A Test-bed for the Evaluation of Business Process Prediction Techniques

(Invited Paper)

Suraj Pandey, Surya Nepal, Shiping Chen
CSIRO ICT Centre
Marsfield, NSW, Australia
Email: {firstname.lastname}@csiro.au

***Abstract*—Business process prediction technologies are being increasingly used by organisations to provide timely feedback to their customers and improve their overall productivity. In order to provide valuable information to both customers and system managers, the timings of business processes need to be forecast with high accuracy and efficiency. In particular, organisations require to predict the process and event flows, recognize their patterns, and forecast the total time it would take for a workflow to complete, in order to meet the Service Level Agreements (SLAs) signed with the customers.**

In this paper, we focus on the prediction models that could be used for forecasting time to completion of business processes by analysing historical event logs. First, we propose a service oriented architecture that provides a test-bed for carrying out predictions on business processes. Second, we propose a Hidden Markov Model (HMM) based prediction technique that produces a model based on event logs, and compare it against existing prediction models. Finally, we describe an implementation of the system, where we simulate the execution of a business process and obtain predictions using both the proposed and existing prediction techniques.

***Keywords*-Business process simulation, prediction techniques, hidden Markov model**

I. Introduction

Nowadays, most business processes are supported by information systems that store data about the process execution in logs [8]. These historical data play an important role in forecasting information about future business activities. Prediction information not only guides the end-users of the state/time-line of their processes, but plays a significant role when determining the quality of service and level of SLA satisfaction at the service provider's side.

Most business processes are complex, and understanding their flow is even more difficult. Thus, simulation in advance is required for complex, high-impact business processes [9]. In order to carry out the simulation of executing complex business processes, logging information, extracting knowledge and forecasting information on future events, the simulation framework needs to provide several features, such as design, execution, analysis, prediction, monitoring, and optimization of business processes. The system also needs to meet the workflow workload requirements and integrate a varying number of external resources (human and computing), so that the simulation environment can closely match the real world execution scenarios. In particular, the prediction component of the simulation system should provide an extensible mechanism for plugging in prediction techniques so that all types of business process workloads can be analysed for accurate and timely forecasting.

Realizing the importance of the prediction in Business Process Management (BPM) systems, this paper proposes a generic system architecture for business process simulations, where various prediction models can be used to analyse stored historical log data. It incorporates an extensible framework for the prediction component, where both existing and new prediction techniques can be implemented.

The proposed layered architecture is unique in several aspects. It facilitates the integration of a semantics layer to define templates for business processes and associate management and operational information based on process ontologies. The workflow templates generated using this knowledge-base form the execution instances for an individual end-user, which are then simulated using the Business Process Engine (BPE). The use of a message queue for enabling interactions between internal and external entities by using existing interchange standard (e.g., messages defined using XML Process Definition Language (XPDL)) makes the system highly portable [6]. The prediction engine works seamlessly with all these components in a time efficient and scalable manner. The work presented in this paper focuses on describing these components and experimenting the capability of the proposed architecture through a prototype implementation.

One of the important and useful information that organisations need to analyse is related to the prediction of the path that processes could follow during the execution of business processes [3]. Based on the likelihood of paths being followed, the overall time to completion could be estimated more accurately. Identification of the likelihood of paths could be achieved through critical paths [7], dependency detection [8] [3], exceptions [5], and so forth.

Predictions could also be obtained using historical event logs. Aalst *et al.* [12] presented a method for predicting the remaining time until the completion of business processes by building an Annotated Transition System (ATS). ATS is an abstraction of event logs from past executions. Their method learns the model, constructs the ATS, and uses it for predictions. In this paper, we implemented the annotated transition system (described in Section III). We used sequence

COLLABORATECOM 2011, October 15-18, Orlando, United States

DOI 10.4108/icst.collaboratecom.2011.247129

abstraction to predict remaining time of business processes.

Path mining and ATS based systems heavily rely on the existence of historical event logs and assume that the logs contain all the paths and patterns that would be traversed while executing business processes. In reality, not all the paths and events occur with equal likelihood. Business process patterns and their likelihood of occurrences are dependent on several factors, such as resource availability and their characteristics, resulting in the processes changing patterns over time. The assumption that the system is in steady state is not very helpful for dynamic processes. To overcome these limitation, we propose a method to fit the historical data onto a model and update the model as and when required. We use a Hidden Markov Model (HMM) based fitting to learn the patterns of workflows existing in historical event logs.

The contributions of this paper can be summarized as follows:

1) An architectural framework that facilitates the simulation of business processes and prediction techniques.
2) A HMM based prediction model for predicting time to completion values of business processes.
3) A prototype test-bed that implements the proposed architecture and the prediction techniques for simulation.

The rest of the paper is organized as follows: we start by describing three basic prediction techniques in Section II; Section III describes the system architecture that provides a test-bed for simulating business processes; Section IV presents the experimental results using the prediction models described in Section II; and finally, Section V concludes the paper by providing insights into potential future work.

II. Prediction Models

In this section, we first present the existing techniques commonly used in predicting time for business processes. They are: descriptive statistics, multivariate regression, and annotated transition system based methods. They have been used for prediction of future states of business processes based on information collected from historical data. We then present our approach of using the HMM for prediction of business processes, and identify its benefits and weaknesses as compared to the existing techniques. We present the prediction results obtained by using each of these techniques in Section IV.

In all these prediction techniques, we are interested in the events generated by tasks and their timestamps. Let $E = [e_1, e_2, ..., e_m]$ be a set of events of a workflow instance, with m as the time length of the observations, and $t^e = [t_1, t_2, ..., t_n]$ the n measurements of the timestamps for an event $e \in E$.

A. Descriptive Statistics

Descriptive statistics summarize the historical log data using numerical descriptors that include mean and standard deviation. We calculate the mean, minimum, maximum, and standard deviation of the timestamps taken from event logs for every event (generated by tasks) registered by the BPE during a simulation.

The numerical descriptors are defined by the following equations:

$$AvgT : \overline{t^e} = \frac{\sum_{i=1}^{n} t_i^e}{n} \tag{1}$$

$$Stdev : \frac{\sum_{i=1}^{n} (t_i^e - \overline{t^e})^2}{n-1} \tag{2}$$

$$MinT : mininum\{t_1, t_2, ..., t_n\} \tag{3}$$

$$MaxT : maximum\{t_1, t_2, ..., t_n\} \tag{4}$$

Equations (1)-(4) simply applies the respective function to all the timestamps for a particular event, irrespective of the business process template it was generated from. They do not take into account the dependencies between tasks.

If we do not know the states (set of events), then the best prediction is the mean $\overline{t^e}$, but the variability in this prediction is high. However, if we do know the states, then the prediction can be given by the regression fit [4].

B. Multivariate Regression

Regression analysis is used for explaining or modeling the relationship between the timestamps (the response variable) and the sequence of events (input variables) taken from event logs. The objective of using regression is to predict the future observations based on the modelled relationship between the input variables. Since we do not have enough data to estimate the relationship between the input variables, we usually have to assume the relationship to be linear, to start with. Let us assume that RT is the Remaining-Time variable that depends on the set of events that have already occurred.

$$RT = \beta_0 + \beta_1 e_1 + \beta_2 e_2 + \ldots \beta_n e_n + \epsilon \tag{5}$$

where $\beta_i, i = 0, 1, 2, ..., n$, are unknown parameters for the fit, and ϵ is the error. We will rely on least squares estimation of β, as the Gauss-Markov theorem states that it is the best linear unbiased estimate [4]. The confidence intervals are based on the assumption of normal errors. To verify that the least squares estimates are acceptable, we test the normality using the Q-Q plot (Quantile-Quantile plot).

In Equation (5), we have assumed (for simplicity) that the relationship between tasks is linear. However, this is not the case in real business processes. Tasks are dependent on each other based on the control flow and data flow, which in turn depend on the availability of resources, capability of resources, and so forth. The relationship (regression function) is different for every business process, which is why the precision of the results is always questionable. Aalst *et al.* [12] have demonstrated that their annotated transition system based approach outperforms regression models in terms of efficiency and precision.

C. Annotated Transition System

Aalst *et al.* [12] proposed an *annotated transition system* (ATS) based on process mining. They built the ATS by mapping partial traces of event logs onto the states of the transition system. The states of the transition system could be

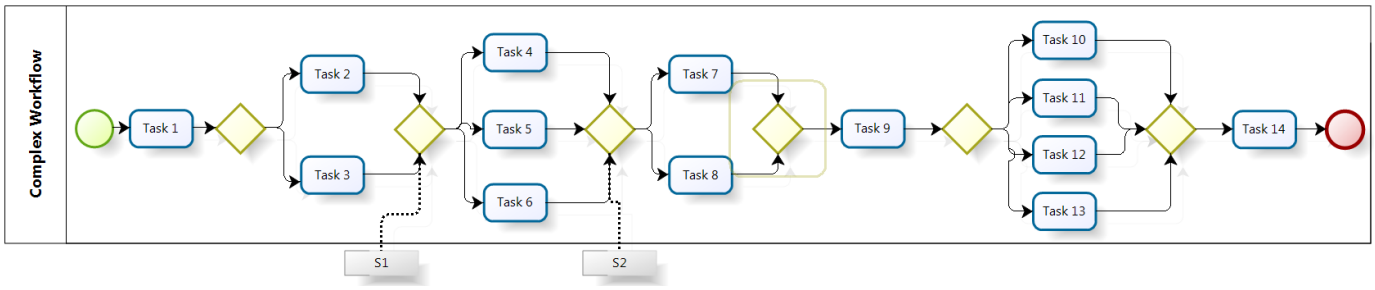

Fig. 1. An example workflow that contains basic workflow patterns. (Resembles the travel agency process in [8])

defined based on the type of business process being simulated (*started, executing, completed, failed, etc.*).

To understand how the remaining time is calculated in the ATS system, let us assume each task in the workflow, depicted in Figure 1, takes 2 seconds to reach the *completed* state. As soon as the tasks complete, the completion events are recorded in the event log. A log entry is retrieved from historical data for each task that we are interested, in the sequence given by the workflow. The following sequence describes each task's completion time using a superscript:

$$< Task_1^0, Task_2^2, Task_4^4, Task_7^6, Task_9^8, Task_{10}^{10}, Task_{14}^{12} >$$

The total time taken for this sequence is 12 seconds (time difference between the last task and the first task in the sequence). If the current state is $Task_4$, the remaining time to completion is $(12 - 4 = 8)$. Similarly, if the current state is $Task_{10}$, the remaining time to completion from this state is $(12 - 10 = 2)$. For calculating average remaining time, we computed the remaining time value for each matching log record (sequence of paths recorded in the log, which have the current state in them) and divided the result by the number of matching records.

D. Hidden Markov Model

ATS provides a more precise mechanism to predict time values for business processes than descriptive statistics and regression, however, it relies heavily on the availability of information on task sequences, as it is not a model fitting historical data. The prediction accuracy cannot be relied on when the number of events being recorded are few in number, which is true even for descriptive statistics. Moreover, the model assumes a *steady state* based process mining. In steady state, the processes are not dependent on factors such as: time of the year, resource availability, resource characteristics, cost of operations, and so forth. Business processes change over time and their flows are dependent on probability of occurrences of several other related events. Our approach is to use HMM to describe these probabilities and build a fitting model.

A HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states. Unlike the ATS, this method computes a set of HMM parameters after analysing the log data (training set). The model thus formed can then be applied to unseen log data for prediction purposes.

A HMM for predicting time of business processes can be defined as follows:

States E: Each event that gets recorded from an execution of a workflow instance is modeled as a HMM state. HMM states are different to the states of a workflow and its tasks.

Observations O: In any HMM, during the time of recording of the states, each state can have a set of observation values. In this paper, we are only considering two states: *completed* denoted by "1" and *other state* denoted by "0". Hence, the observation set is: $O_t = \{1, 0\}$.

Transition Matrix $A_{N \times N}$: Each element a_{ij} denotes the probability of transition from state i to state j. In our system, the transition probability is the probability of a parent event emitting a "completed" event and changing its current state to one of its immediate children.

Observation Emission Matrix B: in which $b_{e_j}(O_t)$ denotes the probability of observing O_t in state $e_j \in E$. This matrix gets calculated during the training phase of the HMM.

Initial Probability $\pi_{N \times 1}$: in which π_i denotes the probability of being in state i when the time $t = 1$. Regarding the fact that in business processes, the process starts from some initial state, we have:

$$\pi_i = \left\{ \begin{array}{ll} 1 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \end{array} \right\}$$

Using the above defined parameters, a HMM λ can now be represented as:

$$\lambda = (\pi, A, B) \tag{6}$$

A HMM assumes that the hidden state e_i is dependent only on e_{i-1}, and so is the observation at the same time as e_i. Similarly, in the context of a workflow, a task will have control and/or data dependencies only with its immediate parent. Having the Equation (6), the questions we need to answer for predicting time values for business processes are:

1) How to tune parameters (π, A, B) to find a model λ that best matches the observations of O for the series of events in E? In the context of business processes, how do we compute a HMM so that its parameters represent the event logs?
2) How to compute the probability of the occurrence of a specific sequence of observations, $P(O|\lambda)$ for the

series of events in E? For business processes, given a sequence of tasks and their event logs, how do we find the probability of occurrence of the same series of tasks in the future, and eventually the time values (e.g., remaining time) for sequences of tasks?

The unknown parameters of a HMM can be found using the Baum-Welch algorithm. The BaumWelch algorithm can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and observation probabilities) of a HMM, when given only observations as training data. Another method is to use the Viterbi training algorithm, which, for an initial HMM and a given sequence of observations, infers near optimal parameters to the HMM. The trained model can then be used for predicting the occurrence of specified sequence of observations (events). Blum *et al.* [1] constructed a HMM from process logs and used Viterbi algorithm to estimate the most likely sequence of states in the model.

For predicting the time values for the specified sequence of task events, we propose a weighted average of the probabilities given by the trained HMM. We assign the weights for each sequence as the time values given by the ATS model (remaining time in this paper), and compute the weighted sum, described in greater detail in Section IV-E. The weighted sum given in Equation (7) is not the optimal way for predicting time values. We are exploring the ways of relating the probability of occurrence of sequences with the time values recorded for these sequences, as part of our future work.

III. System Architecture

Before delving into the specifics of the system architecture and the prediction techniques, we briefly describe relevant work on BPM. Ko *et al.* [6] presented a survey on existing BPM languages, standards, and notations, and identified their strengths and limitations. They described the usage of XPDL in both process design and process enactment stages of the BPM life cycle. Following their guidelines, our system architecture adopts XPDL as the interchange standard and BPMN as the graphical standard.

Existing tools, such as *ProM*[1] and *YAWL*[2], provide support for the automated discovery of simulation models based on event logs. Aalst *et al.* [11] thoroughly studied these existing simulation approaches and identified their limitations. They pointed out the fact that predictions are not restricted to time, but can also be related to costs, resource availability [2], etc. Our system architecture takes into account the possibility of addition of additional parameters to the prediction techniques.

A business process execution (simulation) environment should provide basic functionalities to describe any business process, instantiate workflows out of business process templates, simulate the executions of workflow instances, predict process parameters (e.g., remaining time to completion, average time for a process, etc), and store and access past data (events, computed prediction values, etc.) as and when needed. Such an environment must also be able to simulate a large number of processes in parallel and, at the same time, facilitate end-user (customers) requests. In order to provide these basic features, we propose an architecture for business process simulation, as depicted in Figure 2.

The system comprises of three layers: semantics, core/middleware, and persistence.

A. Semantics

The components in this layer are responsible for determining the way workflows are defined and instantiated using predefined business process templates. A business process is defined using a BPMN-XPDL schema. The semantics associated with business processes are part of the ontology. When a new workflow instance is created, its elements take specific values and types, as defined by the mapping between the business process the workflow belongs to and its associated semantics. Some parts of the instantiated workflow need to be updated using data generated by the prediction engine.

Once the right kind of workflow templates are identified, they are instantiated and submitted to the middleware component for execution.

B. Core/Middleware

The core part of the system, where the actual simulation is carried out, comprises of: a message queue, a Business Process Engine (BPE), prediction algorithms, and a BPM service.

Business Process Management Service: The BPM service links the end-users to the entire system through Web services. Users, reporting tools, or any external component interested in business process monitoring, subscribes to the events from the BPM service. The BPM service provides the latest updates on the workflows it is running on their behalf. The BPM service may directly access the storage and obtain stored data on workflow states and past predictions on those states. Our design facilitates the reporting of data to customers either synchronously or asynchronously. In the former case, the BPM service requests the prediction engine to immediately carry out predictions using the latest set of events, in which case, it waits until it receives results from the prediction engine. In the latter case, it issues a request to the prediction engine, but does not wait for the results to be completed on the fly. Instead, it obtains the recently computed prediction results from the storage.

Message Queue: A Message Queue (MQ) facilitates the execution system by providing a publish-subscribe based queuing system. Any form of information, including events and messages, that need to be exchanged between components are published by the underlying components in the queue. These messages are then queued awaiting the subscribers to eventually consume them, asynchronously. In our system, the MQ is predominantly used by the business process and prediction engine, as depicted in Figure 2. The BPE publishes any events/messages resulting from the execution of workflow instances being simulated. The BPM service subscribes to

[1]http://www.processmining.org
[2]http://www.yawlfoundation.org

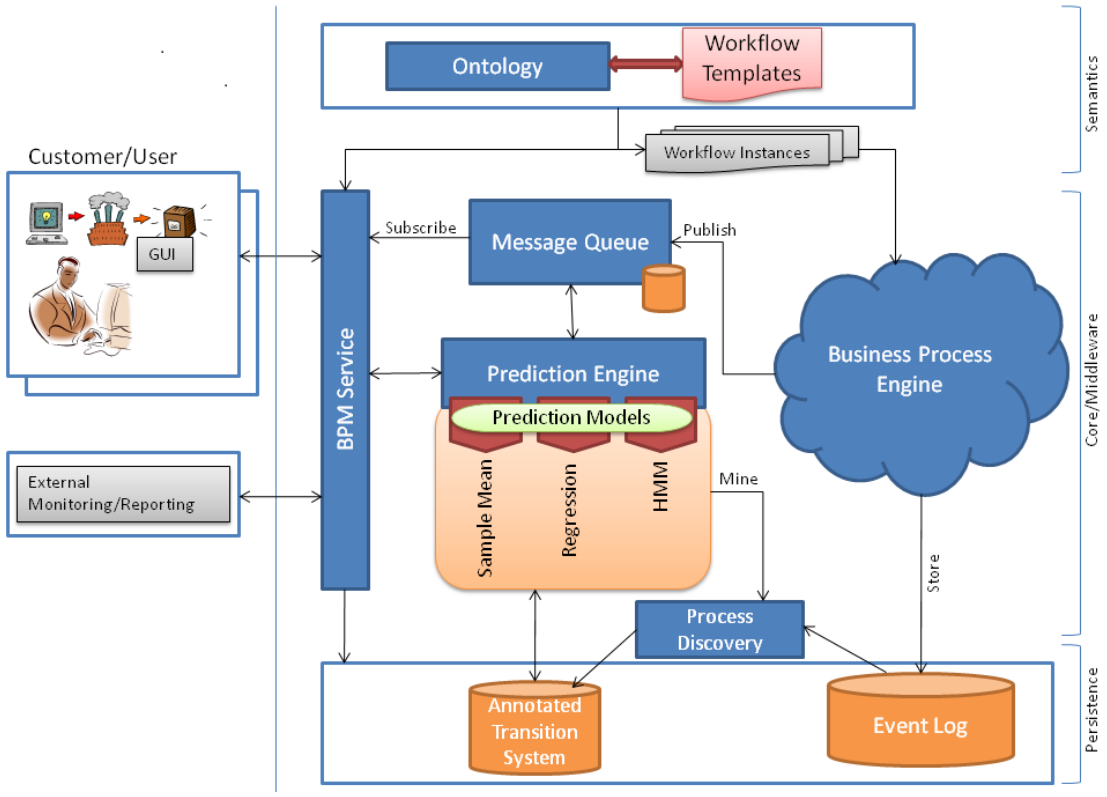

Fig. 2. System architecture for work tracking and prediction.

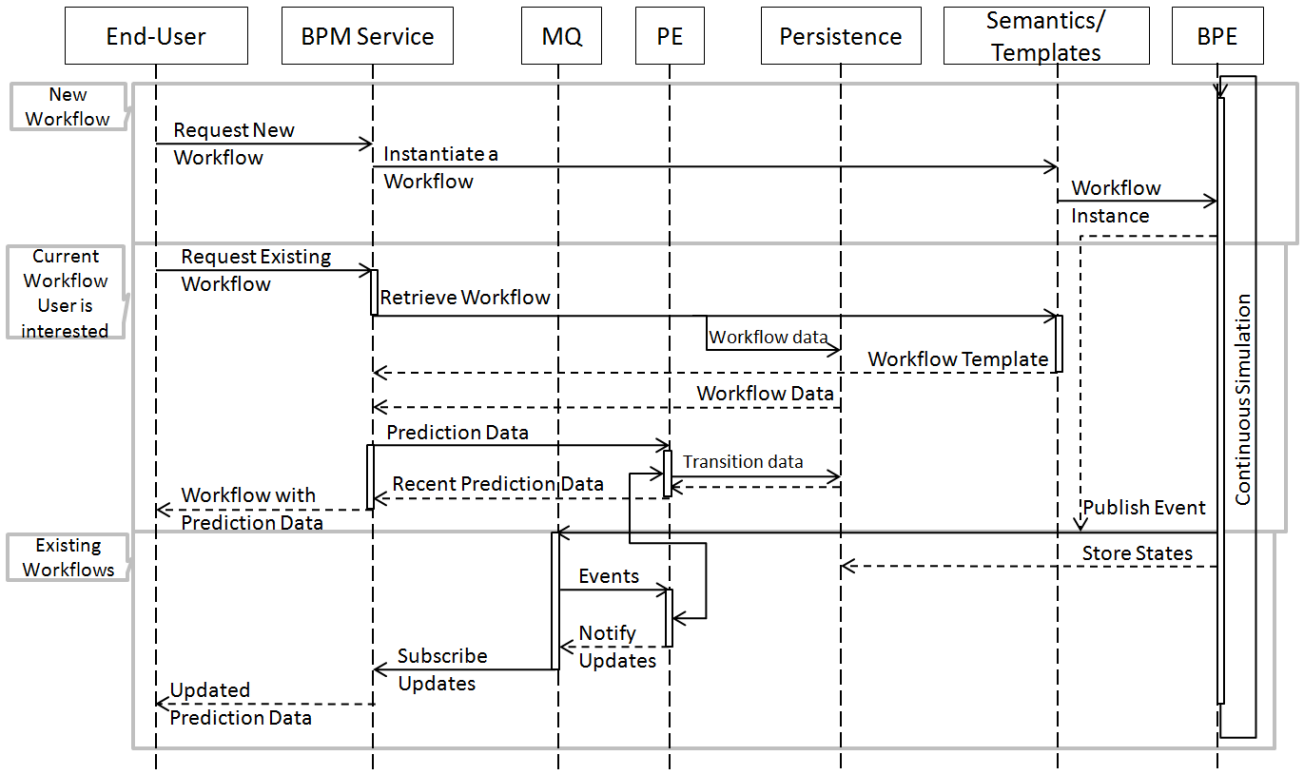

Fig. 3. A sequence diagram showing the flow of messages between system components during the simulation of business processes.

these events and notifies any user-side applications about event/data availability. Similarly, the prediction engine subscribes to the events and performs predictions. After prediction results are available, the BPE publishes them to the MQ for the BPM service to consume. In this way, the MQ plays a major role in connecting the components asynchronously.

Business Process Engine: The BPE simulates the flow of events based on the simulation parameters provided to it at the run-time. These parameters are associated with the workload, which could be: the number of workflow instances to run at a time, types of workflow templates to be used, types of events to generate, and so forth. Based on these parameters, the BPE constructs the workflow instances and starts executing them in parallel. While in execution, BPE publishes all specified events/messages to the MQ. It preserves the execution order of tasks and workflows defined in the instantiated business processes. The events/messages published are timestamped according to this logical control and data flow. The BPE also provides capability to control the generation of user-defined types of events/messages associated with each business process.

The BPE stores all the events and messages generated during the simulation as event logs through the persistence layer. This log forms the historical data for the prediction engine.

The event processing functionality in our prototype implementation is based on the open source complex event processing engine, Esper[3]. Esper provides support for triggers (messages that can then be used as events for the workflow) on workflow event patterns. Using SQL queries, we register our interest on event patterns of any workflow instance, and the Esper's lightweight processing engine forwards them to the MQ. The simulation of business proceses is thus carried out using event stream processing through Esper.

Prediction Engine: The prediction engine (PE) is a pluggable component that is responsible for predicting time information of business processes, such as the remaining time to completion, average time taken for execution, start time, and so forth. It does this with the help of prediction models implemented as algorithms. Some of these prediction algorithms are described in Section II. In order to carry out predictions based on the stored historical data, we need to build an information system. This can be achieved using a process mining component that extracts specific data and forms a transition system. This transition system is then annotated with timing values that help in representation and analysis of prediction values. The design of this ATS is the same as the system proposed by Aalst *et al.* [12]. The ATS consists of records of events with annotations that represent one or the collection of: *elapsed time* (e.g., the average time to reach a particular task), *execution time* (e.g., the average time taken to execute the current task, averaged over a fixed period of log data), *remaining time* (e.g., the average time to reach the end of the current workflow, calculated from historical executions).

[3]http://esper.codehaus.org/

For more detailed analysis of the process mining technique, and the annotation system, we refer the reader to Aalst *et al.* [12].

In this paper, we have implemented the transition system using a database system. The events/task states are stored in one table and annotations (time values) for each event in another table. Each event's attributes are *elapsed time*, *execution time*, and *remaining time*, which are stored as columns in the annotation table.

The processing time taken by the PE depends on: a) the size of the historical data to be analysed, and b) the number of requests received by the PE for computing predictions. The updating of the transition system occurs every time an event is registered at the MQ, which is subscribed by the PE. The PE recalculates the annotated values for each of the associated tasks that are related to the event just registered. After the transition system is updated, the PE publishes the updated values to the MQ, which is then interpreted as prediction values by the subscribers. The BPM service could also directly trigger the transition system update process whenever the end-users request for an updated prediction values. In this latter case, the PE returns the latest values computed and stored in the system.

Figure 3 shows a sequence of message exchanges between the system components for delivering updated prediction data to the end-users. The figure shows how an end-user retrieves prediction data for a new workflow, currently monitored workflows, and past workflows. When the end-user requests to create a new workflow from an existing business process template, the BPM service retrieves the template from the semantics layer, which is then forwarded to the BPE for execution. When the end-user requests information for a selected workflow instance, the BPM service retrieves the instantiated workflow template from the semantics layer and its associated data from the persistence layer. It then requests the PE for the prediction data. The PE has two options: a) return the recently stored prediction data and wait for the next round of prediction process to start and notify the PE about the number of requests coming from end-users, or b) re-compute the prediction data and send the computed results to the BPM, in which case the end-user will need to wait for the updated data to be available. If the PE follows the first choice, it notifies the BPM service about the availability of updated prediction data only after it completes the computation, which will then get pushed to the end-users. The workflow instances that are being observed by the end-user and are configured to receive updates periodically will receive the prediction data from the BPM service as soon as they are updated by the BPE. The BPE is constantly carrying out workflow simulation that registers events to the MQ. As the PE is a subscriber of these events, it starts the prediction process as soon as an event (or a group of events) is registered at the MQ. This triggered computation guarantees that the prediction results are updated as and when activities change state, irrespective of the end-user requesting for them.

C. Persistence

The persistence layer is used for storage of events, processes states, prediction data, and the ATS data. Individual schemas are defined for each of these types. For instance, events data are formatted in conformance to events schema and stored. Similarly, the annotated transition data have their own schema, as discussed in earlier sections. We implemented the persistence layer using MySQL database.

IV. Experiment and Results

In this section, we present the prediction results given by descriptive statistics, multivariate regression, the ATS, and the proposed HMM methods. We use a synthetic event log obtained via executing the workflow depicted in Figure 1 on our simulation test-bed.

In general, the simulation parameters are application context dependent. Fundamentally, these parameters describe the number and type of business processes to be simulated. When a business context is added to a workflow, relevant parameters are introduced to distinguish processes. For example, application documents submitted by customers could be instantiated into different types of workflows based on the customer's background; and the workflow could be processed by specialized group of people. Although our proposed system architecture is capable of handling these complex variations on business processes, we only use one workflow template for reporting the prediction results in this paper.

When generating the event logs, we used 1000 concurrent threads that record the events occurring from one workflow template. Essentially, each thread is responsible for one instance of the workflow depicted in Figure 1. The time for running these threads is fixed at 120 seconds. As soon as the 1000 threads complete executing, the event log creation ceases, irrespective of the time (120 seconds) elapsed. This event log is used by each of the prediction techniques, independently. The interpretation of the results and their significance is described in Section IV-F.

A. Application Workflow

Aalst *et al.* [13] described a number of workflow design patterns that focus on control-flow and events, which are the underlying design principles and execution models of business workflow approaches. Some basic patterns are: sequential, parallel split, and synchronization. In this paper, we use these basic control flow patterns to study the prediction techniques.

Figure 1 depicts the control flow of a complex business process. It contains the sequential, parallel, and synchronization patterns. This workflow resembles the travel agency workflow presented by Schonenberg *et al.* [8] in both control flow and the number of tasks. However, we name the tasks from *Task 1* to *Task 14*, not characterising to any specific application as we are interested in the performance and capability of the test-bed. Two annotations "S1" and "S2" are added in the workflow to track the current state of the workflow. For example, after either *Task 2* or *Task 3* has completed, the current state is "S1".

TABLE I
Average values of timestamps of events obtained using Equations (1)-(4).

Current State	AvgT	MaxT	MinT	Stdv	Average Remaining Time
Task 1	13636	28800	3600	4259	13572
Task 2	11336	25200	3600	4636	11250
Task 3	12776	21600	3600	3515	12876
S1	10582	25200	0	4120	10548
Task 4	8938	14400	0	3223	9000
Task 5	7589	18000	0	3429	7673
Task 6	9523	18000	3600	3720	9337
S2	7055	18000	0	3579	7020
Task 7	5314	10800	0	3355	5274
Task 8	5336	14400	0	3262	5305
Task 9	3418	10800	0	2577	3384
Task 10	1575	3600	0	1786	1694
Task 11	2110	7200	0	2759	2040
Task 12	1872	7200	0	2067	1800
Task 13	1523	7200	0	2040	1466
Task 14	302	766	0	1902	0

We assign uniformly distributed timestamps to each task in the workflow. This is achieved by generating random sequence of timestamps between two points in time (*SecureRandom* class from Java) and assigning them to the tasks in the workflow in sequence. This preserves the dependency constraints (tally between parallel tasks is resolved randomly).

B. Descriptive Statistics

Table I lists the average, maximum, minimum and the standard deviation values for the execution times of each task of the workflow depicted in Figure 1. These data are averaged over all the event logs obtained from the historical data.

When the value given by MinT for a task is "0", practically the task took no time for execution and/or it was simply skipped. In practice, this type of scenario is common when tasks in a workflow do not need to be executed. Similarly, the values given by MaxT vary significantly in the range between 28800 and 7200, reflecting the variability in execution times of tasks in real environments.

C. Annotated Transition System

In Table I, we also list the average remaining time of the workflow for any current state between *Task 1* and *Task 14* (inclusive).

The average remaining time value is calculated after the process mining component creates the ATS using event logs. The annotations for each event is created using a sequence abstraction (See [12] for more details on types of abstractions) and using the "completion" event only. The sequence abstraction preserves the control dependencies between the tasks and provides a meaningful abstraction when predicting information on paths of a workflow.

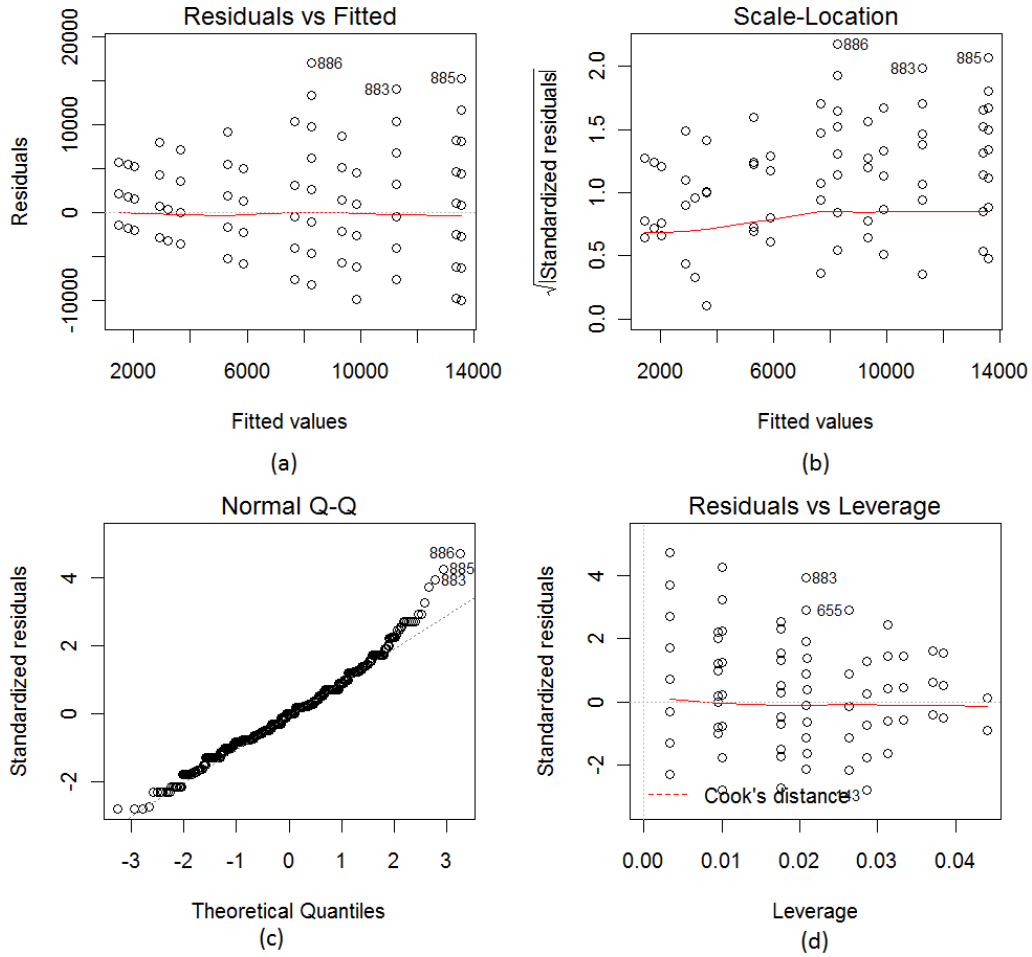

Fig. 4. Analysis of the fit given by multivariate regression.

D. Multivariate Regression

We make the *remaining time* as the dependent variable upon the various states of tasks (*Task 1* to *Task 14*). Using R's[4] linear model fitting, we obtain prediction of remaining time for any given task sequence. For instance, if the task sequence is T_1, T_3, T_6, T_9, then the predicted value denotes the remaining time for completion given that *Task 9* has already completed. We fix the confidence interval at 0.95 and plot the predicted values at different states, depicted in Figure 5.

Before accepting the result of the linear regression, we evaluate its suitability by examining the residuals. We check to see if the residual errors are random and normally distributed. We also examine if the model is sensitive to one or two cases (event sequences in our experiment). This is carried out by using R's *plot()* command and the results are depicted in Figure 4. The plot in the upper left shows the residual errors plotted versus their fitted values. The residuals are randomly distributed around the horizontal line representing a residual error of zero. The plot in the upper right shows the square root of the standardized residuals as a function of fitted values, which also shows randomness. The Q-Q plot in the lower left shows the closeness to normality. Finally, the plot in the lower right shows the measure of each point's importance in determining the regression result. Smaller (close to zero) distance of Cook's distance shows that removing observation has little effect on the regression result.

[4]http://www.r-project.org/

E. Hidden Markov Model

In this paper, we used the following set of values to initialize the HMM given by Equation (6).

- Observation vector has the values "1" and "0" that corresponds to either "completed" or "other state", respectively. $O = 1, 0$.
- Transition and observation matrices are initialized using Dirichlet distribution [10].
- Training data set is a series of observations taken from the event log.
- Baum Welch algorithm is used to train the HMM [10].

The beta distribution can be used to model events which are constrained to take place within an interval defined by a minimum and maximum value (e.g., to describe the time to

completion of a task). The Dirichlet distribution is the multidimensional generalization of the beta distribution. Hence, we initialize both the transition and observation probability matrices A and B, using the random Dirichlet distribution.

The historical log data contains a path that was executed in the past from the workflow depicted in Figure 1. If the tasks that were completed were: $T_1, T_2, T_5, T_7, T_9, T_{11}, T_{14}$, the observation assigns "1" for each task completed, and "0" otherwise. This can be represented as the observed values: $1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1$. If the path in the log was $T_1, T_2, T_5, T_7, T_9, T_{10}, T_{14}$, the observed value would change to: $1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1$[5].

After we obtain the trained HMM, we use it to calculate the probability of occurrence of the path that is of interest to the end-user. For instance, if the user desires to calculate the probability of occurrence of paths that contain *Task 7* or *Task 8*, the paths for query could be several for the workflow pattern shown in Figure 1. Four of the possible paths $Path_i$, where $i \in N$ and their corresponding HMM representations are:

1) $Path_1 = T_1, T_2, T_4, T_7, T_9, T_{13}, T_{14}$
   HMM Observation = $(1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1)$
2) $Path_2 = T_1, T_3, T_6, T_8, T_9, T_{10}, T_{14}$
   HMM Observation = $(1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1)$
3) $Path_3 = T_1, T_2, T_5, T_7, T_9, T_{11}, T_{14}$
   HMM Observation = $(1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1)$
4) $Path_4 = T_1, T_3, T_4, T_8, T_9, T_{12}, T_{14}$
   HMM Observation = $(1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1)$

We use the *Forward* algorithm (P_F) to calculate the probability of occurrence of a given observation sequence. We use this probability value to predict the remaining time for a given sequence of probable tasks $(T_k, T_{k+1}, \ldots)$ using Equation (7).

$$\frac{AvgT(T_k)\sum_{i=1}^{n} P_F(Path_i) + AvgT(T_{k+1})\sum_{i=1}^{n} P_F(Path_i) + \ldots}{\sum_{i=1}^{n} P_F(Path_i)} \tag{7}$$

In the above example scenario, the paths that could be taken after any of the tasks *Task 4* to *Task 6* complete is either *Task 7* (T_k) or *Task 8* (T_{k+1}). The average *remaining time* to completion can now be obtained from the sample mean function $AvgT$ (described in Section II-A) using Equation (7).

In Figure 5, we compare the *remaining time* values obtained using the annotated transition system alone [12] and the time calculated by HMM using the average time of each task (Equation (7)).

F. Discussion

In Figure 5, we have plotted the predicted remaining time for the sample workflow (Figure 1) using: Maximum Time (MaxT), multivariate regression (MReg), annotated transition system (ATS), and hidden Markov model (HMM). The remaining time is obtained when the current execution state of the workflow is *Task 1*, intermediate states (*S1* or *S2*), or *Task 9*.

[5]Notice the change of value for positions 10 (completed) and 11 (other state)

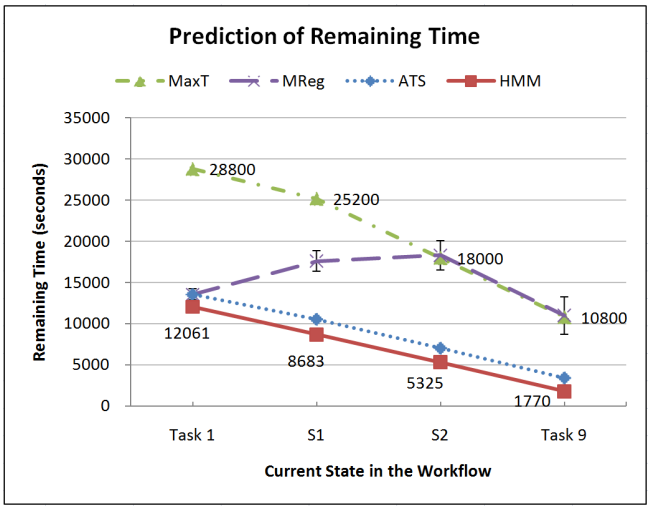

Fig. 5. Predicted remaining time of a workflow when using: Maximum Time (MaxT), Multivariate Regression (MReg), Annotated Transition System (ATS), and Hidden Markov Model (HMM)

Not surprisingly, the MaxT method gave the highest predicted values for the remaining time to completion for all the tasks in the workflow. As MaxT sums the real values from the historical log data (not based on a fitting), these values provide the upper-limit to the time values when comparing against other results obtained using model-based predictions.

The remaining time predicted by the MReg method was in low values for the beginning states of the workflow and approached the values given by MaxT for the states near the completion. This trend could be due to the linear model we constructed during the regression phase, where the independent variables (tasks) were *linearly* summed to obtain the dependent variable *time*. The assumption that the tasks were independent is also an over simplification to obtain the model. Nevertheless, the predicted values remained within the upper-limit defined by MaxT. We also plotted the error margins for MReg by taking the difference between the predicted value and the value (minimum and maximum) existing in the event log. The errors increased in value towards the end states of the workflow, which shows that the MReg based fitting does not take into account workflow state dependencies.

We implemented the ATS based prediction method as described in [12]. The predicted times for each task linearly decreased as the tasks completed from *Task 1* through *Task 9*. This is an expected behaviour, assuming that the process that is observed is in *steady state*. The interesting parameter to notice is the slope of the ATS line in Figure 5, which is similar to the MaxT's prediction. As the ATS is based on real values obtained from log data, and not on a model, the similarity of slope values for ATS and MaxT validates the implementation of the simulation system.

The predicted remaining time values given by the HMM is the lowest amongst all the other approaches. This can

be attributed to the characteristics of HMM. Its prediction technique is based on model fitting (training its parameters based on historical data). In addition, in HMM, the current state is dependent on its immediate predecessor, which enables it to incorporate dependencies defined in a workflow into its model. These two characteristics have enabled the HMM to predict the remaining time as closely as the ATS and well below the upper-limit defined by MaxT. The values obtained are not the optimal values, as the HMM based method is highly dependent on the values chosen during the construction of its transition matrix, observation probabilities, and initial states. Nevertheless, HMM provided a model based fitting and is not susceptible to minor fluctuations to the event logs.

V. Conclusion and Future Work

In this paper, we proposed an architecture that facilitates the simulation as well as time prediction of business processes. We implemented four prediction techniques: descriptive statistics, regression, annotated transition system, and hidden Markov model as plug-in components in the simulation system. We also evaluated the techniques using a simulated business process and compared the predicted remaining time values given by each technique. These values followed the expected trend (decrease in remaining time as tasks get completed), thus validating the simulation system. We also found that hidden Markov model based prediction provides better predictions as compared to regression and the annotated transition system.

Business process predictions are not just driven by time, but also the availability of human (and computing) resources and their capabilities, existing process queues, operational costs, etc. As part of our future work, we are interested in studying the impact of these parameters when predicting remaining time to completion of business processes. We would also like to validate the algorithms by using real-world application use-cases.

References

[1] T. Blum, N. Padoy, H. Feuner, and N. Navab. Workflow mining for visualization and analysis of surgeries. *International Journal of Computer Assisted Radiology and Surgery*, 3:379–386, 2008.

[2] J. Brocke and M. Rosemann. *Handbook on Business Process Management*, volume 1 of *International Handbooks on Information Systems*. Springer, 2009.

[3] J. Cardoso and M. Lenic. Web process and workflow path mining using the Multimethod approach. *Int. J. Bus. Intell. Data Min.*, 1:304–328, March 2006.

[4] J.J. Faraway. *Practical Regression and Anova using R*. July 2002.

[5] D. Grigori, F. Casati, U. Dayal, and M. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 159–168, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[6] R.K.L. Ko, S.S.G. Lee, and E.W. Lee. Business process management (BPM) standards: a survey. *Business Process Management Journal*, 15(5):744–791, 2009.

[7] I. Leong, Y. Si, S. Fong, and R. P. Biuk-Aghai. Critical path based approach for predicting temporal exceptions in resource constrained concurrent workflows. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '09, pages 711–717, New York, NY, USA, 2009. ACM.

[8] H. Schonenberg, J. Jian, N. Sidorova, and W.M.P. van der Aalst. Business trend analysis by simulation. In *Proceedings of the 22nd international conference on Advanced information systems engineering*, CAiSE'10, pages 515–529, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, 2003.

[10] A. Stolcke and S.M. Omohundro. Hidden Markov Model Induction by Bayesian Model Merging. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 11–18, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[11] W.M.P. van der Aalst. Business Process Simulation Revisited. In *Enterprise and Organizational Modeling and Simulation*, volume 63 of *Lecture Notes in Business Information Processing*, pages 1–14, 2010.

[12] W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450 – 475, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.

[13] W.M.P. van der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14:5–51, July 2003.