Classification Techniques for Conformance and Performance Checking in Process Analysis

Hind Chfouka[1], Andrea Corradini[1] and Roberto Guanciale[2]

[1] Department of Computer Science, University of Pisa, Italy
[2] School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract. Standard process analysis techniques, like conformance checking or performance evaluation, are enabled by the existence of event logs that trace the process executions and by the presence of a model that formally represents the process. Such analysis techniques use only part of the huge amount of data recorded in event logs. In this paper the goal is to exploit this data to extract useful information for conformance checking and performance analysis. We present an approach that using standard classification technique, explores how data influence process behaviors by affecting its conformance or performance.

1 Introduction

Today, many Information Systems that support the concept of business process record all events occurring during the process execution in event logs. An event log represents a trace of the process behavior that can be observed and analyzed in order to tune it with the business objectives pursued. Process Mining provides a set of techniques for process discovery and analysis. Through the knowledge extracted from event logs, process discovery allows one to construct a process model. Process analysis instead assumes the existence of a model and it consists of checking the conformance and performance of the process executions with respect to it.

The work presented in this paper is motivated by the observation that existing event logs are rich of data, but such data is used only in part by the existing process analysis techniques. Since data is considered as a valid source of knowledge, developing techniques based on data analysis can provide advantages for process discovery and analysis. This fact has already been observed several times in the literature, and it brought to the development of various approaches where machine learning techniques are applied to workflow or process discovery (see e.g. [4, 5, 8, 2]).

Here instead, we consider more specifically process analysis. Our goal is to find a way to transform the data recorded during the process activities in knowledge useful for the analysis. In particular, we try to determine how data may affect the conformance and performance of process executions. Identifying this influence can provide qualitative information about the causes of possible anomalies in the process behaviors, facilitating the task of taking corrective measures.

In order to analyze data contained in event logs, we propose an approach based on machine learning [10], where we exploit *classification* techniques to find patterns on data in presence of which conformance errors occur. The same approach has been extended to evaluate how data influences process efficiency, by combining classification problems with performance evaluation: this is just summarized briefly in the paper due to size constraints. Classification is a well-known machine learning technique which consists of identifying to which category, among a given set, a new observation belongs. This is done on the basis of a training set of data containing observations whose category membership is known.

After the presentation of a case study in Section 2, Section 3 introduces some background concepts about process analysis. First the log replay algorithm is briefly described, then conformance analysis is introduced using the business process of Section 2. In Section 4 our approach to process analysis based on classification is presented. First a few basic concepts about the classification technique are described, next the conformance analysis based on classification is presented in detail through an illustration with our case study. A possible extension of the approach for performance analysis is briefly discussed presented. Finally, Section 5 describes the implementation of our approach provided as *ProM6* [12] plug-ins, as well the methodology to follow in order to analyse processes using our approach.

2 The case study: a sale business process

A business process can be seen as a collection of activities occurring within an organization that lead to a specific goal. There are several formalisms for representing business processes. In this paper we represent them using Petri nets [7, 11] since this makes possible to exploit some existing process analysis techniques that are based on such nets. However, in a business management context, Petri nets can be not expressive enough and usually process models are presented as workflows, for example using a standard notation for the business process modeling: BPMN (Business Process Model and Notation) [6]. Transforming a BPMN model into a Petri net is possible thanks to a mapping technique presented in [1]. Figure 1 presents the Petri net model for a sale process.[3] This business process abstractly represents a fragment of the procedure followed in a commercial organization for managing orders received from clients. In general this procedure includes several activities and each of them involves a specific department of the organization. We assume that the sale process begins with the notification of an order from a client, so the first activity is simply called *Order*. After the order is received, the sale process continues with some activities that can be done in parallel, since they do not present any dependency. The *FinancialCheck* activity

[3] The Petri net of Figure 1 is obtained by transforming a BPMN model using the algorithm of [1], and mapping each activity to a pair of start/end transitions. The resulting net has been simplified by removing the unnecessary *invisible transitions* used to encode the *Join* or *Fork* gateway [1].

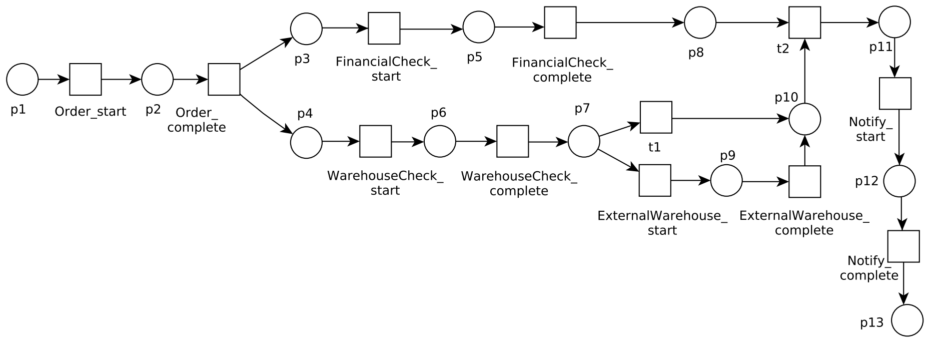


Fig. 1. Petri net for a sale process

represents the financial analysis of the order (i.e verifying the financial situation of the client). Concurrently, the *WarehouseCheck* activity checks the availability of merchandise requested by the order and, in case of a shortage, starts a supplying procedure. This is done through the activity *ExternalWarehouse*. After the completion of the parallel activities, a synchronization is needed before continuing the sale procedure with the final activity called *Notify*, by which a result regarding the acceptance of the order is communicated to the client.

3 Process Analysis

Process analysis is performed using the Petri net representing the process and an event log recording the related executions, also called *process instances*. The basic building blocks of event logs are *events*, that record the execution of the process actions. Therefore an event e can be seen as a tuple including the action recorded, a corresponding timestamp, and possibly other attributes describing relevant information about the state of the process. Events that belong to the same process instance are grouped in a *trace*, where events are ordered by timestamp. An *event log* is a set of traces, recording the activities performed by a system during a finite number of process executions. In this paper the following assumptions are made:

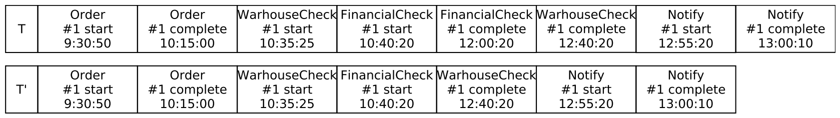
- All traces are instances of the same process.
- For each action there exists a corresponding transition in the net that will be denoted, for simplicity, by the same name of the action.

3.1 Log replay algorithm

The key algorithm exploited to analyze a Petri net model with respect to an event log is the *log replay* algorithm (see e.g. [9]). Given a Petri net model and an event log as input to the algorithm, for each trace in the log, the algorithm starts by placing one token in the start place of the net. For each event in the trace the corresponding transition is fired assuming a non-blocking behavior, then the marking of the net is updated. *Non-blocking replay* means that whenever a firing of a disabled transition is needed, the algorithm enables the transition either by creating artificial tokens in the pre-set or, if possible, by firing some invisible transitions; the non-determinism in this procedure is resolved with a suitable cost function. For each trace, the result of the log replay execution is a map indicating which transitions of the model have been fired for each event in the trace, and additionally which tokens were artificially created and which remained upon completion of the trace. The output of the log replay algorithm can be used to check the conformance of the traces.

3.2 Conformance Analysis

The goal of the conformance analysis is to check if a trace complies with the Petri net modeling the business process. Conformance problems can be discovered by analyzing tokens artificially created during the replay *(the missing tokens)* and tokens not consumed *(the remaining tokens)*. Figure 2 presents two different traces, T and T', of the business process presented in Section 2, where only the actions and the timestamps are reported. The log replay of trace T, which is compliant with the Petri net, terminates with a single remaining token in the end place $\{p13 \rightarrow 1\}$ and without reporting missing tokens.



T	Order #1 start 9:30:50	Order #1 complete 10:15:00	WarhouseCheck #1 start 10:35:25	FinancialCheck #1 start 10:40:20	FinancialCheck #1 complete 12:00:20	WarhouseCheck #1 complete 12:40:20	Notify #1 start 12:55:20	Notify #1 complete 13:00:10

T'	Order #1 start 9:30:50	Order #1 complete 10:15:00	WarhouseCheck #1 start 10:35:25	FinancialCheck #1 start 10:40:20	WarhouseCheck #1 complete 12:40:20	Notify #1 start 12:55:20	Notify #1 complete 13:00:10

Fig. 2. Conforming (T) and non-conforming (T') traces

The log replay of the trace T' terminates with remaining tokens $\{p5 \rightarrow 1, p13 \rightarrow 1\}$ and one missing token $\{p8 \rightarrow 1\}$. The missing token is created artificially and this fact witnesses a wrong execution of the event *Notify_start*. In fact, in T' this event is executed before the termination of the activity *FinancialCheck*: this is interpreted as a non-conformance to the process model.

4 Classification for Process Analysis

Exploiting machine learning techniques for process analysis is encouraged by the presence of a huge amount of data recorded as attributes in event logs. The implicit information contained in those data could be significant for the process behavior analysis. In fact, this potential information could contain an explanation for the deviations discovered during conformance analysis, and for the performance level provided by the process. For this reason, we are interested in discovering how the process data may influence the conformance and the performance of the process executions. This is done with an approach based on *classification*, a classical machine learning technique able to detect patterns on data in correspondence to which the process assumes a specific behavior.

4.1 Classification: basic concepts

In a classification problem [10], data is represented by a collection of records (called instances), and each of them is characterized by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is a set of attributes and y is a special one called *target attribute*. The value of the target attribute is a label identifying the class to which the record belongs. The goal is to learn a classification model that maps each attribute set $\mathbf{x}$ to one of the predefined class labels y.

A classification model can be useful for both a descriptive and a predictive purpose. In fact it provides an explanatory tool to distinguish between objects of different classes, and at the same time it can be used to predict the class label of new records. In this way the model acts as a black box that automatically assigns a class label to an attribute set of a new record.

There is a general approach to solve a classification problem from a given dataset that is independent from the specific classification model. Each classification technique uses a learning algorithm to identify a model that best fits the relationship between the attribute sets and the class labels of the input data. The generated model should both fit the input data well, and correctly predict the class labels of new records. Finding the right trade-off between this two goals is the most delicate part of a classification technique, and in general of any machine learning task.

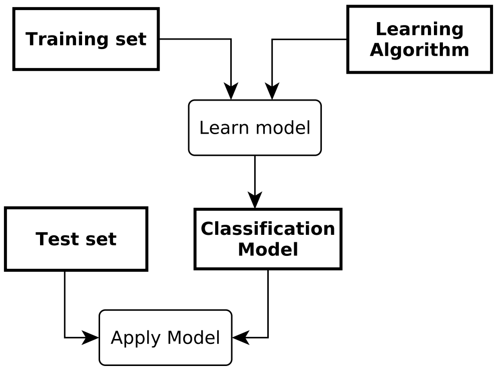


Fig. 3. General approach for solving a classification problem.

Figure 3 summarizes the general approach for solving a classification problem [10]. A *training set* consisting of records whose class labels are known must be provided to the learning algorithm, and it is used to build the classification model. This model is then applied to a *test set*, which consists of records with unknown class labels. This step allows to estimate the *model accuracy* using one of the standard approaches. For example, the accuracy can be represented by the percentage of records correctly predicted with respect to the total number of records in the test set.

Many classification models can be applied to a classification problem. For our purpose we choose the *decision tree* model since the rules detected by the classification are shown explicitly, providing an overview of how process data affects the process behavior. Furthermore, this choice is encouraged by the simplicity of decision trees. In a decision tree, a class label (possible value of the target attribute) is assigned to each leaf node. The non-terminal nodes, which include the root and the internal nodes, contain attribute test conditions to filter records that have different characteristics.

4.2 Classification for Conformance Checking

In order to identify the possible causes of non-conformance, a classification problem can be formulated. The classification dataset is extracted from the process data: for each process instance, relevant information is extracted and included as a record in the dataset. For the process presented in Section 2, a record includes the following attributes: an instance identifier, a client identifier, the client typology (new or consolidated client), the sales manager responsible for the order, the financial officer who conducts the financial evaluation activities, the warehouseman responsible for the warehouse checking, the supplying responsible name in case of a provision, and finally the result communicated to the client for the order issued. Futhemore, as target attribute we associate with each record the conformance result for that instance, which is a boolean value that is extracted from the output of the log replay. All these attributes are discrete and contribute to build the dataset presented in Table 1 of the classification problem. In this preliminary work, in order to explore the applicability of existing machine learning techniques to process analysis, we just used synthetic data generated to emulate process executions.

In order to build the classification problem, the attributes characterizing the process are extracted from the log, while the conformance attribute is computed using the conformance checking algorithm. The Petri net of Figure 4 shows the output of the conformance analysis executed on the event log: edges are labeled with the number of times they were activated, places are labeled with the number of remaining and missing tokens.

The resulting data highlight that there are 37 out of 250 process executions with conformance errors. The presence of 37 missing tokens in $p11$ represents the number of times that the log replay algorithm forced the execution of transition *Notify_start*, even if it was not enabled, in order to mimic a corresponding event in the trace.

Moreover, 37 remaining tokens in place $p3$ indicate the number of times that the financial check activities were not performed. Taking into account the

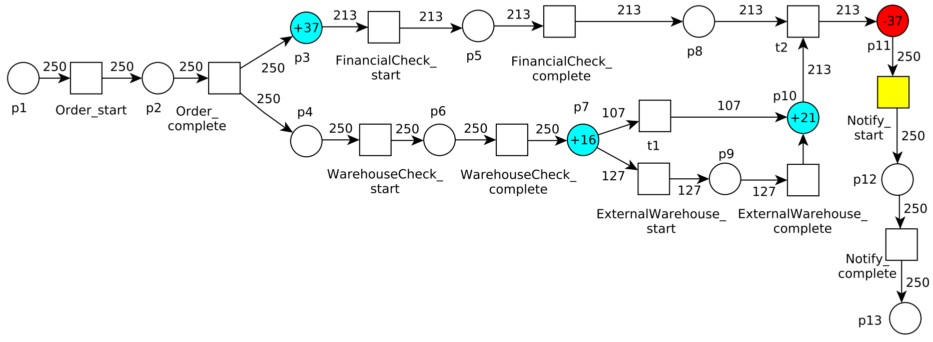


Fig. 4. Petri net: conformance results.

application scenario, we can conclude that 37 instances of the process are not compliant with the sale policy since they did not execute the financial check activity.

OrdIde	CltIde	CltType	SalMan	FinOff	WrhsMan	SupplyResp	OrdResut	Conf
1	20	consolidate	Marco	Mary	Alex	Gianni	positive	no
2	15	new	Anna	Johann	Roberto	Mario	positive	yes
3	10	consolidate	Maria	Mary	Alessio	Gianni	negative	no
10	18	consolidate	Johann	Mary	Roberto	Gianni	positive	yes
...	...	...	...	...	...	...	....	...

Table 1. Dataset for the conformance analysis.

The results of the conformance checking enable the use of existing machine learning tools, allowing us to mine the decision tree of Figure 5 as classification model for the sale process. The resulting decision tree describes a data pattern in correspondence to which a process instance could present conformance errors: order managed by the sales manager *Mary* and received from consolidate clients of the organization may not respect the standard sale procedure.

From the analysis done for the sale process based on both classification and log replay results, a new scenario of the sale procedure has emerged: orders done by the consolidate clients of the organization are not checked from a financial point of view. This fact can be accepted as a valid behavior in a sale procedure, and consequently an extension of the business process model is needed to include the new scenario (*model extension activity*). Alternatively, the misbehaving scenario can be considered as an anomaly in the sale procedure. In this case, since the analysis provides accurate information about the conformance error, corrective measures can be taken in order to prevent such errors in the future.

The classifier constructed with this approach can be used also in a predictive way. Given a trace recording a process execution the decision tree can be used

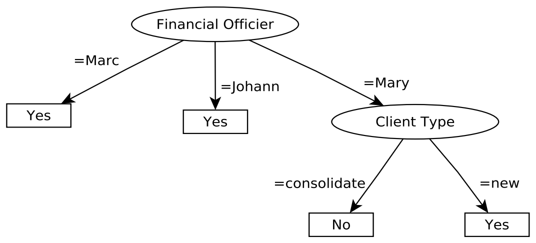


Fig. 5. Decision tree for the conformance analysis.

to predict the conformance result. This brings an advantage in terms of the time required by the analysis, since the replay of a process instance takes more time than its classification using a decision tree. Moreover, it is worth noting that whenever the set of attributes needed for the classification is known before the completion of a process execution, the non-conformances predicted by the decision tree can be used to promptly alert the stakeholders and activate proper countermeasures.

4.3 Extension of the approach to Performance Analysis

Performance analysis of a process can be carried exploiting log replay. Since an event log contains timestamps, during the replay of a trace it is possible to compute (for each place of the Petri net model) some performance measures such as: *synchronization time* (i.e. time interval between arrival of a token in the place and enabling of a transition in the post-set of the place), *sejour time* (i.e. the time interval between arrival and departure of tokens) and *waiting time* (i.e. the time interval between enabling of a transition in the post-set of the place and token departure).

Performing an analysis based on the measures computed by log replay gives just a *quantitative* information about the process performance. In order to understand the *causes* of performance anomalies that can affect the process behavior, one can explore the process data. Discovering how data attribute can influence process performance provides useful information in analyzing and optimizing the process services. For example, discovering data patterns in correspondence to which some activities need more time for completion than others, helps in making decisions about resources distribution to the process activities or in scheduling activities. In addition to the completion time, one could analyze a process under more complex performance metrics such as the synchronization time. A process with parallel branches and synchronizations can present bottleneck activities that lead to increase the execution time of other activities and consequently the completion time of the entire process. To find out the possible data influences on synchronization time, the approach presented in Section 4.2 for conformance analysis through classification can be easily extended to performance analysis. For example, a classification problem can be formulated for each synchronization point of the process. In this way, the classifier obtained in correspondence to a synchronization point classifies the process instances based on their attribute value and regarding the synchronization time of the point taken into exam.

5 Implementation with ProM

In order to experiment with some business process prototypes, the approach presented in this paper has been implemented as a set of plug-ins that integrates *ProM6* [12], an open-source framework implementing Process Mining tools, and *Weka* [3], a data mining framework providing machine learning tools. We can classify the plug-ins developed in three categories. The first one includes plug-ins for building the dataset needed for the classification:

– `Generate Instances with Conformance`: this plug-in takes as input an event log and its conformance results computed by log replay. It returns as output a training set as shown in Table 1 in the format needed by the Weka classification tool. Note that the plug-in extracts *all* the attributes data recorded in the event log. If necessary, the dataset resulting can be subject to some *feature selection* techniques before using it as a training set for the classification task.
– `Generate Instances to Classify`: given an event log, the plug-in generates a dataset of instances that can be classified using a classification model.

The second category includes plug-ins for the classifier generation and utilization as a predictive model:

– `Generate Classifier`: given a training set, the plug-in generates a decision tree model using the $J48$ algorithm, an implementation of the algorithm $C4.5$ provided by Weka.
– `Classify Instances`: given a set of non-classified instances and a classifier model, this plug-in simply classifies the instances according to the classifier.

Finally, the last category includes a set of plug-ins useful for the serialization, deserialization and visualization of the resources used by the previous plug-ins, such as dataset and classifiers.

5.1 Methodology

This last paragraph is dedicated to a description of the methodology to be followed to analyze a business process under the approach presented in this paper.

It should be clear that the analysis presented in the previous sections assumes the presence of a Petri net model of the business process to be analyzed. The model can be expressed in BPMN as well, but in this case a transformation into a Petri net can be done through the techniques described in [1]. The classification analysis needs also an event log tracing the process behavior over multiple instances. Given a process model M and an event log L, the methodology to be followed for the conformance analysis with the approach presented in Section 4.2 can be summarized by these steps:

1. **Log replay analysis**: this phase consists of an execution of the log replay with the model M and the log L as inputs. The results consist of two structures characterizing both the conformance and the performance of the traces in L. Let us call them *ConfResult* and *PerfResult*.
2. **Training set construction**: this phase aims at building the training set for the classification technique. This is done through an execution of the plug-in `Generate Instances with Conformance` with the event log L and the conformance result *ConfResult* as inputs. Let us call the obtained training set TS.
3. **Classifier building**: in this step the classification model (decision tree) for the conformance analysis is built using the plug-in `Generate Classifier` with the training set TS as input. Let us call the obtained decision tree DT.
4. **Using the classifier**: in this step the goal is to exploit DT and the log replay result *ConfResult* to establish the corrective measures in case of anomalies. Note that both DT and *ConfResult* can be visualized as in Figures 5 and 4 into ProM6 thanks to the plug-ins for resources managing that integrate the framework. DT can also be used in a predictive way to establish the conformance of new process instances without needing a log replay execution. Given an event log containing new process instances, a dataset can be built through the plug-in `Generate Instances to Classify` and then classified by the plug-in `Classify Instances` according to the model DT.

For a performance analysis based on the extension summarized in Section 4.3, the same methodology can be followed modifying step 2. The construction of the training set is done through a plug-in (analogous to `Generate Instances with Conformance`) that takes as input *PerfResult* obtained by step 1 and an indication about the synchronization point chosen for the analysis.

6 Conclusions and Future Work

We presented some preliminary results of a research activity aimed at applying machine learning techniques, i.e. classification algorithms, in the realm of process analysis. The idea is to try to identify some correlation between the data value of certain variables in the process instances of event logs, and the results of conformance checking. A successful identification of such values could allow to predict the conformance result of an instance during execution. The same technique was applied to performance checking as well, but only briefly summarized here because of space constraints. The approach is implemented with a combination of plug-ins of the Process Mining framework ProM and of the classification engine Weka. Some guidelines about their use are presented in a concluding section. The approach has been successfully tested on synthetic event logs only. An obvious continuation of this work includes experimentations with real event logs: this will allow us to evaluate the scalability of the approach and its robustness in presence of noise.

Classification techniques in the Process Mining field have already been explored in [8] and more recently in [2]. In those papers the idea is to explore how data influences the case routing of process flow execution, by assigning a classification problem with each decision point in the model. In particular, [2] improves significantly the results of [8] by exploiting some recently developed *alignment techniques* between process instances and the process model. We intend to explore how far these techniques can be applied fruitfully in our framework as well.

References

1. R. Bruni, A. Corradini, G. L. Ferrari, T. Flagella, R. Guanciale, and G. Spagnolo. Applying process analysis to the Italian e-government enterprise architecture. In M. Carbone and J.-M. Petit, editors, *WS-FM*, volume 7176 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2011.
2. M. de Leoni and W. M. P. van der Aalst. Data-aware process mining: discovering decisions in processes using alignments. In S. Y. Shin and J. C. Maldonado, editors, *SAC*, pages 1454–1461. ACM, 2013.
3. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.
4. J. Herbst. A machine learning approach to workflow management. In R. L. de Mántaras and E. Plaza, editors, *ECML*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2000.
5. L. Maruster. *A machine learning approach to understand business processes*. PhD thesis, Eindhoven University of Technolog, 2003.
6. OMG. Business Process Model and Notation, http://www.bpmn.org/, 2011.
7. C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962.
8. A. Rozinat and W. M. P. van der Aalst. Decision mining in ProM. In S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer, 2006.
9. A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.
10. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, May 2005.
11. W. M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
12. W. M. P. van der Aalst, B. F. van Dongen, C. W. Günther, A. Rozinat, E. Verbeek, and T. Weijters. ProM: The process mining toolkit. In A. K. A. de Medeiros and B. Weber, editors, *BPM (Demos)*, volume 489 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.