Information Systems 57 (2016) 1–19

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Analysis and improvement of business process models using spreadsheets

Jorge Saldivar [a,*], Carla Vairetti [b], Carlos Rodríguez [a], Florian Daniel [a], Fabio Casati [a], Rosa Alarcón [c]

[a] University of Trento. Via Sommarive, 9 I-38123 Povo (TN), Italy
[b] Facultad de Ingeniería y Ciencias Aplicadas. Universidad de los Andes. Mons Alvaro del Portillo 12445, Santiago, Chile
[c] Pontifícia Universidad Católica. Av. Vicuña Mackenna 4860, Santiago, Chile

ARTICLE INFO

Article history:
Received 23 July 2014
Received in revised form
20 October 2015
Accepted 30 October 2015
Recommended by: M. Weske
Available online 10 November 2015

Keywords:
Business process analysis
Spreadsheets
Business process simulation

ABSTRACT

Software in general is thoroughly analyzed before it is released to its users. Business processes often are not – at least not as thoroughly as it could be – before they are released to their users, e.g., employees or software agents. This paper ascribes this practice to the lack of suitable instruments for *business process analysts*, who *design* the processes, and aims to provide them with the necessary instruments to allow them to also *analyze* their processes. We use the spreadsheet paradigm to represent business process analysis tasks, such as writing metrics and assertions, running performance analysis and verification tasks, and reporting on the outcomes, and implement a spreadsheet-based tool for business process analysis. The results of two independent user studies demonstrate the viability of the approach.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

The analysis of a piece of software, e.g., an algorithm or a mobile app, is a highly technical and daunting task typically performed by *developers* or *testers* who have the necessary technical background to know what to analyze and how. What is important is that the piece of software is analyzed by someone with the right skills, tools and methodologies.

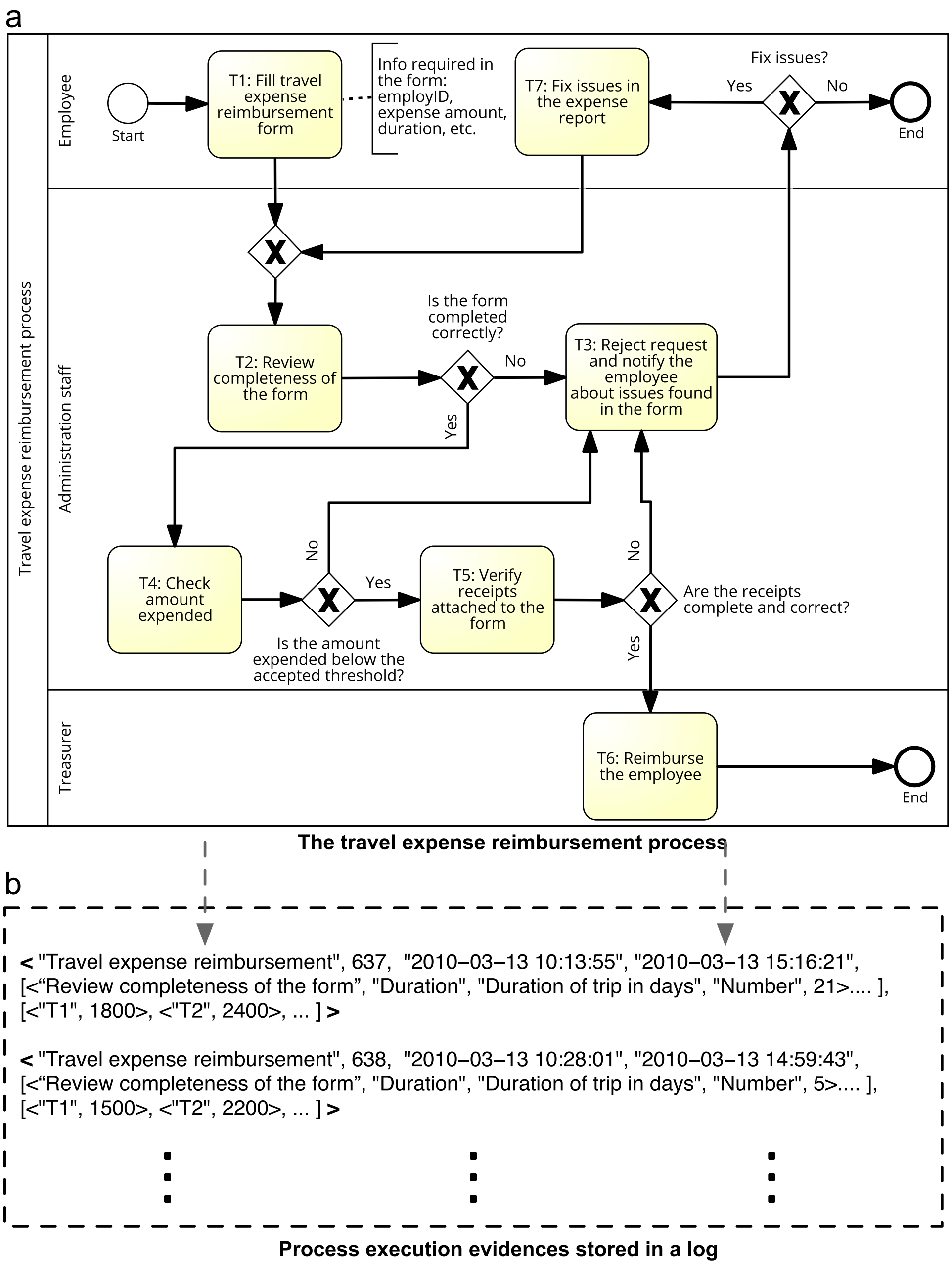
Interestingly, when it comes to *business processes* (BPs) this is not common practice. In fact, the *BP analysts*, who design the processes to be executed, often do not have the necessary instruments to analyze their artifacts, i.e., the *business process models*.

In the context of Business Process Management Systems (BPMSs), the tasks in the process models are typically implemented using web services [1]. The web services can be either fully automated or it can provide a web application that allows human operators to perform the tasks through suitable user interfaces. For this type of business processes, which implementation requires involving developers, the analysis is, therefore, done again by the developers, if at all. This in turn means that the concerns of the actual owners of the artifacts, the BP analysts, may not be properly taken into account before implementing and running the production processes. Identifying issues at this late stage of the process lifecycle can be time-consuming and costly.

Let us consider, for example, the *travel expense reimbursement process* in Fig. 1(a). Furthermore, let us assume that the process is currently in use in a service-based BPM system and that some problems have been identified by the BP analyst of the company. More concretely, he has

[*] Corresponding author.
E-mail address: jorge.saldivargalli@disi.unitn.it (J. Saldivar).

http://dx.doi.org/10.1016/j.is.2015.10.012
0306-4379/© 2015 Elsevier Ltd. All rights reserved.


a
Travel expense reimbursement process
Employee
Administration staff
Treasurer
Start
T1: Fill travel expense reimbursement form
Info required in the form: employID, expense amount, duration, etc.
T7: Fix issues in the expense report
Fix issues?
Yes
No
End
T2: Review completeness of the form
Is the form completed correctly?
No
Yes
T3: Reject request and notify the employee about issues found in the form
T4: Check amount expended
No
Yes
Is the amount expended below the accepted threshold?
T5: Verify receipts attached to the form
No
Are the receipts complete and correct?
Yes
T6: Reimburse the employee
End
The travel expense reimbursement process
b
Process execution evidences stored in a log

```
< "Travel expense reimbursement", 637,  "2010–03–13 10:13:55", "2010–03–13 15:16:21",
[<"Review completeness of the form", "Duration", "Duration of trip in days", "Number", 21>.... ],
[<"T1", 1800>, <"T2", 2400>, ... ] >

< "Travel expense reimbursement", 638,  "2010–03–13 10:28:01", "2010–03–13 14:59:43",
[<"Review completeness of the form", "Duration", "Duration of trip in days", "Number", 5>.... ],
[<"T1", 1500>, <"T2", 2200>, ... ] >

      .                    .                    .
      .                    .                    .
      .                    .                    .
```

Fig. 1. BPMN model of a travel expense reimbursement process and a possible execution log.

noticed that with the current resources assigned to operate this process, only 70% of all the reimbursement requests are processed on time. The BP analyst would like to change the process in order to improve its performance without the need of having to increase the amount of resources assigned to the process. In addition to this, he has also noticed that the amount of many reimbursement requests are far below the operational costs of having to run the BP to process the request and that, in such cases, it may just be better to immediately reimburse the employee without having to run the whole process and incur in costs that are not justified by the requested amount.

Before investing the necessary effort for implementing and deploying changes in the process, the BP analyst needs to find answers to key questions, such as *how many reimbursement requests, per quarter of the year, fall within the 30% of requests that are not processed on time, what should the value be for the amount requested under which the request is immediately reimbursed, and whether all these requests can be reimbursed without exceeding the maximum amount of 15K euros imposed by the accounting department.* These are business questions that require the possibility to try different process execution scenarios that reproduce different execution outcomes. The BP analyst needs to be able to specify the typical behavior per quarter of the year, check whether the new *fast track* reimbursement will comply with the constraints above, analyze and visualize the results to propose a fine tuning of the process, and communicate with the software developer working on the implementation of the process.

These tasks can be done manually if the BP is simple and the number of issues to be analyzed are small. Otherwise, analyzing a BP can turn into a daunting task that requires automation, programming and IT skills. BP analysts usually do not have these skills, and, in practice, BPs are therefore mostly analyzed by software developers that, by nature, focus more on implementation than on business aspects.

If the BP analyst nonetheless wants to analyze a given process, he needs to communicate his analysis goals, requirements and configurations to a software developer, who is able to implement and run the analysis on behalf of the analyst. Once analysis results are ready, the developer needs to communicate them back to the BP analyst, who in turn may ask for a re-run of the analysis under new settings, and so on. Understanding well a process may thus require several iterations between the analyst and the developer. This is not optimal and suffers from the same difficulties already extensively reported in the literature on software engineering in general and requirements analysis in particular, such as ineffective communication channels, inexpressive notations, and its reliant nature [2–4]. We propose therefore an approach that enables the BP analyst to analyze BP models on his own, with less reliance on and intervention of software developers.

We rely on *spreadsheets* to accomplish this. Created in 1979, spreadsheets are nowadays a common business tool and the most widely used end-user programming environment [5]. Scaffidi and colleagues had estimated that only in the United States, by 2012, more than 55 million people would have been using spreadsheets at the workplace, mainly for business purposes [6]. Considering the ubiquity of electronic spreadsheets in today's business landscape, these tools represent an ideal environment to build powerful user-centric solutions, such as BP analysis instruments that target BP analysts.

This paper describes a spreadsheet-based approach for business process model analysis that:

- maps the problem of business process performance analysis and verification to the problem of configuring and analyzing data in common *spreadsheets*;
- enables the *generation* of analysis spreadsheets from an extended business process model editor for BPMN process models [7];
- enables the BP analyst to define own *metrics, assertions* and *analysis reports*;
- automates the *simulation* of BP executions and generates *process execution logs*.

Two independent *user studies* demonstrate the viability of the approach, which was implemented in a prototype *tool for spreadsheet-based BP model analysis*, and a detailed *qualitative analysis* of the state of the art in BP model analysis highlights the benefits of the tool.

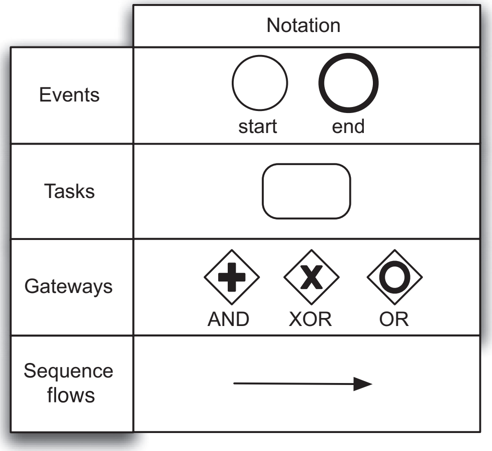


Fig. 2. BPMN elements related to control-flow specification.

Before outlining the details of the approach (Section 3), next we formalize the context and problem statement of the work. In Sections 4–6, we then explain the design, execution and analysis of BP models, respectively. In Section 7, we report on how we implemented our prototype tool, which we assess in Section 8. We conclude the paper with a discussion of related works and our final considerations on the results achieved.

2. Preliminaries and background

2.1. Business processes

As illustrated in Fig. 1(a), in practice, BPs are typically expressed through process models. In this paper, we represent processes using BPMN [7], a BP modeling notation is widely used both in industry and academy. The core elements of BPMN related to control-flow specification include events, tasks, gateways and sequence flows [8] (see Fig. 2). *Events* can be used to signal the start (*start event*) and end (*end event*) of a process. *Tasks* represent atomic activities to be performed as part of the process. *Gateways* are routing constructs that determine the execution flow of the process. They can be one of *AND gateway* (for creating concurrent execution flows), *XOR gateway* (to select one of a number of mutually exclusive flows), or *OR gateway* (to select any number of flows from the set of all outgoing flows). *Sequence flows* are used to represent the ordering relationship between any two elements (events, tasks and gateways) presented before. BPMN includes a richer set of elements, but in this paper we focus only on the ones presented here.

For notational convenience, we define a *business process model* as a tuple $BP = \langle pid, start, end, N, E, P \rangle$, where pid is a unique identifier, $start$ and end are the events that represent the start and end of a process, respectively, $N = T \cup G$ is the set of nodes of the process, with T being the set of tasks and G being the set of gateways of the process, $E \subseteq N \times N$ is the set of edges that connect pairs of nodes, and P is the set of properties that store business data produced and consumed during the execution of *BP*. A task $t \in T$, $t = \langle tid, tname \rangle$ is an activity of the process, where *tid* is a unique identifier and *tname* is the name of the task. A gateway $g \in G$, $g = \langle gid, C, gtype \rangle$ is a control flow node, where *gid* is a unique identifier, *C* is the set of conditions that controls the gateway, and $gtype \in \{AND, XOR, OR\}$ is the type of the gateway that derives from *C*. Each condition $c \in C$ is a tuple $c = \langle cid, e, expr \rangle$, with *cid* being a unique identifier, $e \in E$ being an outgoing edge of the gateway, and *expr* being a Boolean expression over process properties in *P* specifying the condition to follow the outgoing edge *e*. Process properties are of type $p = \langle nid, pname, pdesc, datatype, pvalue \rangle$, with *nid* being the identifier of the node producing a value for *p*, *pname* being the name, *pdesc* being the description, *datatype* being the data type, and *pvalue* being the value of the property (*pvalue* may be empty at the design time).

The BP model we introduce here is mapped to BPMN as follows. *start* and *end* in our model corresponds to the *start event* and *end event* in BPMN, respectively. Tasks *T* and gateways *G* map to the *gateways* and *tasks* in BPMN, respectively. Furthermore, the element $gtype \in \{AND, XOR, OR\}$ in a gateway $g \in G$ determines whether *g* refers to an *AND*, *XOR* or *OR gateway* in BPMN. Edges *E* in our model corresponds to the sequence flows in BPMN. Finally, the conditions *C* and properties *P* in our model are represented in BPMN through attributes associated to gateways and the process itself, respectively. It is worth noticing that the formalization and mapping introduced here are simple and straightforward, and that they are tailored to the notation needs of this paper. The interested reader can find a more detailed treatment of the formal semantics and analysis of BPMN process models (e.g., using Petri Nets) in Dijkman et al. [8,9].

The execution of a *BP* is a business process instance (or process instance for short). A *business process instance bpi* is a concrete case of operation of *BP* and can be represented as a tuple $\langle pid, piid, startTs, endTs, PI, TD \rangle$, where *pid* identifies the process model *BP*, *piid* identifies the process instance, *startTs* and *endTs* are the start and end timestamps of the execution, respectively, *PI* is the set of process properties produced by the execution of *BP*, and *TD* is a set of task durations $td = \langle tid, d \rangle$, with *tid* being the task identifier and *d* being the execution time of the task.

Process instances are typically tracked in the form of a process execution log [10] for later inspection and analysis. We define a *process execution log* as a set of process instances $L = \{bpi_j\}$. For example, Fig. 1(b) shows a possible log of the travel expense reimbursement process with two process instances. This representation differs from the more common, event-based representation of process logs, in that it proposes an already aggregated view on execution events. As we will see later, this choice helps us to simplify the presentation of process execution data to the BP analyst.

2.2. Business process analysis

The term business process analysis has a broad meaning and includes many different types of analyses such as simulation, diagnosis, verification and performance analysis [11]. In this paper, we focus our analysis on the combination of the last two. More specifically, we take the *dynamic* perspective of verification and performance analysis, i.e., we run our analysis based on the execution of the business process models.

From this dynamic perspective, verifying a business process model means analyzing whether or not the behavior of its instances matches a given expected behavior. For example, in the scenario described in the introduction, the BP analyst may want to verify whether, under different execution conditions, the sum of the amounts for the reimbursement processed using the *fast track* option is kept under 15K euros in each quarter of the year. In order to perform this verification, the BP analyst needs to be able to (i) specify the *expected behavior* of the business process, (ii) provide the inputs for the process, run it, and track its *observed behavior*, and (iii) *analyze* the expected and observed behavior in order to verify if they match. We call the joint realization of these tasks *business process verification and performance analysis*. For conciseness, in the rest of the paper we refer to this simply as *BP analysis* and explicitly refer to verification and performance analysis when needed.

The *expected behavior* of the process is partly specified by the process model *BP*. However, the process model provides only static, structural information about the process; if instead the object of the verification are the dynamics and data produced by the execution of the process as we discuss here, additional constructs are necessary. This is where the *performance analysis* part comes into play. More concretely, we use *metrics*, i.e., measures that capture the performance of a process starting from process execution evidences. Formally, we can define a metric as a function $m(L) = val$, where *L* is the process execution log and $val \in \mathbb{R}$ is the metric's value. Although this definition allows for the computation of cross-process metrics, i.e., of metrics computed over execution evidence from different process models, for simplicity in this paper we limit our attention to single-process metrics only.

The availability of metrics further allows one to express expected behavior in terms of conditions over metric values. Such conditions can be expressed as *predicates*, which are Boolean statements over a metric *m*. Formally, we can represent a predicate as a function $pred(L, m) = bool$, where *L* is a process execution log, *m* is a metric and $bool \in \{true, false\}$ holds the evaluation of the predicate. Predicates can be combined using standard logical operators, such as *AND*, *OR* and *NOT*, to build assertions. An *assertion* can be defined as a function $a(L, Pred) = bool$, where *L* is a process execution log, *Pred* is a set of predicates and *bool* is as defined before. Assertions allow one to write arbitrarily complex combination of predicates to specify and check the expected behavior of a process.

In order to assess the behavior of a process, we need to run the process and record its *observed behavior*. For already implemented processes or services, this behavior can be extracted from the log *L* of the process. For processes or services that have not yet been implemented, we need to find the way of generating *L* by exercising the process model *BP*. We will get back to this issue in Section 2.3

The *analysis* of *BP* now requires evaluating the assertions and metrics over the collected execution evidence *L* and visualizing the respective outcomes. Doing so requires setting up a suitable analysis report. We define an *analysis report* as a function $r(L, M, A) = V$, where L is a process execution log, M is a set of metrics, A is a set of assertions and V is a set of tables and charts that summarize the analysis outcomes. For example, $v \in V$ can be a pie chart that shows the percentages of *true* and *false* evaluations of an assertion $a \in A$ over the set of process instances in L. Such reports serve not only as a means to convey the outcomes to the analyst but also as a communication tool between the analyst and the software developer implementing the process.

2.3. Business process simulation

One way to obtain the event log L when it is not possible or convenient to run the real process to generate it is to use *business process simulation* [12], which mimics the execution of process instances, given a business process model BP and a suitable configuration. We propose to use this approach to obtain the process execution log L.

We define a *BP simulator* as a function $BPsim(BP, SS) = L$, where BP is a process model, SS represents the settings used to simulate the BP, and L is the process execution log generated by the simulation. BP simulation thus enables the BP analyst to mimic different process execution scenarios and obtain corresponding execution evidences.

2.4. Problem statement

The problem we aim to solve in this paper is devising an approach that enables the BP analyst to verify and analyze the performance of business processes without the need for software development skills. The first goal is to enable the BP analyst to write own metrics M and assertions A, to obtain a process execution log L, and to design own analysis reports r, so as to be able to autonomously analyze the behavior of a business process BP. The second goal is to do so in a fashion that allows the BP analyst to easily discuss his findings with the software developer in charge of implementing processes. Our hypothesis is that mapping the BP analysis problem as defined in this paper to the design of a spreadsheet calculation allows us to achieve both goals at the same time, in particular, given that spreadsheets are omnipresent in business and well-known by average BP analysts [47].

3. Spreadsheet-based business process analysis

We specifically consider the case of service-based BPs, where activities are executed by web services; human actors are hidden behind web service interfaces. Obtaining a log file for this type of BPs requires either invoking real web services (if such are available and do not have any persistent side effects) or simulating web service invocations (if the web services do have side effects or are not available at all). We assume that the BP analyst is capable of designing coarse BP models using the Business Process Modeling Notation (BPMN) and that he is familiar with spreadsheet tools like Microsoft Excel or Google Spreadsheets. We also assume that there is a software developer implementing the process and its web services, starting from the coarse BP models.

3.1. Requirements

Given these assumptions and the above problem statement, we identify a set of *functional requirements*. We group them into categories that correspond to the BP analysis phases (Section 2) they are related to:

Specification of expected behavior:

- **R1:** The solution shall enable the design of the BP model BP, along with its process properties P.
- **R2:** The solution shall enable the writing of metrics M and assertions A over the process execution log L.
- **R3:** The solution shall enable the storage of metrics and assertions for later reuse.

Obtainment of observed behavior:

- **R4:** The solution shall enable the configuration and simulation of BP to obtain a corresponding log L.
- **R5:** The solution shall enable the use of existing implementations of web services used by BP that do not produce unwanted side effects.
- **R6:** The solution shall enable the configuration and simulation of web services used by BP that do have unwanted side effects or that do not exist yet.
- **R7:** The solution shall enable the storage of the generated log L.

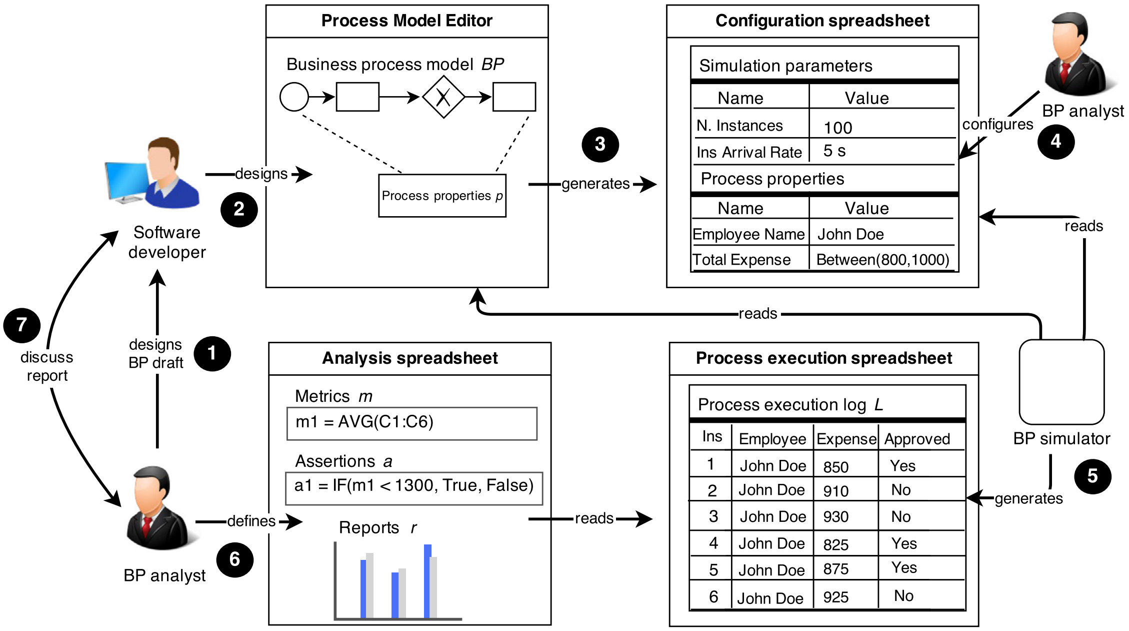
Analysis and reporting:

- **R8:** The solution shall enable the creation of analysis reports r based on BP, M, A and L.
- **R9:** The solution shall enable the storage of reports for future reuse, e.g., for re-running the analysis under different conditions.
- **R10:** The solution shall enable the sharing of BP analysis configurations and results with other stakeholders (e.g., with software developers).

The implicit, *non-functional* requirement is that the solution's tools that target the BP analyst shall not need any software development skills.

3.2. Approach

The overall approach proposed in this paper takes into account the fact that there are tasks that require specific technical skills that BP analysts may not have and that may prevent them from being able to analyze BPs. For example, the detailed design of *executable* process models in BPMN and the configuration of the more technical aspects is usually out of the reach of typical BP analysts. The design of executable BP models may in fact require the developer to use a larger set of modeling constructs than introduced in Section 2.1 (e.g., events or messages); the analyst only needs to master the subset of constructs introduced in Section 2.1 to be able to run his analysis. We therefore



Name	Value
N. Instances	100
Ins Arrival Rate	5 s

Name	Value
Employee Name	John Doe
Total Expense	Between(800,1000)

Ins	Employee	Expense	Approved
1	John Doe	850	Yes
2	John Doe	910	No
3	John Doe	930	No
4	John Doe	825	Yes
5	John Doe	875	Yes
6	John Doe	925	No

Fig. 3. Spreadsheet-based approach to BP analysis.

propose to separate the tasks related to (i) the design and configuration of executable process models, and (ii) the analysis of BPs. Task (i) is assigned to software developers, while task (ii) to BP analysts (see Fig. 3). This separation of duties is not only practical and realistic, but it also leverage on the skills and interests of each role.

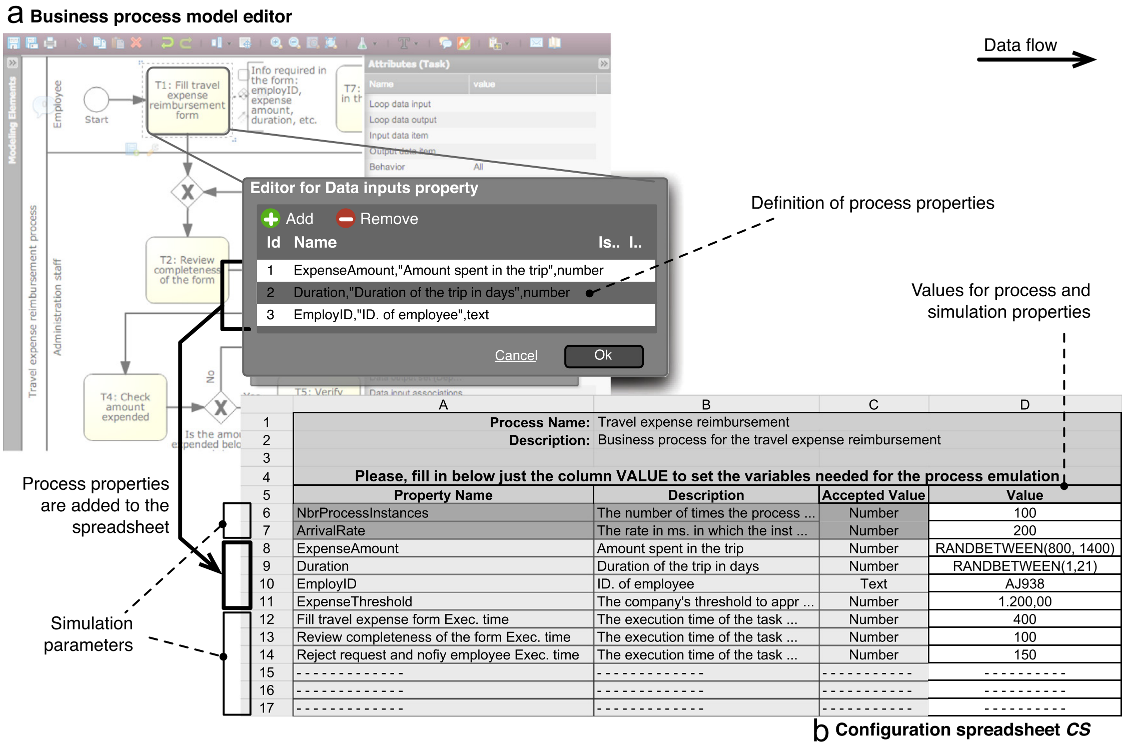
In order to approach the requirements discussed in Section 3.1, we leverage on BPMN and the *spreadsheet paradigm* to provide an approach for the analysis of BPs. BPMN is used to model executable BPs. The spreadsheet is used as interface toward the BP analyst and as communication instrument between the analyst and the developer. Simulation is used to safely generate behavioral information for those web services of the service-based BP that may have persistent side effects in the system or do not yet have a readily usable implementation. Fig. 3 illustrates our approach.

Starting from a *draft of BP* (step 1), the *software developer* refines and implements the process (2) using an extended *BPMN editor* (**R1**). This produces *BP*, including the set of process properties *P* that can be used for analysis. Given these ingredients, the BPMN editor generates a so-called *configuration spreadsheet* (3), which contains the process properties and a set of simulation parameters. Simulation parameters are used to configure the dynamics of the simulation (**R4**); they include parameters such as the number of process instances to be simulated, the rate at which instances are to be generated, and the execution time of simulated web services (**R6**). Process properties *P* are used to configure business data for different process execution scenarios; they are associated to the nodes of *BP* and may refer to both real or simulated web services (**R5**, **R6**). Activities of *BP* that refer to real services are marked as such and pre-configured by the developer in the extended BPMN editor; the BP analyst can configure the behavior only of simulated services. He does so simply by editing the spreadsheet and defining values for the simulation parameters and process properties (4). Once the simulation is configured, the *BP simulator* reads the configuration and *BP* and runs the simulation, mimicking the web service behaviors defined by the BP analyst and invoking existing web services. As a result, it generates (5) a process execution log *L* that contains the observed behavior, which is again stored as a *process execution spreadsheet* (**R7**). The actual verification and performance analysis is again done by the BP analyst using an *analysis spreadsheet* (6). In this spreadsheet, the analyst defines the metrics *M* and assertions *A* over the generated log *L* as standard spreadsheet functions (**R2**). The spreadsheet automatically performs the necessary calculations, and allows the BP analyst to define charts or tables for the visualization of results (*V*), turning the analysis spreadsheet into a report *r* that can easily be saved (**R3, R8, R9**) and shared with the developer (**R10**) (7).

In the following, we detail how processes are modeled, how the simulation is performed, and how predicates, assertions and analysis reports are calculated.

4. Business process modeling and simulation configuration

Setting up a BP analysis requires a suitable design of *BP* and the configuration of the simulation to be performed. The definition of the process properties *P* by the developer and their configuration by the BP analyst play an important role in setting up the BP analysis. The relevance of process properties its twofold: First, gateway conditions that determine the control flow of a process are defined over process properties. That is, they determine the runtime scenarios of process instances. Second, they are the starting point for the design of metrics and, hence, for the


a Business process model editor
Data flow
Modeling Elements
Employee
Start
T1: Fill travel expense reimbursement form
Info required in the form: employID, expense amount, duration, etc.
T7: in th
Attributes (Task)
Name
value
Loop data input
Loop data output
Input data item
Output data item
Behavior
All
Travel expense reimbursement process
Administration staff
T2: Review completeness of the form
T4: Check amount expended
No
Is the amount expended belo
T5: Verify
Data input associations
Definition of process properties
Values for process and simulation properties
Process properties are added to the spreadsheet
Simulation parameters
b Configuration spreadsheet CS

Editor for Data inputs property

Add Remove

Id	Name	Is..	I..
1	ExpenseAmount,"Amount spent in the trip",number		
2	Duration,"Duration of the trip in days",number		
3	EmployID,"ID. of employee",text		

Cancel Ok

	A	B	C	D
1	Process Name:	Travel expense reimbursement		
2	Description:	Business process for the travel expense reimbursement		
3				
4	Please, fill in below just the column VALUE to set the variables needed for the process emulation			
5	Property Name	Description	Accepted Value	Value
6	NbrProcessInstances	The number of times the process ...	Number	100
7	ArrivalRate	The rate in ms. in which the inst ...	Number	200
8	ExpenseAmount	Amount spent in the trip	Number	RANDBETWEEN(800, 1400)
9	Duration	Duration of the trip in days	Number	RANDBETWEEN(1,21)
10	EmployID	ID. of employee	Text	AJ938
11	ExpenseThreshold	The company's threshold to appr ...	Number	1.200,00
12	Fill travel expense form Exec. time	The execution time of the task ...	Number	400
13	Review completeness of the form Exec. time	The execution time of the task ...	Number	100
14	Reject request and nofiy employee Exec. time	The execution time of the task ...	Number	150
15	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
16	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
17	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -

Fig. 4. The instruments used for BP modeling and simulation configuration.

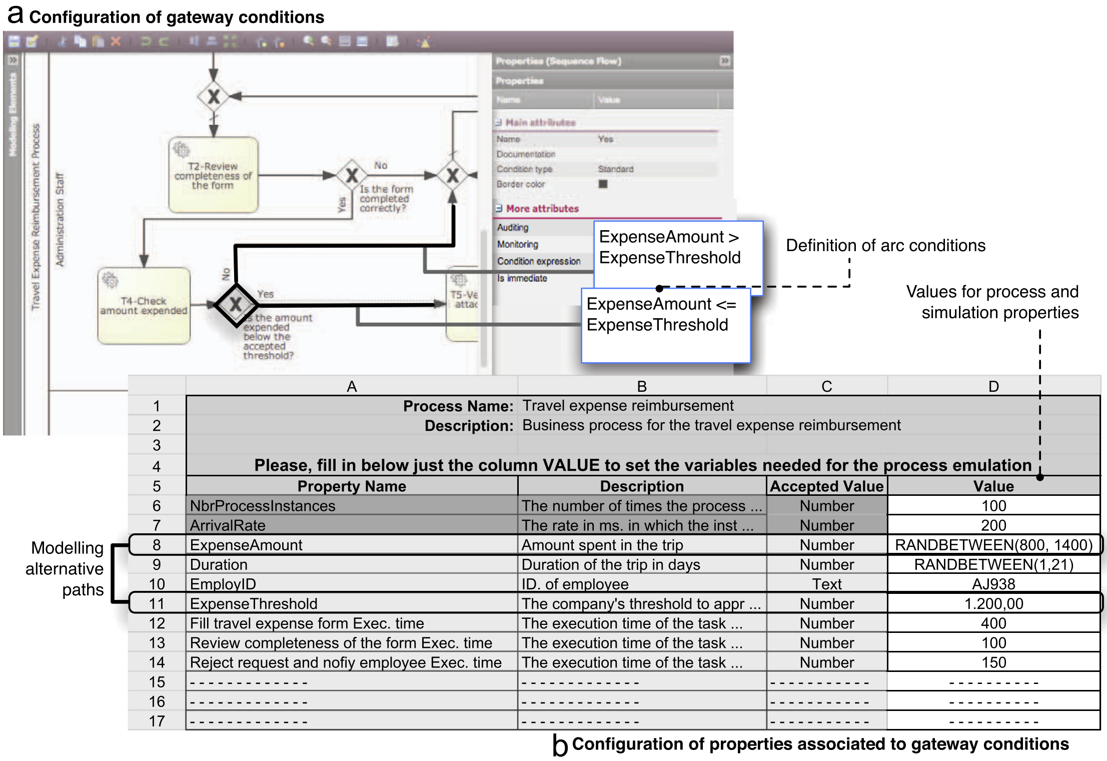
actual design of the verification and performance analysis. The simulation parameters allow the BP analyst to define the time behavior of simulated tasks and the number of task instances to be generated.

The developer models BPs using BPMN [7], which is a standard process modeling notation targeting both BP analysts and software developers. The BPMN constructs presented in Section 2.1 allow him to associate process properties to tasks. For example, Fig. 4(a) shows the definition of three process properties for the task *Fill travel expense reimbursement form*, namely, *ExpenseAmount*, *Duration* and *EmployID*. Each property requires a *name*, *description* and *datatype*, separated by commas, matching the model $p = \langle nid, pname, pdesc, datatype, pvalue \rangle$ introduced before. The value for *nid* is automatically derived by selecting a task in the process editor; *pvalues* are defined by the BP analyst using a *configuration spreadsheet CS*. In the same vein, by using standard BPMN constructs the developer can define conditional rules to control the execution of the process. Conditions are defined over process properties. Fig. 5(a) shows the conditions that regulate the execution flow over the outgoing arcs of the highlighted decision point. Each condition is set by the developer, who defines a Boolean expression, e.g., $ExpenseAmount \le ExpenseThreshold$, over each gateway's outgoing arc.

A *spreadsheet* is a bi-dimensional array *s* where each element *s*(*i*,*j*), with *i* representing the column index and *j* the row index, represents a cell. A cell $s(i,j)$ can contain one of (i) a value that consists of an alpha-numeric datum, such as in $s(i,j) \leftarrow$ "*AJ*487", (ii) a reference to a different cell, such as in $s(i,j) \leftarrow s(p,k)$, or (iii) a formula that combines functions, values and references to other cells such as in $s(i,j) \leftarrow sum(13, s(p,k))$.

The *configuration spreadsheet CS* imports the process properties *P* of *BP* for their configuration. To do so, we can follow the simple rule of mapping each property $p_j \in P$ to one spreadsheet row, like in $CS(1,j) \leftarrow p_j.pname$, $CS(2,j) \leftarrow p_j.pdesc$ and $CS(3,j) \leftarrow p_j.datatype$. Fig. 4(b) shows how the properties are represented in the spreadsheet. The spreadsheet's cells are indexed using letters for columns and numbers for rows. For example, the process property *ExpenseAmount* associated to the BP model is mapped to row 8 in the spreadsheet using the mapping $\texttt{CS(A}, 8) \leftarrow$ "`ExpenseAmount`", $\texttt{CS(B}, 8) \leftarrow$ "`Amount spent in the trip`", and $\texttt{CS(C}, 8) \leftarrow$ "`Number`". The last cell (`CS(D,8)`) is used to set the values of the property. The rest of the properties are mapped following the same mapping logic.

To define values for properties, the BP analyst can use a constant value, as in the cell `CS(D,10)`, or he can choose to write a formula that generates values for the cell. For example, the cell `CS(D,8)` uses the function `RANDBETWEEN(800, 1400)` to compute random values between 800 and 1400. By assigning non-constant values to the properties involved in the conditional expressions of gateways, the BP analyst can model the execution of alternative paths. As Fig. 5(b) illustrates, the random function `RANDBETWEEN`, used to define the values of the property *ExpenseAmount*, is what models the conditional execution of either the *Yes* or *No*-labeled outgoing arcs of the gateway highlighted in Fig. 5(a). The values obtained from these formulas are computed for each process instance at BP simulation time.



a Configuration of gateway conditions

Modelling Elements
Travel Expense Reimbursement Process
Administration Staff
T2-Review completeness of the form
No
Yes
Is the form completed correctly?
T4-Check amount expended
No
Yes
Is the amount expended below the accepted threshold?
T5-Ve attac
Properties (Sequence Flow)
Properties
Name
Value
Main attributes
Name
Yes
Documentation
Condition type
Standard
Border color
More attributes
Auditing
Monitoring
Condition expression
Is immediate
ExpenseAmount > ExpenseThreshold
ExpenseAmount <= ExpenseThreshold
Definition of arc conditions
Values for process and simulation properties
Modelling alternative paths

	A	B	C	D
1	Process Name:	Travel expense reimbursement		
2	Description:	Business process for the travel expense reimbursement		
3				
4	Please, fill in below just the column VALUE to set the variables needed for the process emulation			
5	Property Name	Description	Accepted Value	Value
6	NbrProcessInstances	The number of times the process ...	Number	100
7	ArrivalRate	The rate in ms. in which the inst ...	Number	200
8	ExpenseAmount	Amount spent in the trip	Number	RANDBETWEEN(800, 1400)
9	Duration	Duration of the trip in days	Number	RANDBETWEEN(1,21)
10	EmployID	ID. of employee	Text	AJ938
11	ExpenseThreshold	The company's threshold to appr ...	Number	1.200,00
12	Fill travel expense form Exec. time	The execution time of the task ...	Number	400
13	Review completeness of the form Exec. time	The execution time of the task ...	Number	100
14	Reject request and nofiy employee Exec. time	The execution time of the task ...	Number	150
15	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
16	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
17	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -

b Configuration of properties associated to gateway conditions

Fig. 5. Modelling and configuring gateway nodes.

Simulation parameters are configured similar to process properties. The parameters we consider are three: (i) the number of process instances to be simulated, (ii) the arrival rate for process instances, and (iii) the task duration for each task $t \in T$ in the process model *BP*. These properties do not need to be explicitly defined by the developer or BP analyst. They are added automatically to the spreadsheet at its generation time. Fig. 5(b) shows that rows 6 and 7 are filled with the parameters *NbrProcessInstances* and *ArrivalRate*, respectively. From row 12 on, rows are filled with parameters that define the duration of the tasks. Values for these parameters are defined like values for process properties.

Basic nondeterministic human-behaviors, such as task completion time, can be modeled directly by the BP analyst on the spreadsheet. He can set, for example, a fixed or a normally distributed task completion time. Modern spreadsheet software offers a vast collection of built-in mathematical and statistical functions that enable BP analysts to model the dynamics of human-based tasks by approximating it via mathematical or statistical formulas. Predefined function can be employed also to define process properties, conditional statements and simulation parameters of almost any complexity. If more complex human interventions are required the software developer needs to implement additional pieces of software, e.g., a text recognition algorithm.

The results of the business process modeling and analysis design are a business process model *BP* and a configuration spreadsheet *CS* that are consumed by the BP simulator.

Our approach supports only the BPMN constructs introduced in Section 2.1. Tasks of type service, exclusive, parallel and inclusive gateways and, sequence flows are supported, while events of types different from *start* and *end* are not yet considered in this work.

5. Business process simulation

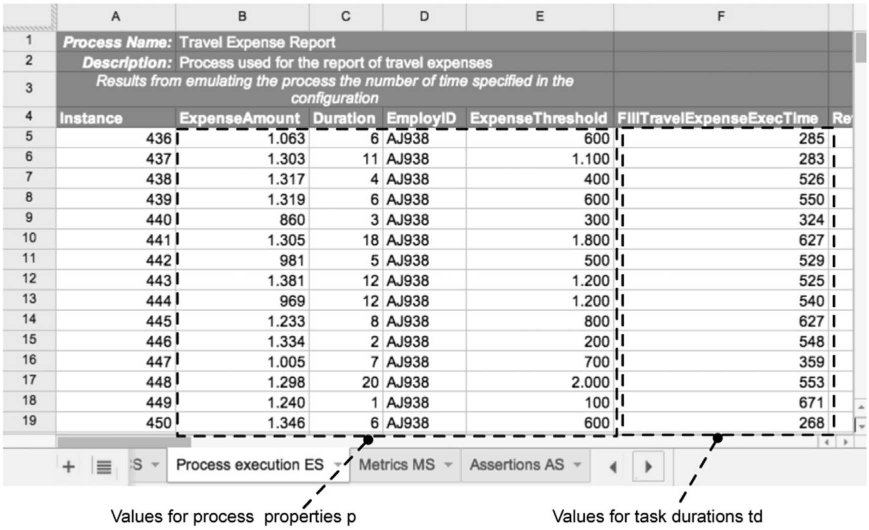
The BP simulator is in charge of simulating the execution of *BP* based on the configurations provided in *CS*, thereby producing a process execution log. Refining our definition of Section 2, the BP simulator can be seen as $BPsim(BP, CS) = ES$, where *BP* is the process model, *CS* is the configuration spreadsheet and *ES* is the resulting process execution spreadsheet (the log in spreadsheet format).

The *process execution spreadsheet ES* is a spreadsheet that holds process execution data that results from the simulation of *BP*. Each tuple in *ES* represents a business process instance *bpi*, and each cell within the tuple stores the runtime values of the elements of *bpi* as defined in Section 2.1. The idea of using this representation, as opposed to an event-based representation, is to keep the querying of business process instances simple and intuitive for the BP analyst and to avoid the need for writing complex event aggregation logics to reconstruct process instances. Thus, to store a business process instance bpi_j in *ES*, where the associated *BP* has a number of *k* properties and *l* tasks, we map the elements of bpi_j to *ES* as $ES(1,j) \leftarrow bpi_j.piid$, $ES(2,j) \leftarrow bpi_j.P[1].value$, …, $ES(k+1,j) \leftarrow bpi_j.P[k].value$, $ES(k+2,j) \leftarrow bpi_j.TD[1].dur$, …, $ES(k+l+2,j) \leftarrow bpi_j.TD[l].dur$. In other terms, we store the *piid* in the first column, followed by all the process properties of *BP* and by the durations of the tasks participating in the BP.

Fig. 6 shows an example of how *ES* looks like for our travel expense reimbursement process. Using letters to index columns, we have that row 8 holds the process instance bpi_{439}. The mapping is done as follows: *piid* is mapped to the first column $\mathtt{ES(A,8)} \leftarrow \mathtt{439}$, then, we have the mapping for the process properties *ExpenseAmount*, *Duration*, *EmployID* and *ExpenseThreshold* as $\mathtt{ES(B,8)} \leftarrow \mathtt{1.319}$, $\mathtt{ES(C,8)} \leftarrow \mathtt{6}$, $\mathtt{ES(D,8)} \leftarrow$ “$\mathtt{AJ938}$” and $\mathtt{ES(E,8)} \leftarrow \mathtt{600}$, respectively. Finally, we have the task durations, of which we show in Fig. 6 only the one corresponding to the task *Fill travel expense reimbursement form* as $\mathtt{ES(F,8)} \leftarrow \mathtt{550}$.

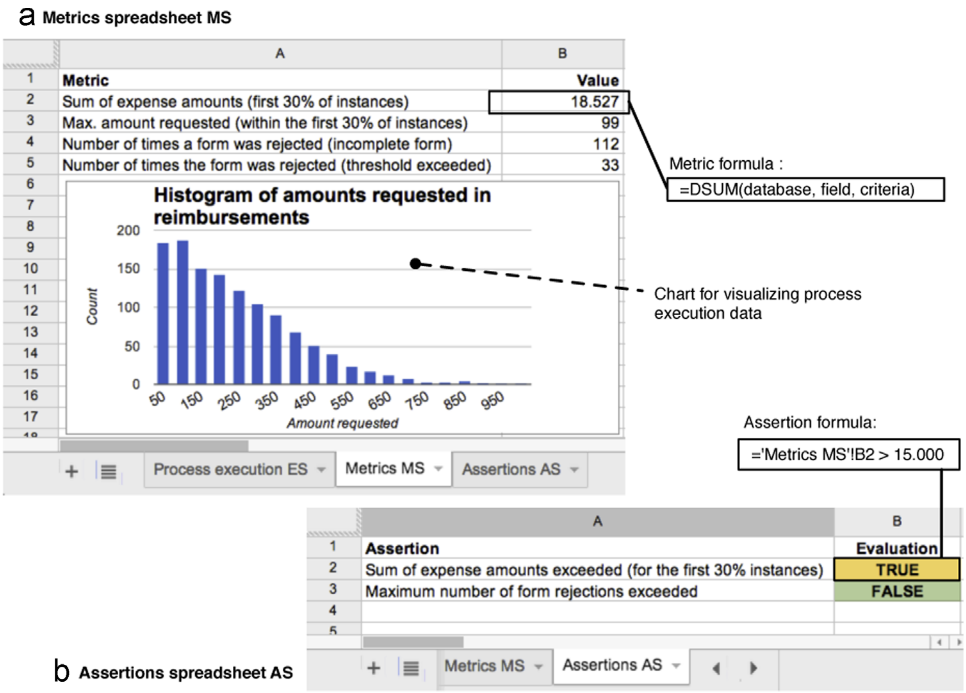
6. Analysis and visualization of results

Recall the definition of metrics as a function $m(L) = val$, where L is the process execution log and $val \in \mathbb{R}$ is the metric's value. Within a spreadsheet, m corresponds to a



	A	B	C	D	E	F	
1	**Process Name:**	Travel Expense Report					
2	**Description:**	Process used for the report of travel expenses					
3	*Results from emulating the process the number of time specified in the configuration*						
4	Instance	ExpenseAmount	Duration	EmployID	ExpenseThreshold	FillTravelExpenseExecTime	Re
5	436	1.063	6	AJ938	600	285	
6	437	1.303	11	AJ938	1.100	283	
7	438	1.317	4	AJ938	400	526	
8	439	1.319	6	AJ938	600	550	
9	440	860	3	AJ938	300	324	
10	441	1.305	18	AJ938	1.800	627	
11	442	981	5	AJ938	500	529	
12	443	1.381	12	AJ938	1.200	525	
13	444	969	12	AJ938	1.200	540	
14	445	1.233	8	AJ938	800	627	
15	446	1.334	2	AJ938	200	548	
16	447	1.005	7	AJ938	700	359	
17	448	1.298	20	AJ938	2.000	553	
18	449	1.240	1	AJ938	100	671	
19	450	1.346	6	AJ938	600	268	

Fig. 6. Process execution spreadsheet *ES* containing logged process progression information.



a Metrics spreadsheet MS

	A	B
1	**Metric**	**Value**
2	Sum of expense amounts (first 30% of instances)	18.527
3	Max. amount requested (within the first 30% of instances)	99
4	Number of times a form was rejected (incomplete form)	112
5	Number of times the form was rejected (threshold exceeded)	33

Metric formula :

=DSUM(database, field, criteria)

Chart for visualizing process execution data

Assertion formula:

='Metrics MS'!B2 > 15.000

b Assertions spreadsheet AS

	A	B
1	**Assertion**	**Evaluation**
2	Sum of expense amounts exceeded (for the first 30% instances)	**TRUE**
3	Maximum number of form rejections exceeded	**FALSE**
4		
5		

Fig. 7. Designing analysis reports: spreadsheets for defining (a) metrics *m* and (b) assertions *a*.

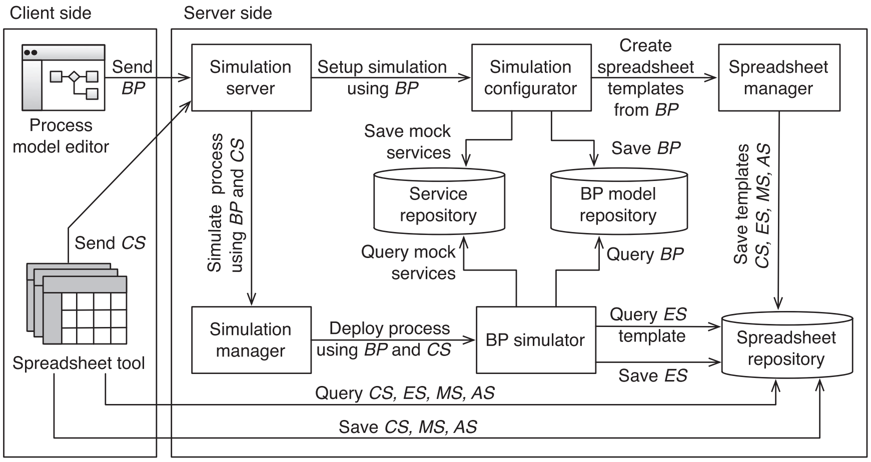


Fig. 8. Architecture of the proposed solution.

formula that can be specified using the standard spreadsheet functions provided by the adopted spreadsheet tool, L corresponds to the process execution log ES, and val corresponds to the output produced by the spreadsheet formula. Fig. 7(a) shows the *metrics spreadsheet MS* we use for computing metrics.

In this example, if we consider Google Spreadsheets[1] as our spreadsheet tool, the BP analyst computes the metric *Sum of expense amounts (first 30% of instances)* using a combination of the spreadsheet functions `SORT(range, sortColumn, isAscending, sortColumn2,isAscending2)` and `DSUM(database, field, criteria)`. The function `SORT(…)` sorts the process instances in ascending order based on the expense amount. The function `DSUM(…)` sums the amount requested for first 30% number of instances that appear in the sorted list. Due to the lack of space to fully explain the use of the aforementioned spreadsheet functions, we put in Fig. 7(a) only the reference to the formula used to compute the sum.

An assertion was defined as a function $a(L, Pred) = bool$, where L is the process execution log, $Pred$ is a set of predicates and $bool \in \{true, false\}$. In turn, a predicate is a function $pred(L, m) = bool$, where L, m and $bool$ are as defined before. In order to define assertions in a spreadsheet, we use the standard logical operators provided by spreadsheet tools. Fig. 7(b) shows an example of an *assertions spreadsheet AS* that can be used to compute assertions. For instance, the spreadsheet formula '`Metrics MS`'!`B2` > `15.000` checks whether or not the sum of expenses for the first 30% of the instances exceeds the maximum amount allocated for the *fast track* reimbursement. This assertion is composed of a single predicate that compares the metric *Sum of expense amounts (first 30% of instances)* (cell `MS(B, 2)`) with the maximum amount allowed (i.e., 15K euros). While this is a simple assertion, the BP analyst can construct fairly complex ones by combining logical operators (such as `AND`, `OR` and `NOT`) and predicates.

Also the definition of *analysis reports* relies on the built-in data visualization tools (charts and tables) provided by the spreadsheet. Recall that reports are of the form $r(L, M, A) = V$, with L being the process execution log and M and A being the sets of metrics and assertions, respectively. V are the visualization widgets (e.g., charts or tables) used to construct the report. Charts and tables conveniently summarize the results obtained from the computation of metrics and assertions. For example, Fig. 7(a) uses a simple bar chart that plots an histogram of the expense amounts requested. This enables the visual analysis of its distribution. The dataset for this chart was prepared using metrics computed with standard spreadsheet formulas (the details are skipped in this paper). The exact design of the report is up to the BP analyst, who knows best how to design the report so as to most effectively communicate his findings to the developer.

Thus, using metrics, assertions and charts, operationalized with the help of the built-in functions of the spreadsheet tool, the BP can analyze the outcomes of the simulations generated by different configuration settings that reproduce the various execution scenarios in study. Back to our reimbursement process, we can see in Fig. 7 that under the configuration settings of the simulation parameters, the BP analyst can learn that reimbursing all 30% of the of lowest request amounts is not possible (the total sum of amounts exceeds the budget), and that, for example, either the target percentage should be lowered, the budget for the *fast track* reimbursement option should be incremented, or the company should still tolerate a delay in a (fewer) number of reimbursement requests.

7. Implementation

Fig. 8 illustrates the functional architecture of the prototype of our solution. On the client side, we have the *process model editor* used to design BPs and the *spreadsheet tool* used to work with the spreadsheets *CS*, *ES*, *MS* and *AS*. On the server side, we have all the components that are in charge of configuring the environment for the simulation and analysis of BPs. When a process model *BP* is ready for

[1] http://docs.google.com/spreadsheet/

simulation, it is sent to the *simulation server*, which is in charge of managing the requests for BP simulations. This request is forwarded to the *simulation configurator*, which takes the model *BP* and performs three tasks: First, it creates the mock services that mimic the tasks in *BP* at simulation time and saves them in the *service repository*. Then, it stores *BP* into the *BP model repository* for future use. Finally, it requests the *spreadsheet manager* to create the templates for the spreadsheets *CS*, *ES*, *MS* and *AS*, which are stored into the *spreadsheet repository* for reuse in the following phases.

Back to the client side, the BP analyst can use the *spreadsheet tool* to configure *CS* and send it to the *simulation server* for the simulation of *BP*. The *simulation server*, in turn, forwards *BP* and *CS* to the *simulation manager*, which is in charge of managing the deployment of *BP* (using the configurations in *CS*) into the *BP simulator*. The latter queries *ES*, the mock services and *BP* from the *spreadsheet*, *service* and *BP model repositories*, respectively, simulate the BP, and store the obtained process execution data into *ES*. The resulting *ES* is stored back into the *spreadsheet repository* and made available, together with *MS* and *AS*, to the *spreadsheet tool* for analysis.

The current implementation of our analysis suite uses Signavio[2] as *process model editor*, Google Spreadsheets[3] as *spreadsheet tool*, and Activiti[4] as internal engine of the *BP simulator*. Signavio has been extended to enable the generation of the configuration spreadsheet. Google Spreadsheets has been extended to interface with the simulation back-end, acting as user interface of the BP analyst for simulation and BP analysis. Both extensions are implemented via JavaScript; the back-end components are implemented as standard web applications using Java. The screenshots in Figs. 4–7 show the look and feel of the prototype at work. At http://goo.gl/v4k2Yj we show a video of the tool in action. The source code of the tool can be downloaded from https://sites.google.com/site/ssbptester/.

8. User studies

We ran two user studies to validate the viability of the proposed approach. First, we assessed the suitability of spreadsheets with real BP analysts, then the whole approach with master students. We summarize both studies next; details of the study (scenarios, questionnaires, raw data) can be found at http://sites.google.com/site/ssbptester.

8.1. Business process analysis with spreadsheets

The objective of this study was to understand whether our approach achieves the first goal of our problem statement, i.e., enabling BP analysts to analyze business processes. This study specifically focused on the configuration of the BP simulation and the analysis of process execution data. The participants to the study were three employees of the Paraguayan subsidiary of DHL, who operate as BP analysts at an everyday basis. Each BP analyst participated separately in a one hour session, conducted within the premises of the company. All participants were familiar with spreadsheets; only one of them knew Google Spreadsheets. None of them had a background in computer science. The user study was structured as a *usability test* followed by a *retrospective probing* in the form of a semi-structured interview [15].

For the usability test, participants were introduced to our tool and watched a video exemplifying the features of the tool. Then, they were presented with a analysis scenario based on a simplified version of the BP model shown in Fig. 1(a), provided with a suitable configuration spreadsheet *CS*, and asked to analyze the BP according to the scenario. For the retrospective probing, we asked participants questions about their experience, thoughts and actions after the usability test.

All participants agreed that the BP analysis tasks were easy to understand, but in some cases difficult to perform. The main reason for this was that, while participants did not have problems in writing simple spreadsheet formulas like sums, averages and standard deviations, they faced difficulties in defining complex formulas that involved conditional and statistical distribution functions. This problem was exacerbated by the fact that the nomenclature of the functions in Google Spreadsheets differs from the one found in Microsoft Excel, which they were more acquainted with. The language of the tool also contributed to the issue: the spreadsheets for the analysis of the BP were all in English, and so were the names of the functions. Although all participants declared good knowledge of English, they were used to work with spreadsheets in Spanish, which made it sometimes difficult to find even functions they knew. These problems were however easily overcome with a small help from the person running the study.

As for the general feeling and mood after the study, two of the participants said they felt comfortable, while the third one said the experiment was long and stressful. All three agreed that they had to learn new concepts and terminology they were not familiar with regarding both BP simulation (e.g., arrival rates) and spreadsheet functions (e.g., conditional and statistical distribution functions). All participants agreed that using spreadsheets for analysing BPs is useful and close to their working experience for two main reasons: (i) they are familiar with spreadsheets, and (ii) spreadsheets are suitable to analyze data in a tabular format, helped by the pre-built spreadsheet formulas, filters and charts. When participants were asked if they would use the tool, all of them responded positively and stated that the approach would indeed be effective in helping them to autonomously analyze BPs, provided they have a good working knowledge of the spreadsheet's predefined functions.

Although this study was conducted with only three BP analysts and, hence, does not have statistical relevance, we nevertheless consider the study a good indicator for the suitability of using our spreadsheet-based approach for

[2] http://code.google.com/p/signavio-core-components.

[3] https://docs.google.com/spreadsheet.

[4] http://www.activiti.org.

analyzing BPs. Participants intuitively understood their tasks and unanimously agreed on the viability of the approach.

8.2. Modeling, analyzing and reporting

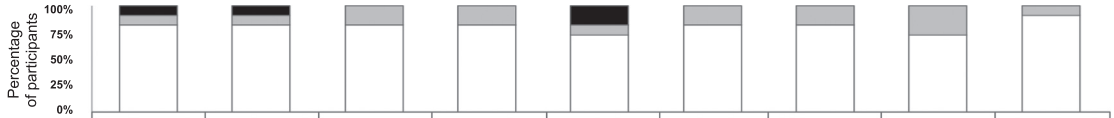
The second study aimed to provide end-to-end coverage of our approach and statistical tests. The study therefore also aimed to understand whether the approach facilitates the discussion of findings between the BP analyst and the software developer. To do so, we involved a total of 22 M.Sc. students taking part of a BPM course at the University of Trento, Italy, all of them with a background in computer science, BP modeling (BPMN) and spreadsheets. The study lasted around 1 h and 15 min and it was carried out in the laboratory of the university.

This study was also structured as a *usability test* with *retrospective probing*. We again offered a training session in the form of a live demo to introduce the tool to the students. The retrospective probing took the form of an online survey. Students were paired up, one playing the role of the BP analyst and one the role of the software developer. Each role was assigned a number of tasks related to the role, specified in two scenarios based on a simplified version of the BP model in Fig. 1(a). The first scenario consisted in performing one analysis cycle as presented in Fig. 3. The second scenario asked the BP analyst to communicate a change in the BP model to the software developer and to analyze the modified BP again.

To collect feedback, we prepared two surveys with 26 and 19 questions for the BP analyst and software developer, respectively. The questions were answered using a Likert scale of 1 (strongly agree) to 5 (strongly disagree) [15]. We consider answers 1 and 2 as *positive answers*, 3 as *neutral answer*, and 4 and 5 as *negative answers*. The survey included also open questions that allowed participants to provide free feedback. The results of the survey are reported in the form of descriptive statistics using the mean and median of the sample. We also use a two-sided hypothesis test to test the significance of the answers for each single question. The test makes use of a *Wilconox, signed rank test* with a significance level of $p=0.05$ and a null hypothesis $H_0: \eta = 3$ (where η represents the median) and an alternative hypothesis $H_A: \eta \neq 3$. In other words, the null hypothesis is that participants of the study have a neutral answer for each question, against the alternative hypothesis that they are lean towards positive or negative answers.

Software developers: Fig. 9 shows the feedback from software developers regarding their overall experience. Most questions were answered positively (which is confirmed by our tests: $p-values$ are lower than 0.05), with the exceptions of the questions regarding the time taken by the experiment and the overall satisfaction with the experience, in which we obtained rather neutral answers (for these two questions, the $p-values$ were slightly higher than 0.05). Fig. 10 shows the feedback regarding the BP model editor. Most questions were again answered positively, confirmed by our hypothesis tests. The exceptions are the questions related to the easiness for setting up process properties, the similarity of the BP editor's UI w.r.t. the original version of Signavio and the preference of our tool over the others, for which we obtained rather neutral answers (we cannot reject H_0). One participant pointed out positively that the joint use of the BP model editor with Google Spreadsheets “permits teams to work together in real-time,” while another participant recommended to improve the debugging functionalities of the editor “in order to be able to find errors in the BP model before running the BP simulation.”

BP analysts: Fig. 11 summarizes the overall experience by BP analysts. All questions were answered positively and confirmed by our statistical tests. Regarding the use of spreadsheets, Fig. 12 reports that most questions were also answered positively. The three exceptions we have are the questions regarding the comfortability with the use of spreadsheets, the similarity of the UI of the spreadsheet w.r.t. to the original version of the tool, and the preference of the proposed tool over the others. For these questions, we have a split preference on the answers and the average results yield a neutral answer (with $p-values$ equal to 0.097; 0.135 and 0.172, respectively, for each question). The reason for this neutrality, despite the positive answers in the previous questions, may be motivated by both the paradigm shift in the approach used to analyze BPs and the fact that our tool is a prototype implementation, and thus, not yet meant to be ready for use in a production environment.



	"I felt comfortable"	"It took a reasonable amount of time"	"It was according to my skills"	"I was able to easily follow all the tasks"	"I am overall satisfied with the experience"	"The tasks were easy to understand"	"I felt confident in doing the given tasks"	"I felt skillful enough to perform the tasks"	"I was successful in setup the properties"
■ Negative	1	1	0	0	2	0	0	0	0
▨ Neutral	1	1	2	2	1	2	2	3	1
□ Positive	9	9	9	9	8	9	9	8	10
Mean	1,909	2,090	1,545	1,636	2,272	1,818	1,727	1,727	1,636
Median	2	2	1	1	2	2	2	1	2
W-value	3,5	8,5	0	0	9	0	0	0	0
p-value	0,014	0,052	0,006	0,007	0,052	0,007	0,007	0,010	0,005

Fig. 9. Survey results for questions regarding the overall experience of software developers.

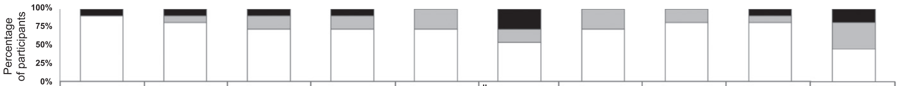
8.3. Discussion of results

The results of both user studies provide good evidence of the potential of our approach. Our observations testify that the interaction between the BP analysts and the developers were well-disposed and facilitated by our approach, confirming the suitability of the approach for collaborative BP analysis. The company's BP analysts were more inclined towards the use of mainstream spreadsheet tools, such as Microsoft Excel, given their familiarity with such tools. BPM students, instead, appreciated more the use of Google Spreadsheets, due to its suitability for real-time collaboration. Given their higher familiarity with conditional expressions and complex statistical functions compared to the BP analysts, the BPM students felt much more comfortable in using spreadsheets to analyze the process execution log. This suggests that, in order to bring our tool to a real setting, it is necessary to make sure BP analysts have the necessary training in using spreadsheets. However, it is important to note that all participants in both user studies easily understand the mapping of the BP analysis problem to the problem of analyzing data in spreadsheets. Once participants figured out the right spreadsheet functions to use, they were able to easily define metrics and assertions over the process execution log organized into process instances. In conclusion, we accept our hypothesis that mapping the BP analysis problem to the design of a spreadsheet calculation enables the BP analyst to analyze BPs autonomously.

9. Qualitative analysis

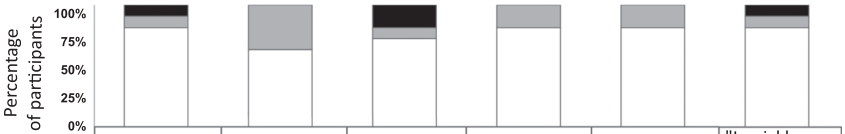
We complement the above user studies with a qualitative analysis of our tool (Spreadsheet-based BP Analyzer). The analysis consists in a comparison of our tool against state of the art solutions used for BP analysis and simulation.

The analysis includes today's most representative commercial tools in the realm of BP analysis as well as academic and open source solutions. In particular, we considered Websphere Business Modeler (v. 6.2) [16], the licensed solution of IBM for simulation and analysis of BPs, TIBCO Business Studio (v. 3.9) [17], the proposal of TIBCO Software Inc. for modeling, analysis, and simulation BPs. Free alternative solutions are also included in our analysis. In concrete, we consider BizAgi Modeler [18], the free solution offered by BizAgi to document and analyze BPs, Bonita Open Solution (v. 5.6) [19], the open-source proposal of BonitaSoft for modeling and simulating BPs represented in BPMN, and Adonis Community Edition [20], the free modeling and simulation BPM tool offered by BOC Group. Two academic tools are also present in the comparative analysis. The first one, Signavio Process Editor (v. 9.0) [21], started as an academic project and recently turned into a commercial solution. The proposal targets business practitioners in the context of modeling and simulation of BPs. The second one, SimQuick [22], is an entirely academic BP simulation tool that uses spreadsheets and Microsoft Excel macros to enable the design and simulation of BPs.



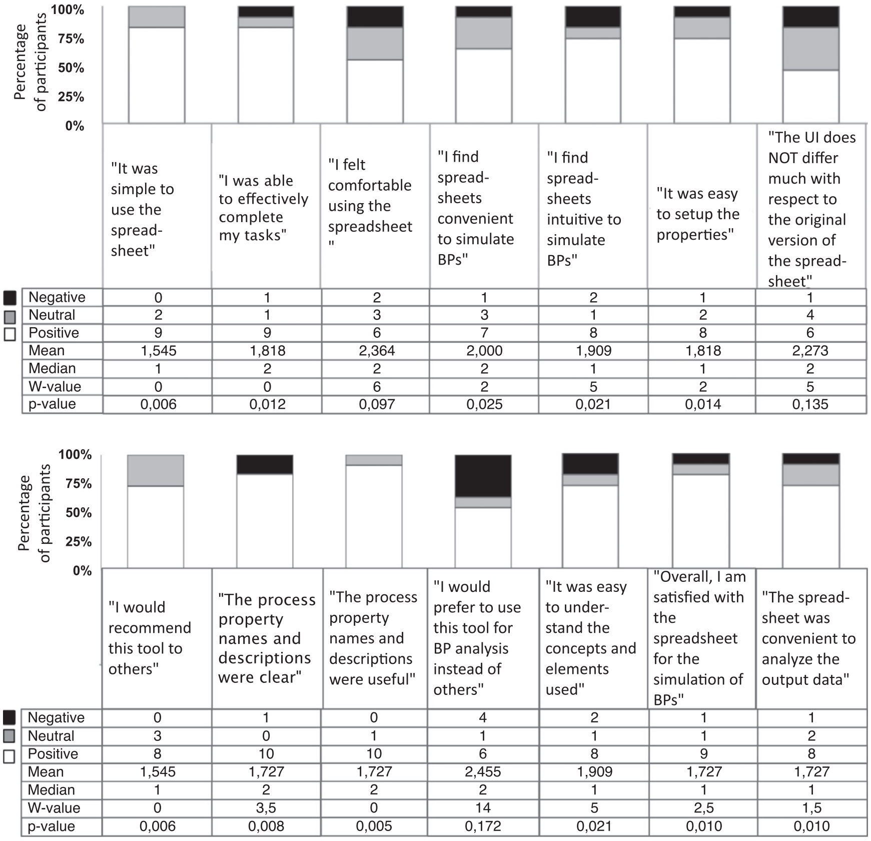
	"It was simple to use the software for designing processes"	"I was able to effectively complete my tasks using the software"	"I felt comfortable using the software"	"It was easy to setup process properties for the BP simulation"	"It was easy to generate the spreadsheet from the BP model"	"The UI does NOT differ much with respect to the original version of Signavio"	"It was easy to understand the concepts concepts introduced in Signavio"	"Overall, I am satisfied with the software"	"I would recommend this tool to others"	"I would prefer to use this tool for BP analysis instead of others"
■ Negative	1	1	1	1	0	3	0	0	1	2
▨ Neutral	0	1	2	2	3	2	3	2	1	4
□ Positive	10	9	8	8	8	6	8	9	9	5
Mean	1,818	1,545	1,909	2,000	1,818	2,636	1,909	2,000	2,000	2,636
Median	2	1	1	2	2	2	2	2	2	3
W-value	8,5	1,5	6	6,5	0	15	0	0	8	8,5
p-value	0,028	0,006	0,046	0,057	0,012	0,390	0,012	0,006	0,047	0,386

Fig. 10. Survey results for questions regarding the BP model editor.



	"I felt comfortable"	"The analysis took a reasonable amount of time"	"The analysis was according to my skills"	"I was able to easily follow all the tasks"	"I am overall satisfied with the experience"	"I quickly acquired the necessary knowledge to perform the tasks"
■ Negative	1	0	2	0	0	1
▨ Neutral	1	4	1	2	2	1
□ Positive	9	7	8	9	9	9
Mean	1,818	2,091	1,909	1,636	1,727	1,727
Median	2	2	1	1	2	1
W-value	3	0	5	0	0	2,5
p-value	0,012	0,019	0,021	0,007	0,007	0,010

Fig. 11. Survey results regarding the BP analysts' overall experience.



	"It was simple to use the spreadsheet"	"I was able to effectively complete my tasks"	"I felt comfortable using the spreadsheet"	"I find spreadsheets convenient to simulate BPs"	"I find spreadsheets intuitive to simulate BPs"	"It was easy to setup the properties"	"The UI does NOT differ much with respect to the original version of the spreadsheet"
■ Negative	0	1	2	1	2	1	1
▨ Neutral	2	1	3	3	1	2	4
□ Positive	9	9	6	7	8	8	6
Mean	1,545	1,818	2,364	2,000	1,909	1,818	2,273
Median	1	2	2	2	1	1	2
W-value	0	0	6	2	5	2	5
p-value	0,006	0,012	0,097	0,025	0,021	0,014	0,135

	"I would recommend this tool to others"	"The process property names and descriptions were clear"	"The process property names and descriptions were useful"	"I would prefer to use this tool for BP analysis instead of others"	"It was easy to understand the concepts and elements used"	"Overall, I am satisfied with the spreadsheet for the simulation of BPs"	"The spreadsheet was convenient to analyze the output data"
■ Negative	0	1	0	4	2	1	1
▨ Neutral	3	0	1	1	1	1	2
□ Positive	8	10	10	6	8	9	8
Mean	1,545	1,727	1,727	2,455	1,909	1,727	1,727
Median	1	2	2	2	1	1	1
W-value	0	3,5	0	14	5	2,5	1,5
p-value	0,006	0,008	0,005	0,172	0,021	0,010	0,010

Fig. 12. Survey results regarding the use of spreadsheets for BP analysis.

9.1. Comparison framework

The comparison framework used to conduct the analysis is based on five categories of functionalities, namely, *BP modeling*, *simulation configuration*, *simulation*, *analysis*, and *reporting*. These five categories represent the typical phases of the BP design and analysis process proposed by the analyzed tools. The framework in particular aims to highlight those concerns that are related to the BP analyst inside this BP design and analysis process.

Although BP modeling is not the focus of our work, it was included in the analysis given its crucial role in the proposed process lifecycle. The *BP modeling* capabilities of the tools are analyzed under three dimensions: *notation*, i.e., which BP modeling notation is supported by the tool, *model verification*, i.e., what types of modeling errors can be detected, and *debugging*, i.e., what types of features are available to find and fix the modeling errors.

The analysis of functionalities offered by the tools for the *BP simulation configuration* is mainly focused on the statistical facilities offered by the tools to configure *process tasks*, *resources*, and *domain-specific parameters* (e.g., expenses, interest rate). We also analyze the variety of domain-independent variables (e.g., number of simulation instances, instances arrival rate, loading period) that are possible to setup with the tools.

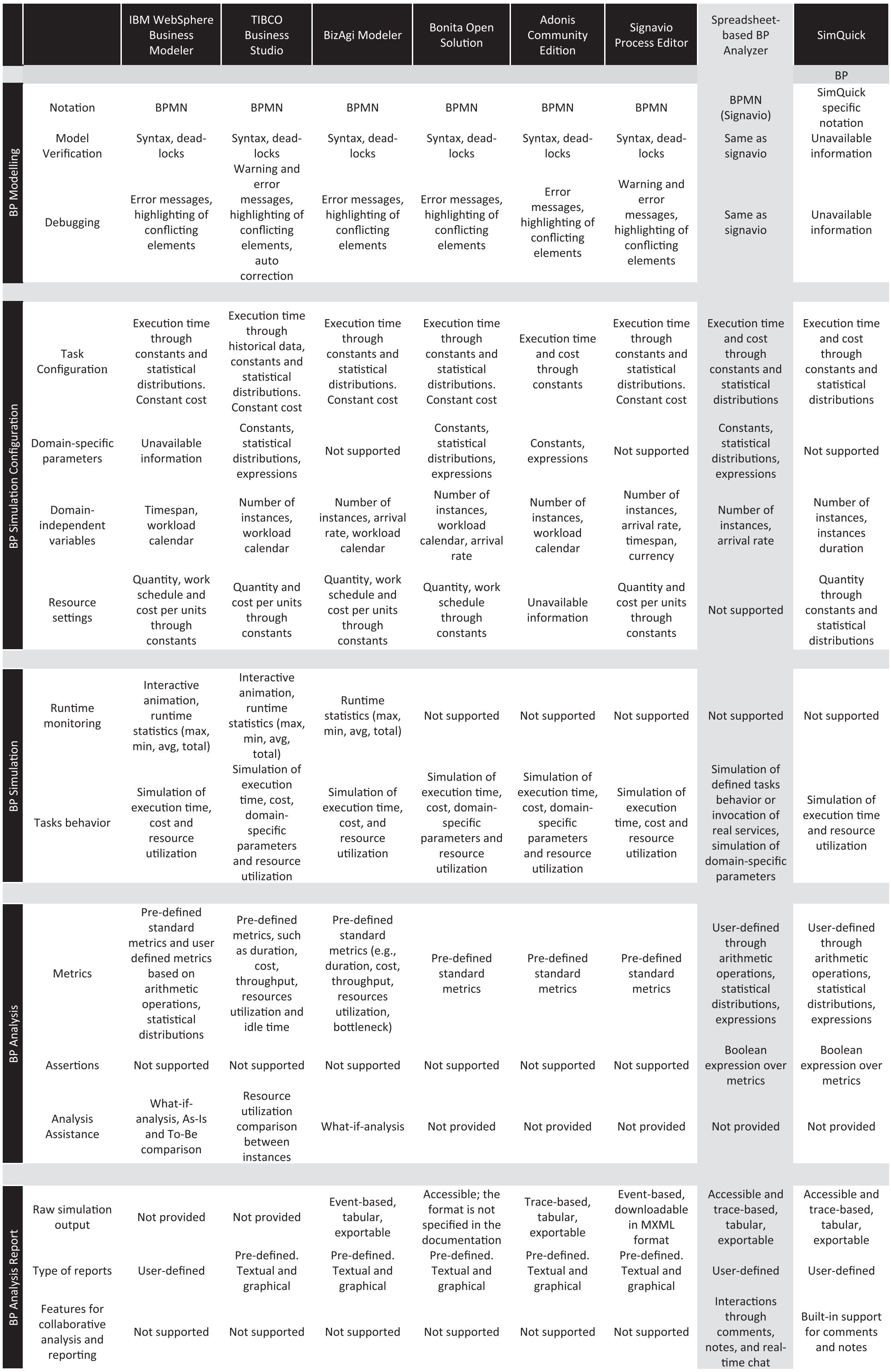
We capture the *simulation* capabilities of the tools through two dimensions, namely, *runtime monitoring* and *task behavior*. The first one is related to the features offered by the tools to monitor the execution of the simulation, e.g., animation and runtime statistics, while the second one is associated to the simulation capacities of the tools w.r.t. the simulation of task behavior.

Regarding the *analysis* capabilities of the tools, we focus on understanding the flexibility of the tools in giving BP analysts the freedom to write their own *analysis instruments* (metrics and assertions) and the power of the tools in *assisting analysts* in obtaining information about the behavior of the process.

The *reporting* dimension includes the flexibility of the tools to create *custom analysis reports*, the features offered to foster *collaborative analysis and reporting*, and whether the *simulation output* is accessible and in which format.

Because not all the tools could be installed for their study, we exclusively base our analysis on the official documentation provided for each solution. We therefore highlight the situations in which the documentation provides insufficient information to draw a conclusion about a particular dimension.

Fig. 13 presents the analysis in a table where columns represent the description of each tool. The gray-shaded column contains the descriptions of our Spreadsheet-based BP Analyzer.



		IBM WebSphere Business Modeler	TIBCO Business Studio	BizAgi Modeler	Bonita Open Solution	Adonis Community Edition	Signavio Process Editor	Spreadsheet-based BP Analyzer	SimQuick
BP Modelling	Notation	BPMN	BPMN	BPMN	BPMN	BPMN	BPMN	BPMN (Signavio)	BP SimQuick specific notation
	Model Verification	Syntax, deadlocks	Syntax, deadlocks	Syntax, deadlocks	Syntax, deadlocks	Syntax, deadlocks	Syntax, deadlocks	Same as signavio	Unavailable information
	Debugging	Error messages, highlighting of conflicting elements	Warning and error messages, highlighting of conflicting elements, auto correction	Error messages, highlighting of conflicting elements	Error messages, highlighting of conflicting elements	Error messages, highlighting of conflicting elements	Warning and error messages, highlighting of conflicting elements	Same as signavio	Unavailable information
BP Simulation Configuration	Task Configuration	Execution time through constants and statistical distributions. Constant cost	Execution time through historical data, constants and statistical distributions. Constant cost	Execution time through constants and statistical distributions. Constant cost	Execution time through constants and statistical distributions. Constant cost	Execution time and cost through constants	Execution time through constants and statistical distributions. Constant cost	Execution time and cost through constants and statistical distributions	Execution time and cost through constants and statistical distributions
	Domain-specific parameters	Unavailable information	Constants, statistical distributions, expressions	Not supported	Constants, statistical distributions, expressions	Constants, expressions	Not supported	Constants, statistical distributions, expressions	Not supported
	Domain-independent variables	Timespan, workload calendar	Number of instances, workload calendar	Number of instances, arrival rate, workload calendar	Number of instances, workload calendar, arrival rate	Number of instances, workload calendar	Number of instances, arrival rate, timespan, currency	Number of instances, arrival rate	Number of instances, instances duration
	Resource settings	Quantity, work schedule and cost per units through constants	Quantity and cost per units through constants	Quantity, work schedule and cost per units through constants	Quantity, work schedule through constants	Unavailable information	Quantity and cost per units through constants	Not supported	Quantity through constants and statistical distributions
BP Simulation	Runtime monitoring	Interactive animation, runtime statistics (max, min, avg, total)	Interactive animation, runtime statistics (max, min, avg, total)	Runtime statistics (max, min, avg, total)	Not supported	Not supported	Not supported	Not supported	Not supported
	Tasks behavior	Simulation of execution time, cost and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost, and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost and resource utilization	Simulation of defined tasks behavior or invocation of real services, simulation of domain-specific parameters	Simulation of execution time and resource utilization
BP Analysis	Metrics	Pre-defined standard metrics and user defined metrics based on arithmetic operations, statistical distributions	Pre-defined metrics, such as duration, cost, throughput, resources utilization and idle time	Pre-defined standard metrics (e.g., duration, cost, throughput, resources utilization, bottleneck)	Pre-defined standard metrics	Pre-defined standard metrics	Pre-defined standard metrics	User-defined through arithmetic operations, statistical distributions, expressions	User-defined through arithmetic operations, statistical distributions, expressions
	Assertions	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Boolean expression over metrics	Boolean expression over metrics
	Analysis Assistance	What-if-analysis, As-Is and To-Be comparison	Resource utilization comparison between instances	What-if-analysis	Not provided	Not provided	Not provided	Not provided	Not provided
BP Analysis Report	Raw simulation output	Not provided	Not provided	Event-based, tabular, exportable	Accessible; the format is not specified in the documentation	Trace-based, tabular, exportable	Event-based, downloadable in MXML format	Accessible and trace-based, tabular, exportable	Accessible and trace-based, tabular, exportable
	Type of reports	User-defined	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	User-defined	User-defined
	Features for collaborative analysis and reporting	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Interactions through comments, notes, and real-time chat	Built-in support for comments and notes

Fig. 13. Qualitative analysis of our tool (Spreadsheet-based BP Analyzer) and state-of-the-art business process analysis and simulation solutions.

9.2. Analysis

BP modeling: All the tools offer similar modeling functionalities. Almost all of them have BPMN editors equipped with features that perform automatic model checking (syntax validation and deadlocks verification). They also provide model debugging functions through warning and error messages and by coloring the conflicting elements. In addition, TIBCO provides features to semi-automatically fix simple syntax errors. SimQuick is an exception within this category because it has its own modeling notation implemented through Microsoft Excel macros. In our case, the full modeling functionalities are based on Signavio Core Components, the free and open-source version of Signavio Process Editor and thus our modeling features are equivalent to that of Signavio.

BP simulation configuration: No big differences are appreciated when comparing the features for configuring the simulation. All the tools enable the use of statistical distributions to configure the duration of the tasks. Spreadsheet-based BP Analyzer and SimQuick allow for the use of distributions to set up the costs associated to the execution of the process tasks while the rest of the tools allow only for the use of constant values to define task execution costs. TIBCO provides features to set up tasks duration using historical data.

Half of the tools, namely TIBCO, Bonita, Adonis, and Spreadsheet-based BP Analyzer, offer the possibility to model, through constants, expressions, and even by statistical distributions, the value of domain-specific parameters, such as interest rate, expenses, return of investment. All the tools enable the definition of domain-independent variables. In SimQuick and Spreadsheet-based BP Analyzer a couple of variables can be configured, i.e., number of simulation instances and arrival rate, while in the rest of the tools a large variety of parameters can be defined, such as timespan in which the instances are created and workload calendar.

Almost all the tools, excluding ours, enable the definition of quantity, work schedule and cost per units of the resources associated to the process execution. Being a tool that is constructed on top of a spreadsheet editor (Microsoft Excel), SimQuick provides the chance to employ statistical distributions to model resources while in the other tools only constant values can be used.

BP simulation: Only TIBCO, WebSphere and BizAgi support runtime-monitoring functions. The first two provide interactive animation features that allow users to follow the execution of the simulation step-by-step. In addition, all of them display descriptive statistics at runtime, e.g., average tasks duration, most expensive simulation instance, number of current idle resources and average waiting time. The main advantage of Spreadsheet-based BP Analyzer in relation to the others is the possibility to go from 0% to 100% on the simulation of the behavior of the process tasks. In other words, Spreadsheet-based BP Analyzer provides the flexibility to choose whether a task in the BP should be simulated or carried out by a real web service. This feature, which is not available in any other state-of-art tools that offer the chance of simulating the execution time, cost, domain-specific process parameters and resource utilization of the tasks, provides the possibility to verify whether the real services operate as expected.

BP analysis: Spreadsheet-based BP Analyzer and SimQuick are the only ones that fully empower BP analysts to define their own metrics and assertions. In order to provide such flexibility, they leverage on the power of the spreadsheet's built-in functions. Similarly, WebSphere provides user-defined metrics but from the documentation is not clear whether this functionality is available to measuring simulation data or real data.

The rest of the tools offer standard, predefined metrics, e.g., duration, cost, throughput and resource utilization, through which the analyst can get information about process behavior. Also, only SimQuick and Spreadsheet-based BP Analyzer are the only ones that give BP analysts the chance of writing assertions on top of the metrics computed over the process execution data.

A useful and important feature is the possibility to assist analysts in the evaluation of their business processes. Only TIBCO, IBM WebSphere and BizAgi offer additional analysis functions. WebSphere and BizAgi provide features to conduct what-if-analysis while TIBCO enables the comparison of resource utilization between simulation instances. In addition, IBM WebSphere provides features that ease the comparison between the ideal BP model (to-be) and the current version of the model (as-is).

BP analysis report: Most of the tools provide predefined reports. However, SimQuick and Spreadsheet-based BP Analyzer offer the users the possibility to create their own reports. By exploiting the graphical and analytical features of spreadsheets editor programs (Excel, Google Spreadsheet) SimQuick and Spreadsheet-based BP Analyzer allow BP analysts to draw custom reports of any complexity. Following a more restricted approach, WebSphere also offers user-defined report features. By employing a drag-and-drop designer, BP analysts are able to build custom reports which are based on a limited set of predefined elements, such as text-fields, tables, and summary statistics (counts, sums, and averages).

The spreadsheet editors used by SimQuick and Spreadsheet-based BP Analyzer offer, in addition, built-in functionalities that enable the possibility to add comments and notes on the reporting documents, which can facilitate the communication between BP analysts and developers. In the case of Spreadsheet-based BP Analyzer this communication may even happen in real-time thanks to the chat functionalities of Google Spreadsheet.

Neither IBM Websphere nor TIBCO provides the raw output of the simulations, which limits the possibility of running further BP analyses. The rest of the tools do allow users to access the simulation output promoting the use of alternative tools to further analyze BPs. In this sense, BizAgi provides the output in a tabular, exportable and event-based format; Signavio in an event-based, downloadable MXML format [10], while Adonis, Spreadsheet-based BP Analyzer, and SimQuick offer the raw simulation output in a tabular, trace-based and exportable scheme. Bonita's documentation states that simulation outputs can be downloaded but their content and format are not specified.

9.3. Discussion

The above analysis helps us to understand better the distinctive and innovative aspects of the Spreadsheet-based BP Analyzer. The combination of BP simulation and a spreadsheet-based UI (Google Spreadsheet) accompanied with a standard BPMN process model representation equips also BP analysts with limited technical skills with a single tool that enables them to simulate, verify and analyze the performance of BPs through personalized instruments (metrics, assertions and custom reports). The use of a full-fledged BP engine to run the simulation enables the flexible invocation of both simulation web services that mimic the behavior of other web services as well as real, production web services. This turns the Spreadsheet-based BP Analyzer into an instrument that can be used both by the BP analyst for his BP verification and simulation and by the developer for the testing of how real web services perform inside a simulated process. As an add-on, the live collaboration support by Google Spreadsheet offers a new and dynamic communication alternative that can ease the interaction between BP analysts and developers.

Two limitations of the Spreadsheet-based BP Analyzer w.r.t. to some of the tools presented in this comparative analysis are the lack of support for the simulation of human resource utilizations (given our focus on service-based processes) and the lack of additional analysis aids such as what-if-analyses. We plan to extend the Spreadsheet-based BP Analyzer with these functionalities in future work.

10. Related work

The analysis of business process has triggered many research efforts, yielding a variety of different approaches. In the following, we discuss those related works that fall into the context of this paper, which includes service-based BP testing, BP simulation, compliance checking, process mining and modeling and testing spreadsheet.

The problem of *service-based BP testing* took significant relevance with the SOA and its use to support the operation of BPs. In particular, many approaches have been proposed to address this problem in the context of BPs represented with BPEL [23]. For example, some of the works in this context are dedicated to perform *unit tests* of web service compositions. Unit tests in this context means testing each web service and their corresponding interfaces, i.e., each operation offered and invoked by the service [24,25]. A side problem associated to the testing of BPEL processes is the generation of test cases. Works like Garćа-Fanjul et al., [26], Yuan et al. [27], and Yan et al. [28] propose approaches for the generation of test cases using techniques from model checking, graph search and concurrent path analysis. Other types of tests performed on service-based BPs include regression testing [29] and integration testing [30]. All these approaches require special software testing and development skills.

BP simulation has been employed for the purpose of testing BPs. For example, Aguilar et al. [31] propose a BP simulation methodology to analyze the performance of financial BPs in unforeseen, potential situations. In the context of service-based BPs, Narayanan and McIlraith [32] propose the use of simulation to test the preservation of properties (e.g., safety conditions) associated to the services responsible for the BP execution by combining Petri-Nets and DAML-S. Chandrasekaran et al. [33] use simulation to monitor and analyze the performance of individual web services involved in a BP. Tan and Takakuwa [34] and Wynn et al. [35] use simulation to evaluate the impact of BP re-engineering tasks on process performance.

In the context of *process mining* [10], techniques such as conformance checking and process discovery have been employed to check whether or not a BP behaves as expected. *Conformance checking* [36,37] verifies whether the traces of execution of a BP conform with a given BP model. In order to do so, the approach proposes to *replay* the real process execution data on the BP model to detect if there are mismatches between the two. Conformance checking therefore checks if the event log structurally matches the process model, or, in other words, it checks if the control flow that underlies the event log matches that of the process model. It therefore only considers the structure of the BP model as the specification of the expected behavior and does not focus on process-specific metrics. Moreover, while the main use case of conformance checking consists in using a real event log to check against a predefined process model *a posteriori* (i.e., after a real execution of the process), in our case we use a simulated log to check *a priori* (i.e., before the real execution of the process) if a BP behaves as expected.

Process discovery is the task of inferring a BP model from process execution data [38,39]. Testing a BP with process discovery can be done by first inferring the BP model from the process execution data and then comparing if the inferred model corresponds to the expected model. This comparison can be done either manually or using automatic techniques such as those based on BP similarity [40]. There are a set of commercial (e.g., ARIS PPM, HP BPI, and ILOG JViews) and academic (e.g., EMiT, Little Thumb, InWoLvE, Process Miner, and MinSoN) process mining tools. The main goal of the academic tools is to extract knowledge from event logs for discovering processes. The commercial tools are more oriented to the design, analysis and optimization of BPs using for example, charts and dashboards. In addition, HP BPI can discover a BP from event logs. Our approach differs from these process mining techniques in that they are meant to be used *after* the real process has been executed. This contradicts our purpose of using BP testing as an instrument to prevent unwanted behaviors. Moreover, these two approaches focus only on the structure of the BP, while our approach tests the dynamics and data produced by the execution of the BP through the use of user-defined metrics and assertions.

Compliance checking is the problem of verifying whether a BP model or its execution adheres to a set of compliance rules (i.e., the *expected behavior*) that typically emerge from laws, regulations and standards. This problem has been addressed both statically and dynamically. In *static compliance checking*, only the model of the BP is checked against the compliance rules. For example, Liu et al. [41] address the problem by expressing the BP model in Pi-Calculus and the corresponding compliance rule in Linear Temporal Logic. Using this representation, model checking techniques are used to check whether the BP model complies with the compliance rules. Governatori et al. [42] propose a logic-based formalism to represent both the semantics of contracts and compliance checking procedures. The formalism used is Format Contract

Language, which is based on Deontic Logic and helps in representing and checking contrary-to-duty obligations in contracts. In *dynamic compliance checking*, BP execution evidences are used to check for compliance. Works by Rodríguez et al. [43] and Silveira et al. [44] propose the use of the so-called Key Compliance Indicators (KCIs) to measure the compliance level of service-based BPs from process execution data, e.g., to measure the fulfilment of Service-Level Agreements (SLAs). In a similar approach, Casati et al. [45] and Sayal et al. [46] propose to warehouse process execution data to enable the monitoring, analysis and reporting on the performance of BPs, e.g., to check the duration of process execution instances when they are constrained in time. The approach we present in this paper is similar to dynamic compliance checking, with the difference that we enable the use of simulated data, next to real data, to check different execution scenarios. Our approach can thus be used for simulation-based compliance checking if process properties provide access to the data necessary to express compliance concerns (assertions).

A topic that is also related to our work is that of *spreadsheet modeling and testing*. Here, the focus is put on modeling and testing the spreadsheet content itself. In particular, spreadsheet testing and debugging is important because it positively influences spreadsheet accuracy [48]. Burnet et al., who coined the term “end-user software engineering”, proposed an approach to support assertions in spreadsheets [49]. The assertions are built on top of cells to check the execution of formulas contained in cells. The approach provides the possibility to create assertions by the end-user, following an abstract syntax that is implemented through both graphical and textual concrete syntaxes. Hermans proposes Expector, an Excel-based tool for helping users in improving their testing practices, e.g., by helping them in achieving better testing coverage, more meaningful names for outcomes of the testing, among other features [50]. In the same paper, the author presents an interesting study on the use of testing within spreadsheets. They found out that 8.8% of the spreadsheets from the EUSES corpus [51] contain testing formulas that use only the spreadsheet's built-in functions. Rothermel et al. present a methodology for the test adequacy criterion in form-based visual programs (the authors place spreadsheets under this category) [52]. In their methodology, the authors propose to check for the definition-use adequacy of a test suite based on the *all-uses* data flow adequacy criterion. The prototype, implemented in the research language Forms/3 [53], provides visual feedback to the users about the “testedness” of their spreadsheet. The research works presented above focus more on modeling and testing the spreadsheets itself. Our approach, instead, focuses on testing an external artefact with the help of spreadsheets. Yet, the contributions made in these works can complement our solution because they can help us to improve the accuracy of the spreadsheets we used for BP analysis.

11. Conclusion

In this paper we approached a relevant and timely issue in today's business process management practice, i.e., that of analysing processes. We specifically emphasized the role of the BP analyst, not only in providing input for the design of processes but also in analyzing them. In order to enable BP analysts to perform own analyses without the need for programming skills, we conceived a technique that is specifically tailored to the average skills of BP analysts. The intuition we followed for the implementation of the technique is to adopt common spreadsheets as abstraction and user interface for both setting up analyses and computing analysis reports.

As confirmed by our user studies, the spreadsheet abstraction has indeed the potential to enable BP analysts to perform fairly complex analyses autonomously and to effectively discuss findings with software developers, so as to iteratively improve their models. The qualitative analysis of the approach complements the user studies with a discussion of its strengths and weaknesses compared to the state of the art in BP model analysis.

The positive results we obtained encourage further extensions of the approach. Specifically, we would like to allow BP analysts to also obtain a concrete feeling of how their processes behave if deployed in a real BP system by emulating web services and visualizing process progress in a process monitoring dashboard. Comparing log data produced during the analysis of a process with real log data obtained after the deployment of the process would further enable the fine-tuning of the simulation/emulation. This, in turn, would improve analysis precision and turn the simulator into a viable tool, for example, to produce synthetic data for the testing of process mining algorithms.

Acknowledgments

We thank the team of DHL Paraguay for its generous collaboration with the user-study sessions.

References

[1] M. Weske, Business Process Management: Concept, Languages, Architecture, Springer, Berlin, Germany, 2007.

[2] J.M. Bhat, M. Gupta, S.N. Murthy, Overcoming requirements engineering challenges: lessons from offshore outsourcing, IEEE Softw. 23 (5) (2006) 38–44.

[3] A. Sutcliffe, User-centred Requirements Engineering, Springer Science & Business Media, London, UK, 2012.

[4] K. Wiegers, J. Beatty, Software Requirements, Pearson Education, Redmond, Washington, USA, 2013.

[5] M. Burnett, C. Cook, G. Rothermel, End-user software engineering, Commun. ACM 47 (9) (2004) 53–58.

[6] C. Scaffidi, M. Shaw, B. Myers, Estimating the numbers of end users and end user programmers, in: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE, Dallas, Texas, USA, 2005, pp. 207–214.

[7] Object Management Group (OMG), Business Process Model And Notation (BPMN) Version 2.0, ⟨http://www.omg.org/spec/BPMN/2.0/⟩, 2011.

[8] R.M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Inf. Softw. Technol. 50 (12) (2008) 1281–1294.

[9] R.M. Dijkman, M. Dumas, C. Ouyang, Formal Semantics and Analysis of BPMN Process Models using Petri Nets, Queensland University of Technology, Technical Report, 2007.

[10] W.M.P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, Berlin, Germany, 2011.

[11] W.M. Van Der Aalst, A.H. Ter Hofstede, M. Weske, Business process management: a survey, in: Business Process Management, Springer, Berlin, Germany, 2003, pp. 1–12.

[12] W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, N. Russell, Business Process Simulation: How to Get it Right, BPM Center Report BPM-08-07, BPMcenter. org, 2008.

[15] J. Lazar, J.H. Feng, H. Hochheiser, Research Methods in Human–Computer Interaction, Wiley, Chichester, West Sussex, UK, 2010.

[16] IBM, Tutorials and Samples for WebSphere Business Modeler version 6.2, IBM Corp., ⟨http://www-01.ibm.com/support/docview.wss?uid=swg27013902⟩, 2009.

[17] TIBCO, TIBCO Business Studio, TIBCO Software Inc., ⟨https://docs.tibco.com/pub/business-studio-bpm-edition/3.9.0/doc/html/index.html⟩, 2014.

[18] BizAgi, Bizagi Process Modeler—User Guide, BizAgi, ⟨http://download.bizagi.com/docs/modeler/2904/en/Modeler_user_Guide.pdf⟩, 2015.

[19] Bonitasoft, Bonita Open Solution—Simulation Guide, ⟨http://www.bonitasoft.com/system/files/download/bos-5.6-simulation-guide.pdf⟩, 2011.

[20] BOCGroup, Adonis Community Edition Getting Started Series, BOC Group, ⟨http://en.adonis-community.com/welcome/webinars-and-tutorials/adonisce-getting-started-series⟩, 2015.

[21] Signavio, Feature Overview Signavio Process Editor, ⟨http://www.signavio.com/docs/en/pe-features.pdf⟩, 2015.

[22] D. Hartvigsen, Simquick, Process Simulation With Excel, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 2004.

[23] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S.Thatte, et al., Business Process Execution Language for Web Services, ⟨http://goo.gl/kpBKUS⟩, 2003.

[24] P. Mayer, D. Lübke, Towards a BPEL unit testing framework, in: Workshop on Testing, Analysis, and Verification of Web Services and Applications, ACM, New York, New York, USA 2006, pp. 33–42.

[25] Z.J. Li, W. Sun, B. Du, BPEL4WS unit testing: framework and implementation, Int. J. Bus. Process Integr. Manag. 3 (2) (2008) 131–143.

[26] J. García-Fanjul, J. Tuya, C.De La Riva, Generating test cases specifications for BPEL compositions of web services using SPIN, in: WS-MaTe 2006, 2006, p. 83.

[27] Y. Yuan, Z. Li, W. Sun, A graph-search based approach to BPEL4WS test generation, in: International Conference on Software Engineering Advances, IEEE, Tahiti, French Polynesia, 2006, p. 14.

[28] J. Yan, Z. Li, Y. Yuan, W. Sun, J. Zhang, BPEL4WS unit testing: test case generation using a concurrent path analysis approach, in: ISSRE 2006, IEEE, Raleigh, North Carolina, USA, 2006, pp. 75–84.

[29] Z.J. Li, H. Tan, H. Liu, J. Zhu, N.M. Mitsumori, Business-process-driven gray-box SOA testing, IBM Syst. J. 47 (3) (2008) 457–472.

[30] A. Bucchiarone, H. Melgratti, F. Severoni, Testing service composition, in: Proceedings of the 8th Argentine Symposium on Software Engineering (ASSE07), 2007.

[31] M. Aguilar, T. Rautert, A. Pater, Business process simulation: a fundamental step supporting process centered management, in: Winter simulation, ACM, Phoenix, Arizona, USA, 1999, pp. 1383–1392.

[32] S. Narayanan, S. McIlraith, Analysis and simulation of web services, Comput. Netw. 42 (5) (2003) 675–693.

[33] S. Chandrasekaran, J. Miller, G. Silver, B. Arpinar, A. Sheth, Performance analysis and simulation of composite web services, Electron. Mark. 13 (2) (2003) 120–132.

[34] Y. Tan, S. Takakuwa, Predicting the impact on business performance of enhanced information system using business process simulation, in: Winter simulation, IEEE Press, Washington, D.C., USA, 2007, pp. 2203–2211.

[35] M.T. Wynn, M. Dumas, C. Fidge, A.H. Ter Hofstede, W.M.P. van der Aalst, Business process simulation for operational decision support, in: BPM, Springer, Berlin, Germany, 2008, pp. 66–77.

[36] A. Rozinat, W.M.P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Inf. Syst. 33 (1) (2008) 64–95.

[37] W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, E. Verbeek, Conformance checking of service behavior, ACM Trans. Internet Technol. 8 (3) (2008) 13.

[38] W.M.P. van der Aalst, T. Weijters, L. Maruster, Workflow mining: discovering process models from event logs, IEEE Trans. Knowl. Data Eng. 16 (9) (2004) 1128–1142.

[39] H.R. Motahari-Nezhad, R. Saint-Paul, F. Casati, B. Benatallah, Event correlation for process discovery from web service interaction logs, VLDB J. 20 (3) (2011) 417–444.

[40] R. Dijkman, M. Dumas, L. García-Bañuelos, Graph matching algorithms for business process model similarity search, in: BPM, Springer, Berlin, Germany, 2009, pp. 48–63.

[41] Y. Liu, S. Muller, K. Xu, A static compliance-checking framework for business process models, IBM Syst. J. 46 (2) (2007) 335–361.

[42] G. Governatori, Z. Milosevic, S. Sadiq, Compliance checking between business processes and business contracts, in: EDOC, IEEE, Hong Kong, China, 2006, pp. 221–232.

[43] C. Rodríguez, D. Schleicher, F. Daniel, F. Casati, F. Leymann, S. Wagner, SOA-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes, Serv. Oriented Comput. Appl. (2013) 1–18.

[44] P. Silveira, C. Rodríguez, F. Casati, F. Daniel, V. D'Andrea, C. Worledge, Z. Taheri, On the design of compliance governance dashboards for effective compliance and audit management, in: ICSOC/ServiceWave 2009 Workshops, Springer, Stockholm, Sweden, 2010, pp. 208–217.

[45] F. Casati, M. Castellanos, U. Dayal, N. Salazar, A generic solution for warehousing business process data, in: VLDB, VLDB Endowment, 2007, pp. 1128–1137.

[46] M. Sayal, F. Casati, U. Dayal, M.-C. Shan, Business process cockpit, in: VLDB, VLDB Endowment, 2002, pp. 880–883.

[47] Deloitte, Spreadsheet Management: Not What you Figured, Online: ⟨http://www2.deloitte.com/content/dam/Deloitte/us/Documents/audit/us-aers-spreadsheet-ebrochure-070710.pdf⟩, White paper, Deloitte, 2009.

[48] S. Kruck, Testing spreadsheet accuracy theory, Inf. Softw. Technol. 48 (3) (2006) 204–213.

[49] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, C. Wallace, End-user software engineering with assertions in the spreadsheet paradigm, in: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, Portland, Oregon, USA, 2003, pp. 93–103.

[50] F. Hermans, Improving spreadsheet test practices., in: CASCON, 2013, pp. 56–69.

[51] M. Fisher, G. Rothermel, The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms, in: ACM SIGSOFT Software Engineering Notes, vol. 30, ACM, New York, New York, USA, 2005, pp. 1–5.

[52] G. Rothermel, L. Li, C. DuPuis, M. Burnett, What you see is what you test: a methodology for testing form-based visual programs, in: Proceedings of the 20th International Conference on Software Engineering, IEEE Computer Society, Kyoto, Japan, 1998, pp. 198–207.

[53] M.M. Burnett, A.L. Ambler, Interactive visual data abstraction in a declarative visual programming language, J. Vis. Lang. Comput. 5 (1) (1994) 29–60.