DOBS: Simulating and Generating Data for Model Assessment and Mining - Technical report -

Kreshnik Musaraj[1] and Florian Daniel[2] and Mohand-Said Hacid[1] and Fabio Casati[2]

[1] Universite Claude Bernard Lyon 1; LIRIS, UMR CNRS 5205, 8 Boulevard Niels Bohr, 69622 Villeurbanne CEDEX, France {kmusaraj, mshacid}@liris.cnrs.fr
[2] Department of Information Engineering and Computer Science, University of Trento, Italy {daniel, casati, musaraj}@disi.unitn.it

Abstract. Model assessment and model mining are two key processes which are crucial in software design and engineering, service-oriented architectures and business process management. Model assessment evaluates the compliance of a model towards its specification before or after implementation. Model mining allows the extraction of the implemented model from its activity logs and without prior knowledge on the model itself. This paper presents DOBS, a model evaluation and data generation tool for the improvement and testing of model compliance and correctness and for assisting the process of mining state diagrams or flowcharts, such as business processes, web services etc. DOBS is a continuous time/discrete event simulator that allows the design, simulation and testing of a behavioral model before its expensive implementation, or to check and evaluate an existing real-world model such as a business process or web service for compliance requirements towards specifications. The data generation feature allows to analyze the output as well as to test mining methods on large amounts of realistic high-quality data. Experimental results show the efficiency and effectiveness of DOBS in modeling and analyzing diagram behavior, as well as the huge production capacity of realistic configurable data.

Keywords: Modeling, analysis, evaluation, optimisation, workflow, web service behavior, compliance, business protocol

1 Introduction

Model assessment (also referred to as model compliance) addresses the issue of conformance towards a policy, standard, law or technical specification that has been clearly defined [30]. This also includes but is not limited to conforming towards business regulations and stated user service requirements [22]. As the authors in [32] point out in their survey, compliance toward regulations is finding more and more attention in the eyes of the research community. Applications of model assessment include workflow checking, protocol verification,

and constraint validation [15]. Factors that motivate model compliance utility are: cost of the implementation prototype just for assessing the model, cost of re-implementing once that assessment results are negative, risk of testing on real-world already deployed systems, complexity of designed systems which prohibits exhaustive static verification and validation. There are critical flaws which obviously appear only during execution behavior analysis, be it on real implementation or simulation.

Model mining from activity logs is an extended research domain with important industrial applications and consequences. Its theoretical roots are automata learning and grammar inference [4,2,20]. More concrete and oriented applications are mining of workflows, business processes and software specifications, business protocol discovery and web application behavior extraction. Applications include: post-mortem monitoring, checking the equivalence between the specification and implementation (this particular application is mandatory for critical systems before deploying them online or in business platforms.), obtaining the specification if it does not exist, checking for security flaws, verifying that constraints in the execution flows are satisfied, checking if the designed model is correct, complete and finite (i.e. no deadlocks, infinite loops, bottlenecks), verify performance parameters on given parts of a model. Yet data for mining is extremely hard to get mainly for confidentiality reasons, or because the data is used for commercial purposes.

Model assessment and model mining merge together in synergy, especially regarding the tasks on (i) checking the equivalence between the specification and implementation and (ii) obtaining the specification if it does not exist. In this double-sided context we present **DOBS** (**D**ynamic **M**odel **B**ehavior **S**imulator), a modeling-and-testing generator tool which allows the expert of business processes,web services, or any other dynamic behavior-based systems, to design, test and simulate a behavioral model such as a business process or a web service / application. It also allows to generate activity data, of both low and high level of abstraction, from the simulation execution and testing. DOBS can thus be used as a testing or data generation tool, or both. It can improve model compliance while helping decision support during all stages of model assessment evaluation life-cycle. DOBS utility for compliance analysis may vary depending on the context of its usage since critical systems require both pre/post assessment, whereas nominal systems should in principle require only pre-implementation assessment.

Let us introduce an example scenario that illustrates the current need for a model simulator and data generator like DOBS. A banking institution wants to develop a new online service for its clients. Before starting the implementation work, which is resource and time consuming, the developing team needs to undergo several steps of design and model improvement, without having to go through the development of prototypes for each modification brought into the platform. Thus, the team models several versions of the protocol using the graphical interface of DOBS. Simulations are run in order to assess which version best fits the specified needs, as well as to detect potential flaws such as bottlenecks, security breaches, incorrect execution sequences of messages and

events. This includes drawbacks from poor design, or simply missing features that are not accounted for. Performance evaluation can take place during this step in order to determine which configuration of the protocol model is the most efficient and ergonomic. Meanwhile the team also wants to test its mining techniques for assessing the compliance of the prototype towards the initial model. This is a common practice in critical applications since it provides further proof that the traces of the simulated model do confirm the expected behavior or not. Therefore, the data output from simulations can be utilized in order to assess the model itself, as well as to check the mining techniques in case their efficiency needs to be confirmed.

One should also note that by combining the DOBS enhanced finite-state machine(FSM) representation with the source blocks generating data that behave according to user-defined statistical parameters, it is possible to obtain with DOBS more than just the corresponding automata of a given model, but also other richer and semantically equivalent representations such as Petri nets, probabilistic FSMs or Markov Models, that may be more suitable to the model context or expert expectations.

2 Evaluation criteria and data issues

In this section we give a particular attention to data-related hot spots. First of all, we define the elementary units of what we call data from activity logs or traces. We assume throughout this paper that the terms *log* and *trace* describe more or less the same concept despite technical considerations (which go beyond the scope of this paper). Data inside logs is considered in this paper to originate from very diverse models, for example business processes, medical workflows, web service (WS) business protocols, service-oriented architectures (SOA) etc. Thus, we can explicit units that represent the data which is logged into activity traces into the following set: **T**asks, **O**perations, **M**essages, **A**ctivities, and **E**vents. In the following, all these interaction units will be referred to as a **TOMAE**. We also underline the difference between an event in the context of DOBS and the definition of an event in the context of business processes. An event in DOBS is a generic concept associated to the activation of any TOMAE associated to an a transition of an automaton: the notion of event encountered in business processes and workflows is a particular case of the definition of an event in DOBS. Therefore, in the following when we mention an event we refer to the more generic DOBS definition of it. Now let us pursue with our analysis.

Next, we explicit the concept of a model. A model in this report refers to the *conceptual* model of the behavior of a system, and/or the semantics of the TOMAE flow associated with this system. The model is thus an abstract representation of the control flow of charts, workflows, and business protocols, that employs the state diagram notation.

Data that is to be used for mining purposes is subject to numerous issues and difficulties that jeopardize the whole mining process and even its implementation. The first problem is how to obtain the raw data itself. For example, during QDB

2009 [24] participants pointed out that authentic data sets are extremely hard to obtain for a researcher. Indeed, only researchers employed in R&D departments, or those participating in particular collaborations involving partners from data-generating fields (or which are in possession of these data sets) such as medical centers, large businesses, government agencies, service providers etc. can make usage of such datasets. All the other researchers are faced with a continuous and disturbing lack of data. This concerns not only real-world data but also synthetic sources such as generators, simulators, etc.

The problem itself is not the mere existence of real data sources. The real problem is the extremely limited access to those sources. The limited access to data unavoidably leads to a very negative consequence. Since the results of model mining heavily rely on characteristics which are inherent to data, this implies that the discovered model will be influenced by the input data considered. Specific patterns in models appear because of the underlying hidden properties and correlations which also exist in data. As it is clearly shown in [31,9], data has a huge impact on the mining itself. Without proper information, which characterizes "well-grown" logs and their very large size, mining methods will fail to test some of their most important characteristics such as completeness, quality, soundness of the results as well as scalability and performance issues. This concerns both existing methods and yet-to-come approaches and mining algorithms.

A straightforward solution to scarce datasets is to generate them. Despite being an apparently easy solution, yet generating data requires particular criterias to be met, in order to comply with what would be expected from real datasets. Indeed, generated data has in general much lower volumes compared to real world sets. Moreover, these synthetic datasets do not have the characteristics of real ones, for example statistical distribution, noise nature and level, content diversity, and so on. Consequently, we can summarize the following main criteria (among many others) :

1. Quality of the model behavior

The *completeness* criterion requires the model execution to cover all existing transitions, in other words every component of the behavior model is to be explored. Completeness of the data contained into the logs is a factor that heavily influences the analysis and mining results. The completeness criterion requires the model execution to cover all existing transitions, in other words every component of the behavior model is to be explored. For example, it is obvious that if particular TOMAEs are missing from the traces, simply because they are not considered by the logging application, then entire sequences that are followed during the diagram execution will be severely damaged or even lost because of incoherences due to missing TOMAEs in those sequences. Using its user-specified simulation speed or by analyzing output activity traces, the actual activation of all existing paths that can be followed for that given model can be immediately verified, either by direct observation of the running simulation model (see Figure 3), or by a very simple and fast analysis of activity traces, which uses high-performance code. This avoids design or verification scenarios

in which the control flow and the constraints inside the model contain design errors that might prevent particular model execution instances from executing correctly. Finding such flaws in a model using conventional techniques is extremely costly in terms of temporal and financial ressources, as underlined by the authors in [3].

Scalability expresses the capacity of DOBS in performing in both a rapid and reliable way when simulating large, complex, and multiple-instance models. This measure also concerns the data generation and processing capabilities of DOBS. Consistency on the other hand, assures that the designed model behaves according to the specification, and does not show unexpected, undefined, or non-deterministic behaviors. This is obtained through default runtime rules on transition selection that avoid any ambiguity inside the automata. Nevertheless, default runtime rules can be translated into partially simulated models, deadlocks, and generation of incomplete sequences of TOMAE. This is the reason why the completeness criterion is mandatory to be tested, along with scalability, as opposed to consistency, that is automatically avoided by default deterministic rules. We also include temporal consistency, that ensures for temporal data to correctly follow the time logic specified in the model.

Correctness of the model is an additional property to be verified. The same debugging GUI and functionalities make it possible to easily identify event sequences that occur in disregard of constraints defined over control sequences (>=, <> etc.), security and privacy constraints etc. This avoids implementing (or not identifying in existing models) faulty or non-existing constraints with disastrous consequences because of security threats or other fallout issues following the non-compliance toward the specified execution logic.

2. Properties and quality of generated data

Sheer *size* is an obvious property of activity logs and traces, since in realistic scenarios logging takes place during extended periods of time (months, years).

Temporal constraints ensure for temporal data to correctly follow the time logic specified in the model. For example, timestamps are supposedly processed and logged in the required order without undesired value alteration and overlapping. This should allow nevertheless for desired noise introduction using predetermined techniques. That is why the inclusion of imperfections is also to be considered, since all implemented logging tools will be responsible for flaws during data recording. These imperfections include noise (in a broad sense), errors, uncertainty on the data values itself, and any other data alteration process that implies potential errors or undesired data modification.

Statistical properties characterizing real activity data are also a key parameter. Traces output from different systems will exhibit different statistical behavior, following different distributions each one having its own parameter values. Thus, mimicking these stochastic models during data generation is mandatory for using these traces in a useful and profitable manner. Moreover, these properties are also to be verified in order to check that generated data values do

conform to expectations, therefore increasing the probability of a positive data and behavior assessment.

We also include the requirement that generated data must reflect in the most realistic manner (to the maximal extent possible) the actual behavior of a real world implementation of the model, even if the latter might not even exist, thus its real behavior being yet unknown. This feature calls for the design of a generator of a high level of abstraction, but which must also provide the capability of being configured at a very low level when it comes to its internal parameters, so that it can model virtually every model in the range, while at the same time offering a fine-tuning that allows the incorporation of particularities of the real model into the generator. This would provide a good mimicking of what would be considered satisfactory data.

The core of DOBS is a discrete event simulator that allows for the reproduction of the behavior of dynamic models. DOBS makes it possible through its graphical user-interface to model very complex structures in an extremely short time compared to the amount of time required for implementing source-code-based mock-ups or prototypes. In the case of an existing running model, DOBS allows to check the conformance between the specification model using its GUI, and the real system. Moreover this can be used to play what-if scenarios in case of a future update, allowing experts to assess problematic points, and the impact of every modification on the evaluation criteria. while avoiding the expensive and time-consuming of implementing changes on the software/real application level. In particular, DOBS allows users to assess several criteria of the studied model behavior.

At its basic level, DOBS implements an accurate, but also generic representation of a dynamic model. Modeling business processes, web service protocols, or software systems is a tricky and heavy-duty task where the slightest error can lead to many problems which are too numerous to be exhaustively enumerated. The correct simulation and data generation for a realistic model faces two main difficulties: (i) the very complex behavior of such systems and (ii) the extremely large quantities of data they are supposed to generate. Moreover, realistic simulation of processes and web services requires accounting for simultaneous multiple instances execution (up to millions of instances), employing time clocks that may feature asynchrony or not, and guaranteeing data constraints, type and other attribute-value restrictions. All these features combined together lead to restrictive requirements in terms of computational complexity. DOBS modeling interface offers the necessary complexity for designing very rich models with enough high expressive power to capture the dynamics of a real system, yet simple enough to allow for rapid design, simulation and data analysis and a user-friendly graphical interface for design, configuration, debugging and testing. DOBS models a dynamic workflow as a finite-state machine. The simulated model behavior is provided by the interaction of its states and super-states via labeled transitions, which constitute the basic elements of the DOBS behavior controller. The choice of employing the automata representation was motivated by their ability of efficiently modeling and computing a dynamic model's behav-

ior. Despite the FSM theoretical considerations in modeling certain models such as Petri nets or Markov chains, yet this is valid only for FSM in its original form [10]. These limitations do not concern DOBS, which uses an enhanced and more abstract form of a FSM with an expressive power equalling that of Petri nets and Markov chains. More specifically, each model has a set of input blocks, a set of controllers (the automata modeling the dynamic behavior) and the group of output blocks. When a model is run, events trigger the transition from one state to another, provided that the corresponding transition conditions and constraints are met. Once a transition is selected, all the associated statements in its label are executed.

3 Architecture of DOBS

The architecture of the DOBS tool is given in Figure 1 which depicts the conceptual schema of its main components: the Graphical User Interface (GUI), the Block Library (BL), the Simulator Controller(s) (SC), the Data Generator(s) (DG), the Log Pre-processing Library (LPL), and the Model Explorer (ME).

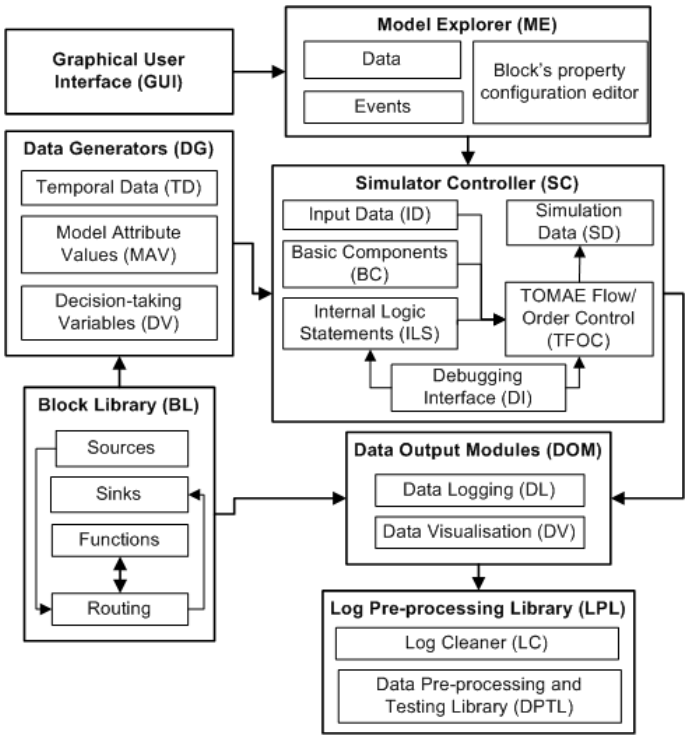

Fig. 1. Conceptual model of DOBS

- The **GUI** component is ubiquitous and the starting point for every simulation step. It mainly allows users to load a simulation model from a file and save it, to design and run a model from scratch by means of Block Library(BL) and Model Explorer(ME) modules, to modify simulation parameters or to update different model blocks including the controller behavior, and finally to start/stop simulations. The GUI can also be launched via a non-interactive command-line interface.
- The **Model Explorer (ME)** is responsible for defining and configuring all the data variables that the model is going to employ as of input, output or internal type, as well as events which are going to be triggered during runtime of the model simulation. This component includes also an editor for configuring the properties of each block in the model; this concerns also the controller(s) existing in the designed instance and the associated data and events.
- The **Simulator Controller (SC)** component implements the behavior specific to the targeted model. SC has six main sub-components which are described as follows:

1. The Input Data (ID) receives all incoming data from outside the SC and eventually initializes and/or prepares them for further usage.

2. The Basic Components (BC) block provides the elementary modules that will compose the automata whose execution represents the model's behavior. These modules include states, transitions, super-states, junction points, user-written functions, etc.

3. The Internal Logic Statements (ILS) is composed of single and optional statements such as variable instantiation, arithmetics and so on. These statements are edited and inserted directly into the transition labels of the automata, and executed if and only if the event identifying the transition is triggered and the associated condition returns the boolean **true** value. Despite being optional and their relative simplicity, ILSs are a key element that radically improves the features which can be obtained during a simulation in terms of data diversity and behavioral complexity. For an illustration of ILS, see Figure 3.

4. The TOMAE Flow/Order Control represents the core of SC. It implements the user-specified behavior by means of transition connections between states, enforcing transition constraints over determined values, probabilistic transition selection employing the input random user-defined distribution generated by DG. It can also define a more specific runtime behavior of the simulator, such as the verbose or silent output on the console, which can be used in real-time monitoring applications, state-bound triggering of events or data operations that are of great interest and usefulness in the case of business process simulation.

5. The Debugging Interface (DI) offers functionalities that allow a quick detection of human errors made during the modeling phase in DOBS. If an error is detected, for example an incorrect ILS on a transition, state inconsistency, transition conflict, or data range, DI highlights the part(s) of the model presenting the conflict and provides semantically rich information that allows users to quickly identify both the location and the source/cause of bugs. This interface also allows (a) to define breakpoints at chart entry, event broadcasts and

other points, (b) to enable/disable the graphical animation during the simulation and (c) to define a simulation delay. The simulation delay makes it possible to run the simulation at different speeds. This feature is very useful when visually monitoring the execution, or for demonstration purposes.

6. The Simulation Data (SD) is in charge of all data that is of interest to the model from the designer point of view. This does not include meta-data, internal variables or generated values from sources in Block Library (BL), all of which are employed solely for satisfying the requirements of the DOBS' inner mechanisms in order for the model to be simulated correctly and output data to be handled according to the users' expectations. The content which is output by the SD module contains everything that will be entered as input for the Data Logging (DL) module.

- The **Data Generators (DG)** module, as its name clearly indicates, has the task of grouping all blocks whose function is to generate all the necessary data that will be used and processed during a simulation on DOBS. The data issued by DG can be divided into two main categories: (i) data for the "visible" part of the model, i.e. that will constitute the basis for the model's activity output, (ii) data employed for model parameters' configuration, debugging, decision-making during simulation runtime, but yet without interest for the activity traces. DG is composed of three sub-components:

1. The Temporal Data (TD) provides realistic time values that are associated to TOMAE occurrences or state and transition activations in any given point of the model flowchart. DOBS uses continuous time values and the time interval can be defined by the user as finite (fixed-duration simulation) or infinite (very long duration of the simulation). Since users can specify both the minimal time unit and fixed-length delays, employing the infinite time interval has experimentally proven to be extremely useful for simulating models and generating activity data volumes that would require months, and even years, to collect in a real working platform [21]. The continuous time values are obtained by a digital clock of double-type precision. Nevertheless, DOBS also offers the feature of using discrete time values through integer-based counters (long format integer type) as well as limited intervals. Multiple independent clocks and counters can be integrated in the same model. They can be synchronized or not, and their respective parameters are configured separately, all these criteria being decided by the user during design time.

2. The Model Attribute Values (MAV) constitute the set of data that will be associated to every attribute of an existing TOMAE in the flowchart. The interval values for a given attribute can be of any type: enumerated sets used for generating TOMAE names and labels; (un)limited discrete-value sets useful for generating values that can represent any string of characters, for example URLs, identifiers, keys or any other desired usage.

3. The Decision-making Variables (DV) are part of a particular, yet extremely important group of DG. These variables bear the decision of TOMAE selection

in multiple-choice scenarios. In other words, when several transitions exit the same state, or when more than one TOMAE can be executed following the precedent one, it will be the task of a DV to generate the value that will be used to discriminate the selected transition or TOMAE to be followed on the next simulation step. The values for a DV are generated using uniform (LIST OTHERS) statistical distribution. The distribution type for these values is also user-specified. Note that the distribution type defined here will deeply affect the output data of the simulation, since the percentages of selected model paths will influence the occurrence rate of all attributes associated to all the transitions and states (in other words of the TOMAEs) included in their respective paths. This allows for simulating models representing Petri nets and Markov models and is therefore a feature that greatly enhances the generic capability of DOBS.

- **The Block Library Module (BL)** provides all the elementary blocks that will compose every model. Three sub-components constitute this module. They are categorized based on their functionalities:

1. The Source module includes all blocks that are responsible for data, value and noise generation. Among these we can mention: the *Band-limited White Noise* for generating normally distributed random numbers that are suitable for use in continuous or hybrid systems; the *Digital Clock* for producing current simulation time at the specified rate; the *Counter Limited* that wraps back to zero after it has delivered the specified upper limit; the *Random Number* that provides a normally (Gaussian) distributed random set of values with a non-repeatable output if the seed is modified; the *Uniform Random number* that provides a uniformly distributed random signal for the same seed conditions as the *Random Number*.

2. The Sinks module constitutes the set of blocks acting as the output interface. Among the existing ones, the most relevant are the *Display* for numeric display of input values, and the *To Workspace* block that writes the input to a specified array or structure in the main workspace. However, it is necessary to underline that for consistency reasons, data is not available until the simulation is stopped.

3. The Functions module is the most flexible part. It is composed of both pre-defined and new user-written functions that enhance the capabilities existing library blocks in BL. An editor allows to enter the function code or to modify it.

4. The Routing module is composed of block that channel the data and other values between the model components. The most important blocks of this module are (i) the *Demultiplexer* that splits either (a) vector signals into scalars or smaller vectors, or (b) bus signals produced by the Mux block into their constituent scalar, vector, or matrix signals (ii) the *Multiplexer* used for multiplexing scalar, vector, or matrix signals into a bus and (iii) the *Manual Switch* whose output toggles between two inputs by double-clicking on the block. These blocks allow users to design lighter models which are not visually overloaded with simple connections that quickly overload the GUI.

- The objective of the **Data Output Module (DOM)** is to ensure the appropriate handling of the output incoming from the SC module. More precisely, its two components Data Logging (DL) and Data Visualization (DV) deal respectively with (i) recording the SC simulation data by utilizing the correct data type storage format which comes in the form of arrays, matrixes, cell arrays, and symbolic values, and (ii) provide the appropriate data visualization interfaces by using either numerical displays for direct value reading, or plotting functions for observation of patterns or statistical study. Examples are depicted in Figure 4 and 5 that are described in detail in Section 5.

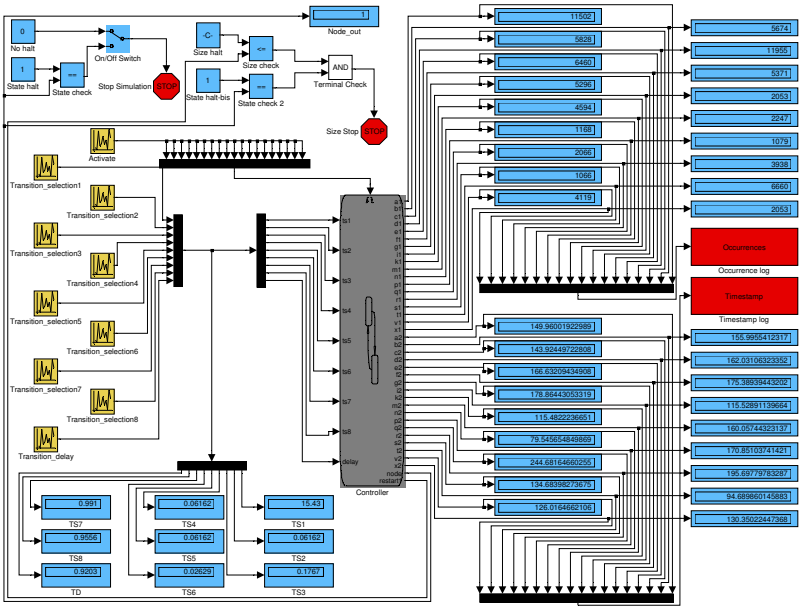

Fig. 2. Main window view of DOBS

- The **Log Pre-processing Library (LPL)** constitutes the final stage of the DOBS usage flow, and deals with the crucial task of (i) cleaning logs from redundant and other irrelevant data c.f. Log Cleaner (LC) sub-component and (ii) reorganizing data structures and manipulating data content in order to make it fit for further usage, as well as testing its properties in order to ensure that the output corresponds to the users' expectations before feeding the output to furthermore processing and analysis steps; this is carried out by the Data Preprocessing and Testing Library (DPTL) sub-component. Testing also includes visualization techniques, an example of which is illustrated in Fig.5. When we speak of data redundancy, we refer to the fact that the DL sub-component is

capable of logging data at a higher speed rate that the SC running simulation clock itself, thus recording multiple occurrences of the same event. Even if this phenomenon can be avoided by setting a lower logging speed rate for DL, experiments show that this is a useful feature that ensures that eventually no data loss can occur. Moreover, since the LC sub-component shows an extremely high performance, applying the LC with redundant logs provides both no-loss logs in a very short time, therefore combining both positive requirements without presenting data inconsistency risks.

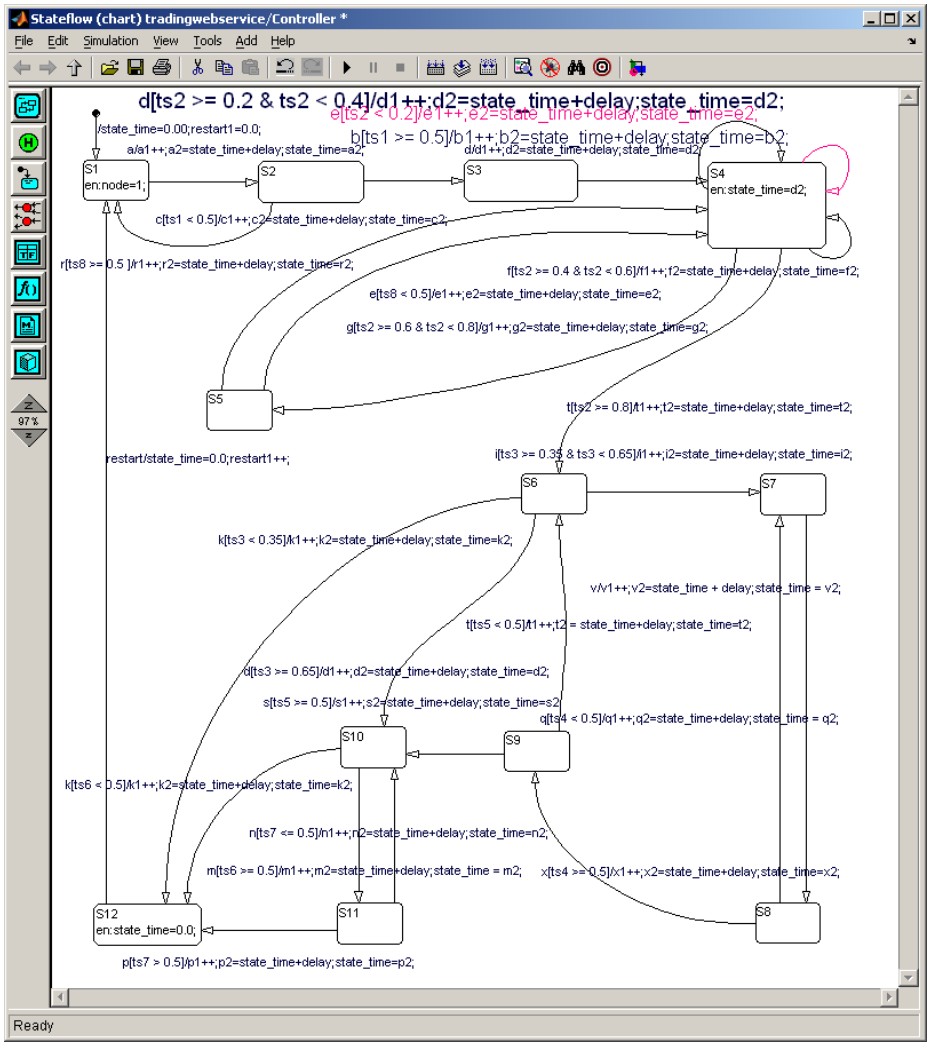

Fig. 3. Behavior designer interface - SC component

Figure 2 depicts the main interface of DOBS. The rectangle blocks in light blue are the numerical displays, the square ones in yellow are the numerical random value generators, the square blocks in light blue stand for constants and logical operators for execution control, and finally the rectangle grey block in the center is the SC module. By double-clicking the SC block, the SC design interface opens, an illustration of which is given in Figure 3. One should observe that DOBS has the very strong capability of being deterministic (yet allowing the implementation of the parallelism encountered in many workflows) thus avoiding inconsistencies and is capable of going through *every* transition thus providing a solid guarantee for exhaustive execution completion and completeness of data, as we will see in Section 4.

DOBS is implemented in the Matlab Simulink environment [16], and the LPL module is written in the Matlab programming language. The Matlab Simulink environment was selected mainly because it has the required features for an efficient implementation of DOBS. Among these features we can mention the set of existing blocksets and toolboxes. The choice was also influenced by the opportunity to graphically design dynamic models using the Simulink environment. This software package is well known for modeling, simulating, and analyzing dynamic systems. The final reason for this choice is the existence of very numerous libraries that are made freely available from many academic researchers. Nevertheless, since Matlab is not free software we also explored other candidates. Indeed, Scilab [27] offers a quite interesting option, with its Scicos code generator whose functionalities are unfortunately quite limited compared to those of Simulink, notably regarding the dynamic simulation capabilities. Octave [8] on the other hand is even more limited since it only offers a command-line interface thus allowing only for the LPL library to be tested upon it.
The implementation of DOBS is composed of two distinctive parts. The first one is composed of all the modules that appear in the GUI, and that are provided by the Simulink tool itself. More precisely, it consists of the definition of adequate parameters for these modules, as well as choosing the required set of modules. Another component of the implementation consists of the source code in which are written functions, and the TFOC, ILS, and DV modules. In conclusion, this section of DOBS is implemented using the graphical programming language (GPL) of Simulink. The second part of the implementation consists of Matlab functions written specifically for DOBS in the Matlab dataflow programming language (DPL). Virtually every component in the conceptual schema in Figure 1 exploits the capabilities of the Simulink GPL, except the LPL component. The latter, including Functions and DV sub-components are implemented using only the Matlab DPL. Since systems can also be defined to be multirate, i.e., have different parts that are sampled or updated at different rates, this allows for simultaneously simulating several models at a time which is a useful feature.

4 Experiments and testing

This section presents an experimental evaluation of DOBS based on several key criteria. The main objectives of the experiments were to check that DOBS meets its objectives provided in Section 2. We briefly recall them: (i) the quality of the model behavior in terms of completeness, scalability, and consistency, and (ii) the properties and quality of generated data in terms of temporal consistency, pattern coherence, and statistical properties.

DOBS was used to generate data for (i) the WatchMe scenario [22] (see Figure **??**) in order, to assess compliance restrictions (ii) the Drug Dispensation process [23] (see Figure **??**) for process mining based on uncertain data, and (iii) the fictitious commercial web service TradingWS (illustrated in Figure **??**) for client service behavior simulation. The WatchMe scenario describes the business process of an online multimedia delivery system. The Drug dispensation process describes the business process followed during the delivery of drugs in the medical domain. The TradingWS web service describes the business protocol of an online shopping service very similar to eBay. For all three scenarios transition selections are randomly chosen following a uniform distribution at generation time. TOMAE inter-arrival times were independently configured in order to assure that no correlation occurred during the simulation. This is in fact a required condition for a realistic simulation.

Table 1 illustrates the evolution of processing time required by DOBS to generate a fixed number of instances during the simulation of the Trading Web service. The outcome of the experimental results in this Table, given in the second and third column, are based upon the variation of the parameter given by the number of completed instances in the first column. One can notice the linear progression of the time required for simulation and generation versus both the number of messages and instances. A more visible form of this result is given in Figure 6.

Table 1. Performance metrics of simulated messages from TradingWS web service

# Instances	Time (sec.)	# Generated events
500	69	7485
1000	135	14778
2500	331	36080
5000	652	71207
10000	1473	132562
25000	3409	353549

We show in Table 2 the experimental results on the selection rate of multiple transitions. This corresponds to the criteria of completeness and consistency. The results aim at showing that all of the considered messages were executed according to the specified behavior in Figure 3. This behavior relates to the selection conditions defined in the TFOC sub-module. As the third column of

this Table shows, the divergence between the expected selection rate and the experimental rate is significantly low. This non-zero divergence corresponds to what is expected from a real execution of the model. In addition, messages **d**, **r**, **g**, and **f** have the highest level of divergence. This is due to their loop-based behavior and this provides the expected proof that the pattern coherence criterion is respected during simulation.

Table 2. Statistical metrics of simulated messages from TradingWS web service

Message type (abbreviation)	Selection rate in mutiple-choice transitions (%)	Estimated difference with expected rate
loginOK (b)	0.50556	$+5.561 \times 10^{-3}$
loginFail (c)	0.49443	-5.561×10^{-3}
browseProducts (d)	0.22849	$+28.49 \times 10^{-3}$
addToList (e)	0.20552	$+5.527 \times 10^{-3}$
order (r)	0.55121	$+51.21 \times 10^{-3}$
viewDetails (g)	0.17361	-26.39×10^{-3}
deleteFromList (f)	0.16776	-31.27×10^{-3}
confirmProductList (t)	0.20236	-2.361×10^{-3}

Figure 4 provides the temporal distribution of several events of the WatchMe process workflow, when using a relative timeline, i.e. a clock reset to zero at the beginning of each model instance simulation. One should expect from the execution of the model that: (i) events are executed in the correct order, (ii) timestamps are included in the defined interval, and (iii) for a given event ID, its occurrences are distributed along the defined interval. This Figure inidicates that these expectations are eventually met, when the three previously mentioned conditions are studied from the view of the BPMN model defining the process. For example, the TOMAE having the $ID = 1$ (value on the y axis) is the first TOMAE to be executed, thus its temporal interval (measured on the x axis) is the narrowest and it also has a very high occurrence density on that interval. The TOMAE having the $ID = 12$ is the last TOMAE to be executed, and thus its interval right limit is the highest, and its occurrence density considerably lower.

Figure 5 shows that TOMAEs are correlated during the simulation as expected. On the right hand of the chart one sees that the dynamics of the two considered messages are indeed correlated. This derives from the structure of the WatchMe protocol that connects these messages. If we take a closer look at this Figure, we see that an abrupt change in the number of occurrences of the event type $LoginSuccess$ (in blue), is followed by a proportionally drastic drop in the same number for the event type $SearchMedia$ (in green). The visual pattern on the right provides further proof that the temporal constraints are not violated during log generation and transformation, since the events are ordered in a sequential fashion from the temporal perspective. Both plots were obtained

using the LPL module for data cleansing and visualization.
Figure 6 depicts the temporal performance of DOBS during generation of increasing quantities of data, with the impact of both instance and event numbers. From this Figure it can be deduced that the two variables, namely the numbers of instances and events, have a very similar impact upon the performance of DOBS. Figure 7 demonstrates that the statistical properties of data (blue plot) fit the theoretical estimation of the statistical distribution (in red), that in this case is the Exponential distribution. The extensive analysis of these Figures allows to deduce more information than what we mention up to this point, yet this analysis is beyond the scope of the present work.

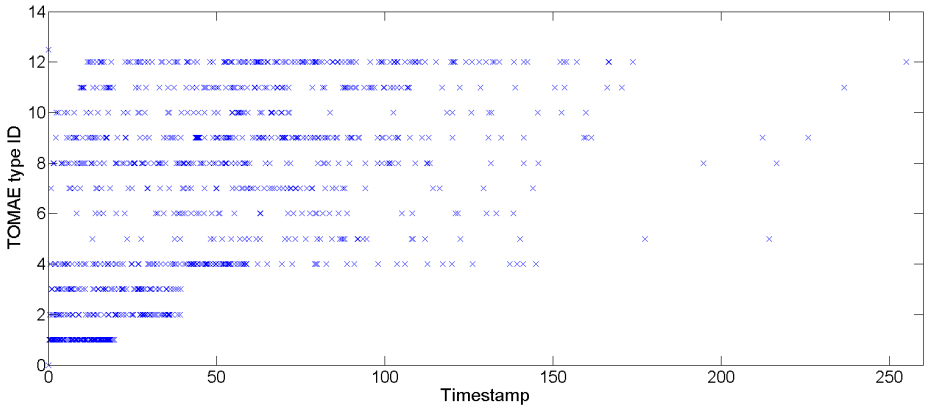

Fig. 4. Timeline sequencing of simulated TOMAEs from the WatchMe scenario [22]

Additional evidence of the advantage of using DOBS for simulation and data generation purposes was provided by internal surveys in our team which showed that while the development of a BPMN model took in average 3-5 man days, it took on the other side several man weeks to complete the same development task on the same model by implementing solely the data generation capabilities in the Java programming language.

5 Related work

The work presented in this paper is positioned in a domain crossroad connecting model assessment (including process compliance) [26,13,14], model mining [25,7,28,1,6,11,18,29], grammar inference [5,4,2,20], and diagram behavior simulation [3]. To the extent of our research results on preceeding tools, all the existing attempts to design and build tools that might achieve similar goals to those of DOBS present limitations since they are designed to deal with particular situations, hence suffering from non-generic functionalities which severely

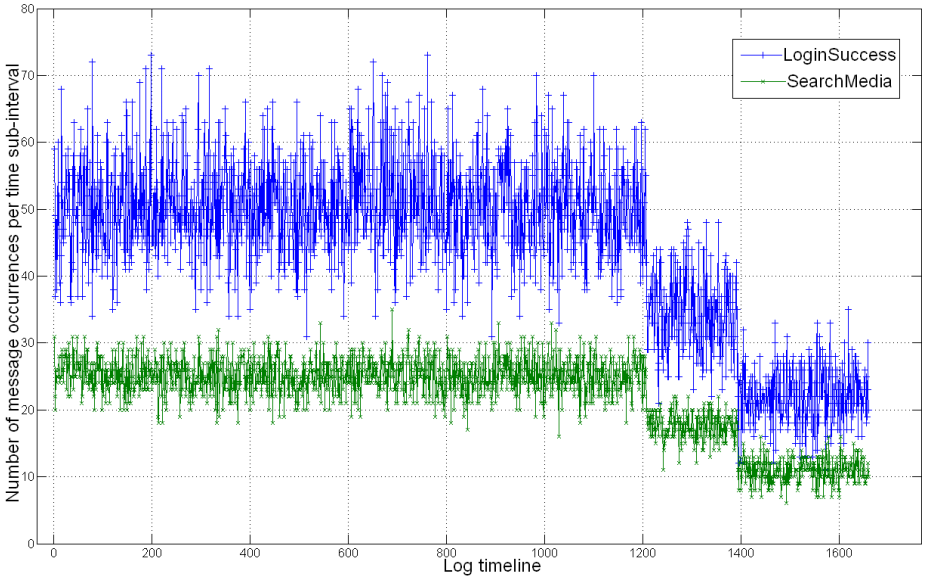

Fig. 5. Temporal distribution of simulated messages from the WatchMe scenario [22]

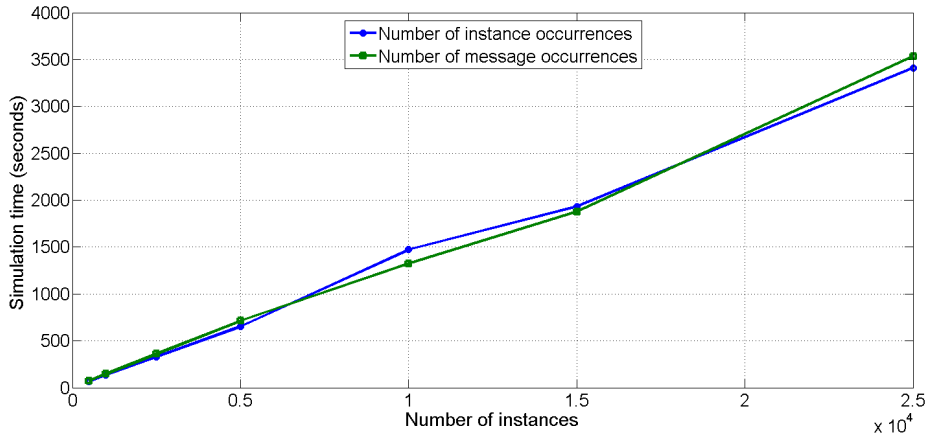

Fig. 6. Scalability measures versus number of instances and events generated for TradingWS.

restricted their extensive usage. Several tools address the issue of simulation and data generation of state-diagrams.

In [3] the authors present SYMIAN, a decision support tool for the improvement of incident management performance. This tool tests corrective measures for the IT support organization by improving performance measures. Since this simulation tool is targeting the performance optimization of IT management

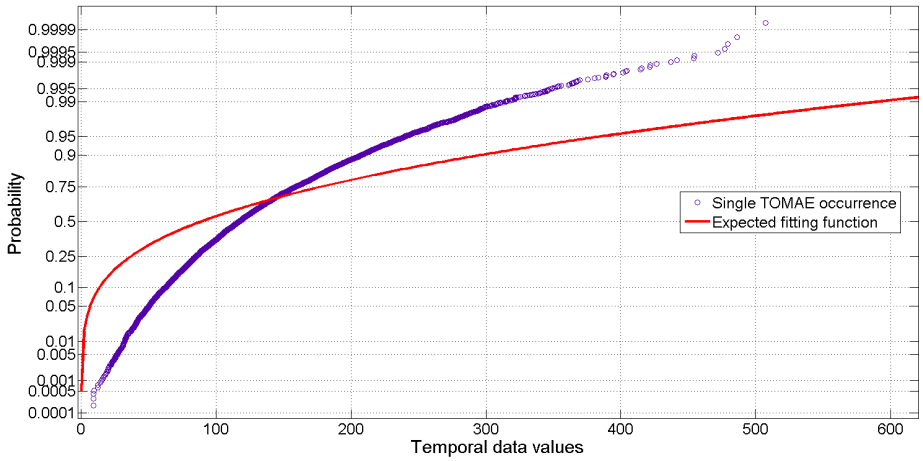

Fig. 7. Cumulative distribution function of simulated messages (blue) of TradingWS and theoretical fitting function (red)

processes in a very precise manner, it is thus not possible to employ it for more generic goals such as the ones of DOBS. The main difference between DOBS and SYMIAN is the target objective: the former addresses assessment analysis and data generation, while the latter considers only performance issues. In that sense, SYMIAN can be seen as a potential application case of the more universal DOBS. The Sage-Combinat toolbox [19] also offers interesting capabilities in exploring weighted automatons, which corresponds indeed to one of the many features of DOBS. This toolbox runs on the former MuPad application, which no longer exists since it was actually acquired by The Mathworks and incorporated into the Symbolic Math Toolbox for Matlab [16]. Yet, exploring finite-state machines is a very narrow application of Sage-Combinat, which, as an algebraic combinatorics tool, has objectives that extremely diverge from the scope of the domains considered in this paper. Nevertheless, this toolset has the important property of guaranteeing that all the transitions of a given automata are explored. We showed in the precedent section that this property is indeed fundamental, this is why a particular attention was given to the fact that DOBS could offer the same guarantee. Also, since the application which served as a running platform for Sage-Combinat is no longer officially available, this severely limits any future usage.

The authors in [5] provided a command line utility for generating sequences of words belonging to a user-specified regular grammar associated to a probabilistic automata. The capabilities of this generator are quite limited since it can provide only data for testing grammar inference algorithms, and it did not aim at providing any generic and more complete logs. Thus, this application is not of any interest in the mining domain. However, this tool might have been of use in order to check that sequences of events generated from DOBS actually

complied with the probabilities associated to a transition in the designed model. Yet, since this verification functionality is already included in DOBS, this usage case is no longer needed.

With acknowledgment to all these works, DOBS incorporates a new and innovative approach that provides for the first time a proposal and implementation framework for modeling the inner mechanisms of state diagrams in the context of processes, web services and software, thus supporting behavior analysis and data generation for these systems.

6 Conclusions and future work

Simulation of business processes, web service business protocols, and other structures based on state diagrams for assessment analysis and data generation for mining applications is a complex task. Nevertheless achieving these objectives is very helpful for assisting and allowing these analysis and mining applications to be tested. This paper described and detailed the DOBS tool for for the improvement and testing of model compliance and correctness and for assisting the process of mining state diagrams or flowcharts.

In the future, DOBS will be enhanced with the incorporation of more explicit compliance concerns, in order to test, for high-level semantic constraints, for example security, privacy, protocol specified actions etc. An on-going effort is addressing the future improvement that will be the automatic synthesis of state diagrams based on the traces of their behavior. This will allow users not only to design a model from scratch using the GUI interface, but also to have an automatically built model which will extract all the model structure and the corresponding parameters directly from the activity logs, if the user is in possession of the latter.

A particular interest will be accorded to incorporating other statistical models as a basis for model design. For example, statistical distributions for the inter-arrival time between TOMAEs in a model will include the exponential and Erlang distributions which can be very helpful in designing telecommunication-based systems, extending therefore the potential applications for DOBS. Another important feature that will be added is to provide DOBS as a service which will be accessible over the web.

Finally, DOBS will be completed with adapter modules in order to link its input entry point with BPEL or other object models, such as for example the BPMN Modeler for Eclipse [12]. Metrics for assessing the similarity and divergence between models are also part of future capabilities that will be integrated into DOBS.

References

1. Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *EDBT '98*, pages 469–483, Valencia, Spain, Mar 1998. Springer.

2. Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269, Sep 1983.
3. Claudio Bartolini, Cesare Stefanelli, and Mauro Tortonesi. Symian: A simulation tool for the optimization of the it incident management process. In *DSOM '08: Proceedings of the 19th IFIP/IEEE international workshop on Distributed Systems: Operations and Management*, pages 83–94, Berlin, Heidelberg, 2008. Springer-Verlag.
4. A. Biermann and J. Feldman. On the synthesis of finite state machines from samples of their behavior. *IEEE Transactions on Computers*, 21(6):592–597, 1972.
5. R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Proc. 2nd International Colloquium on Grammatical Inference - ICGI '94*, volume 862, pages 139–150. Springer-Verlag, 1994.
6. Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, July 1998.
7. Schahram Dustdar, Robert Gombotz, and Karim Baïna. Web services interaction mining. Technical Report TUV-1841-2004-16, Technical University of Vienna, Vienna, Austria, Sep 2004.
8. GNU. Octave. `Available at http://www.gnu.org/software/octave/`.
9. David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining*. MIT Press, Cambridge, MA, USA, 2001.
10. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
11. San-Yih Hwang and Wan-Shiou Yang. On the discovery of process models from their instances. *Decision Support Systems*, 34(1):41–57, Dec 2002.
12. Intalio Inc. Soa tools bpmn modeler. `Available at http://www.eclipse.org/bpmn/` .
13. Dimitris Karagiannis. A business process-based modelling extension for regulatory compliance. In *Multikonferenz Wirtschaftsinformatik*, 2008.
14. Marwane El Kharbili, Sebastian Stein, Ivan Markovic, and Elke Pulvermueller. Towards a framework for semantic business process compliance management. In *GRCISÕ08 Workshop at CAiSEÕ08 - Governance, Risk and Compliance: Applications in IS*, June 2008.
15. Elisabetta De Maria, Angelo Montanari, and Marco Zantoni. Checking workflow schemas with time constraints using timed automata. In *OTM Workshops*, pages 1–2, 2005.
16. The Mathworks. Matlab and simulink. `Available at http://www.mathworks.com`.
17. Kreshnik Musaraj, Dider Devaurs, Fabien De Marchi, and Mohand-Said Hacid. Timed transition discovery from imperfect web service logs: A stochastic approach (extended version). Technical Report RR-LIRIS-2008-007, LIRIS UMR 5205 CNRS/Université Claude Bernard Lyon 1, Villeurbanne, France, 2008. Available at: http://liris.cnrs.fr/∼kmusaraj/musaraj-al-extended.pdf.
18. Hamid R. Motahari Nezhad, Régis Saint-Paul, Boualem Benatallah, and Fabio Casati. Protocol discovery from imperfect service interaction logs. In *ICDE '07*, pages 1405–1409, Istanbul, Turkey, Apr 2007. IEEE.
19. CNRS France & others NSF USA. Sage-combinat: Extensible toolbox for computer exploration in algebraic combinatorics. `Available at http://wiki.sagemath.org/combinat`.

20. Rajesh Parekh and Vasant Honavar. Grammar inference, automata induction, and language acquisition. In *Handbook of natural language processing*, pages 727–764. Marcel Dekker, Inc., New York, USA, 2000.
21. G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *The First International Conference on Scalable Information Systems*, 2006.
22. European Commision Seventh Framework Programme. Compliance-driven models, languages, and architectures for services. `Available at http://www.compas-ict.eu/`.
23. European Commision Seventh Framework Programme. Managing assurance, security and trust for services. `Available at http://www.master-fp7.eu/`.
24. QDB09. In *7th International Workshop on Quality in Databases at VLDB'09*, 2009.
25. Motahari Nezhad Hamid Reza. *Discovery and adaptation of process views*. PhD thesis, Computer Science and Engineering, Faculty of Engineering, UNSW, Sydney, Australia, 2008.
26. Shazia Wasim Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In *BPM*, pages 149–164, 2007.
27. Open Source. Platform for numerical computation. `Available at http://www.scilab.org`.
28. Wil van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, Sep 2004.
29. Wil M.P. van der Aalst and A.J.M.M. Weijters. Process mining. In *Process-Aware Information Systems: Bridging People and Software through Process Technology*, pages 235–256. John Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2005.
30. Wikipedia. Regulatory compliance. `Available at http://en.wikipedia.org/wiki/Compliance_(regulation)`.
31. Ian H. Witten and Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann series in data management systems, USA, 2005.
32. Ernst & Young. European fraud survey 2009. `http://www2.eycom.ch/publications/items/fraud_eu_2009/200904_EY_European_Fraud_Survey.pdf`.

A DOBS Simulation scenarios

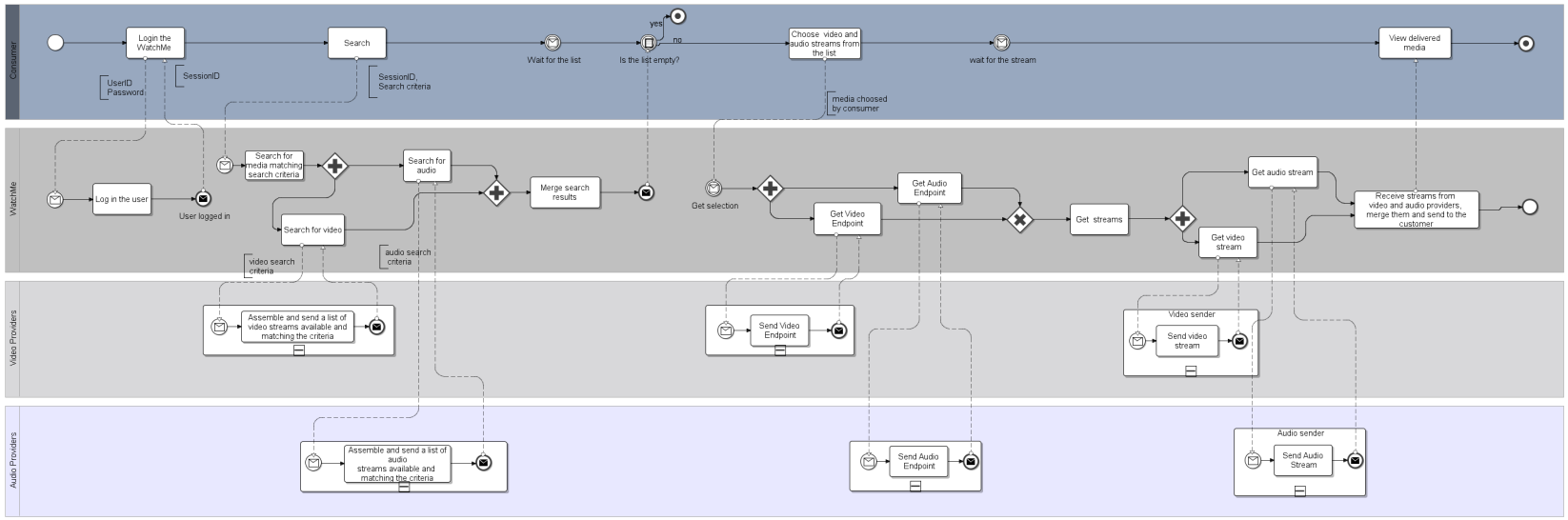

Fig. 8. The BPMN of the *WatchMe* scenario

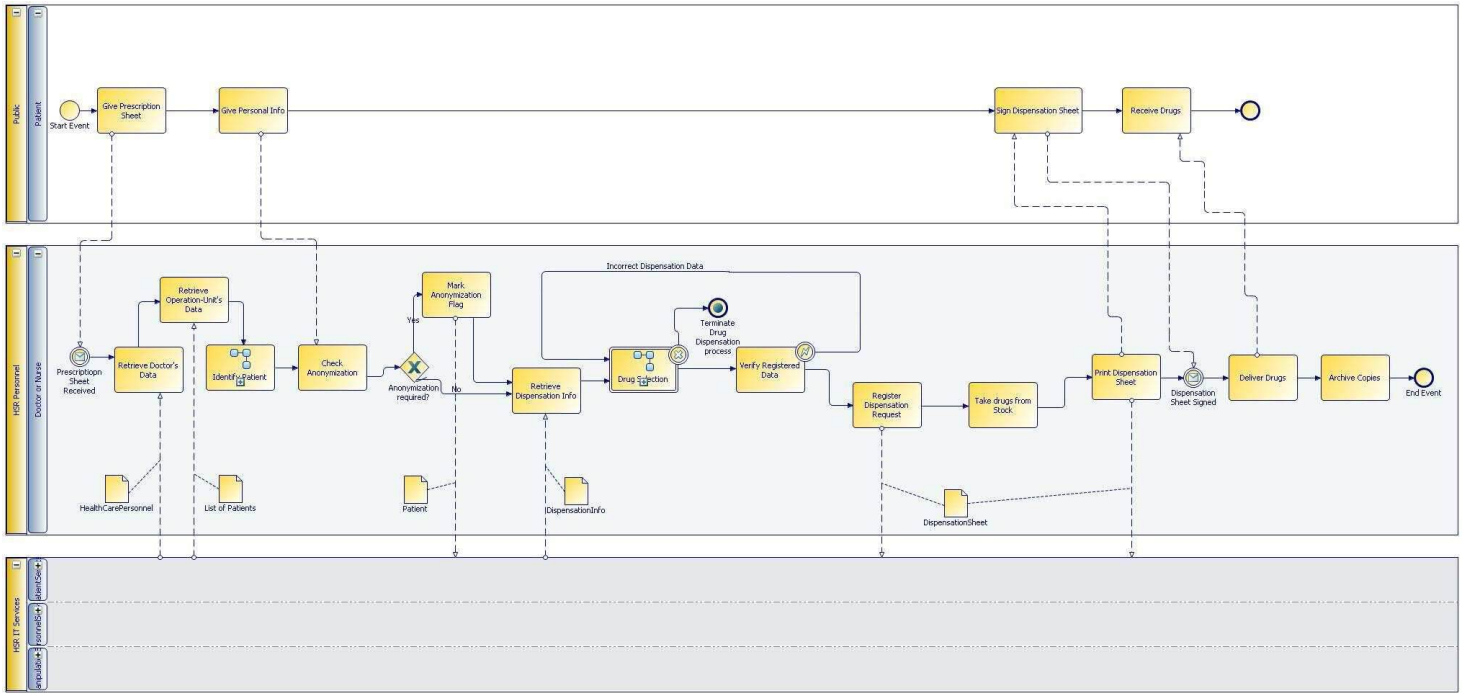

Fig. 9. The BPMN of *Drug Dispensation* process

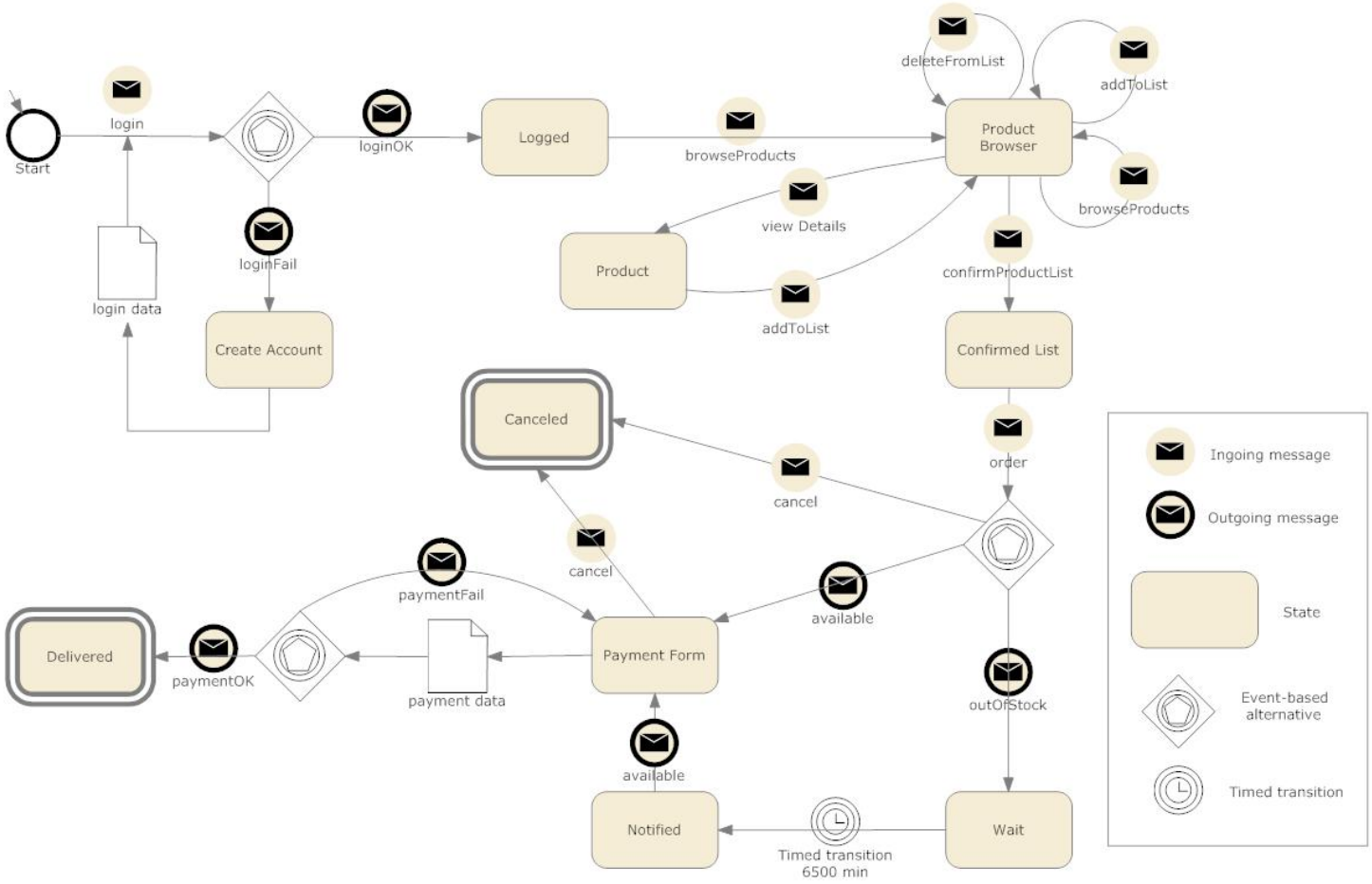

Fig. 10. The business protocol of $TradingWS$ service