Predicting Workflow Tasks at Run-time

Thomas Eisenbarth and Bernhard Bauer

University of Augsburg, Germany,
{eisenbarth,bauer}@ds-lab.org,
WWW home page: http://www.ds-lab.org

Abstract. Modeling, executing and analyzing business processes gained much attention both in research as well as industry over the last years. Deregulation, globalisation and increasing competition demands continuous improvement both regarding quality and speed in design, implementation and throughput times in business processes.
Our goal in this paper is to present an approach to predict workflow states at run-time. We show a use cases where this prediction can help to decrease throughput time of business processes.

Key words: Business Process Management, Workflows, Prediction

1 Introduction

Due to increasing global competition as a result of globalized industries and simultaneous profit-oriented shareholders, enterprises today demand for high automation and quality as well as fast execution and completion of daily tasks. Be that in departments responsible for production, sales or administration.

Utilizing a formal foundation, enterprises today often express recurring work in a graphical definition in form of business process models[1]. Regularly these graphical models are transformed into lower-level languages to be executed in business process management systems (BPMS) [13, 6]. For every one of those recurring processes, one instance of the model is instantiated and runs within a executing container software until it finishes or is aborted due to a failure. Using this kind of specification for workflows it is possible to limit mistakes of employees by predefining when, by whom and how information is gathered, processed and distributed in a workflow. Tasks where humans get involved in workflows are referred to as *human tasks*. This way enterprises achieve sustainable quality of work.

These first steps in office automation shortly lead to significant interest in industry. Therefore, shortly after workflow systems evolved there was ongoing research on how to improve design and modeling, management and execution of workflows in order to enhance (most-often) business applications [12]. These efforts include

[1] We do not make any difference between business process and workflow in this paper and use the terms synonymously.

– automatic or semi-automatic modeling,
– faster and more reliable execution of workflow instances,
– lower throughput time,
– less abortion of running instances,
– detailed analysis of finished processes e.g. to find out bottlenecks.

This work is part of the research area Business Process Intelligence (BPI) aiming at applying measurement and analysis techniques in the area of business process management. This primarily includes the execution phase of business processes and makes use of analysis, prediction, monitoring, control and optimization.

The objectives we're aiming at with the approach presented in this paper are faster execution, less overhead, less surprise and smoother flow of business processes at run-time with a special focus on human task-centric process models. Therefore, we focus on processes that are not fully automated but contain human tasks such as approval, input submissions, human decisions, and so on. We present an approach to predict (likely) upcoming tasks in workflows which is primarily useful for human tasks as employees anticipate future work and e.g. try avoid bottlenecks by delegating work, preparing preliminaries and so on. The prediction is based on currently running or completed workflow instances.

2 Basics

In [2] Georgakopoulos et al. defined a *workflow* as a "collection of tasks organized to accomplish some business process (e.g., processing purchase orders over the phone, provisioning telephone service, processing insurance claims). A task can be performed by one or more software systems, one or a team of humans, or a combination of these. Human tasks include interacting with computers closely (e.g., providing input commands) or loosely (e.g., using computers only to indicate task progress)." IN [17] the Workflow Management Coalition (WfMC) defines workflow as: "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

Furthermore, a WorkFlow Management System (WFMS) is defined as: "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of Information-Technology (IT) tools and applications."

Systems executing business processes are called Business Process Management Systems (BPMS). In [16] Weske et al. define a BPMN as "a generic software system that is driven by explicit process designs to enact and manage operational business processes".

A graphical modeling notation common in industry is Business Process Model and Notation (BPMN) [5]. There is a successor (BPMN 2.0) which is in beta status currently. It aims to enable non-IT (but domain-experts) to design business processes. Thereafter those models are mapped to executable code such as Web Services Business Process Execution Language (WS-BPEL) although this mapping is challenging [9].

As already indicated earlier we do not differentiate between workflow and business process in this paper and use the terms interchangeable. The ideas presented in this paper is applicable to both workflows and business processes.

2.1 Workflow states

In this section we will first have a look at the different states that workflow elements usually traverse while a model is in execution.

Workflow model tasks traverse several phases when a model gets instantiated as shown in Figure 1. We will refer to the tasks as A, B and so on.

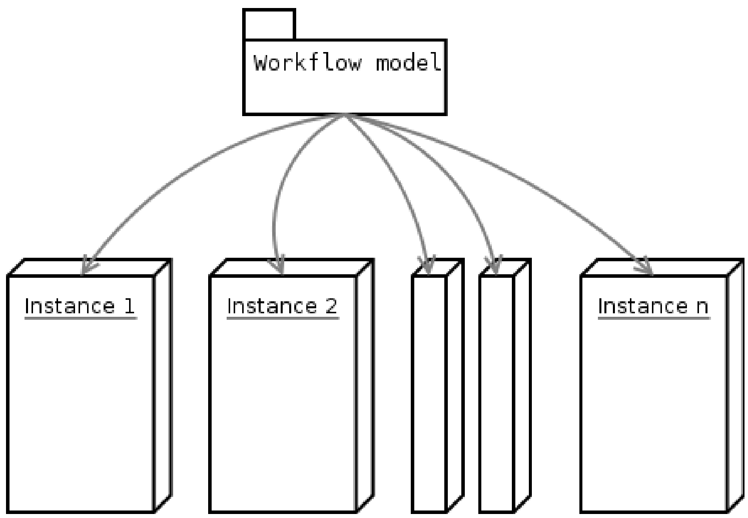

Fig. 1. Workflow model and its instances

- `Sleep`
  State of a task while not being executed and waiting for activation.
- `Start`
  Usually (and most commonly) when a preceding task finished (assuming a sequence pattern) the following task is started. But there are other cases that can lead to the start of a workflow task e.g. a Parallel Split Pattern or a special event.
- `Run`
  While the task is running and the defined activity gets accomplished.

– `Finished`
  At the moment the work of the task is done, the task switches to the state `Finished` and the workflow engine possibly enables a following task, evaluates a subsequent control pattern, etc.

In a very straightforward workflow as shown in Figure 2 in which no exceptions happen that could lead to an abortion of a running workflow both tasks A and B walk through these states once:

– Process model gets instantiated, A and B are at state `Sleep`
– A is put into state `Start` by the workflow engine as the start event is triggered.
– A is running until the task is completed.
– A is put into state `Finished`
– B is put into state `Start`
– B is running until it's completed, the workflow engine finishes the workflow as the next element is the end state.

For human tasks the most interesting state to know in advance is `Start` as this is the point of time where the task will require the person to accomplish work to allow to process instance to continue. The other states (`Run` and `Finished`) are consequences of the start of a task. That is why we will look at how to predict the start of tasks in the following.

2.2 Integration of human tasks in execution engines

Current implementations of BPMS often use a web-based component to access and interact with processes either to start new process instances or work on running instances. These human tasks typically include giving approval to a request or high invoice amount, enter specific information, make a manual decision and so on. It is quite obvious that the performance regarding e.g. throughput and response time of processes that involve human tasks are dependent on those tasks being completed quickly. As mentioned those tasks are presented to process-involved people in form of a list on a web portal or the like in order to get completed.

We propose to indicate another tasks list of **probably upcoming tasks** in order to enable people involved in workflows to estimate upcoming work, react in good time if bottlenecks might appear or simply to avoid that a task will be assigned to someone that is unavailable (and therefore will not be able to complete the task).

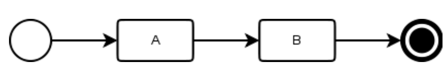

Fig. 2. Simple workflow example

2.3 Workflow patterns

In order to model parallel execution of tasks in workflows, to navigate through workflows based on decisions and to synchronize branches accordingly many notation languages such as BPMN [5] or YAWL [14] implement (at least parts of) control-flow workflow patterns [11, 15]. As these patterns have great influence on our approach and the we shortly introduce five basic workflow patterns:

1. *Sequence* is defined as a task in a process that is enabled after the completion of a preceding task in the same process. This means that the state of B changes to "Start" as soon as the preceding task (A) finished. This is shown in figure 2.
2. *Parallel Split* starts two process actions in parallel after the completion of a preceding one i.e. start B and C simultaneously after a preceding task A finished.
3. *Synchronization* joins two branches into one.
4. *Exclusive Choice* splits a single branch into two or more branches. In contrast to Parallel Split only (and exactly) one subsequent branch is followed. Which branch is activated depends on some decision mechanism. In practise decisions depending on order amount, customer state, etc. affects the control-flow of processes.
5. Simple Merge Simple Merge unites two or more branches into a single subsequent branch.

Although there are more patterns we confine to these as the functionality should be clear enough to understand the influence on this paper.

In the following, let there be a graph G representing a process model such that $G = (T, E)$ while T being a set of tasks (or "process actions") and further model elements such as control flow gateways and the like. Let E be a set of edges connecting tasks $t \in T$.

3 Predicting upcoming tasks

In general workflow engines change the state of process actions reacting on workflow patterns or events. We will examine the tasks of our prediction approach for workflow state changes in the following.

Before discussing the details, we have to define $m >= 1$ which is the number of (possibly upcoming) tasks that should be predicted and $o = [0..1]$ which is the global threshold to indicate whether we assume a task to get started. The higher o is set, the more likely the prediction is correct but the fewer tasks will be predicted at all.

We aim to calculate the set possible next tasks $\mathcal{M}$ of a single task $t \in T$ in a running instance of a workflow model using the following function:

$$\mathcal{P} : t \rightarrow \mathcal{M} \tag{1}$$

3.1 Step 1: Collect data

Our approach is based on historical process data. Therefore, we collect a set of (completed or currently running) sequences that included task t. We call this set a Instance Prediction Set (IPS).

It is theoretically possible that there is no data available to fill a IPS. This is especially the case if

- I is the first running instance ever.
- I is the first instance that reaches task t.

Moreover, let n be the the cardinality of IPS.

$$n = |IPS| \tag{2}$$

Please note that it is not possible to apply our approach if there are no process instances started in the past ($n = 0$). That is why we assume that $n >= 1$ which simply means that there was at least one instance of the examined model that was enacted in the past. This assumption is a fairly small limitation in our opinion.

For a formalization of this part see algorithm 1.

Algorithm 1 Collect historic data

```
Require: Task t
Require: History data set H
IPS ← []
for all sequences s in H do
   for all tasks t_2 in s do
      if t_2 == t then
         IPS ← IPS + s
      end if
   end for
end for
return IPS
```

Please note that the supply of the historical data is not focus of this paper. See related work and section 5 for more details.

3.2 Step 2: Calculate probability

Having collected relevant data in step 1, we describe the second step in the following that calculates the probability of tasks to be started as well as the estimated start date of the task.

In order to do so, we walk through all sequences of tasks stored in the IPS. Additionally we step through every sequence and increase the counter for every succeeded task at the respective position. We formalized this step in algorithm 2.

Afterwards we filter the tasks based on the constant o we defined earlier and obtain the final list of predicted tasks as described in algorithm 3.

Algorithm 2 Calculate successors

Require: Task t
Require: Instance Prediction Set IPS for task t
Require: Number m of tasks to predict
Require: $n \leftarrow IPS.size()$
$i \leftarrow 0$
$M \leftarrow []$
for all sequences s in IPS **do**
  for all tasks t_2 in s **do**
    if $i >= m$ **then**
      continue
    end if
    $R[i][t_2] \leftarrow R[i][t] + 1$
    $i++$
  end for
end for
return R

3.3 Discussing utilized parameters

At a first glance it might seem as if the more instances the IPS contains, the better the results will be. Basically this is true due to plain statistical reasons. Though, as we remarked in the introduction, there are reasons why we intentionally exclude data and limit the instances included in the prediction: At the time workflow I is running and the prediction takes place, the world changed to when the other instances in IPS were running. Lets assume a workflow contains the decision whether an employee is equipped with a PC or Macintosh. This might change at some point in time because management decides that only the design apartment should be equipped with Macintosh. That is why we think that a manageable amount of recent process instances will not only provide a adequate but good basis for run-time prediction.

3.4 Example

We show this model in an example. Figure 3 shows a basic workflow model. We will assume $o = 0.5$ and $m = 3$.

As stated before we cannot predict anything without at least one instance that run (or is still running) of this model. Figure 4 shows the path the first instance walked through the model.

Given a second instance of the example workflow model with a slightly different path outlined in Figure 5.

Algorithm 3 Run-time state prediction

Require: Task t
Require: Number m of tasks to predict
Require: List of successors R

```
for i = 0; i <= m; i++ do
    for all t in R[i] do
        x ← R[i]/n
        if x >= o then
            M[i] ← M[i] + t
        end if
    end for
end for
return M
```

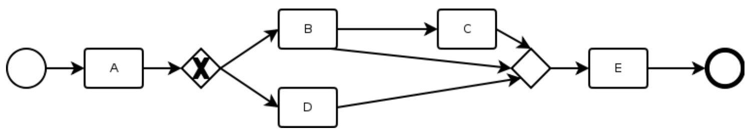

Fig. 3. Basic workflow model example

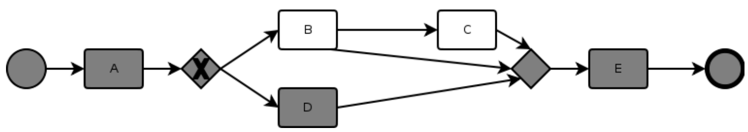

Fig. 4. Path of an instance of the sample model

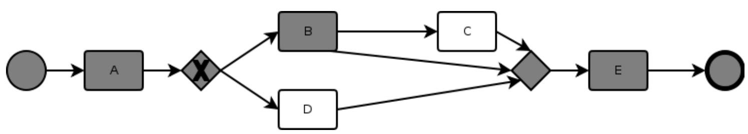

Fig. 5. Path of an instance of the sample model

We can shorten up the paths of these instances by referring the tasks the instances passed:

- Instance 1: `Start | A D E | End`
- Instance 2: `Start | A B E | End`

Given we launch another (third) instance of our model, after the `Start` (which is indicated by the circle in BPMN used in our example) the state of the following task `A` is set to `Start`. At this time we want to predict possible upcoming tasks. As we have two instances in our instances prediction base we obtain the following tasks and probabilities:

- $R[1][B] = 0.5$
- $R[1][D] = 0.5$

- $R[2][C] = 0$
- $R[3][E] = 1$

Given a forth instance is run and is contained in the IPS. The instance has the same signature like the first one (`Start | A D E | End`).

- $R[1][B] = 0.33$
- $R[1][D] = 0.66$
- $R[2][C] = 0$
- $R[3][E] = 1$

Utilizing the algorithm 3 with $o = 0.5$ we would rate the tasks D and E as predicted and present those to involved employees accordingly.

Another interesting figure was not respected so far: Using our approach we can easily get the abortion rate of our workflow model by looking at the probability to reach the end of our workflow. As we denominated the end state as "End" so far, we say that if the calculated probability for the final task is 1 the probability of a abortion of the workflow is not given at all while if the result was 0 no workflow at all reached the end state which would indicate a high abortion rate. Needless to say that there must be a reasonable amount of instances in the instance prediction base to get a useful result.

3.5 When to do prediction

As we focus on long-running workflows as described in the introduction, we want to be able to refine our prediction results even during the run-time of workflow instances and use the information of other currently running instances. That is why a prediction performed each time before a task is started is most effective although it surely costs most performance.

4 Related Work

A good overview about research work done in the area of business process analysis and business process improvement examining design-time as well as run-time analysis is given by van der Aalst in [12].

There is a lot of work done in several sub-disciplines such as simulation, monitoring and process mining on log files and new techniques that got applied to business process management in the last years are used for analysis and improvement [7, 8].

So far prediction in business processes and workflows was done in special cases, e.g. for error detection and prediction [4], which implies validation of models. Another approach by Rozinat et al. in [10] is using simulation to support business decisions. The motivation given for simulating processes is very similar to ours but mainly aims at executives having the possibility to react on the simulations' results. The method builds on historical data as we do, but does not have a direct and automatic support for running process instances for people directly involved in processes as in our approach.

Other prediction was utilized for predicting duration of tasks in order to estimate whether a order will be delivered in time, service level agreements (SLA) are violated and the like [1].

5 Conclusion and Future work

In this paper we have introduced first ideas about prediction of tasks during run-time and formalized the central ideas around our approach. Although we have not yet finished our implementation and therefore currently do not have any evidence or evaluation we feel certain our approach helps improving business process performance.

Therefore, we will work on the implementation and an evaluation in the near future. Additionally, we are think about integrating the approach into a graphical modeling language in order to support the full life-cycle of business processes. Two decisions at design-time have to and should be made by domain experts: Decide whether to enable prediction for single tasks or not: There are certainly tasks where the knowledge that they might get activated soon entails no additional value or might be even risky in some cases. The second domain-centric decision is at which degree of certainty a task should be recognized as predicted.

Although we have limited the prediction to human tasks in this paper there might be other areas where the approach is adaptable, e.g. in orchestration where automatic execution is prevailing. One implementation strategy for this could be to define an additional phase for tasks before the start phase. This prepare phase could — if it is possible which depends on the tasks — handle necessary preparation.

Additionally we plan to integrate temporal aspect such as the point of time a predicated task is estimated to start.

Although the supply of the historical data is not part of this paper, we figured out two possibilities to gain such data. One way is to make use of log files produced by process engines while executing processes as described in 4. Another possibility would be to actively maintain the prediction data (IPS and concrete task prediction) for all or selective tasks. For maximum performance utilizing a in-memory database might be useful for this purpose [3].

References

1. M. Castellanos, F. Casati, M. Sayal, and U. Dayal. Challenges in business process analysis and optimization. In C. Bussler and M.-C. Shan, editors, *TES*, volume 3811 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2005.
2. D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. In *DISTRIBUTED AND PARALLEL DATABASES*, pages 119–153, 1995.

3. E. P. C. Jones, D. J. Abadi, and S. Madden. Low overhead concurrency control for partitioned main memory databases. In *SIGMOD*, 2010.
4. J. Mendling, H. M. W. Verbeek, B. F. van Dongen, W. M. P. van der Aalst, and G. Neumann. Detection and prediction of errors in epcs of the sap reference model. *Data Knowl. Eng.*, 64(1):312–329, 2008.
5. OMG. Business process modeling notation, v1.2. OMG Specification, January 2009.
6. C. Ouyang, M. Dumas, and A. H. M. T. Hofstede. From BPMN Process Models to BPEL Web Services. In *In Proceedings of the 4th International Conference on Web Services (ICWS06*, pages 285–292. IEEE Computer Society, 2006.
7. C. Pedrinaci, J. Domingue, and A. K. A. de Medeiros. A core ontology for business process analysis. In M. Hauswirth, M. Koubarakis, and S. Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag.
8. C. Pedrinaci, D. Lambert, B. Wetzstein, T. van Lessen, L. Cekov, and M. Dimitrov. Sentinel: a semantic business process monitoring tool. In *OBI '08: Proceedings of the first international workshop on Ontology-supported business intelligence*, pages 1–12, New York, NY, USA, 2008. ACM.
9. J. Recker and J. Mendling. On the Translation between BPMN and BPEL. In J. Krogstie, T. Halpin, and H. E. Proper, editors, *Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'06), held in conjunctiun with the 18th Conference on Advanced Information Systems (CAiSE'06), Luxembourg, Luxembourg, EU*, pages 521–532. Namur University Press, Namur, Belgium, EU, 2006.
10. A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge. Workflow simulation for operational decision support. *Data Knowl. Eng.*, 68(9):834–850, 2009.
11. N. Russell, Arthur, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical report, BPMcenter.org, 2006.
12. W. M. P. van der Aalst. Challenges in business process analysis. In J. Filipe, J. Cordeiro, and J. Cardoso, editors, *ICEIS (Selected Papers)*, volume 12 of *Lecture Notes in Business Information Processing*, pages 27–42. Springer, 2007.
13. W. M. P. van der Aalst and K. Bisgaard Lassen. Translating unstructured workflow processes to readable BPEL: Theory and implementation. *Inf. Softw. Technol.*, 50(3):131–159, 2008.
14. W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.
15. W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.
16. M. Weske, W. M. P. van der Aalst, and H. M. W. Verbeek. Advances in business process management. *Data Knowl. Eng.*, 50(1):1–8, 2004.
17. WfMC. Terminology glossary, 1999.