

BP2SAN

From Business Processes to Stochastic Automata Networks

Kelly Rosa Braghetto
Department of Computer Science – University of São Paulo
kellyrb@ime.usp.br

March, 2011

Contents

1	Introduction	1
2	Instructions for Use	2
2.1	Installation and Execution	2
2.2	Conversion Parameters	2
2.3	Input Files	4
2.3.1	BPMN Models	4
2.3.2	Resources Declaration	6
2.3.3	Requirements Declaration	7
2.4	Output Files	9
3	Extracting Performance Indicators from the Generated SAN Models	9

1 Introduction

BP2SAN is a software tool that automatically converts business processes modeled as annotated BPMN (*Business Process Model and Notation*) process diagrams [4] to SAN (*Stochastic Automata Networks*) [5, 6].

BPMN is a graphical notation of OMG (Object Management Group) created to standardize the representation of business processes. Despite being able to support business users in different phases of the business process life cycle, BPMN models have no formal semantics. They are not well suited to qualitative and quantitative analyses. Furthermore, BPMN diagrams do not provide mechanisms to quantify the computational/human effort required to perform the activities, nor the capacity of work of shared resources and their access policies. This deficiency hinders the use of BPMN for performance evaluation.

SAN is a structured Markovian formalism that enables us to build stochastic models in a compositional approach. Created to attenuate the well-known state space explosion problem associated with the Markovian formalisms, SAN can be applied in the modeling of parallel and distributed systems with large state spaces. It is very efficient regarding the memory consumption, in addition to provide the concept of functional rates - that can facilitate the modeling and help reduce the size of the state-space. More than support verification, SAN models support the performance evaluation of business processes via numerical analysis.

The **BP2SAN** tool is able to convert a subclass of the BPMN process diagrams to SAN models that reflect the modeled control-flow of the business processes. In addition, when information about the resource requirements of the BPMN process diagrams are provided to the tool, the automatically generated SAN models contemplate the resource management policy associated to the process (or, in other words, they are models ready to provide performance indices). The input files for the **BP2SAN** tool must be BPMN models textually specified using the DOT Language and (optionally) files describing the resources/requirements associated to the BPMN models. These last files must contain declarations in a language specifically created for resources/requirements specification in the **BP2SAN** tool. The generated SAN models are textually expressed using the syntax accepted by the PEPS tool – a numerical solver for SAN models [2]¹.

Some details about the conversion algorithm implemented in **BP2SAN** can be found in [1].

2 Instructions for Use

2.1 Installation and Execution

BP2SAN was developed in the Java platform and requires Java 6 (or superior)² to run.

To execute the tool, please follow these two steps:

1. Download and uncompress the zip file with the tool. This will generate a folder with the file `bp2san.jar` and 2 folders: `lib` and `examples`.

`lib` contains the third-part libraries required by **BP2SAN**.

`examples` contains examples of BPMN models that can be used as input for **BP2SAN** and the SAN models resulted from the conversion.

2. At DOS or UNIX command prompt, enter in the folder of the tool and execute the command

```
java -jar bp2san.jar
```

This will open a graphical user interface, that allows the selection of input files and conversion parameters, as well as the visualization of the models.

Warning: `bp2san.jar` depends on the `lib` folder to be properly executed; `lib` must be in the same folder as `bp2san.jar`.

2.2 Conversion Parameters

Figure 1 shows a screenshot of the graphical interface of the **BP2SAN** tool. In the figure, it is possible to see in the left panel the input parameters required by the tool, indicated by the values in the following fields:

- “Input file” – the DOT file with the BPMN model of the business process to be converted. Section 2.3.1 describes the format this file must have;

¹PEPS is available at <http://www-id.imag.fr/Logiciels/peps>.

²Java is available at <http://www.java.com>.

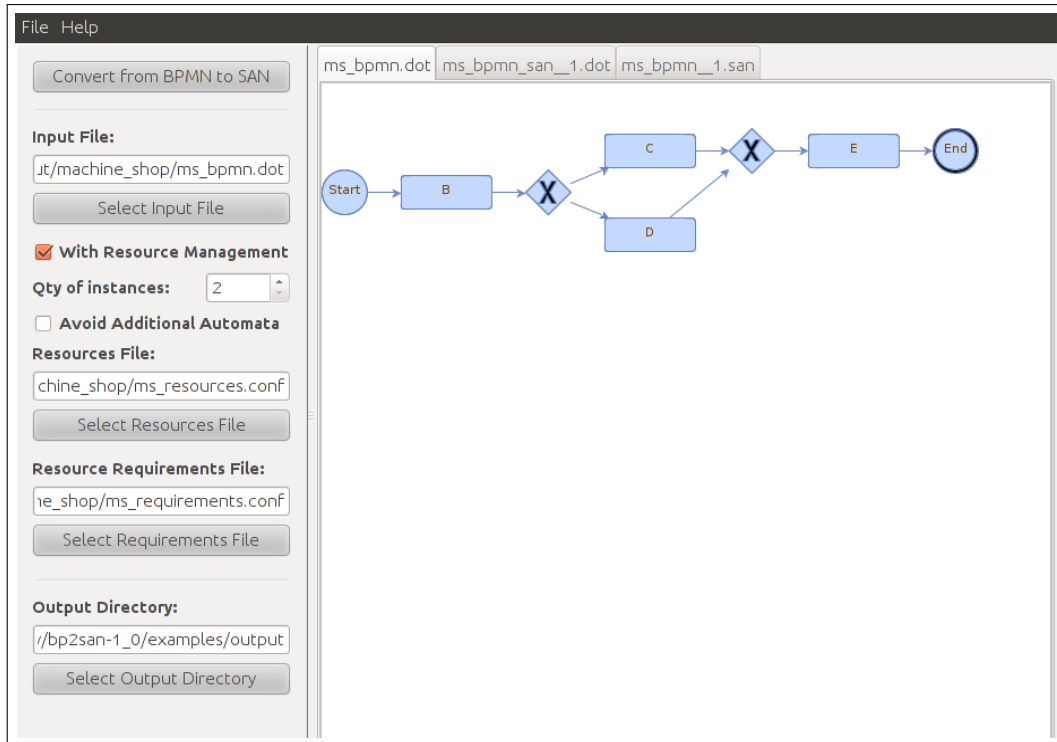


Figure 1: Screenshot of the BP2SAN tool.

- “With Resource Management” – check this box when the resource management associated to the business process should be take into account during the conversion. To generate a SAN model considering resource management, the following parameters will be also required:
 - “Qty of Instances” – number of instances executing in parallel that will be considered in the SAN model;
 - “Avoid Additional Automata” – resources of type *random choice* (see Section 2.3.2) can be expressed in the SAN model as additional automata + functional rates or only with functional rates. Check this box when the additional automata should be avoided when it is possible to avoid them.
 - “Resources File” – a file containing the description of the resources used in the business process. Section 2.3.2 describes the format this file must have;
 - “Resource Requirements File” – a file containing the description of the resources requirements of the activities of the business process. Section 2.3.3 describes the format this file must have;
- “Output Directory” – the directory where the generated SAN models will be saved.

In the right panel in Figure 1, a graphical view of the BPMN model selected to be converted is shown. In addition, after the conversion, a pair of tabs is opened for each one of the generated SAN models: one to show a graphical view of the SAN model and other to show the textual representation of the SAN model (fore more details, see Section 2.4).










	Start Event		Exclusive Gateway		Sequence Flow
	End Event		Parallel Gateway		Association
	Atomic activity		Inclusive Gateway		Annotation

Table 1: Basic flow and connecting objects of BPMN.

2.3 Input Files

2.3.1 BPMN Models

BP2SAN is able to deal with a subclass of the models that can be represented as a BPMN Process diagram. Table 1 introduces the BPMN objects that can be used in the model accepted as input by the tool. These objects are very important to process modeling and with them we are able to express a rich class of business process models.

To be accepted by BP2SAN, the BPMN Process diagram must be a *well-defined* business process graph. A well-defined business process graph respects the following properties:

- a start event vertex can have only one output vertex and no input vertices;
- an end event vertex can have only one input vertex and no output vertices;
- an activity vertex can have only one input vertex and only one output vertex;
- each gateway vertex has one of the following labels: “+”, for a parallel gateway; “○”, for an inclusive gateway; and “×”, for an exclusive gateway;
- a gateway vertex can perform a role of divergence or a role of convergence (but not the two roles at the same time). As a consequence, a gateway can have only one input vertex and more output vertices (case of divergence), or can have only one output vertex and more input vertices (case of convergence);
- for all vertex v of activity or gateway: (i) there exists a path from one start event vertex to v , and (ii) there exists a path from v to one end event vertex;
- each output edge of an exclusive/inclusive gateway vertex must have an associated probability value;
- the sum of the probabilities of the output edges of an exclusive gateway vertex must be 1;
- an exclusive gateway does not converge (join) parallel sequence flows;
- a parallel gateway does not converge (synchronize) alternative sequence flows;
- an inclusive gateway only converges (merges) sequence flows originated by another inclusive gateway. In addition, there is an one-to-one correspondence between the diverging and the converging inclusive gateways.

If a not well-defined BPMN graph is provided as input to BP2SAN, then the tool will return an error message and will not enable the conversion of the model.

Figure 2 shows an example of well-defined BPMN model.

A BPMN model must be passed to BP2SAN in a textual description in the DOT language [3] (<http://www.graphviz.org/doc/info/lang.html>). The syntax of the DOT file accepted by BP2SAN is given by the following grammar (specified in Backus Normal Form):

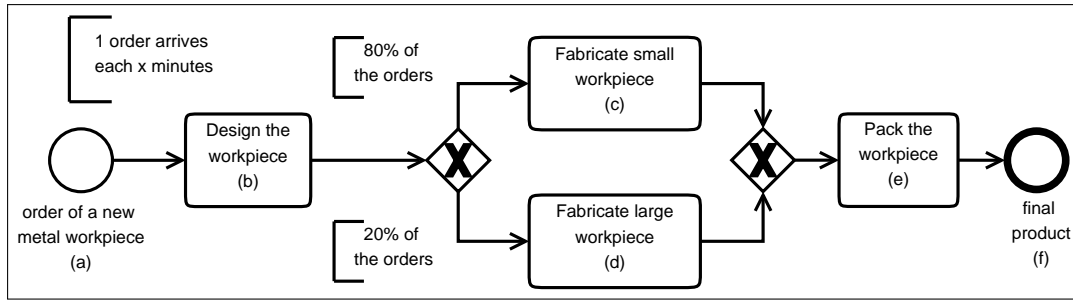


Figure 2: BPMN model of the fabrication process of a small machine shop.

```

<bpmn-graph> ::= "digraph" <model-name> "{" <stmt-list> "}"
<model-name> ::= <string>
<stmt-list> ::= <stmt> [";"] [<stmt-list>]
<stmt> ::= <node-stmt> | <edge-stmt>
<node-stmt> ::= <node-id> "[" {<event-attr-list> |
                                <activity-attr-list> |
                                <gateway-attr-list>} "]"
<activity-attr-list> ::= <activity-label> "," <activity-shape> |
                          <activity-shape> ["," <activity-label>]
<activity-label> ::= "label=" <activity-label-value>
<activity-label-value> ::= <string>
<activity-shape> ::= "shape=box"
<event-attr-list> ::= <event-label> "," <event-shape> |
                      <event-shape> ["," <event-label>]
<event-label> ::= "label=" <event-label-value>
<event-label-value> ::= <string>
<event-shape> ::= "shape=circle"
<gateway-attr-list> ::= <gateway-label> "," <gateway-shape> |
                       <gateway-shape> ["," <gateway-label>]
<gateway-label> ::= "label=" <gateway-label-value>
<gateway-label-value> ::= <and-gateway-label> |
                          <xor-gateway-label> |
                          <or-gateway-label>
<gateway-shape> ::= "shape=diamond"
<and-gateway-label> ::= "+"
<xor-gateway-label> ::= "X"
<or-gateway-label> ::= "0"
<edge-stmt> ::= <node-id> <edgeRHS> [<edge-probability>]
<edgeRHS> ::= "->" <node-id> [<edgeRHS>]
<edge-probability> ::= "[" <label> <decimal-number> "]"
<node-id> ::= <unquoted-string> | <quoted-string>

```

In Figure 3, we have the DOT file of the BPMN model of Figure 2.

From this example, it is possible to see that the definition of a BPMN model in DOT is very simple. All the nodes (vertex) that appear in the BPMN model must be “declared” in the DOT file with a statement that defines the vertex type. On the case of the BP2SAN tool, only three types of nodes can be created:

```

digraph MachineShop {
  A  [shape=circle,label="Start"];
  F  [shape=circle,label="End"];

  B  [shape=box,label="Design the piece"];
  C  [shape=box,label="Fabricate small piece"];
  D  [shape=box,label="Fabricate large piece"];
  E  [shape=box,label="Pack the order"];

  G1 [shape=diamond,label=X];
  G2 [shape=diamond,label=X];

  A -> B -> G1;
  G1 -> C [label=0.80];
  C -> G2;
  G1 -> D [label=0.20];
  D -> G2 -> E -> F;
}

```

Figure 3: Example of BPMN Process diagram defined using DOT language.

- activity – represented by a node whose shape is “box”;
- event – represented by a node whose shape is “circle”;
- gateway – represented by a node whose shape is “diamond”. A gateway in turn has one of the following types:
 - parallel gateway (or AND-gateway) – indicated by the label “+”;
 - exclusive gateway (or XOR-gateway) – indicated by the label “X”;
 - inclusive gateway (or OR-gateway) – indicated by the label “O”.

An edge from a node n_1 to a node n_2 in the the BPMN model is represented in the DOT file by the statement $n_1 -> n_2$. A probability value must be associated with each output edge of an exclusive or inclusive gateway node. In the example of Figure 3, node G1 is a divergent inclusive gateway; for this reason, there is a probability value associated with each one of its output edges: $G1 -> C$ [label=0.80] and $G1 -> D$ [label=0.20].

2.3.2 Resources Declaration

We define *resource* as something required to accomplish an activity of the business process model. It can be either a physical entity (as processors, memory, printers, human beings, organizations, etc.) or a virtual entity (as libraries, web services, databases, etc.).

For BP2SAN, a *resource* is a quadruple

$$R = [resource\ ID, quantity, work\ capacity, access\ discipline]$$

where:

- *resource ID* is an unique identifier that specifies the type of the resource being modeled;

- *quantity* is a positive integer number expressing how many units of the resource are available to be accessed;
- *work capacity* is a positive decimal number defining the average quantity of work that one unit of the resource is able to process per unit of time;
- *access discipline* is a strategy that determines how the activities are assigned to the resource.

BP2SAN accepts two kinds of access disciplines – *random choice* and *time sharing*. They are described in the following:

- *random choice* – an instance of activity will be randomly selected between the others waiting to access the resource;
- *time sharing* – the using time of the resource will be equally divided between all the instances of activities requesting access.

The resources declaration is a text file containing a resource list whose declaration respects the following syntax (specified in Backus Normal Form):

```

<resource-list> ::= <resource> [";"] [<resource-list>]
<resource> ::= "[" <definition> ["," <description>] "]"
<definition> ::= <id> "," <quantity> "," <work-capacity> "," <type>
<description> ::= <quoted-string>
<id> ::= <string>
<quantity> ::= <integer-number>
<work_capacity> ::= <integer-number> | <decimal-number>
<type> ::= "random_choice" | "time_sharing"

```

Figure 4 shows an example of the content of a resource declaration file.

```

[Designer, 4, 0.125, random_choice];
[CNC1, 2, 6.0, random_choice, "CNC Machine for small metal pieces"];
[CNC2, 1, 2.0, random_choice, "CNC Machine for large metal pieces"];
[Painter, 1, 10.0, time_sharing, "Painting machine"]
[Packer, 1, 1.0, random_choice]

```

Figure 4: Example of resources declaration.

2.3.3 Requirements Declaration

An activity of a business process may require the access of one or more resources to be performed. A resource may be required by one or more activities. As instances of activities (of either a same type or different types) can be executed parallelly in business process management systems, resources can be concurrently accessed.

The execution time of an activity is related to its resources requirements and may vary with the workload of the system. This is why it is very important to consider the resource requirements of the business process when its SAN model is generated.

A *single resource requirement (SRR)* of an activity can be expressed by a pair [*resource id, quantity of work*] where:

- *resource id* is the identifier of the resource type required by the activity;
- *quantity of work* is a positive decimal number that determines how much units of work the activity will require of the resource.

The *resources requirements (RR)* of an activity can be described by an expression created using SRRs and two logical operators for composition: AND or OR. In a RR expression, the AND operator has a higher precedence than OR; parentheses may be used to force the order of operations.

BP2SAN assumes that, when an activity requires more than one resource, it only will be executed when *all* its required resources are available. In an analogous way, the used resources will be released all together, at the end of the execution of the activity.

A valid requirements declaration file for BP2SAN must respects the following syntax:

```

<requirements-set> ::= <activity-requirements> [";"] [<requirements-set>]
<activity-requirements> ::= "{" <activity-id> "," <requirements-expression> }"
<requirements-expression> ::= <disjunctive-expression>
<disjunctive-expression> ::= <conjunctive-expression>
                               [<or> <disjunctive-expression>]
<conjunctive-expression> ::= <unary-expression> [<and> <unary-expression>]
<unary-expression> ::= <single-requirement> | "(" <requirement-expression> ")"
<single-requirement> ::= "[" <resource-id> "," <required-work> "]"
<activity-id> ::= <string>
<resource-id> ::= <string>
<required-work> ::= <integer-number> | <decimal-number>

```

Figure 5 shows an example of the content of a requirements file.

```

{B, [Designer,0.5] AND [Designer,0.5]};
{C, ([CNC1,60] OR [CNC2,30]) AND [Painter,60]};
{D, [CNC2,20] AND [Painter,40]};
{E, [Packer,1]};

```

Figure 5: Example of resource requirements declaration.

2.4 Output Files

Given a valid model in BPMN, the BP2SAN generates one or more SAN that model the same behavior of the business process model provided as input.

Each SAN models generated by the tool are stored in two formats:

- a text file with extension `.san`, with the textual representation of SAN models used as input by the PEPS tool. The PEPS tool is able to numerically solve the SAN model and provide performance indicators extracted from it;
- a text file with extension `.dot`, with the automata of the model represented in the DOT language. With this DOT file it is possible to obtain a graphical view of the SAN model using other visualization softwares, such as Graphviz (<http://www.graphviz.org>). It is important to mention that the DOT file does not include all the information of the SAN model; the rates associated to the events can not appear in the dot file.

3 Extracting Performance Indicators from the Generated SAN Models

We can extract performance indicators from a SAN model by defining numerical functions over the global state space of the system and integrating them with the stationary probability distribution of the model.

Using a SAN solver such as the PEPS tool, we are able to obtain the stationary probability distribution of the model (i.e., the probability of each state of the system in the long-run behavior).

The BP2SAN tool include at the end of the generated SAN files (extension `.san`), in the section of the file called *results*, some performance indicators defined in terms of functions in the format required by PEPS. The PEPS tool returns the values for the indicators defined in the section *results* together with the stationary probability distribution, after solving the SAN model. The performance indicators automatically included by BP2SAN are:

- utilization of resource (defined for each resource of the process model) – gives the percentage of the time in which the resource is in use;
- resource throughput (defined for each resource of the process model) – gives the average number of units of work processed by the resource per unit of time;
- average number in use (defined for each resource of the process model) – gives the average number of units of the resource simultaneously in use;
- activity throughput (defined for each activity of the process model) – gives the average number of executions of the activity per unit of time;
- probability of the initial state – this probability can be used to calculate the service time of the business process (i.e., the average time required for the complete execution of an instance).

Consider the probability of the initial state as π_{s_0} . Knowing that

$$\pi_{s_0} = \frac{\text{the time between order arrivals}}{\text{the time between order arrivals} + \text{the service time}}$$

and that the time between order arrivals is $\frac{1}{\text{rate of } e_s}$ (where e_s is the start event of the business process), the service time is given by the formula:

$$\frac{1 - \pi_{s_0}}{\text{rate of } e_s \times \pi_{s_0}}.$$

For more details about how to use PEPS to solve SAN models and extract other performance indicators, please refer to the user guide of the tool: <http://www-id.imag.fr/Logiciels/peps>.

References

- [1] Kelly Rosa Braghetto, Joao Eduardo Ferreira, and Jean-Marc Vincent. From Business Process Model and Notation to Stochastic Automata Network. Technical Report (Reference Number: RT-MAC-2011-03), University of São Paulo, March 2011. [Available at: [http://www.ime.usp.br/~kellyrb/files/fromBPMNtoSAN\(technical_report\).pdf](http://www.ime.usp.br/~kellyrb/files/fromBPMNtoSAN(technical_report).pdf); accessed March-2011].
- [2] Leonardo Brenner, Paulo Fernandes, Brigitte Plateau, and Ihab Sbeity. PEPS2007 – stochastic automata networks software tool. In *Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the*, pages 163–164, 2007.
- [3] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Softw. Pract. Exper.*, 30:1203–1233, September 2000.
- [4] OMG. Business process model and notation (BPMN), version 2.0, 2010.
- [5] Brigitte Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *SIGMETRICS PER*, 13(2):147–154, 1985.
- [6] Brigitte Plateau and Karim Atif. Stochastic automata network for modeling parallel systems. *Software Engineering, IEEE Transactions on*, 17(10):1093–1108, 1991.