Learning Business Rules for Adaptive Process Models

Hans Friedrich Witschel, Tuan Q. Nguyen, Knut Hinkelmann
Fachhochschule Nordwestschweiz FHNW
Olten, Switzerland
hansfriedrich.witschel@fhnw.ch, nguyen.quoctuan@students.fhnw.ch, knut.hinkelmann@fhnw.ch

***Abstract*—This work presents a new approach to handling knowledge-intensive business processes in an adaptive, flexible and accurate way. We propose to support processes by executing a *process skeleton*, consisting of the most important recurring activities of the process, through a workflow engine. This skeleton should be kept simple. The corresponding workflow is complemented by two features: firstly, a task management tool through which workflow tasks are delivered and that give human executors flexibility and freedom to adapt tasks by adding subtasks and resources as required by the context. And secondly, a component that learns business rules from the log files of this task management and that will predict subtasks and resources on the basis of knowledge from previous executions. We present supervised and unsupervised approaches for rule learning and evaluate both on a real business process with 61 instances. Results are promising, showing that meaningful rules can be learned even from this comparatively small data set.**

Keywords – business process intelligence; process mining; knowledge work; workflow management

I. Introduction

Offering the best possible support for the execution of core business processes is a mission-critical requirement for many companies. Such support can be offered, e.g., through workflow management systems (WfMS), which help to automate process executions. The advantages of automating business processes with WfMS consist in the increased consistency, traceability and hence quality of process executions as well as gains in efficiency.

WfMS [1] require a model of the business process; such business process models are traditionally created in a top-down way by modeling experts, an approach which brings with it various problems:

- Agility: process models need to adapt changing conditions. This is hard to achieve if modeling requires an expert.
- Cost: modeling business processes is time-consuming, especially if a great accuracy should be achieved.
- Accuracy: process models should be consistent with reality in the sense that they either describe accurately how a process is executed or – in the case of automation through WfMS – that the execution really follows the model (i.e., no "by-passes" occur). Such consistency is almost never achieved because the cost would be too high in many situations.

These problems are especially serious in processes which we call *knowledge-intensive*, i.e., ones that involve complex decisions at runtime (e.g. exceptional situations, highly variable situations) and can include a wide variety of resources depending on the context.

In this paper, we present an integrated approach to modeling and managing business processes which addresses the above problems through a combination of top-down modeling – resulting in a simple *process skeleton* – with bottom-up techniques: end-user contributions and automatic *learning of business rules.*

The process skeleton includes what is stable – in most business processes, even if they are very loosely structured and highly knowledge-intensive, there exists a kernel of tasks that always has to be executed. This skeleton is usually easy to identify and can thus be modeled quickly and at low cost.

Then, the skeleton is combined with business rules that predict subtasks and resources based on the context (e.g. the attributes of previous activities in the process) and on historical user behavior. By observing user behavior, we ensure both accuracy and agility: because they are based on usage history, the rules reflect how the process is executed in reality – and new rules can be learned at any time.

Thus, taken together, the process skeleton and the business rules form an *agile process model* that is adaptive and maintainable at low cost. It is the aim of this paper to show that it can also be accurate, i.e. that useful business rules can be learned (semi-)automatically to complement the process skeleton. These rules are meant to *support* human knowledge workers who execute a process – they can still choose to ignore the resulting recommendations and act according to their own experience.

In the rest of this paper, we will first examine related work in section II and outline our specific contribution. After giving some background regarding an existing approach that we build on in section III, we will outline our method for learning business rules in section IV, describe our experimental setup for evaluating in section V and present the results of that evaluation in section VI. Finally, section VII concludes the paper.

II. Related work

In [2], three dimensions of change for business processes are introduced:

- Dynamism: adapting the model (at design time) to evolutions in the execution of a business process
- Adaptability: reacting to exceptional circumstances at runtime
- Flexibility: being able to execute on a partially specified model (where the full specification is made at runtime)

The need for dynamism has been recognized by researchers a long time ago and has resulted in the emergence of the new research area of business process reengineering, see, e.g., [3].

The accuracy of process models which are created by human experts is a related problem: the complexity of most business processes often leads to discrepancies between the model and reality [4]. Therefore, process mining has been proposed as a means to discover process models from usage data: traces of real process executions, e.g. recorded actions in event logs, are the basis for learning process models automatically [3][5]. Besides the discovery of process models, checking the conformance of a model with traces of real executions is another discipline in process mining [6]. Finally, van der Aalst [7] has proposed various ways of enhancing process models, e.g. with information about execution times and relationships between human executors. Learning (decision tree) classifiers to predict decisions in the case of choices is another kind of process model enhancement [8]. It is related to our approach since a decision tree can be represented as a set of business rules that predict subsequent activities. In [9], the history of process executions is used to learn a model that recommends activities in a running process instance.

However, decision mining and process mining in general focus exclusively on the control flow, i.e., activities of a process model are treated as atomic units, their internal structure, e.g., resources or subtasks, are not considered or predicted. In [10], we have investigated how possible subtasks of process activities can be recommended based on the analysis of informal traces of work (email). However, in that approach we did not build on an existing WfMS to collect traces of process execution.

The challenges regarding adaptability and flexibility result from the fact that the traditional distinction between design time and run time in workflow management [11] can be less strictly followed in cases where the flow of action (traditionally modeled at design time) is largely determined by input that is received only at runtime.

Modeling languages that are based on business rules (e.g. [12]) have been claimed to be more flexible and expressive than graphical models – in terms of flexibility, it is possible to create *partial* process descriptions with business rules, and the expressiveness of rules is higher because they can take into consideration the run-time process context. Although these advantages are confirmed by [13] in a comparative evaluation of the two approaches, the authors of that study also find that specifying workflows solely through business rules has the drawback of requiring more technical knowledge and being more difficult to understand and maintain by humans than a graphical model.

Various other solutions have been put forward to achieve flexibility: as discussed in [1] and [10], trying to model all possible choices at design time is rarely possible and results in complex and unwieldy models.

Therefore, combinations of process models and business rules have been put forward. By separating the business logic from the process logic and representing it as rules, the complexity of process models can be reduced. The KISS approach [15] combines semantically enhanced process models with business rules to increase their flexibility. A similar approach is taken in [1][14] where a combination of a core process model (similar to our proposed skeleton) with *pockets of flexibility* is proposed – in which predefined or new process fragments are inserted into the core model at run time via rules (contained in so-called *build activities*). But again, this approach concentrates on the control flow, and does not consider e.g. resources used in process activities.

Other approaches for ensuring flexibility in processes consist in giving more freedom and responsibility to the human executors of processes. Examples of such endeavours are case handling [16] and task management approaches based on task patterns [17][18]. The task pattern approach relies on collaborative development of process knowledge. A combination of process skeletons with pattern-based task management is proposed in [19].

However, the idea of having at most a very coarse process model (“flexibility by granularity” [2]) and leaving details of process execution to humans is problematic since it fails to adequately support these persons in their work. In general, as pointed out in [9], as the flexibility of execution increases, the support offered by process-aware information systems usually decreases.

Finally, there are approaches that also propose to model families of business processes on a coarse level and then make them configurable such that they can be adapted to the concrete needs of a situation or company [19][20]. The focus of these approaches is, however, not so much on flexibility for individual processes, but on the *re-use* of models of recurring processes across companies.

A. *Contribution*

Our approach builds on and extends previous work as follows: We start from the notion of a *process skeleton*, similar to the core process model in [2]. That model is deployed as a workflow, but its activities can be handled flexibly by the executors in a *task management* framework (see [17][19]) by adding resources and subtasks at their own discretion. That handling is captured in an event log, which serves as the basis for *learning business rules* that recommend resources and subtasks in later process executions. This learning approach is similar to decision mining [8], but goes beyond it since it predicts task features, not only the control flow.

All in all, our proposed approach combines the advantages of previously introduced ones to respond to the challenges mentioned earlier: It reduces the *cost* of modeling since it only requires a rough process skeleton to be deployed. Furthermore, it ensures *flexibility* and *adaptability* by giving responsibility to humans (task management) and triggering rules based on the context. On the other hand, it offers adequate *support* to knowledge workers through recommendations. By learning the rules from real process executions, this approach also ensures the *accuracy* of the resulting model.

III. BACKGROUND: THE KISSMIR APPROACH

As outlined in the previous section, this work builds on the KISSmir system, as described in [19]. KISSmir is based on a process skeleton that contains those activities of a business process that are always executed. The process skeleton is deployed as a workflow – human executors are assigned tasks that they need to complete.

The tasks are loaded into a task management application where they can be modified by adding resources (persons or documents), notes, subtasks or statements of problems that occur during execution.

Task executors are supported by recommendations of resources, subtasks and problem/solution statements that have been contributed by other persons during execution of the same activity (see [19] for details).

Any modifications of tasks, e.g., addition of resources, subtasks or problem statements, be it by copying from the task pattern or manual creation, are logged by the system.

IV. LEARNING BUSINESS RULES

The learning of business rules that we propose in this work is based on the KISSmir system logs. It consists in an initial preprocessing, followed by either supervised or unsupervised learning, as described in the following subsections. Both rely on co-occurrence of attributes, e.g. the fact that the presence of a certain problem (exceptional situation) triggers a certain subtask to be executed. We thus understand a business rule as an implication of the form A → B, meaning that the presence of the element A in a case implies that B should also be present.

A. Preprocessing

The data from the KISSmir system logs is transformed into feature vectors, each of which represents exactly one process instance. Formally, a feature vector for a case c is given by

$$\vec{c} = (c_1, \dots, c_k, s_1, \dots s_{l,} p_1, \dots, p_m, r_1, \dots, r_n)$$

The attributes are derived as follows:

- $c_1, \dots, c_k$ are attributes that describe the case as a whole (process variables)
- $s_1, \dots s_l$ describe the subtasks that have been added to *any* of the activities of which the case consists. Each attribute $s_i \in \{0,1\}$ denotes whether a given subtask (from the set of all subtasks in the whole log) is present in the given case ($s_i = 1$) or not ($s_i = 0$).
- In the same way, $p_1, \dots, p_m$ and $r_1, \dots, r_n$ describe the presence of problem statements (p_i) and resources (r_i) in a case.

Subtasks and problem statements are identified by their name. Obviously, the same kind of (sub)task (resp. problem) can be described by different names and it is thus sometimes difficult to match the subtask names that describe the same activity. On the other hand, subtasks are frequently accepted recommendations, which ensures consistent naming across process instances. The set of features used is given by all resources, subtasks and problem statements that actually occurred in the past, not limited to any fixed or predetermined set.

B. Supervised learning

Supervised learning through classification consists in predicting one value of a feature vector from some or all values of the other attributes, i.e., we want to learn a classifier that predicts the presence of a particular subtask s_i in a given case, based on the presence of other subtasks, resources and/or problems in the same case. This is promising since often the presence of an exceptional situation (as documented by problem statements) triggers the execution of a subtask, or one subtask triggers another etc.

1) Learning decision trees

Because of the temporal dependency between activities – as given by the workflow – we do not use the full feature vectors in each case: when we train a classifier for an attribute, say subtask s_i, we prune the feature vectors of all training cases such that they only include the case attributes c_i, plus those subtask, problem and resource elements that have been used in activities prior to or equal to the one to which the current class attribute (s_i in our example) belongs. In our experiments, we used the Weka machine learning library [22] and its implementation of the J48 decision tree learner [23] – we loaded the feature vectors into the tool, which then produced candidate decision trees out of them.

2) Selecting decision trees, deriving business rules

In a real-life scenario, most elements of feature vectors are 0, i.e., subtasks, resources etc. are used in only a few cases. For those attributes that do not strongly depend on the presence of other elements, this strong bias means that their corresponding decision tree will consists of only one node which is root and leaf at the same time and decides for a value of 0 – i.e., the classifier predicts the absence of the element in all cases. Obviously, these are not interesting business rules.

Only attributes whose presence depends significantly on others will result in a decision tree with more than one node. Therefore, choosing all learned decision trees that have more than one node is a simple, but – as we will see later – effective approach for filtering the initial set of decision trees.

Since each decision tree can be expressed as a set of rules, we can easily derive our targeted business rules from the resulting set of trees. We will see later that in practice, most trees are not very deep (two or at most three levels), such that the resulting rules are simple and human-readable.

C. *Unsupervised learning*

Mining association rules is another approach to derive business rules which determine appropriate subtasks for certain problems, i.e., we are looking for co-occurences of problems and subtasks in process instances. For this, we transform feature vectors representing the process instances into *transactions*, where only the attributes with a value of 1 appear, say, e.g., $\{c_2, s_3, s_7, r_4\}$.

1) *Mining associations*

We then use the Apriori algorithm [24] from the Weka library to derive association rules. Because of the known weaknesses of approaches that use support and confidence – confidence does not accurately measure the statistical dependence between antecedent and consequence of a rule [25] – we chose to additionally analyse the co-occurrence patterns in the data using a likelihood ratio measure [26] that is particularly reliable in cases where frequency distributions are skewed (which is the case for our data set). For this purpose, we used the software tinyCC[1] which implements the likelihood ratio measure for computing the significance of co-occurrences of words in sentences of natural language text. We therefore represented the transactions as “sentences” where each sentence is formed of the subtasks, resources or problems that are present in a given case – each such element is a word of the sentence.

2) *Selecting association rules*

For the unsupervised variant of the rule learning, we need to rely on thresholds to filter rules. In case of the Apriori results, we can filter by confidence, in case of the tinyCC results, we can filter by the likelihood ratio values. The threshold needs to be set manually.

In both cases, we need to additionally discard rules that point into the past, i.e., ones where the consequence of a rule appears in an activity that occurs temporally before the activity to which the antecedent of the rule belongs.

V. EXPERIMENTAL SETUP

Within the School of Business at our university, there are two master programmes – “Business Information Systems” (BIS) and “International Management” (IM) – that have a very similar student selection process. The so-called “matriculation process” for the two master programmes, i.e., the process of checking and deciding on student applications and of communicating these decisions to applicants, forms the context of our experiments.

For each of the two master programmes, there is one secretary in the administration office (see the middle layer of the process model below) who performs the majority of activities within the student selection process.

The work of the two secretaries in the matriculation process is supported by an implementation of KISSmir. In this evaluation, we have concentrated on the sub-process “Check application” that consists of four sub-tasks as shown in Figure 1 and that is performed by the secretaries alone.

We logged the execution of 61 instances of this process with the KISSmir tool and then transformed the log data into feature vectors as described above. In this case, the case attributes were the previous degree of the applicant (DEGREE) and her nationality (COUNTRY).

Table 1 shows statistics about the other attributes that occurred in the 61 matriculation cases.

Regarding the number of resources, the statistics are somewhat misleading since 61 of the 64 distinct resources were attached to the tasks automatically by the workflow and contain information about the applicant.

This means that there were only 3 distinct resources being used. Since these were mutually exclusive, we created a single attribute (called WEBNAME below) that had the title of the resource chosen as value.

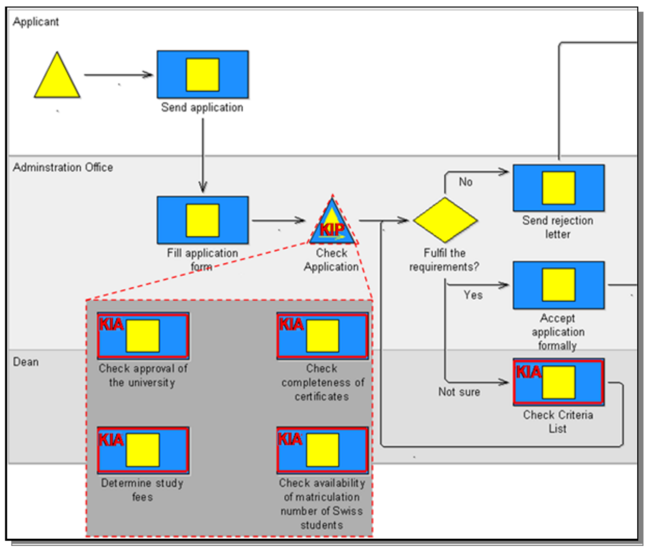

Figure 1. The matriculation process

TABLE 1. STATISTICS OF THE LOG DATA

Measure	Value
Number of cases (process instances)	61
Number of subtasks added by secretaries	34
Number of distinct subtasks	16
Number of problems used	19
Number of distinct problems used	7
Number of resource attachments	314
Number of distinct resource attachments	64

Thus, including the two case attributes, there are 28 attributes in each feature vector (2 case attributes, 16 subtask attributes, 7 problem attributes and 1 resource attribute).

[1] http://wortschatz.uni-leipzig.de/cbiemann/software/TinyCC2.html

VI. Results

A. Supervised Learning

Classifiers are usually evaluated by measures such as accuracy of area under the (ROC) curve (AUC). In our case, however, we are not so much interested in the quality of the classification, but rather in the quality of the *model* that is learned – i.e., the decision trees. Because of the small scale of our experiment and since we know the process rather well, we were able to judge the degree to which the extracted trees made sense.

By applying the procedure for extracting business rules as described above in section 0, we found five non-trivial decision trees. Three of them have only two levels. We will thus describe them directly as rules where the antecedent of the rule corresponds to a value of the root node of the decision tree and the consequence of the rule describes the value of the attribute that should be predicted. For each rule, we add a short description that explains the rationale of the rule in terms of the matriculation process. In each case, we only present the "positive" variant, i.e., the variant that predicts the *presence*, not the absence of the class attribute.

- *Applicant has degree in a complete different area = 1 → Ask the dean = 1.* This rules predicts the presence of the subtask "Ask the dean" (consequence) in cases where an applicant had a degree that was definitely not compatible with the requirements of the master programme (e.g. the student had a Bachelor degree in French language and applied for the International Management master programme, antecedent). This accurately describes the current practice of the secretaries.
- *COUNTRY = Other → Check anabin = 1.* This rule predicts the presence of the subtask "Check anabin" in cases where the applicant is neither from an EU nor from an EFTA country. Anabin is an information platform where university degrees and accreditations can be compared for equivalence. Hence, this rule predicts that the platform should be consulted for students from non-European/non-EFTA countries.
- *Ask the student for a description of the financial situation = 1 → Forward scholarship documents to commission = 1.* This rule predicts the presence of a subtask – namely forwarding an applicant's scholarship application documents to a commission – from the presence of another subtask, as given in the antecedent. When a request for scholarship arrives, the secretaries will first ask the student for a description of his/her financial situation. When that description is received, the documents will be forwarded to the commission.

In addition to these simple rules, two three-level decision trees were discovered. One of them is shown in Figure 2: its leaf nodes (light grey boxes) contain the values of the attribute WEBNAME. That attribute describes which of three letter templates were chosen by the secretaries in their last task – "Accept application formally".

The tree should be read as follows: if a problem statement is present that indicates that the Bachelor degree of a student is still missing because the student is still studying, the student can be accepted, but one needs to use a special letter template in which the applicant is asked to hand in the certificate as soon as (s)he finishes his/her studies. If this is not the case, but the problem "Applicant has degree in complete different area" is present, a letter template should be used in which the student is informed that (s)he needs to take part in a so-called pre-master course to catch up with important foundations for their studies.

Only if none of the two problem statements is present, the standard letter template will be used. Although this tree is largely correct, the two attributes are actually independent, i.e., theoretically – yet very improbably – both could occur together.

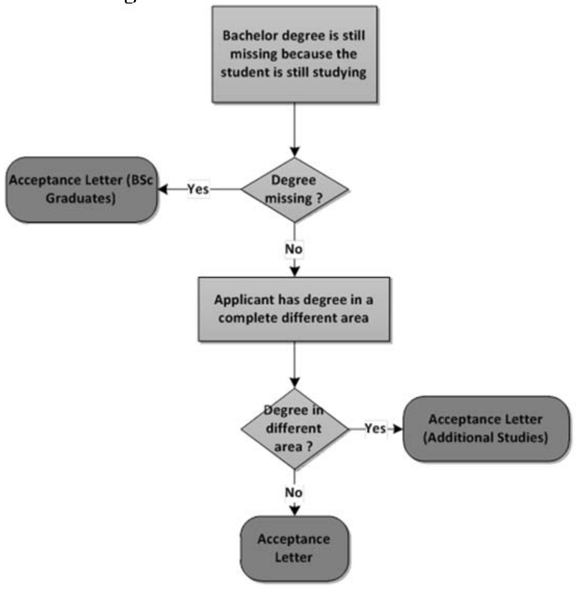
Bachelor degree is still missing because the student is still studying
Acceptance Letter (BSc Graduates)
Yes
Degree missing ?
No
Applicant has degree in a complete different area
Degree in different area ?
Yes
Acceptance Letter (Additional Studies)
No
Acceptance Letter

Figure 2. Decision tree for the attribute WEBNAME in the activity "Accept application formally".

Unsupervised learningTable 2 shows the set of rules extracted with tinyCC using a likelihood ratio threshold of 10.0 and filtered by temporal consistency. They are to be read from left to right; the letters in brackets behind each attribute indicate whether it is a case attribute (c), a subtask (s), problem (p) or resource (r).

Some of these rules – numbers 1, 3 and 6 – are also extracted with supervised learning. The rest can be divided into useful additional rules (4 and 8, e.g., rule 4 will remind the secretaries to forward documents to the commission when a scholarship is requested) and over-generalisations (rules 2, 5 and 7, e.g., rule 2: checking the result of an interview with the student does not always result in using the acceptance letter that notifies about pre-master studies).

All in all, unsupervised learning produces slightly more rules, including more useful rules, at the cost of some noise.

VII. Conclusion and future work

We have presented an approach to flexibly support business processes and to make process models more accurate by learning business rules. The approach starts from a simple process skeleton which is executed through a workflow engine. Human executors are given freedom and support to adapt tasks to their needs. Finally, the log files of this application are evaluated to learn business rules, which can be incorporated into the workflow using a rule engine to recommend subtasks in future process executions.

We have experimentally compared a supervised and an unsupervised approach on a real-life business process and found that both approaches yield meaningful rules even with a small training set. The supervised approach was more reliable, the unsupervised approach had higher recall.

In the future, we plan to implement our proposed approach in an environment with a larger number of participants and cases to test its scalability.

References

[1] W.M.P. v. d. Aalst and K. van Hee, *Workflow Management: Models Methods, and Systems*. MIT Press, 2002.

[2] S. Sadiq, W. Sadiq, and M. Orlowska, “Pockets of Flexibility in Workflow Specifications,” in *Proceedings of ER’01*, 2001, pp. 513 - 526.

[3] K. D. Swenson and K. Irwin, “Workflow technology: trade-offs for business process re-engineering,” in *Proceedings of COCS 95*, 1995, pp. 22-29.

[4] W. M. P. van der Aalst, W. M. . van der Aalst, A. J. M. . Weijter, and L. Maruster, “Workflow Mining: Discovering process models from event logs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, 2003.

[5] J. E. Cook and A. L. Wolf, “Discovering models of software processes from event-based data,” *ACM Trans. Softw. Eng. Methodol.*, vol. 7, no. 3, pp. 215-249, 1998.

[6] A. Rozinat and W. M. P. van der Aalst, “Conformance checking of processes based on monitoring real behavior,” *Information Systems*, vol. 33, no. 1, pp. 64-95, Mar. 2008.

[7] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Verlag, 2011.

[8] A. Rozinat and W. M. P. van der Aalst, “Decision Mining in Business Processes,” 2006.

[9] H. Schonenberg, B. Weber, B. Dongen, and W. Aalst, “Supporting Flexible Processes through Recommendations Based on History,” in *Proc. of BPM’08*, 2008, pp. 51-66.

[10] S. Brander et al., “Refining Process Models through the Analysis of Informal Work Practice,” in *9th International Conference on Business Process Management*, vol. 6896, S. Rinderle-Ma, F. Toumani, and K. Wolf, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2011, pp. 116-131.

[11] D. Hollingsworth, “The workflow reference model.” The Workflow Coalition, 1993.

[12] J. Bae, H. Bae, S.-H. Kang, and Y. Kim, “Automatic control of workflow processes using ECA rules,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 1010-1023, Aug. 2004.

[13] R. Lu and S. Sadiq, “A survey of comparative business process modeling approaches,” in *Proc. of BIS'07*, 2007.

[14] S. Sadiq, M. Orlowska, and W. Sadiq, “Specification and validation of process constraints for flexible workflows,” *Information Systems*, vol. 30, no. 5, pp. 349-378, Jul. 2005.

[15] D. Feldkamp, K. Hinkelmann, and B. Thönssen, “KISS - Knowledge-Intensive Service Support: An Approach for Agile Process Management,” in *RuleML*, 2007, pp. 25-38.

[16] W. M. P. van der Aalst, W. M. . van der Aalst, M. Weske, and D. Grünbauer, “Case Handling: A New Paradigm for Business Process Support,” *Data and Knowledge Engineering*, vol. 53, 2005.

[17] E. Ong, O. Grebner, and U. V. Riss, “Pattern-based task management: Pattern lifecycle and knowledge management,” in *Proc. of WM’07)*, 2007, pp. 357-367.

[18] B. Schmidt and U. V. Riss, “Task patterns as means to experience sharing,” in *Proceedings of ICWL’09*, 2009, pp. 353-362.

[19] H. F. Witschel et al., “A Collaborative Approach to Maturing Process-Related Knowledge,” in *Proc. of BPM’10*, 2010, vol. 6336, pp. 343-358.

[20] M. La Rosa, M. Dumas, A.H.M. ter Hofstede, and J. Mendling, “Configurable multi-perspective business process models,” *Information Systems*, vol. 36, no. 2, pp. 313-340, 2011.

[21] I. Montero, J. Pena, and A. Ruiz-Cortes, “From Feature Models to Business Processes,” in *SCC’08*, 2008, pp. 605-608.

[22] “Weka.” [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka. [Accessed: 18-Jun-2012].

[23] R. Qinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[24] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases,” in *Proceedings of ACM SIGMOD*, 1993.

[25] S. Brin, R. Motwani, and C. Silverstein, “Beyond market baskets: generalizing association rules to correlations,” in *Proc. of ACM SIGMOD*, 1997.

[26] T. Dunning, “Accurate Methods for the Statistics of Surprise and Coincidence,” *Computational Linguistics*, vol. 19, no. 1, 1993.

Table 2. Associations of attributes extracted using tinyCC

Nr	Attribute 1	Attribute 2	Sig
1	Applicant has degree in a complete different area (p)	Ask the dean (s)	26.55
2	Check interview result (s)	Acceptance Letter (Additional Studies) (r)	18.63
3	Bachelor degree is still missing because the student is still studying (p)	Acceptance Letter (BSc Graduates) (r)	17.74
4	How to handle Scholarship requests (p)	Forward Scholarship Documents To Commission (s)	17.39
5	Ask The Dean (s)	Acceptance Letter (Additional Studies) (r)	15.34
6	Applicant has degree in a complete different area (p)	Acceptance Letter (Additional Studies) (r)	10.8
7	Ask The Dean (s)	Check interview result (s)	10.78
8	iso3166#Other (c)	Anabin (r)	10.36