Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers in Industry 53 (2004) 321–343
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Business Process Intelligence

Daniela Grigori[a,*], Fabio Casati[b], Malu Castellanos[b], Umeshwar Dayal[b], Mehmet Sayal[b], Ming-Chien Shan[b]

[a]*Laboratoire PRiSM, CNRS FRE-2510, Université de Versailles St-Quentin en Yvelines, 45 avenue des États-Unis., 78035 Versailles Cedex, France*
[b]*Hewlett-Packard, 1501 Page Mill Road, MS 1142, Palo Alto, CA 94304, USA*

Abstract

Business Process Management Systems (BPMSs) are software platforms that support the definition, execution, and tracking of business processes. BPMSs have the ability of logging information about the business processes they support. Proper analysis of BPMS execution logs can yield important knowledge and help organizations improve the quality of their business processes and services to their business partners. This paper presents a set of integrated tools that supports business and IT users in managing process execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimization. We refer to this set of tools as the *Business Process Intelligence* (BPI) tool suite. Experimental results presented in this paper are very encouraging. We plan to investigate further enhancements on the BPI tools suite, including automated exception prevention, and refinement of process data preparation stage, as well as integrating other data mining techniques.

Keywords: Business Process Intelligence; Workflow mining; Process execution analysis and prediction; Data warehouse

1. Introduction and motivations

Process design and automation technologies are being increasingly used by both traditional and newly formed, Internet-based enterprises in order to improve the quality and efficiency of their administrative and production processes, to manage e-commerce transactions, and to rapidly and reliably deliver services to businesses and individual customers.

In order to attract and retain customers as well as business partners, organizations need to provide their services (i.e., execute their processes) with a high, consistent, and predictable *quality*. From a process automation perspective, this has several implications: for example, the business processes should be correctly designed, their execution should be supported by a system that can meet the workload requirements, and the (human or automated) process resources should be able to perform their work items in a timely fashion. While research in the business process area has been mostly focusing on developing new process models and process automation techniques, little work has been done in the areas of process analysis, prediction, and optimization.

This paper presents a set of integrated tools that supports business and IT users in managing process

* Corresponding author. Tel.: +33-1-39-25-40-53; fax: +33-1-39-25-40-57.
E-mail addresses: daniela.grigori@prism.uvsq.fr (D. Grigori), casati@hpl.hp.com (F. Casati), castella@hpl.hp.com (M. Castellanos), dayal@hpl.hp.com (U. Dayal), sayal@hpl.hp.com (M. Sayal), shan@hpl.hp.com (M.-C. Shan).

0166-3615/$ – see front matter
doi:10.1016/j.compind.2003.10.007

execution quality. We refer to this set of tools as the *Business Process Intelligence* (BPI) tool suite, since it is based on the application of business intelligence techniques to business processes. In fact, Business Process Management Systems (BPMSs) record many types of events that occur during process executions, including the start and completion time of each activity, its input and output data, the resource that executed it, and any failure that occurred during activity or process execution. By cleaning and aggregating process logs into a warehouse and by analyzing them with business intelligence technologies, we can extract knowledge about the circumstances in which high- or low-quality executions occurred in the past, and use this information to explain why they occurred as well as predict potential problems in running processes. Note that in this paper we use a very high-level and user-oriented notion of quality: we assume that it is up to the (business or IT) users to define what quality means to them, and in general which are the characteristics that they want to analyze and predict.

The BPI suite provides several features that offer various levels of automation for the management of process quality:

- *Analysis*: BPI allows users to analyze completed process executions from both a business and an IT perspective. IT analysts will be interested in viewing detailed, low-level information such as average execution time per node or the length of the work queues of resources. Business users will instead be interested in higher-level information, such as the number of ''successful'' process executions, or the characteristics of processes that did not meet the Service Level Agreement (SLA) stipulated with customers. Besides providing a wide set of reporting functionalities, BPI also offers several features to help analysts identify the causes of process execution behaviors of interest. The analysis capabilities of BPI can also be applied to analyze the design of a process model, and in particular for identifying techniques for improving an existing process definition.
- *Prediction*: BPI can derive prediction models and apply such models to running processes, to identify in particular the possibility of exceptions or undesired behavior. As in the case of the analysis, predictions can be made at the IT level (e.g., predicting whether a given computer or application will be involved in the execution), or at the business level (e.g., predict whether a service will be delivered in accordance with the stipulated SLA).
- *Monitoring*: BPI can monitor and analyze running process instances, and inform the user of unusual or undesired situations. Users can view the health status of the system, processes, services, and resources. In addition they can define critical situations (alerts), so that BPI can notify them on the medium of their choice in the event such a critical situation occurs.
- *Control*: Based on process monitoring and predictions, BPI can interact with the BPMS to avoid (or reduce the impact) of foreseen and actual quality degradations, and in particular to help avoid missing SLAs.
- *Optimization*: BPI can identify areas of improvements in business process definitions and in the assignment of resources and services to work activities.

Achieving the above objectives would be very appealing for any company executing business processes in an automated fashion. However, the development of a BPI solution present several challenges, such as:

- Identifying the architecture and the technologies that can deliver the above functionalities, and understand how to apply or modify these technologies to achieve our goals.
- Enabling the definition of concepts and metrics that enable business-level, qualitative analysis of processes.
- Developing techniques to make it easy for analysts to use the tool and extract the knowledge they need, possibly without writing any code.
- Understanding how to interact with the BPMS and with users in order to report and correct critical situations in a timely fashion.

The work on business process intelligence is part of a larger research effort aiming at developing business intelligence techniques and tools for monitoring, managing, analyzing, and optimizing the entire e-business platform, from the web server to the back end applications and the network. In the remainder of

the paper we will refer to this research area as *Business System Intelligence*, or BSI for short. While a BSI solution would certainly be extremely beneficial to many companies, its design and development are a quite complex undertaking. However, the problem can be addressed with a *divide and conquer* approach, by first separately addressing each class of business system components (e.g., the application server or web services platform) and then develop a complete, integrated solution. We have engineered and developed the initial solution focusing ''only'' on business process management systems both because it is very valuable and has a wide applicability in its own right, and because we had the required knowledge, expertise, and access to customers' requirements and customers' data, which is a crucial help in the design and development of such a system.

This paper is structured as follows. Section 2 describes basic concepts on process models, whose execution instances are logged, and process execution logs, which contain raw execution data to be mined for valuable information to improve the quality of processes. Section 3 presents the architecture and functionalities of BPI: Section 3.1 covers the design of the process data warehouse along with the description of the semantic concepts that we have created as the foundation for semantic process analysis; the process mining engine that mines the data in the data warehouse is described in Section 3.2; the BPI Cockpit that enables business users to conduct monitoring and analysis is outlined in Section 3.3. We conclude by discussing related work in Section 4, and by presenting some work in progress as well as our ''wish list'' in Section 5, along with some final remarks.

2. Process model and process logs

This section provides a brief introduction to process models and process execution logs. In particular, we will present the process model and execution log structure of *HP Process Manager* (HPPM), since this is the BPMS on top of which we built the BPI tool suite (that can anyway be ported on top of virtually any process model, with minor modifications). However, the same concepts and techniques are applicable to virtually any other BPMS.

In HPPM, a process is described by a directed graph, that has five different kinds of nodes:

- *Work nodes* represent the invocation of activities (also called *services*), assigned for execution to a human or automated *resource*.
- *Route nodes* are decision points that route the execution flow among nodes based on an associated *routing rule*.
- *Event nodes* denote points in the process where an event is notified to or requested from other processes.
- *Start nodes* denote the entry point to the process. Only one start node is allowed in a process.
- *Complete nodes* denote termination points.

Arcs in the graph denote execution dependencies among nodes: when a work node execution is completed, the output arc is ''fired'', and the node connected to this arc is activated. Arcs in output to route nodes are instead fired based on the evaluation of the routing rules. As an example, Fig. 1 shows the Expense Approval process. In the figure, boxes represent work nodes, diamonds are route nodes, triangles represent start and completion nodes, while pentagons represent event nodes. A process definition can be *instantiated* several times, and multiple instances may be concurrently active. Activity executions can access and modify process variables.

Every work node is associated to a *service description* that defines the logic for selecting a resource (or resource group) to be invoked for executing the work. The service description also defines the parameters to be passed to the resource upon invocation and received from the resource upon completion of the work. Several work nodes can be associated to the same service description. For example, both nodes *confirm with buyer* and *change or abandon* of Fig. 1 are associated to service description *send_email*, executed by the resource *email_server* (both of them send emails to a user).

When a work node is scheduled for execution, the BPMS reads the corresponding service description, executes the resource selection rule associated to the service description, and puts the work item to be performed into the resource's *worklist*. Resources periodically connect to BPMS, pick a work item assigned to them (or to a group to which they are member of), and then execute it. Details on the HPPM process model are provided in [25]. An introduction to BPMS in general is provided in [1].

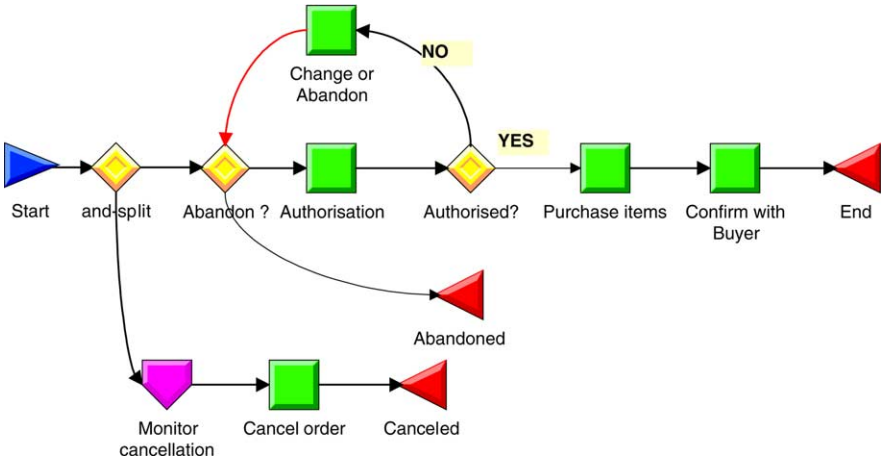

Fig. 1. The Expense Approval process.

BPMSs log information on process, node, and service executions into an *audit log* database, typically stored in a relational DBMS. The audit log database include the following information:

- *Process instances*: Activation and completion timestamps, current execution state (e.g., started, suspended, completed), and name of the user that started the process instance.
- *Service instances*: Activation and completion timestamps, current execution state, and name of the resource that executed the service.
- *Node instances*: Activation and completion timestamps and execution state for the most recent

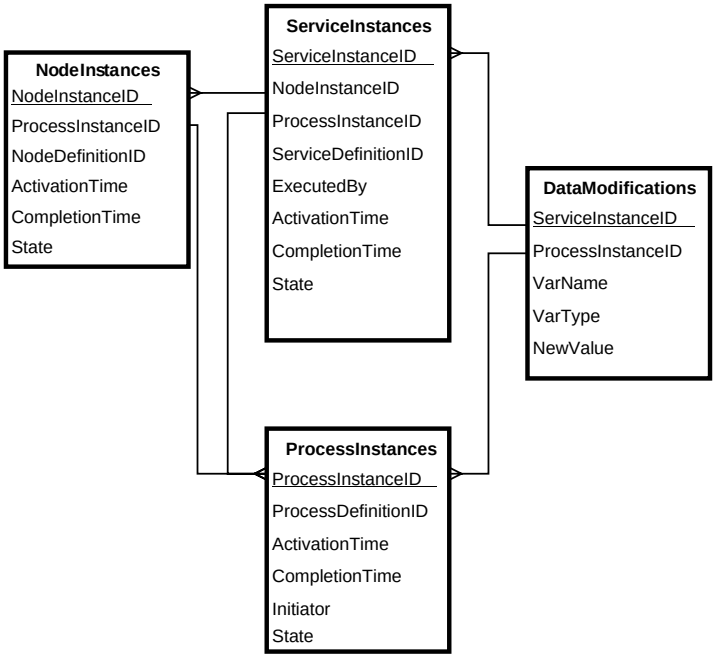

Fig. 2. Database schema of the HPPM audit log.

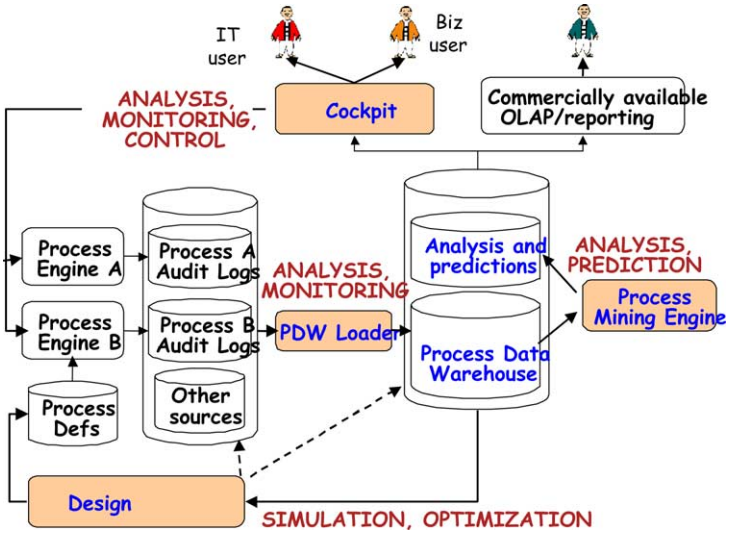

Fig. 3. Architecture of the Business Process Intelligence tool suite.

execution of the node within a process instance (multiple executions are possible if the process contains loops).
- *Data modifications*: New value for each data item every time it is modified.

Fig. 2 shows the audit log tables and their relationships. Underlined attributes denote primary keys, while links among tables denote foreign key constraints. For clarity of presentation, in the figure we have slightly simplified the structure of the tables. In addition to instance execution tables, authorized users can access *definition* tables, that describe the processes, nodes, services, and resources defined in the BPMS. A complete and detailed description of the HPPM audit log database schema is provided in [26].

3. BPI architecture and functionalities

Fig. 3 shows the overall architecture of the BPI suite. The suite includes three main components:

- The *PDW loader* extracts data from process logs, checks them for consistency, computes business-level metrics, and inserts the data into a *process data warehouse*, whose structure enables the easy definition and fast execution of detailed and aggregated reports, and provides a wide range of analysis functionalities.
- The *process mining engine* (PME) applies data mining techniques to data in the warehouse and derives sophisticated models to help users identify the causes of behaviors of interest as well as predict the occurrence of behaviors in running processes. The output of the process mining engine is stored in the "Analysis and Predictions" database. Analysis models help process designers in understanding the impact and behavior of new or modified processes; they are used during the redesign phase suggesting modifications that can increase the value of process quality metrics.
- The *Cockpit* is a graphical interface designed to provide reports to business users. It displays data in the warehouse in a way that is easy to consume, and allows analysts to define a variety of queries in an intuitive way, without writing code. In addition, it informs users of (actual or predicted) critical situations, and interacts with the process engine to avoid (or reduce the impact of) undesired situation.

In the following we describe each component, detailing their architecture and functionalities.

3.1. The process data warehouse, data loader, and core semantic concepts

The process data warehouse (or PDW for short) is the basic component of the BPI architecture.

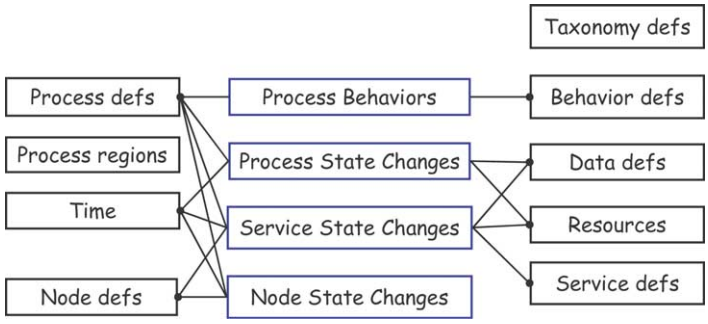

Fig. 4. The PDW schema (overview). Fact tables are in the center column, dimensions are on the side of the picture. For simplicity, links among facts or among dimensions are not shown.

It provides many analysis functionalities on its own, and supports the execution of all the other components in the BPI architecture.

3.1.1. The PDW and the data loader

The PDW is designed to simplify the definition and speed up the execution of many different kinds of queries, generating reports commonly needed by analysts. Indeed, querying the PDW yields response times that are several thousands of times faster than querying the HPPM log tables. The PDW organizes data according to a star schema (see Fig. 4), where process, service, and node state changes are the *facts* to be analyzed, while processes definitions, node definitions, service definitions, data definitions, resources, time, and behaviors (described next) are the *dimensions* under which the fact data can be analyzed. For example, users can examine how many process instances per fiscal quarter are initiated by a certain user (analysis of the *process* fact under the *time* and *resource* dimension), or the variance of the duration of service ''Purchase Items'' during weekends (analysis of the *service* fact under the *time* dimension). In addition, PDW contains a wide set of aggregated information describing typical performance metrics, such as the efficiency of a resource. The PDW Loader (PDWL) collects data from the log and loads them into the PDW. It can be activated periodically (at scheduled interval) or upon request. At loading time, PDWL checks log data for consistency and corrects erroneous information that could make the analysis more complex and error-prone.[1]

[1] This includes, for examples, special date codes inserted in the process or node termination timestamps to denote failures.

As Fig. 3 shows, data in the PDW can be directly accessed with a commercial reporting tool (such as Crystal Reports or Oracle Discoverer). Analysts can also access the PDW through the BPI Cockpit (defined later in the paper) that provides specific visualization techniques to report on business process executions. Details on the process data warehouse are provided in [2].

3.1.2. Semantic process analysis

The features described above are indeed very useful in their own right, and provide many advantages with respect to the state of the art. However, they only allow a low-level kind of analysis, which is not suitable for business users. In fact, while it is possible to retrieve all sorts of data about process and nodes executions, users, are still unable to get answers to *semantic* questions such as ''how many requests have been approved?'', ''what is the average duration of a negotiation with our partners?'', or ''how many processes are *too slow*''. The problem is that the concepts of ''approved'', of ''negotiation'', and of ''too slow'' exist only in the mind of the analysts, and are unknown to the BPMS. Therefore they cannot be measured and reported. To address this issue, we extended the PDW to allow *semantic process analysis*. Specifically, we provide the user with three conceptual tools: *behaviors*, *taxonomies*, and *process regions*.

A *behavior* enables the identification of process instances that have characteristics of interest to the analyst, possibly because they correspond to particularly high- or low-quality executions. Examples of behaviors are process instances lasting more than 10 days, or in which a certain loop is executed more than twice, or in which node ''cancel order'' is not executed.

Behaviors are defined by instantiating *behavior templates*. A template is a parametric definition of a behavior, such as ''*Instances of process P in which a node N has been executed more than T times*''. In order to define a behavior of interest for a specific process, users simply need to instantiate the template, i.e., provide values for the parameters. From the users' perspective, this is as simple as filling out a form. No coding is needed. Multiple specific behaviors to be monitored (on the same or different processes) can be defined for each behavior type, and a process can be analyzed for multiple behaviors. Behavior templates are implemented by means of SQL queries that, for each behavior (i.e., for each set of instantiation parameters, stored in the database), examine process instance data and select the process instances that had the behavior. A rather complete, although not exhaustive, set of templates that correspond to typical requirements of business analysts has been pre-built into PDW. However, if further requirements arise, PDW facilitates the creation of new templates that would only require SQL familiarity in case of complex mappings. Behaviors are at the core of semantic analysis because they constitute the mechanism by which low level technical metrics can be mapped to the high level ones, that exist only in the mind of business users, breaking in this way the existing gap between these two levels.

Analysts can also define *taxonomies* and instruct the PDW to classify process instances according to the defined taxonomies. For example, users can define classes ''successful'' and ''unsuccessful'' for a process, and specify ''success'' criteria. PDW will then automatically classify process instances and show the results to the users. The analysis of process instances based on user-defined classification criteria turns out to be an extremely powerful tool for understanding and improving process execution quality. Taxonomies can be defined by specifying the categories that compose the taxonomy. Each category is then associated to a behavior, with the meaning that the process instance is classified in a given taxonomy if the process instance has the corresponding behavior.

Behavior and taxonomy data can be observed under different dimensions. For example, users can analyze the occurrence of a certain behavior depending on the day of the week that a process started or on the resources involved in a process instance execution. However, a very useful form of analysis consists in analyzing *correlations* among behaviors or taxonomies. Correlations allow analyst to examine when or how often a behavior B (called *effect* behavior in correlation analysis) occurs among process instances that also have a behavior A (the *cause* behavior). In this way analysts can examine cause–effect relationships among behaviors, for example to identify sources of problems and undesired situations.

Fig. 5 shows a report obtained from PDW with Oracle Discoverer, highlighting the correlation among a cause behavior ''total order value >$ 15,000'' and the other behaviors defined for process ''Expense Approval'' that, in this example, are related to the process duration. Bars labeled ''effect bhv percentage'' show the percentage of occurrences of the *effect* behaviors independently of the cause behavior, while the bars labeled ''correlated percentage'' shows the percentage of instances that had the effect behaviors among those who also had the cause behavior, thereby highlighting cause–effect relationships. The figure shows that when the total order value is more than $ 15,000, then the processes typically last longer, and in particular almost two thirds of the time they last more than 10 days (while on average ''Expense Approval'' processes last more than 10 days in less then 8% of the cases). If they need to track the problem of excessive duration further, analysts can then examine other correlations, for example to understand which part of the process contributes to the delays in orders above than $ 15,000, until they are able to pinpoint the source of the problem and therefore devise a solution.

Another PDW concept that supports semantic analysis is the *process region*. A process region is a part of a process that can be treated as a unit from a business analysis perspective. The purpose of process regions is to bridge the gap between the *conceptual* process, designed by business analysts, and the *actual* (implemented) process, which typically include many nodes to implement a single conceptual step. Fig. 6 illustrates this concept. Through the definition of process regions, analysts can get reports at the region level, rather than at the node level.

Regions are defined by selecting the starting node s and a set of ending nodes E in an underlying process P. At most one node e in E can be a non-terminal node of P (i.e., a node with no output arcs). Regions need to

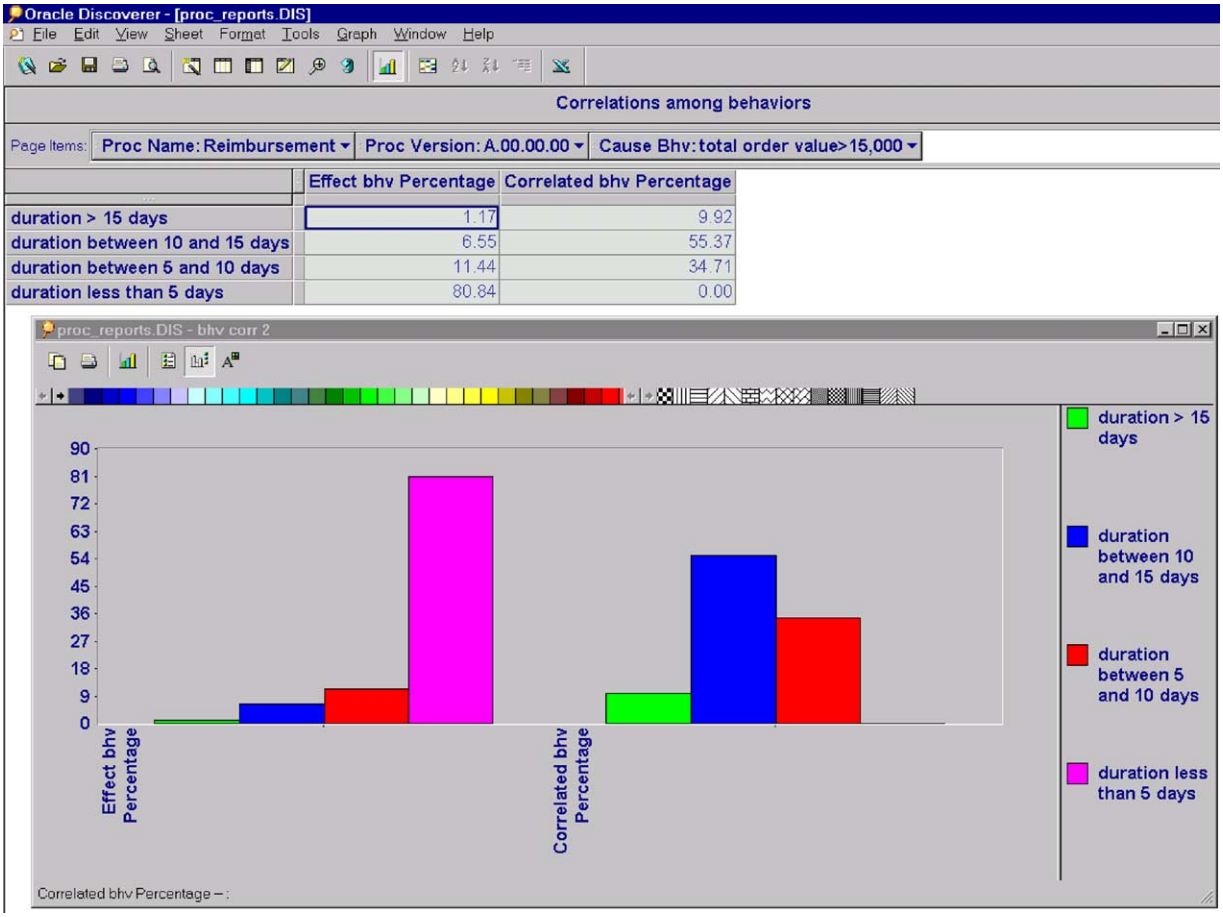

Fig. 5. Example of correlation among behaviors (report obtained with Oracle Discoverer).

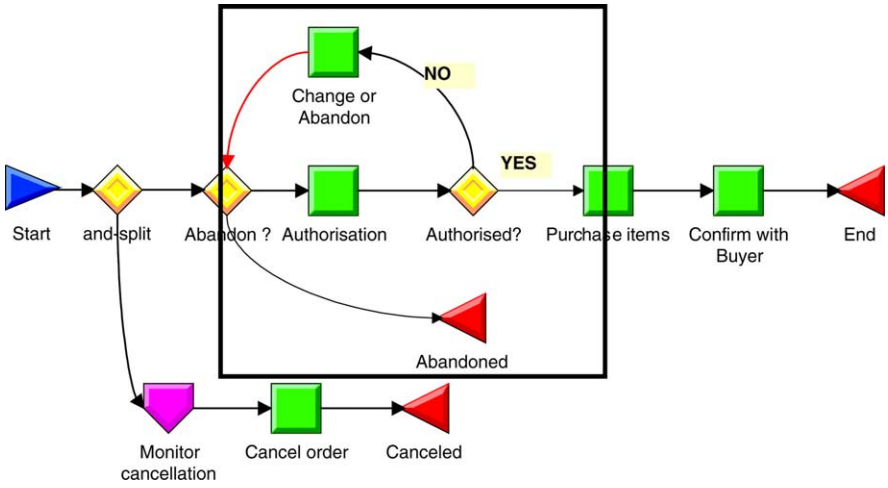

Fig. 6. Example of a process region.

satisfy certain constraints. In particular, s and E must identify a subgraph SG of P, such that:

1. s has only one input arc, and this arc comes from a node outside SG.
2. The only node e in E who is a non-terminal node of P has only one output arc a, that connects e to a node that does not belong to SG.
3. Every other arc in SG must only connect nodes in SG.

3.2. The BPI process mining engine

The PDW allows for a wide range of data analysis functionalities. However, it is a ''passive'' component: it is up to the user to ''guess'' the possible causes–effect relationships among behaviors. The PDW can then help the users in verifying whether the assumptions are correct or not, for example by exploiting behavior correlation analysis.

The *BPI process mining engine* (PME) allows a more ''intelligent'' kind of analysis by executing data mining algorithms on the warehouse data in order to:

- Understand the causes of specific behaviors, such as the execution of certain paths in a process instance, the use of a resource, or the (in)ability to meet service level agreements.
- Generate prediction models, i.e., information that can be used to predict the behavior and performances of a process instance, of the resources, and of the BPMS.

The PME achieves this goal by mapping the behavior analysis problem to a *classification* problem [3], where process instances are the *objects* to be classified, and the classes are ''have'' or ''have not'', respectively, representing process instances that have or do not have a certain behavior. The PME analyzes *labeled* data in the PDW, i.e., data of process instances that have been labeled with the class corresponding to a certain behavior. This data constitutes the training set since it is the data that PME will analyze to derive a set of *classification rules*, i.e., mappings from a condition on the process instance attributes to a class, with the meaning that process instances whose attributes satisfy the condition belong to the specified class. Therefore, classification rules identify the characteristics of the process instances in each class, in terms of values of the their attributes. For each classification rule, the classifier also provides information about the rule's *accuracy*, i.e., about the probability that classifications performed with the rule are correct. Once the characteristics of the process instances in each class have been identified, the user can have a much better understanding of why processes have a certain behavior. For example, analysts may want to identify the characteristics of the ''Expense Approval'' processes that have the behavior ''last more than 15 days''. The PME can process data in the warehouse and return the following classification rule: instances that involve orders for more than $ 18,000 and in which node ''Authorization'' is executed by John *have* this behavior, with 80% accuracy.

We next describe how this is done in the proposed approach. Although Sections 3.2.1 and 3.2.2 focus on behavior *analysis*, we will show in Section 3.2.3 that most of that description applies to the *prediction* of behaviors as well.

3.2.1. Process mining phases for behavior analysis

The approach to analyze why instances of a certain process are affected by a specific behavior is composed of four phases: *process data preparation*, *exception analysis preparation*, and *mining and interpretation*.

The first phase, named *process data preparation*, presents a number of challenges. First, data has to be cleansed, which requires identifying noisy data. For example, some fields adopt meaningless default values when some kind of error takes place, or humans involved in the process may input typos and abbreviations. This noise has to be eliminated before data can be mined, otherwise the mining algorithms will be mislead by it. Ideally, this process should be automatic. For the moment, only anomalies in data that has been entered manually, i.e., typos, abbreviations and misspellings, can be automatically detected and corrected using a tool that we describe in [4].

Once data has been cleansed, we need to prepare a process-specific table, called *process analysis* table, that includes one row per process instance, and where the columns correspond to process instance attributes. One additional column is needed to store labeling information. Note that the need of having a single table with one tuple per element to be considered in the analysis is a requirement posed by virtually any

data mining tool. However, there are a number of issues that complicate this task. On one hand, unlike traditional classification problems, the information about a single object (process instance) in the PDW (data warehouse) is scattered across multiple tables, and each table may contain multiple rows related to the same instance. On the other, we need to define the set of attributes to be used as the behavior descriptor.

Defining which attributes will compose a descriptor is not trivial. Part of the problem is that, even within the same process, different instances may have different attributes. In fact, a node can be activated a different number of times in different instances. The number of such activations is a priori unknown. Hence, not only do we have to identify which are the interesting node execution attributes to be included in the *process analysis* table (that, as said above, needs to contain one tuple per process instance), but also how many node executions (and which ones) should be represented. This issue can be addressed in several ways: for example, we could decide that if a node can be activated multiple times, then we consider for our analysis only a specific node execution (e.g., the first one or the last one). An alternative approach consists in considering all node executions. In this case, the process analysis table must have, for each node, a number of columns proportional to the maximum number of executions of that node, determined by looking at the process instance data in the warehouse. This approach is appropriated when the number of loops is small (less than 10 for example) and does not differ significantly from instance to instance. Otherwise (if the number of loops is high and greatly differs from instance to instance), despite the fact that this technique provides more information to the mining phase, it does not necessarily give better results. In fact, tables generated in this way include many undefined (null) values. Data mining tools use different methods to fill in missing values during data preprocessing step. The undefined values in our case cannot be handled in this way, because they are not due to incomplete data, but they represent values that do not exist. In addition, when classifications are based on a large number of similar attributes that often have null values, it is very difficult to interpret and understand the results. Finally, this approach can be computationally heavy. The approach we followed for these cases consists in inserting two attribute (column) sets for each node that can be executed multiple times: one to represent the *first* execution and the second to represent the *last* execution of that node. This is due to the observation, from several experiments we have conducted on different processes, that the first and last executions of a node in the process have a higher correlation with many kinds of process behaviors, such as those related to process execution time and to the execution of a given subgraph in the process. In addition, in both cases the number of node activations is added as a attribute in the *process analysis* table.

Independently of whether there are different attributes for the same process or not, to choose the set of descriptor attributes, process attributes have to be evaluated to assess their relevance or discriminating power, i.e., their influence or effect on the behavior (class) being analyzed. This corresponds to *feature selection* in machine learning. The problem is to select a relevant subset of features upon which the learning algorithm will focus its attention, while ignoring the rest. Basically, some evaluation function (metric) that can be used on a single feature is used. All the features are independently evaluated, a score is assigned to each of them, and features are sorted according to the assigned score. Then, a predefined number of the best features is taken to form the feature subset. The number of features to select is an experimental issue. The scoring function can be any of the classical ones: information gain, chi-square, expected cross entropy, or mutual information. Some classification techniques, like support vector machines, make use of all the features given to them, in which case it becomes necessary to first select a subset of features. Others, like decision trees, even though they select relevant features as part of their induction process (in decision trees, an evaluation function constitutes the criterion for deciding on splits at the nodes of the tree), in practice they are known to degrade in performance when faced with features that are not necessary for learning the desired classification [5]. Furthermore, when classification rules are used for analysis purposes, there is an added level of complexity where there is no formula to evaluate the relevance of attributes since it is based only on domain knowledge. For example, in one experiment to analyze the behavior ‘‘last more than 15 days’’, the attribute ‘‘duration

of node Authorized'' was found as relevant by the chi-square metric, and it even originated the classification rule ''if duration of node Authorized is more than 15 days, then duration of the process is more than 15 days''. However, this rule did not convey any valuable information, therefore, attribute ''duration of node Authorized'' had to be eliminated from the set of descriptor attributes. In our implementation, we have configured the tool to select the attributes that have shown higher correlations with behaviors in the tests we have performed. In particular, the *process analysis* table includes the following attributes for each process instance:

- *Activation and completion timestamps:* These actually corresponds to multiple columns that decompose the timestamps in hour of the day, day of the week, etc., and with the addition of the *holiday* flag to denote whether the process was instantiated on a holiday.
- *Data items:* Initial values of the process data items, plus the length (in bytes) of each item.
- *Initiator:* Resource that started the process instance.
- *Process instance duration.*

In addition, the process analysis table includes attributes for each node in the process (two sets of attributes are included for nodes that can be executed multiple times, as discussed above):

- *Activation and completion timestamps* (decomposed as described for the process instance timestamps).
- *Data items:* Values of the node output data, plus the length (in bytes) of each item.
- *Resource that executed the node.*
- *Final state of the node* (e.g., completed or failed)
- *Node duration.*
- *Number of activations* of the node in the process instance (this attribute is only included once per node, even if two attribute sets are used for this node, since the value would be the same for both).

The process analysis table is automatically built by a *process analysis preparation* PL/SQL script. This script takes the name of the process to be analyzed as input parameter, and retrieves process definition information from the BPI warehouse. In particular, the script identifies the nodes and data items that are part of the process, and creates the *process analysis* table. Then, the script populates the table with process instance data. Users can also restrict the process analysis table to contain only data about instances started within a time interval.

The *behavior analysis preparation* is the second phase and it joins in a single view the information generated by the previous phase with the behavior labeling information (stating whether the instance is exceptional or not) computed by BPI at behavior definition time. This phase is implemented by process- and behavior-independent PL/SQL code that receives as parameter the name of the process and of the behavior to be analyzed, and generates a process- and behavior-specific view. The view joins the *Process Analysis* and *Process Behaviors* tables (the latter is a process- and behavior-independent table that lists which instances have been affected by which behaviors), to provide a data set that includes process instance attributes as well as labeling information. The obtained view includes all the information required by the classification tool to generate the classification rules.

The third phase is *mining*, which applies data mining algorithms to the data generated by the data preparation phase. A variety of data mining and classification applications are available on the market. Therefore we did not develop our own mining algorithms, but expect that a commercial or off-the-shelf tool be employed, at least in the preliminary versions of the BPI tool suite. At this stage, our contribution therefore is not in the data mining algorithm, but rather in designing and developing the components that map the behavior analysis problem into a classification problem. Further contributions will be to enrich the mining features by integrating new algorithms to perform other analysis tasks, and to quickly adapt to the evolving nature of some processes.

In particular, we typically use *decision trees* [3] for exception analysis. Decision trees are widely used because they work well with very large data sets, with large number of attributes, and with mixed-type data (e.g., continuous and discrete). In addition, they are relatively easy to understand (even by non-expert users), and therefore simplify the *interpretation* phase. A decision tree is learned from the training set (labeled process instances) by a top-down induction algorithm. A splitting criterion is used to determine which attri-

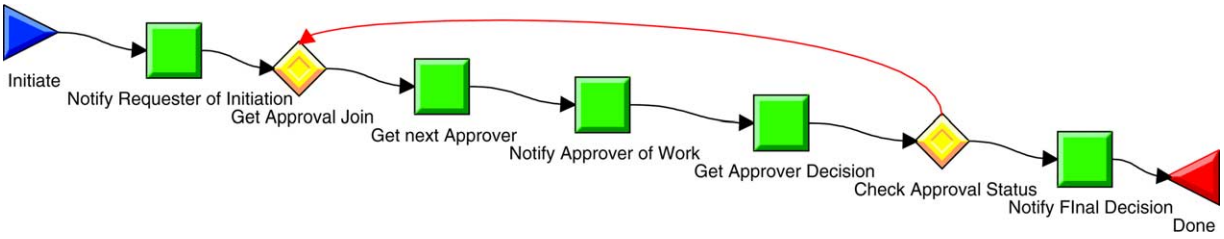

Fig. 7. Simplified view of the Expense Approval process used for our experiments.

bute is the best to split that portion of the training data that reaches a particular node. The splitting process terminates when the class values of the instances that reach a node vary only slightly or when just a few instances remain. With decision trees, objects are classified by traversing the tree, starting from the root and evaluating branch conditions (decisions) based on the value of the objects' attributes, until a leaf node is reached. All decisions represent partitions of the attribute/value space, so that one and only one leaf node is reached. Each leaf in a decision tree identifies a class. Therefore, a path from the root to a leaf identifies a set of conditions and a corresponding class, i.e., it identifies a classification rule. Leaf nodes also contain an indication of the rule's accuracy, i.e., of the probability that objects with the identified characteristics actually belong to that class. Decision tree building algorithms in particular aim at identifying leaf nodes in such a way that the associated classification rules are as accurate as possible. In our experimentations, we used SAS to generate the decision trees.

In the fourth phase, *interpretation*, the decision trees generated by the mining tool in the previous phase, are used by analysts to focus on the leaf nodes that classify instances as having a given behavior. Then, they can traverse the tree from the root to the leaf, to identify which attributes and attribute values lead to the leaf node, and therefore identify the characteristics of the instances exhibiting that behavior. In particular, problems and inefficiencies can be identified, and consequently, addressed and removed.

3.2.2. Experimental results in behavior analysis

We have applied the BPI approach and toolkit to analyze several administrative processes within HP, such as electronic employee reimbursements and requests for disbursement vouchers. These processes are implemented on top of HP Process Manager, and are accessed by hundreds of employees per day. As a representative example, we discuss the results obtained in analyzing the *Expense Approval* process described in Fig. 7, and specifically in identifying the characteristics of instances that take more than 8 days to complete (the average execution time was about 5 days). We had access to 5 months of process execution data, corresponding to approximately 50,000 process instances. About 15% of the instances were affected by this behavior.

After importing the process instance data into the BPI warehouse and having defined the behavior, we ran the scripts described in the previous section to label the instances and generate the exception analysis table. We next used SAS Enterprise Miner (a leading commercial data mining tool) for the generation of decision trees. In the preparation of the decision tree we used chi-square as splitting criteria, and we used the proportion of correctly classified records as assessment value. Sixty percent of records were used as training data while 40% of records were used for validation. These are the parameters that gave us the best overall results.

After several failures, that led us to restructure database schemas and preparation scripts as described earlier in this section, the tool finally produced interesting results. For both simplicity and confidentiality reasons, we do not show the exact decision tree and some details of the results. However, we summarize the main findings in Fig. 8. The results show the following:

- Cases that required many expense approvers were more likely to last longer. In particular, instances lasting more than 8 days typically had more than six approvers.

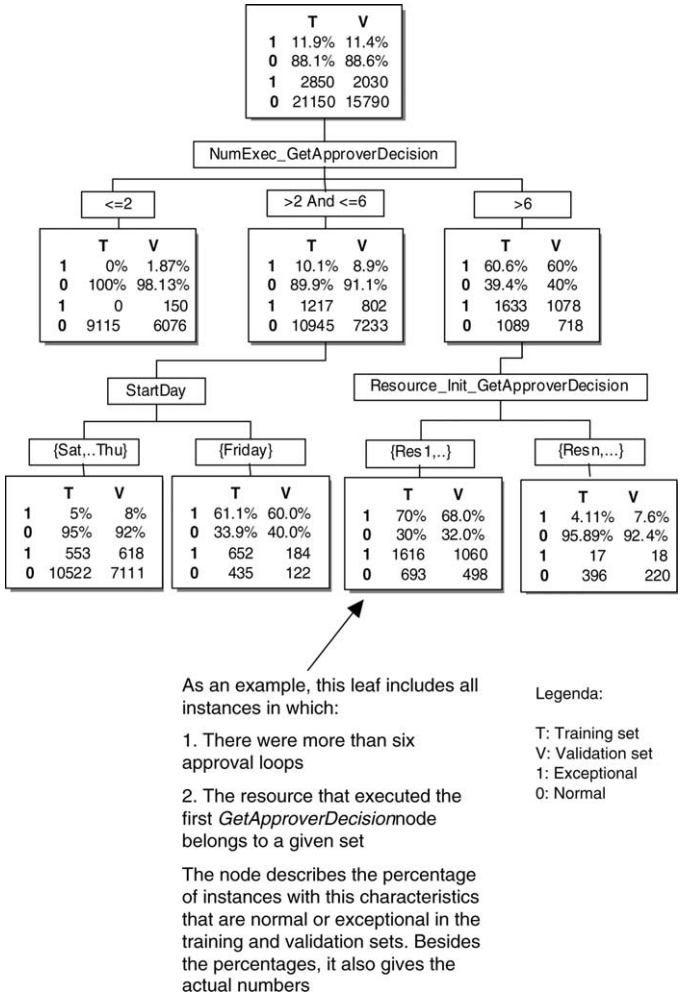

Fig. 8. Simplified decision tree obtained by analyzing the Expense Approval process.

- When, in addition to having more than six approvers, clerical activities[2] were executed by employees in a specific group, then 70% of the instances exhibited the ''more than 8 days'' behavior. The majority of process instances were instead on time when such clerical activities were executed by other employees.
- Process instances started on Fridays were more likely to last longer, since the work was in fact postponed until the next Monday.

[2] The first approver is typically a clerk that verifies that the request is formally correct and that payments can be made.

As it often happened in our analysis, some of the identified correlations are not immediately helpful in order to understand the behavior. For example, the fact that processes with more approvers (and therefore more node executions) last longer is to be expected. However, some of the identified correlations are often useful to isolate bottlenecks and, in general, aspects of the process or of the organization that can be improved. For example, following the discovered correlation between behaviors and the resource group, a further look at the BPI warehouse revealed that employees in that group had more workload than others. Hence, the analysis allowed spotting a problem

and suggesting a possible solution, by reassigning the work appropriately.

Business process intelligence, just like any other business intelligence application, requires care in interpreting the results and in identifying biases due to the kind of data that is available. For example, a problem characteristic of BPI is the *border* effect: typically, the analysis is performed on processes started (or completed, or both) within a certain time window. For example, we may have a data set containing all instances completed in October. If a mining tool analyzes these data, it will determine that instances started in spring lasted longer than those started in the summer or fall. Indeed, the tool will state that the accuracy of this rule is very high. However, the result is only due to how data are collected, rather than to a property of the process: in fact, the data set is polarized with respect to the start date, in that it contains instances started in spring only if they lasted very long, i.e., until October. A formal analysis of this and other typical biases is part of our future research agenda.

3.2.3. Prediction of behaviors

Besides allowing behavior analysis, the PME can also derive prediction rules, used by other BPI components (and specifically by the Cockpit) to speculate about the possible occurrence of a behavior in a running process instance. In particular, we aim at predicting abnormal behaviors as early as possible so that they can be *prevented,* or so that at least adequate expectations about the process execution speed and quality can be set.

Our approach to behavior prediction includes the four phases described in the previous section. However, there are a few differences that must be taken into account. In particular, the *process data preparation* phase must face additional challenges. The problem is that classification rules generated by behavior analysis work very poorly (and may not even be applicable) for predictions about running instances. In fact, we want to classify process instances as having or not having a certain behavior while they are in progress, and possibly in their very early stages. Therefore, the value of some attributes (such as the executing resource or the duration for a node yet to be executed) may still be undefined. If the classification rules generated by the behavior analysis phase include such attributes, then the rules cannot be applied, and the process instance cannot be classified. For example, assume that decision tree-building algorithms have been used in the mining phase. If undefined attributes appear in the branch conditions of the decision tree, then the branch condition cannot be evaluated. The prediction becomes less accurate as the undefined attributes appear in branch conditions closer to the root of the tree, since we can only follow the tree (and improve the classification accuracy) while branch conditions can be evaluated.

We address this issue by modifying the *process data preparation* phase so that it generates several different process analysis tables (that will eventually result in several different classification rule sets), each tailored to make predictions at a specific *stage* of the process instance execution. A stage is characterized by the set of nodes executed at least once in the instance. For example, the process analysis table targeted at deriving classification rules applicable at process instantiation time is prepared by assuming knowledge of only the process instance input data, the starting date, and the name of the resource that started the instance. In this way, only these attributes will appear in the classification rules.

The other phases are executed as discussed in the previous section, with the difference that they are performed once for every table generated by the process data preparation phase. At the same time there is a simplification given by the fact that the predictive attributes (those in the nodes of the decision tree) are definite. The added level of complexity corresponding to domain-based feature selection does not apply here. As explained in Section 3.2.1, in behavior analysis this kind of feature selection is necessary to eliminate attributes from the classification rules when they do not convey any valuable information to the business analyst. In contrast, in prediction, the user does not care about the information provided by the rules themselves, only about whether the given behavior will occur or not for a running process instance, i.e., the prediction itself. Hence, the user will not need to get into a loop (rule analysis, attribute elimination from the feature set when they appear in rules that have no value, and learning of a new classification rule) to fine tune the rules. In addition to the phases common with behavior analysis, behavior prediction also includes a *prediction* and a *reaction* phase.

The *prediction* phase is where predictions on running process instances are actually made. In this phase, classification rules are applied to live instance execution data, to classify the instances and obtain, for each running instance and each behavior of interest, the probability that the instance will be affected by the behavior.

In the *reaction* phase, users or systems are alerted about the risk of abnormal behaviors, and take the appropriate actions to reduce the ''damage'' caused by these behaviors, or possibly to prevent their occurrence. The BPI toolkit can be configured to proactively interact with the BPMS in an attempt to prevent a behavior. Currently, the only allowed form of automated intervention consists in raising the process instance priority for those instances that are likely to be late. The process administrator can specify the level to which the priority can be raised depending on the probability of the process instance being late. Furthermore, it is possible to send events to the process instance. The event is then handled as specified by the process definition. In the future we plan to extend the automatic reaction capabilities to:

- Modify process instance and work node priorities based on the risk and cost of missing SLAs.
- Modify the resource assignment policies so that activities are given to faster resources.
- Influence decision points in the process, so that the flow is routed on certain subgraphs, if this can help avoid the behavior while still satisfying the customers and process goals (although perhaps causing increased process execution costs).

We have conducted a number of experiments to apply this approach for behavior prediction to HP administrative processes. Here we only summarize our findings to predict the behavior ''lasts more than 15 days'' for the Expense Approval process of Fig. 7. A more complete description can be found in [6]. In the Expense Approval process, it was possible to have a good prediction for the instance duration at the very start of the instances. In fact, the resulting decision tree revealed that the length of the process is correlated to the name of the requester (i.e., the creator of the instance) and to the length of data item *Approver* that contained the names of the approvers (and therefore its length indicated the number of loops to be executed). For some combinations of these values, over 70% of the instances were affected by that behavior.[3]

As expected, predictions get more and more accurate as process instance execution proceeds, since more information about the instance becomes available. In particular, very accurate predictions could be made right after the execution of node *Get Approver Decision*. In fact, this activity is the most time-consuming one, and therefore after its execution it is possible to have more information about the likelihood of the process exceeding the acceptable execution time. The decision tree for predicting the duration behavior at this stage of the process shows in fact that if the first execution of node *Get Approver Decision* takes slightly more than 5.5 days, then the instance can be predicted to have the behavior ''lasts more than 15 days'' with 93% probability.

We observe here that the duration of this node also appeared in the rules generated while *analyzing* this exception. However, in that context, we had removed this attribute from the input data set, since it was not particularly helpful in identifying ''interesting'' correlations and in understanding *why* the behavior occurs. In this case instead, our focus is on predictions, and any correlation becomes useful.

In general, we experienced that there are only a few stages in a process instance where the accuracy of the prediction improves, typically after the execution of some ''critical'' nodes. An interesting optimization of our algorithms could therefore be based on the identification of such stages, so that the various behavior prediction phases can be executed for these stages only.

3.2.4. Analysis of decision points

BPI plays an important role in the analysis of decision points in a business process. This analysis can help analysts to redesign processes in order to improve their quality and is realized as follows. Process execution data generated during execution of one or more instances of a business process involving a set of one or more decision points is accessed from the PDW. Based upon these data, PME builds a predictive quantitative model comprising a set of one

[3] Given that only 15% of the instances exhibit that behavior, predicting that behavior with a 70% accuracy is a quite interesting result.

or more rules correlating context data at different stages of the business process with business process outcomes at one or more decision points in the business process. Different contexts are obtained by partitioning the business process into a set of logical stages given by a sequence of process regions (Section 3.1.2). After the business process has been partitioned, classification rules are generated for predicting outcomes of one or more decision points in the business process based upon context data available at the different logical stages. The prediction approach described in Section 3.2.3 is applied for this purpose but instead of predicting process instance behaviors, here we are interested in the prediction of decision point behaviors. For each possible outcome O_i of a decision point and each logical stage, PME learns a classification rule to predict whether the decision point will "have outcome O_i" or not ("having outcome O_i" being the class) for a running process instance.

After correlations between context data at different stages of the business process with business process outcomes at one or more decision points in the business process have been found, they can be presented to process designers in the following format:

> Given a graph G that represents the process definition of business process, after the activities in a subset G′ of the original graph have been completed and for a certain set of values for process variables and time-related data, a particular path P at a given decision point DP will be chosen with *X*% certainty.

This valuable information helps process designers to understand the outcomes of decision points and to redesign process models to improve their quality. For example, if a rule indicates that when an expense approval is submitted on Friday, the outcome of decision point "Authorized?" is "no", then there is no point in going through all the processing done at previous nodes in the process. In such a case, by inserting a new decision point "Submission date" that checks the submission date right at the beginning and if it is Wednesday follows a path that gets joined to the "no" path of the "Authorized?" decision point, the process can be made more efficient. In other situations it can be discovered that a decision point can be moved to an earlier stage of the process or that the path leading to a decision point can be decomposed into two concurrent paths since only some nodes in that path affect the decision taken at the decision point.

For example, the following rule tells us that a particular vendor (called company1) is definitely not a good choice for purchasing a Laser Printer LP1100. This rule can be utilized only when all of the activities in sub-graph G′ are already executed. This means, only after all activities in sub-graph G′ are executed we will have enough data to make a good prediction about the outcome of the decision point. A possible course of action as a result of this rule might be immediately canceling the order from that particular vendor and submitting the order to another vendor.

> Given that all activities in a sub-graph G′ have been completed.
> If vendor = "company1" and product = "Laser Printer LP1100"
> Then outcome "Cancel" will be chosen
> With 95% certainty.

3.2.5. Process mining engine architecture

Fig. 9 details the architecture of the PME. A Process Data Preparation application reads process instance and behavior information from the PDW, selects the features, and outputs the data (training set) in a format suitable for the classifier (typically, a relational table). The classifier then generates the rules and stores them in the "Analysis and Prediction" database. Rules can then be viewed by analysts (for example in the form of a decision tree, as shown in Fig. 8) to understand the causes of behaviors. In some cases, analysts may want to repeat the classification after removing some features from the ones included in the training data set, to force the classifier to focus on specific characteristics in which they are interest (for example, the resources) or to discard features that are meaningless for their analysis as explained in Section 3.2.1.

3.3. The BPI Cockpit

The main goal of the Business Process Cockpit is to enable *business* users to perform *business-level* quality

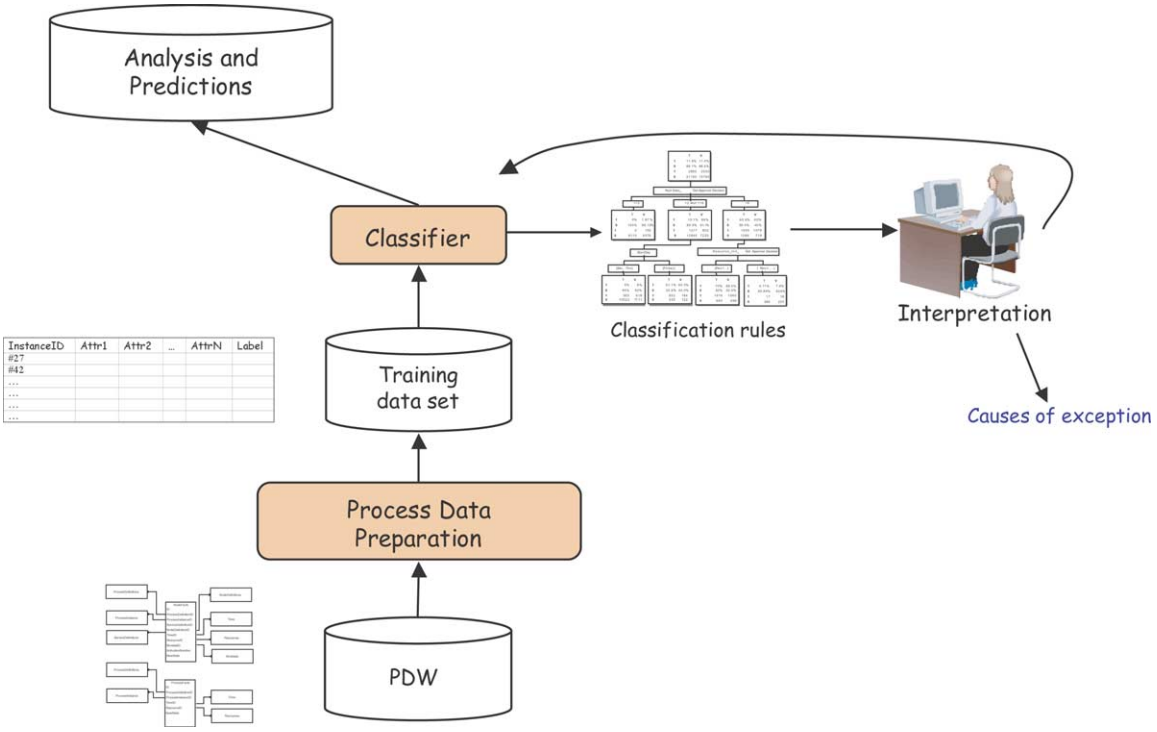

Fig. 9. Architecture of the process mining engine.

analysis, monitoring, and management of business processes. Targeting business users implies keeping the user interface as simple and immediate as possible, without limiting the flexibility and the functionalities provided by the tool. Providing business-level analysis and monitoring involves the development of techniques to enable users to define, monitor, and measure business quality metrics. To this end, the Business Process Cockpit (BPC) provides three main functionalities:

- It displays a variety of reports to business and IT users. Unlike general-purpose reporting tools, the BPC is aware of the semantics of the data in the PDW, and includes visualization concepts and techniques specifically designed to display business process execution data. In addition, the BPC can be used to configure the PDW, for example to define behaviors or taxonomies. Both reporting and PDW configuration can be performed in an intuitive manner, without requiring any code.
- It monitors processes, services, resources, and other process-related entities, and inform users of actual or foreseen quality degradation. BPC can also send notifications to users on the medium of their choice.
- BPC can manage running processes by tuning process and system configuration parameters (such as the process priority) and by notifying *events* to processes.

The BPC visualizes process execution data (and related quality metrics) according to different *perspectives.* A perspective identifies the process entity that is the focus of the analysis. For example, under the service perspective, the user will be able to see statistics and metrics about the web services invoked during the execution of business processes. After discussing with consultants and customers, we have identified the following perspectives as being relevant in business-level analysis:

- *System* (*overall*).

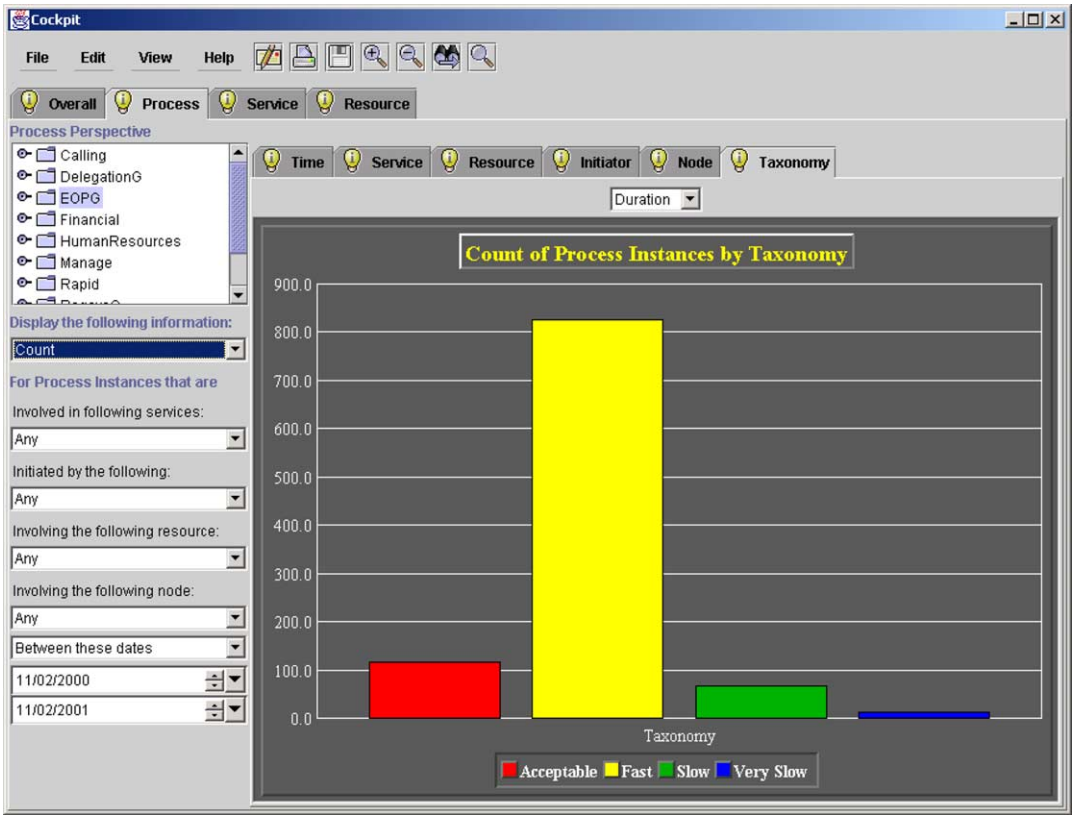

Fig. 10. Screen shot from Business Process Cockpit: process focus and taxonomy perspective.

- *Processes*: Displays information about a specific process or set of processes.
- *Resources*: Shows data related to individual resources or group of human or automated resources, such as the resource rating or performance.
- *Services*: Displays data related to e-services invoked within the execution of processes.

For each perspective, the BPC can present basic statistical information (such as average execution time and performance), value-related information (associated revenues and costs), and behavioral information. In addition, when focusing on a specific perspective, the BPC can then narrow the analysis also based on other perspectives, similarly to what is done in multidimensional analysis. For example, when focusing on a specific *process* (process perspective), it is possible to analyze the process in general, or with respect to specific *time* values (e.g., analyze instances started in a certain time window) or to specific resource values (e.g., analyze only instances in which a specific resource was involved). As an example, Fig. 10 shows the distribution of process instances among the categories of a *duration* taxonomy. The duration taxonomy has four categories: *fast* (defined as processes lasting less than 5 days), *acceptable* (between 5 and 10 days), *slow* (between 10 and 15 days) and *very slow* (more than 15 days). More details on the BPI Cockpit can be found in [7].

4. Related work

In this section we examine prior art in the BPI area and discuss its relationships with respect to the scope and approach presented in this paper. In this presentation we group the related work along the following categories: workflow history management,

workflow mining, and workflow controlling and optimization.

4.1. History management

Several research projects deal with the technical facilities necessary for the logging of the audit trail data. Muth et al. [8] address the problem of workflow history management in virtual enterprises. In order to provide a light-weight architecture as a low cost solution for enterprises, they propose to minimize the workflow history by storing selected and aggregated data. To this end, they extend the workflow specification (expressed in terms of state and activity charts) by means of orthogonal state charts components and corresponding activities for history tracking. In this way the process of tracing a workflow's execution can itself be seen as a workflow. The prototype Mentor-lite implements history management as a workflow on top of a lightweight kernel.

An approach for the tracking of history information in a distributed workflow management system is presented in [9]. The authors focus on the structure and querying of the history, in a fully distributed architecture realized in conformance with Object Management Architecture (OMA) of OMG. Within this context, several strategies for evaluating queries against the workflow history are represented and their execution cost is compared.

The design of a data warehouse for workflow logs is discussed by zur Muehlen [10] and Eder et al. [11]. zur Muehlen [10] develops a taxonomy of different analysis purposes for workflow audit data. This taxonomy helps in determining the relevant data for analysis and their aggregation. They distinguish between workflow monitoring, which describes the analysis of active workflows and workflow controlling, which deals with the post-analysis of completed instances. Other criteria of differentiation are the purpose of the analysis (for technical or business-oriented analysis goals) and the type of user (individual user or organization). The author also discusses data warehouse design issues derived from the integration of the business data accessed by the applications which is not stored by the workflow management system. Our approach and tool allow for all the types of analysis described. Moreover, the knowledge derived from the analysis of past instances can be used for controlling active instances by predicting undesired behavior and trying to avoid it.

Eder et al. [11] focuses on the data warehouse design based on a general workflow metamodel that specifies the relation between the workflow specification description and workflow execution characteristics. The proposed data warehouse hypercube has the following six dimensions: workflows, participants, organization, servers, time and measures. Our model, developed in parallel, has similar concepts, although it includes dimensions that focus on semantic aspects. In particular, the behavior dimension allows a semantic process analysis by mapping low-level measures on high level quality measures defined by business users.

4.2. Workflow mining

Data mining in the context of workflow logs to discover various kinds of information about the workflow instances is addressed in several papers, such as [12–15] (a detailed survey of this research area is provided in [16]). Cook and Wolf [12] and Agrawal et al. [13] propose methods for automatically deriving a formal model of a process from a log of events related to the executions of a process that is not supported by a process management system. The work is limited to sequential processes. Starting from the same kind of process logs, van der Aalst and van Dongen [17] derive a workflow model based on Petri nets and incorporate time information like minimal, maximal and average time spent in a certain stage of the process. This timing information is attached to places of the derived Petri net-based workflow model. They propose also a XML-based format for storing and exchanging workflow logs. Herbst [14] and Herbst and Karagiannis [15] present an inductive learning component used to support the acquisition and adaptation of sequential process models, generalizing execution traces from different workflow instances to a workflow model covering all traces.

Compared to these approaches, our work emphasizes application areas where the workflow process model is known. The mentioned approaches address the design definition phase of the business process reengineering process, whereas our work aims at supporting business-level execution analysis and prediction, as well as process re-design. We start from

existing process definition and analyze process logs in order to improve and reengineer the process model.

To the best of our knowledge, there are no approaches to process execution analysis and prediction based on data warehousing and data mining techniques for Business Process Intelligence. A few contributions exist in the field of exception prediction, however, limited to estimating deadline expirations and based on simple statistical techniques. In the following we summarize these contributions and then we underline the main differences with the approach proposed in this paper.

One of the first contributions to time management is provided by Panagos and Rabinovich [18], and builds on work in the real-time system domain. The authors address the problem of predicting as early as possible when a process instance is not likely to meet its deadline, in order to escalate the problem and take appropriate actions. In the proposed workflow model, every activity in the process has a maximum duration, assigned by the workflow designer based on the activity's estimated execution times and on the need to meet the overall process deadline. When the maximum duration is exceeded, the process is escalated. When an activity executes faster than its maximum duration, a slack time becomes available that can be used to dynamically adjust the maximum durations of the subsequent activity. This activity can take all the available slack or a part of it, proportional to its estimated execution time or to the cost associated to escalating deadline expirations.

The approach of Panagos and Rabinovich [19] also aims at estimating process durations and deadline expirations. Estimates are obtained by taking into account the state of the process and the statistics collected during past executions. In particular, the activity completion time is estimated based on the current load of the system resources and on the average execution time of the activity, calculated on past executions. Estimated activity execution times are then used to estimate the duration of the process, and to decide whether the process should be escalated.

Eder et al. [20] propose a technique for deadline monitoring and management. In the proposed approach, a process definition includes the specification of the expected execution time for each activity. This duration can be defined by the designer or determined based on past executions. In addition, the designer may define deadlines for activities or for the whole process. Deadlines specify the latest allowed completion times for activities and processes, defined as interval elapsed since the process start time. Process definitions are translated into a PERT diagram that shows for each activity, based on the expected activity durations and on the defined deadlines, the earliest point in time when the activity can finish as well as the latest point in time when it must finish to satisfy the deadline constraints. The PERT technique is extended to handle process definitions that include alternative and optional activities. During the execution of a process instance, given the current time instant, the expected duration of an activity, and the calculated latest end time, the progress of the process instance can be assessed with respect to its deadline. This information can be used to alert process administrators about risk of missing deadlines and to inform users about the urgency of their activities.

Our approach differs considerably from the ones presented above. In fact, we aim at predicting any kind of exception, rather than focusing on deadline expirations. In addition, we propose to build prediction models by leveraging data warehousing and data mining techniques, that enable more accurate predictions, based on many characteristics of the process instances, such as data values, resources, the day of the week or hour of the day in which processes or activities are started, and many others. In addition, the approach presented in this paper also allows exception analysis, to help users in understanding and attacking the causes of the exception. A further difference with respect to the above-mentioned approaches is that this paper also presents the architecture and implementation of a *toolkit* for exception analysis and prediction, and illustrates experimental results. This approach and toolkit will be re-used and extended to provide more BPI functionalities.

Finally, we also mention the work by Hwang et al. [21], since it also deals with workflow exception and with the analysis of process execution log. However, the goal of the authors is to support users in *handling* exceptions once they have occurred. Exception handling suggestions are provided by providing the users with information about the way in which similar situations were handled in previous executions. Our goal is quite different, since we aim at analyzing

exceptions to understand why they occur, and in predicting and preventing their occurrence.

4.3. Workflow monitoring and controlling

Several approaches have been developed to monitor and control a workflow execution. For example, zur Muehlen and Rosemann [22] focuses on the analysis of the logged audit trail data, and proposes a tool (called PISA) that integrates the analysis of the audit trail data with the target data obtained from business process modeling tools. They differentiate three views of process monitoring and controlling: processes, involved resources, and process objects. The PISA tool contains data source adapters responsible for the realization of access to operative data sources. An evaluation method library contains procedures that allow evaluation of workflow history data (including the hedonic wage model).

Various graphical representations for the methods of the evaluation method library are stored in the panel library.

The CMI project [23,24] is also concerned with process monitoring, but its focus is geared towards an awareness model instead of an analysis model. The model allows customization of the awareness delivered to each process participant by requiring the specification of awareness roles and the specification of corresponding awareness descriptions.

Several commercial products (Tibco, Categoric, Actimize) offer solutions for business activity monitoring (BAM). BAM is a term coined by Gartner Group that refers to analysis and presentation of relevant and timely information about business activities across multiple applications, data formats, and systems. BAM is still an emerging area and addresses only one piece of the overall BPI challenges, providing a view of historical performance through monitoring business processes.

Several BI and analytic application vendors claim to offer the ability to optimize business processes, but their strengths in reporting and analysis have yet to be aligned in a process context. Vendors like Oracle, Cognos Inc., and Hyperion Solutions Corp., Informatica Corp. have added workflow support to enable event and notification-based support, but they focused merely on adding process metrics to their product architectures for traditional reporting and analysis.

Our approach differs considerably from the research projects and commercial products presented above. We propose an integrated approach and a tool suite for process analysis, prediction, monitoring, control and optimization. More than reporting functions offered by commercial products, our tool supports semantic analysis of processes from a business perspective by transforming low-level measures into business metrics. Another distinctive feature is the use of data mining techniques for building prediction models. These models are used for proactive process control.

5. Conclusions and future work

This paper has presented an approach to support business and IT users in managing process execution quality. The approach has been materialized into a set of tools referred as Business Process Intelligence tool suite since it is based on the application of business intelligence techniques, in particular data mining and data warehouse, to business processes. We have discussed the main challenges we had to face in undertaking this effort, and we have described how we have addressed them in our approach and implementation.

The experimental results have been quite encouraging. Therefore, we plan to put a considerable effort in this research area and address the issues that still lie ahead. In particular, our research agenda includes the refinement of the process data preparation stage to better handle the problem of multiple executions of a node within the same process instance and to select attributes also based on the behavior being analyzed. Improved mechanisms for automated exception prevention are also in our plans.

On the data mining part, we want to integrate other data mining techniques into PME to extend the intelligence of the tool suite to other kinds of analysis and predictions. We also want to incorporate some text mining capabilities to be able to automatically extract valuable information from documents that flow through a process and which could be useful for analysis and prediction. We are working on a seamless integration of data mining algorithms, including decision trees, in order to be able to offer a tool suite that is not tied to licensing of other commercial products for mining data. At the same time, and in order to respond

to near real time requirements, the tool suite should be equipped with algorithms that dynamically adapt to changes in the concepts (rules, patterns) learned without requiring to re-build prediction models from scratch when concept drift takes place.

These steps will contribute to our end goal, that consists in making process mining technologies readily available to users and that can be easily applied, without requiring any data mining expertise from BPI users. Such mining technologies are mainly targeted at *business* users, who will be able to use the BPI suite to analyze and predict the business metrics that they consider significant to assess the quality of their operations.

The work presented in this paper is part of a larger, long-term research effort aiming at developing a Business Process Intelligence solution for BPMSs. Other research objectives in the BPI context include process definition discovery, execution path analysis and prediction, and dynamic system, process, and resource optimization.

References

[1] F. Leymann, D. Roller, Production Workflow, Prentice-Hall, Englewood Cliffs, 2000.
[2] F. Casati, Intelligent Process Data Warehouse for HPPM 5.0, HP Labs Technical Report HPL-2002-120, 2002. http://www.hpl.hp.com.
[3] M. Berry, G. Linoff, Mastering Data Mining, Wiley, New York, 2000.
[4] M. Castellanos, J. Stinger, A practical approach to extracting relevant sentences in the presence of dirty text, in: Proceedings of the First SIAM Workshop on Text Mining, Chicago, IL, 2001.
[5] G.H. John, R. Kohavi, K. Pfleger, Irrelevant features and the subset selection problem, in: Proceedings of the 11th International Conference on Machine Learning ICML'94, 1994.
[6] D. Grigori, F. Casati, U. Dayal, M.C. Shan, Improving business process quality through exception understanding, prediction, and prevention, in: Proceedings of the VLDB'01, Rome, Italy, 2001.
[7] M. Sayal, F. Casati, U. Dayal, M.C. Shan, Business Process Cockpit, in: Proceedings of the VLDB'02, Hong Kong, China, August 2002, 2002.
[8] P. Muth, J. Weissenfels, M. Gillmann, G. Weikum, Workflow history management in virtual enterprises using a light-weight workflow management system, in: Proceedings of the Ninth International Workshop on Research Issues in Data Engineering, Sydney, Australia, 1999, pp. 148–155.
[9] P. Koksal, S. Arpinar, A. Dogac, Workflow history management, SIGMOD Rec. 1 (1998) 67–75.
[10] M. zur Muehlen, Process-driven management information systems—combining data warehouses and workflow technology, in: B. Gavin (Ed.), Proceedings of the Fourth International Conference on Electronic Commerce Research (ICECR-4), Dallas, 2001, pp. 550–566.
[11] J. Eder, G. Olivotto, W. Gruber, A data warehouse for workflow logs, in: Proceedings of the International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002), Lecture Notes in Computer Science 2480, Beijing, China, 17–20 September, Springer-Verlag, Berlin, 2002, pp. 1–15. ISBN 3-540-44222-7, ISSN 0302-9743.
[12] J. Cook, A. Wolf, Discovering models of software processes from event-based data, ACM Transactions on Software Engineering and Methodology 7 (1998) 3.
[13] G. Agrawal, D. Gunopulos, F. Leymann, Mining process models from workflow logs, in: Proceedings of the Sixth International Conference on Extending Database Technology (EDBT), Valencia, Spain, 1998.
[14] J. Herbst, An inductive approach to adaptive workflow systems, in: Proceedings of the CSCW-98 Workshop Towards Adaptive Workflow Systems, Seattle, WA, 1998.
[15] J. Herbst, D. Karagiannis, Integrating machine learning and workflow management to support acquisition and adaptation of workflow models, in: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, IEEE, 1998, pp. 745–752.
[16] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, J. Maruster, G. Schimm, A.J.M.M. Weijters, Workflow Mining: A Survey of Issues and Approaches, Internal Report, 2002. http://www.tmitwww.tm.tue.nl/staff/wvdaalst/workflow/mining/wf-min-surv.pdf.
[17] W.M.P. van der Aalst, B.F. van Dongen, Discovering workflow performance models from timed logs, in: Y. Han, S. Tai, D. Wikarski (Eds.), Proceedings of the International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002), Lecture Notes in Computer Science, vol. 2480, Springer-Verlag, Berlin, 2002, pp. 45–63.
[18] E. Panagos, M. Rabinovich, Escalations in workflow management systems, in: Proceedings of the DART'96, Rockville, MD, 1996.
[19] E. Panagos, M. Rabinovich, Predictive workflow management, in: Proceedings of the NGITS'97, Neve Ilan, Israel, 1997.
[20] J. Eder, E. Panagos, H. Pozewaunig, M. Rabinovich, Time management in workflow systems, in: Proceedings of the BIS'99, Poznan, Poland, 1999.
[21] S. Hwang, S. Ho, J. Tang, Mining exception instances to facilitate workflow exception handling, in: Proceedings of the DASFAA'99, Taiwan, 1999.
[22] M. zur Muehlen, M. Rosemann, Workflow-based process monitoring and controlling—technical and organizational issues, in: Proceedings of the 33rd Hawaii International Conference on Systems Sciences, Wailea, HI, 2000.

[23] D. Baker, D. Georgakopoulos, D. Schuster, A.R. Cassandra, A. Cichocki, Providing customized process and situation awareness in the collaboration management infrastructure, in: Proceedings of the Conference on Cooperative Information Systems, 1999, pp. 79–91.
[24] H. Schuster, D. Baker, A. Cichocki, D. Georgakopoulos, M. Rusinkiewicz, The Collaboration Management Infrastructure, ICDE, 2000, pp. 677–678.
[25] M. Berger, E. Ellmer, D. Merkl, A learning component for workflow management systems, in: Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS'98), vol. 7, 6–9 January 1998.
[26] [HPPM-PD] Hewlett-Packard, HP Changengine Process Design Guide, 4.4th ed., 2000.

Daniela Grigori graduated in automatic control and computer science at the Technical University of Iasy (Romania) in 1991. From 1992 to 1997 she worked as research assistant at Romanian Research Institute for Informatics of Bucarest, involved in projects on information retrieval methods and tools. She got her masters and PhD from the University Nancy I in 1998 and 2001, respectively. Since 2002 she is an associate professor at the University of Versailles. Her main research interests are in process modeling, business process intelligence, and coordination and cooperation models and systems.

Fabio Casati is a senior researcher at Hewlett-Packard Laboratories, Palo Alto, CA. His research interests include workflows, web services, and application management. He has led the development of several applications in this space. He is also author of more than 50 papers in international conferences and journals and has served as organizer and program committee member in many conferences. He is co-author of a book on web services.

Malu Castellanos received her degree in Computer Engineering from the National Autonomous University of Mexico, and PhD in Computer Science from the Polytechnic University of Catalunya (Barcelona). Her current research interests are Data Mining and Text Mining applications. Previously she was an Associate Professor in the Information Systems Department at the Polytechnic University of Catalunya, where she taught database courses and led various projects related to database interoperability. She has a number of papers in international conferences and journals and been reviewer and PC member of several others. She is a coauthor of a book on text mining and a book on multidatabase systems. Currently she is a researcher at HP Labs.

Umeshwar Dayal is Director of the Intelligent Enterprise Technologies Laboratory at Hewlett-Packard Laboratories in Palo Alto, California, USA. His research interests are data mining, business process management, distributed information management, and decision support technologies, especially as applied to e-business. He has published over 110 papers on these topics. He received a PhD in applied mathematics from Harvard University. Dayal is a member of the ACM and IEEE. Contact him at umeshwar.dayal@hp.com.

Mehmet Sayal is currently working at Hewlett-Packard Labs, Palo Alto. He has received his PhD in Computer Science from Northwestern University. His areas of interest include data mining, B2B interactions, workflow management, content delivery, and distributed information systems. His current research is about automatic data correlation and outlier analysis techniques.

Ming-Chien Shan is a program manager in the Hewlett-Packard Laboratories, Palo Alto, CA. He joined IBM DB2 team in 1977 working on query optimization, data definition manager and distributed DBMS. He then joined HP in 1985 and managed various research projects, including object-oriented DBMS, heterogeneous DBMS, workflow and telecom service provisioning. Currently, he is the manager of e-business solutions group. He received his PhD degree in computer science from University of California, Berkeley in 1980. He has published more than 50 research papers and been granted 15 software patents. He has served as Chairperson for ICDE RIDE'2002 and VLDB TES 2002 and 2003 and program committee member in many conferences.