i.e., the length of the log, is implemented as a type of bang-bang controller [[1]]. The window size comes with a lower and upper boundary. The controller changes the number of event sequences considered in the log by +1 or -1 based on the following conditions:

1. Increase time window for inferring norms by 1 (i.e. how many event sequences to consider) if as-is failure rate is higher than the to-be failure rate and the change of the as-is failure rate is positive.
2. Decrease time window by 1 if as-is failure rate is lower than to-be failure rate and change of the as-is failure rate is not positive.

Other parameters used in the system are the norm threshold and the uncertainty. These two variables are explained in detail in Section 5.3.

4 Experimental results

We conducted three experiments to test the three hypotheses. *The results* are presented in the form of graphs and summary tables. All the graphs show on the x-axis the number of iterations of the simulation. The y-axis red line

[3] Using this approach, recent changes do not get prioritized. For example, if a consultant provides the feedback that AE is beneficial (instead of AD), if the whole history is considered, there might 100 instances of AD being beneficial, but only one instance of AE would be present. Hence, it would take a long time before the count of AE exceeds that of AD.

shows the failure rate at each iteration (a value between 0 and 1).The y-axis for the blue line indicates the size of the considered time window (i.e., the number of instances in the log, based on the upper bound of the time window at each iteration[4]).

4.1 Experiment 1 – Considering the full history evidence

In this experiment, all executed paths in the log are considered in order to identify prohibited actions (i.e. full history is chosen as the evidence). It can be observed from Figure 3 that as the number of iterations increase, the number of entries in the log increases (i.e. size of time window increases). Because of this, all possible prohibited actions can be identified. The red line in Figure 3 shows that as the simulation progresses, the failure rate comes down because all those prohibited actions will not be chosen by the decision-maker. Thus, hypothesis 1 is verified from Figure 2. It also confirms that our simulation behaves correctly for a well-known case.

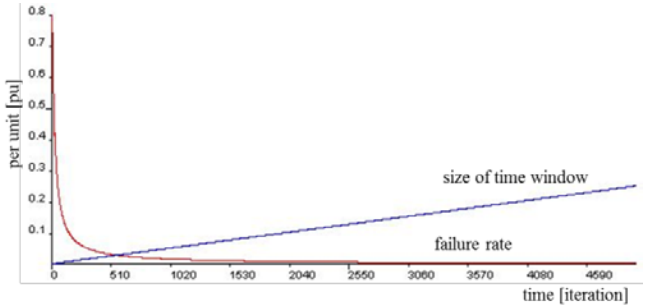

Figure. 2. Log history length is inversely proportional to failure rates

4.2 Experiment 2 – Considering a limited-range of history

The objective of this experiment is to test hypothesis 2. In this experiment, the set-up is analogous to the scenario presented above, but with a small change. The controller is in charge of the time window (as opposed to a infinitely growing window in the previous experiment). The resulting length of the time window is around 100 (lower bound of 10 and upper bound equals 200) when the to-be failure rate is set to 0.1. It can be observed from Figure 4 that there is a cycle of ups and downs in the window size length in blue. This is because once the as-is failure rate's value goes below 0.1 (i.e. the goal is achieved), the window size length is decreased by the controller and hence at some point the value of to-be failure rate increases above 0.1 at which point the controller increases the window size. Because the controller is a bang-bang controller, this pattern repeats, hence, we get a serrated pattern in the blue line.

[4] According following formula:
$y = \frac{number\ of\ instances\ in\ the\ log}{upper\ bound\ of\ the\ time\ window}$

Reducing the to-be failure rate to 0.05 leads to a larger time window of around 200 (upper bound equals 300) as shown in Figure 3. It takes a larger number of observations (i.e. larger window size) to identify prohibitions more precisely because of the reduced failure rate which confirms hypothesis 2. Though the findings so far have been intuitive, they are helpful to demonstrate that the simulation setup and the implementation of the norm inference algorithm work as they should.

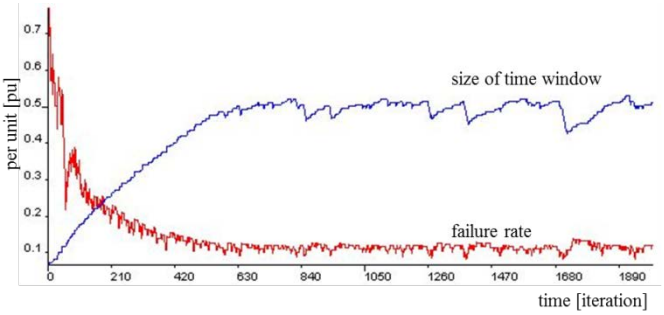

Figure. 3. Simulation results for a target failure rate of 0.1

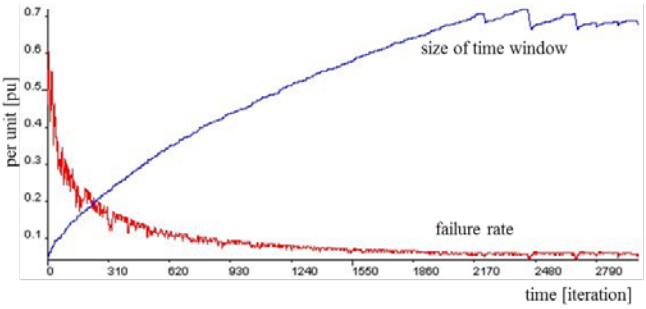

Figure. 4. Simulation results for a target failure rate of 0.05

Table 2 summarizes the resulting time window by varying the target (i.e. to-be) failure rates. It shows that if the failure rate should be decreased in the system, then the time window of the history of cases observed should increase (also verifying hypothesis 2).

Table 2. Window sizes for target failure rates

Target failure rate	Time window
0.05	200
0.1	100
0.15	60
0.2	45

An interesting output is the prohibition norms which are identified for each instance (iteration). The following excerpt shows the results from the early part of a simulation. Output in line 1 shows that the system has identified 5 prohibited sequences. It is interesting to observe that after each iteration new norms may be inferred and added to the existing ones. For example, in iteration 2, one new norm is added (ADI). However, inferred norms may also be eliminated because of

a failure case sliding out of the time window (not shown here).

1. [AE, AF, BD, BF, CE]
2. [ADI, AE, AF, BD, BF, CE]
3. [ADI, AE, AF, BF, CE, CFH]
4. [ADI, AE, AF, BF, CE, CFH]
5. [ADI, AE, AF, BEI, BF,CE,CFH]
6. [ADI, AE, AF, BEI, BF, CE, CFH]
7. [ADI,AE,AF, BEG, BEI,BF,CE,CFH]
8. [ADH,ADI,AE,AF,BEG,BEI,BF,CE,CFH]
9. [ADH,ADI,AE,AF,BEG,BEI,BF,CE,CFH]

4.3 Experiment 3 – Considering uncertainty

Two parameters related to ambiguities which were not considered in the previous experiments are considered here. First, previous experiments assume that there is no noise in the feedback (i.e. no noise in labeling an action sequence correctly into a prohibited sequence). The algorithm in [[17]] inspects failure cases and infers the prohibition norms. It does not consider the notion of erroneous failure cases (i.e. a case where a failure case is erroneously considered as a non-failure). Considering this adds noise to the prohibition norm inference algorithm. In order to accommodate this, we consider a parameter that defines a threshold called *norm threshold* that captures the minimal rate of occurrence for a failure case to be considered significant (i.e. the failure rate crosses a threshold). Second, we model noise by introducing a probability of an action sequence being incorrectly labeled called *uncertainty*. Uncertainty is defined as the probability that a failure case is accidentally treated as a success case or a success case is accidentally treated as a failure. Hence, an uncertainty of 0.1 means that in average every tenth failure/success case is treated erroneously. This behavior models social feedback characteristics where the feedback is noisy. Below, we present the results of two experiments conducted to investigate the impact of these two variables.

Impact of norm threshold: Our implementation takes an absolute number as an input where norm threshold is the count of number of failure cases. Once the failed sequence happens certain number of times, then the norm inference mechanism will identify the prohibited actions and use this in the feedback for the forthcoming iterations.

Table 3 shows the results of changes to the time window as inferred by the controller when the norm threshold parameter is introduced. The threshold for the minimal rate of occurrence of a failure case is set to 1, 2 and 4 respectively.

Table 3. Time window ranges based on varying target failure rates and norm thresholds

Test case	Target failure rate	Norm threshold	Time window
1	0.05	1	200
2	0.05	2	325
3	0.05	4	680
4	0.1	1	100
5	0.1	2	160
6	0.1	4	340
7	0.15	1	60
8	0.15	2	95
9	0.15	4	230
10	0.2	1	45
11	0.2	2	70
12	0.2	4	160

An observation of the data presented in Table 3 is that the higher the threshold for the number of occurrences, the longer is the window size. For example, when the norm threshold is 4, it takes almost thrice the size of the window when the norm threshold is 1 for the same target failure rates to be achieved (compare test cases 1 and 3). The explanation of this phenomenon is that, since a failure needs to occur more than once, a longer time period needs to be considered for larger thresholds. Alternatively, it can be said that, if the time window remains the same and the norm thresholds were varied, then the to-be failure rates (target failure rates) reached for higher norm thresholds will be higher (compare test cases 1 and 4). A look at the evolution of prohibition norms for test case 12 shows this clearly (given in the excerpt below).

1.[CD]
2.[AF, CD]
3.[AF, CD]
4.[AF, CD]
5.[AF, CD]
6.[AF, CD, CE]
7.[AF, CD, CE]
8.[AF, CD, CE]
9.[AF, CD, CE]
10.[AF, CD, CE]
11.[AF, CD, CE]
12.[AE, AF, CD, CE]
13.[AE, AF, CD, CE]
14.[AE, AF, CD, CE, CFH]

Compared to the previous excerpt the inference takes longer (i.e. fewer norms are identified early on) because of higher failure rates (0.2 in this case), however, the identified norms tend to be more stable.

Impact of uncertainty: Table 4 shows the effect of the uncertainty parameter. We conducted several experiments to study the effect of uncertainty on the failure rates achieved. Two different fixed lengths of time window were generated (200 and 400). Also the norm threshold was varied (1, 2 and 4).

The first six experimental data sets contain the well-known case with no uncertainty. The second six and the third six experiments had different target failure rates (0.05 and 0.1 respectively). It can be observed from the results shown in the first six rows of Table 4 that the behavior is consistent with results shown in Table 3. The failure rates decrease upon increasing time windows and norm thresholds. However, as soon as uncertainty is introduced the resulting failure rates deteriorate (i.e. they increase). This can be inferred in experiments 7-18. Time window does not have a significant impact because the same failure rates are

observed for varied time windows under same uncertainty (see results of experiments 9 and 10). Also, norm thresholds by themselves do not have a significant impact because different norm thresholds under same uncertainty result in same results (see results of experiments 13 and 14). Also, it is interesting to observe that even a small increase in uncertainty leads to significantly poor results. Moving the value of uncertainty from 0.05 to 0.1 in experiments 7 and 13 increases the failure rates from 15% to 40%.

Table 4. Time window ranges based on varying target failure rates and norm threshold

Exp. No	Uncer-tainty	Norm threshold	Time window	Failure rate (avg)
1	0	1	200	0.06
2	0	1	400	0.03
3	0	2	200	0.09
4	0	2	400	0.0425
5	0	4	200	0.18
6	0	4	400	0.0875
7	0.05	1	200	~0.15
8	0.05	1	400	~0.3
9	0.05	2	200	~0.3
10	0.05	2	400	~0.3
11	0.05	4	200	~0.4
12	0.05	4	400	~0.4
13	0.1	1	200	~0.4
14	0.1	1	400	~0.4
15	0.1	2	200	~0.4
16	0.1	2	400	~0.5
17	0.1	4	200	~0.5
18	0.1	4	400	~0.6

However, we note that the resulting failure rates are still lower than the baseline value of 8/9 (i.e. 0.89) that does not take into account the feedback based on inferring prohibition norms. Hence, hypothesis 3 which states that relaxing the strict assumptions still results in the reduction of failure rates, holds.

4.4 Remarks on experiments

The results presented above show the positive effect of decreased failure rates in the system when introducing the norm inference mechanism. Starting with a simplistic case (experiment 1), we have demonstrated how complexity can be introduced and its impact on the results obtained in experiments 2 and 3. We believe our abstract process model matches many knowledge intensive business processes where decisions need to be made based on information acquired by antecedent process steps. The knowledge worker, e.g., the assessor at the control gate in our abstract example, generally knows her business and can based on her experience make very accurate decisions. In the real-world use of our system, the random generator will be replaced with a knowledge worker and the feedback provided by our system will be used by the human user. When it comes to a real-life production system incorporating our approach, we expect that the number of errors in the system, to start might be lower since humans do not make decisions at random. However, this does not introduce any change to the mechanism we have proposed hence it assumes a pathologically extreme scenario of random decision making and even in that case we have demonstrated how our system is able to bring down the failure rate. Hence, we believe the proposed approach should lead to better decisions. Overall, we believe that social norm inference provides a practical approach without introducing overwhelming complexity.

5 Discussion

This paper has shown how norm inference can be applied to provide decision support in a business process context. We consider the transfer of norm inference, traditionally used in agent-based systems, to a process mining and decision support context as a conceptual contribution of this paper. At a conceptual level the challenges tackled by norm inference in both the fields are the same.

Based on analyzing executed business logs, often suggestions on actions to be performed are made by human decision makers. These actions can be considered as obligations. Obligation norms are not considered in this work as they represent paternalism and will eventually lead to frustration and demotivation [5] depending on the cultural background. So, in this paper, we proposed the employment of a prohibition norm based mechanism where prohibitions are identified in the context of business process execution sequences and those norms are provided as recommendations. This provides flexibility to the human decision makers to make informed choices (i.e. they know not only what is not allowed, but also the consequences).

The simulation results in general verify our three hypotheses. However, there are several areas for improvement. The approach demonstrated solely relies on prohibition norm inference by considering the failure cases. This approach will be improved in a next step by considering the success cases as well and combining both approaches (incorporating prohibitions jointly with obligations instead of treating each one in silo). The encoding of the norm threshold can be more sophisticated. For example, options to consider are a) using punishment probability for violations for each type of failure and b) considering a domain-specific set of the most common failure cases.

We acknowledge that the simulation setup and its assumptions are limiting, but are not over simplistic. The introduction of uncertainty effectively models human behavior (i.e., social feedback that includes noise) without making the whole setup too complex. The modeling of the decision maker, e.g., the assessor at the control gates, as a random generator will need to be replaced with a more human-like model. However, in our simulation, the random generator is a learning agent which learns which actions to execute based on the outputs from the norm inference system.

We note that the results reported in the paper are our initial efforts in this area. We plan to conduct parameter analysis by varying the range of input variables to

systematically study the effects of output variables (i.e. use a parameter sweeping technique). We also plan to employ statistical tests to validate our results. We believe our controller will benefit from not one variable (window size at present), but a range of variables (e.g. norm threshold, domain-specific failure probability) to study the outputs.

We believe our work here has a range of real world applications. For example, the norm inference module can easily be integrated into existing BPM suites, where human decision making capability can be assisted through making the reasons for failures apparent (i.e. if you choose actions AEF to complete a process, then the chances of failure is 75% for the complex case that is being dealt with currently). This can be easily achieved in suites that already support business activity monitoring since the feedback loop is already built-in and our module based on norm inference can be easily added. Apart from this integration, additions to BPM modeling components can be made. For example, an additional gateway type in BPMN can be introduced indicating the consideration of social feedback arising from norms or a special stereotype can be introduced for modeling a task that integrates normative feedback.

5.1.1.1.1 REFERENCES

[1] Artstein, Zvi, "Discrete and continuous bang-bang and facial spaces, or: Look for the extreme points". 1980, SIAM Review 22 (2). pp. 172–185

[2] Dahanayake, A., Welke, A. J., Cavalheiro, G., Improving the understanding of BAM technology for Real-Time Decision Support

[3] Gomes, C., Sperandio, F., Borges, J., Almada-Lobo, B., Brito, A., A Decision Support System for Surgery Theatre Scheduling Problems, ENTERprise Information Systems, Communications in Computer and Information Science Volume 221, 2011, pp 213-222

[4] Grünert, D., Brucker-Kley, E., Keller, Th. (2014). oBPM – An Opportunistic Approach to Business Process Modeling and Execution. International Workshop Series on Business Process Management and Social Software

[5] Hofstede, G. (2011). Dimensionalizing Cultures: The Hofstede Model in Context. Online Readings in Psychology and Culture, 2(1)

[6] Huang MJ, Chen MY, Lee SC (2007) Integrating data mining with case based reasoning for chronic disease prognosis. Expert Syst Appl 32:856–867

[7] Kolodner J (1993) Case-based reasoning. Morgan Kaufmann Inc., California

[8] Liu, S., Duffy, A. H. B., Whitfield, R. I., Integration of decision support systems to improve decision support performance, Knowledge and Information Systems, March 2010, Volume 22, Issue 3, pp 261-286

[9] Meyer, J-JC, and Roel J. Wieringa. "Deontic logic: A concise overview." (1993).

[10] Musen, M.A., Middleton, B., Greenes, R.A., Clinical Decision-Support Systems, Biomedical Informatics, 2014, pp 643-674

[11] Niedermann, F., Maier, B., Radeschütz, S., Schwarz, H., Automated Process Decision Making Based on Integrated Source Data, Lecture Notes in Business Information Processing Volume 87, 2011, pp 160-171

[12] Rebuge, A., Ferreira, D.R., Business process analysis in healthcare environments: A methodology based on process mining, Information Systems, Volume 37, Issue 2, April 2012, Pages 99-116

[13] Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J., Workflow simulation for operational decision support, Data & Knowledge Engineering, Volume 68, Issue 9, September 2009, Pages 834-850

[14] Savarimuthu, B.T.R., Cranefield, S., Purvis, M.A and Purvis, M.K, "Identifying prohibition norms in agent societies", Artificial Intelligence and Law, Volume 21, Issue 1, pp 1-46, 2013.

[15] Savarimuthu, B.T.R, Cranefield, C., Purvis, M.A and Purvis, M.K,. "Obligation norm identification in agent societies." Journal of Artificial Societies and Social Simulation 13, no. 4 (2010): 3.

[16] Ullmann-Margalit, Edna. The emergence of norms. Oxford: Clarendon Press, 1977.

[17] Xu, Hongyun, Bastin Tony Roy Savarimuthu, Aditya Ghose, Evan Morrison, Qiying Cao, and Youqun Shi. "Automatic bdi plan recognition from process execution logs and effect logs." In Engineering Multi-Agent Systems, pp. 274-291. Springer, 2013.

[18] van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)

[19] Vahidov R, Kersten GE (2004) Decision station: situating decision support systems. Decis Support Syst 38:283–303

[20] Van Putten, B.-J., Romeiro, C., Azevedo, L.G. (2011). Decision Support by Automatic Analysis of Business Process Models. Proceedings of the 19th European Conference on Information Systems (ECIS), June 9-11, 2011, Helsinki, Finland