Deployment of the BPMN Verifier: (User's Guide)

Carlos Molina–Jimenez¹ and Wenzhong Sun $(Jim)^2$

¹ School of Computing Science, Newcastle University, UK, carlos.molina@ncl.ac.uk
² Graduated MSc student, School of Computing Science, Newcastle University, UK, jim.sun1983@hotmail.com

Abstract. This document is a walkthrough description of the deployment of the BPMN verifier (version 1.1) that can be used to verify the logical correctness of choreography diagrams written in BPMN (Business Process Management Notation).

In brief, the BPMN verifier is a java software tool that can mechanically build a PROMELA model from a BPMN 2.0 choreography diagram. Such a model can them be extended with LTL correctness requirements (that the designer selects from a template repository) and presented to the SPIN model checker to verify weather the LTLs are satisfied.

This document is aimed at potential users interested in downloading the BPMN verifier (for example, from the GitHub repository), deploying it locally and trying it by means of provided running examples.

It covers the deployment on Windows, Mac and Linux platforms.

1 Introduction

The BPMN verifier is a software tool implemented in Java, for assisting choreography designers in the verification of choreography diagrams written in BPMN (Business Process Management Notation).

This document discusses the latest version version 1.1 of the BPMN verifier which was implemented by Wenzhong Sun (Jim) as a research collaboration with the School of Computing Science of Newcastle University, UK. Version 1.0 was implemented by Jim as part of his as part of his MSc Dissertation completed at Newcastle. The document, discusses implementation features and is available in pdf from [1].

A conceptual view of the BPMNverifier is shown in Fig. 1.



Fig. 1. Functionality of the BPMN verifier.

The tool consists of four main components:

BPMN editor

 $\mathbf{2}$

- BPMN2PROMELA translator
- LTL manager
- SPIN modelchecker

1.1 BPMN Editor

BPMN is a public standard language for modelling business processes and is maintained by the OMG [2]. It covers process models at different levels of abstraction, including choreography diagrams.

Graphically, a BPMN choreography diagram consists of several components including circles, boxes, diamonds and lines, that represent, respectively, events, activities, gateways and execution flows. These diagrams can be created by designers with the help of BPMN editors such as the one shown in the figure. At a lower level, a BPMN choreography diagram is a conventional XML file that complies (or should) with the BPMN 2.0 standard.

1.2 BPMN2PROMELA Translator

BPMN2PROMELA is a software tool that we have implemented in java and automatically translates BPMN 2.0 compliant choreography diagrams to PROMELA models that can be verified by the SPIN model checker.

PROMELA models produced by the BPMN2PROMELA translator can be presented to the SPIN model checker, for verification of general (application independent) safety and liveness properties (for example, absense of deadlocks and non-progress cycles) that SPIN verifies by default. However, application dependent correctness properties (for example, *purchase request is eventually followed by response*) need to be explicitly included in the PROMELA model (discussed below). SPIN accepts correctness properties specified as LTL (Linear Temporal Logics) formula. LTL was originally suggested by Amir Pnueli [3] as a formalism for reasoning about program correctness.

1.3 LTL Manager

The *LTL manager* shown in the figure is a graphical interface that can help choreography designers include LTL correctness properties in the PROMELA model produced by the BPMN2PROMELA translator. It was implemented by us in Java and is a core component of the BPMN verifier. It allows designers to select LTL correctness properties from a repository of typical LTL templates, instantiate them with operations that appear if the choreography diagram under analysis and include them in the PROMELA model produced by the BPMN2PROMELA translator .

1.4 SPIN Model Checker

SPIN one of the most mature and well documented model checkers. It was written by Gerald Holzmann [4] and publicly available for Windows, Mac and Linux [5]. SPIN can be used for mechanically verifying whether a given PROMELA model presented as input satisfies or violates a set of correctness properties.

2 Distribution Files

The source code distribution of the BPMN verifier includes:

- local java classes implemented by us.
- ancillary jar files available from the Internet.
- bpmn choreography examples that can be used as start up examples.

3 Ancillary Software

The user is expected to have the following software installed on his computer.

- Eclipse: the recommended way of launching the BPMN2PROMELA translator is as a Java application from within the eclipse platform. The current version of BPMN2PROMELA is known to operate correctly in eclipse Indigo which can be downloaded for free from eclipse home page [6].
- Spin model checker: Spin can be downloaded for free from its home page [5]. In our experiments we used the latest version of SPIN, namely Version 6.1.0-4 May 2011.
- BPMN 2.0 compliant editor: In our experiments, we have used choreographies produced by the Eclipse BPMN2 modeler that is bundled with the Savara Eclipse tooling and freely available from its home page [7]. The BPMN2 Modeler is part of JBoss Savara project [8].
- SQL Data Base: The BPMN verifier needs a data base for permanently storing LTL templates. Thus the current version expects access to a MySQL data base which can be freely dowloaded from MySQL home page [9].

4 Download the BPMN Verifier

A zip file of the source code is available from our home web page [10]. Yet we strongly recommend to download it from GitHub public revision control system [11] as this copy is guaranteed to include the very latest updates.

4.1 Deployment of the BPMN Verifier from GitHub

To install the BPMN verifier on your local computer from GitHub, go through the following procedure:

- 1. Start eclipse.
- 2. Click on Help->Install New Software.
- Copy and paste the link https://github.com/nclcmj/bpmn2promela into the Work with: box.

5 Deployment of the BPNM Verifier from the zip File

If for some reason you have decided to download the zip file of the BPMN verifier from our home page, go through the following step to deploy it in your local computer.³

- 1. **Store the BPMNverifier.zip:** Save the BPMNverifier.zip file in a folder of your local computer.
- 2. Start eclipse: Start eclipse and switch to the workspace of your choice. For example /Users/Bob/eclipse/workspace.
- 3. Create a Java project: Right click on the blank area under *Package Explorer* view. Point the mouse on the *New* menu and click *Java Project* on sub-menu when it pops up. See Fig. 2.

Pac	kage Explorer 🔀	E 🔄 🗊	~ -	
	New	۲	ß	Java Project
	Show In	Alt+Shift+W ►	2	Project
	Copy Copy Qualified Name	Ctrl+C	₩ ©	Package Class
	Paste	Ctrl+V Delete	C C	Interface Enum

Fig. 2. Create a new java project.

When the New Java Project dialog appears, assign a name the project (e.g. BPMN2Promela or BPMNverifier). In the Use an execution environment

³ Ignore this section if you have (or are planning to) downloaded it from GitHub.

JRE: box select $javaSE{-}1.6$ or the latest available version of JRE. Finally, press the Finish button to complete the project creation. See Fig. 3.

Project name: BPMN2Promela		
Vse default location		
Location: H:\myWorkSpace\BPMN2Pro	omela	Browse
JRE		
Use an execution environment JRE:	JavaSE-1.6	-
O Use a project specific JRE:	jdk17	
O Use default JRE (currently 'jdk17')		Configure JREs
Project layout Use project folder as root for source 	es and class files	
Project layout O Use project folder as root for source O Create separate folders for sources a	es and class files and class files <u>Cc</u>	onfigure default
Project layout O Use project folder as root for source O Create separate folders for sources a Working sets O Add project to working sets	is and class files and class files <u>Cc</u>	onfigure default
Project layout © Use project folder as root for sources © Create separate folders for sources a Working sets □ Add project to working sets Working sets:	es and class files and class files <u>Cc</u>	Select

 ${\bf Fig.~3.}$ Java project created.

4. **Import BPMNverifier.zip:** To import the files included in the *BPMN-verifier.zip* files to populate the *BPMNverifier java project* created above, do as follows. Right click on *BPMNverifier* and select *Import* on the menu, as shown in Fig. 4.



Fig. 4. Import BPMNverifier zipped files into the BPMNverifier project.

Select General-¿Archive File from the options (see Fig. 5) and press the Next button.

Mark Import	
Select Import resources from an archive file into an existing project.	Ľ
Select an import source:	
✓ ➢ General	

 ${\bf Fig.}\,{\bf 5.}$ Select archive file and press Next button.

Press the *Browse* button and locate your BPMNverifier.zip file. Finally, as shown in Fig. 6, press the *Finish* complete this step.

BPMN verifier 7

Import	
Archive file Import the contents of an archive file in zip or tar format from the local file system.	Ę,
From archive file: C:\Users\B1037041\Desktop\BPMN2PromelaV1.zip	Browse
Filter Types Select All	
Into folder: BPMN2Promela	Browse
Cverwrite existing resources without warning	
Kext > Finish	

Fig. 6. Import of BPMNverifier.zip completed.

5. **Build Path:** As shown in Fig. 7, eclipse signals some errors on the newly populated BPMNverifier project (a red box with an X on the project folder).



Fig. 7. Content of the project.

To correct the error, we proceed to import some required jar files. To do this, right click the project folder. Select the **Build Path** menu and select **Configure Build Path** from the submenu. See Fig. 8.

BPMN2Promela			_
🖶 src	New	+	
BPMN	Go Into		
generato			
generato	Open in New Window		
generato	Open Type Hierarchy	F4	
generato	Show In	Alt+Shift+W ►	
b denerato	-	01.0	
b # generato	Сору	Ctn+C	
p denerato	Copy Qualified Name		
🛛 🖶 generato 💼	Paste	Ctrl+V	
🛛 🌐 generato X	Delete	Delete	
🖻 🌐 generato 🔬	Remove from Context	Ctrl+Alt+Shift+Down	-
👂 🌐 generato 🎽	D. H.D. H	Carr Aler Shire Down	9. 11.1.0
🛛 🔠 simulato	Build Path	•	A Link Source
b 🏭 simulato	Source	Alt+Shift+S ►	New Source Folder
b 🖶 simulato	Refactor	Alt+Shift+T ►	(# Use as Source Folder
b # simulato	Import		Add External Archiver
b ## simulato P#	million		Muu External Archives
	-		

Fig. 8. Build class path for the project.

When the **Properties** dialog appears, press **Add Jars...** to select jar files as shown in Fig. 9.

type filter text	Java Build Path	
Resource BPMN2 Builders Choreography Java Build Path Java Code Style Java Compiler Java Editor Javadoc Location Project Facets Project References Run/Debug Settings Task Repository Task Tags TPTP JUnit Validation WikiText	Source Projects A Libraries Order and Export JARs and class folders on the build path:	Add JARs Add External JARs Add Variable Add Library Add Class Folder Add External Class Folder Edit Remove Migrate JAR File
0		OK Can

 ${\bf Fig.~9.}$ Add jars dependencies to the project.

From the **Jar Selection** menu, select all jar files shown in the **Referenced Libraries** folder (see Fig. 10) and press OK to complete this step.

9



Fig. 10. Select jar dependencies.

6. Refresh project: All the needed Referenced Libraries (jar files) are added into the BPMNverifier project. Thus the error should have disappeared by now. If the the red box with the X is still displayed, right click on the project folder and click on **Refresh** from the menu that appears. The BPMNverifier folder should look as shown in Fig. 11.



Fig. 11. BPMNverifier is ready for execution.

7. All the eclipse related files are now in the right place.

6 BPMN Verifier Configuration

The *config.properties* file located within the *src* (see Fig. 12) folder provides pre–run configuration options.

	Java – Eclipse – /Users/ncmj2/eclipse/workspace
] 📬 • 🔛 🗟 🗁] 🏇 • 💽 • 🂁 • 🍕 •] 🖶 🞯 •] 🍕 • [🔌 🦚, 😑 🐁] 🛱] 💯 🗀 🖋 •] 한 • 집 • 🏷 수 • 수 •] 🗧
📕 Package Explorer 😫 📄 🗢 🗖 🛛	
▼ 🕮 src	2
generator.jim.beans	
generator.jim.composite	
generator.jim.convert	
🕨 🌐 generator.jim.entites	
generator.jim.ltl.creator	
🕨 🌐 generator.jim.meta	
generator.jim.sequence.creator	
generator.jim.test	
generator.jim.tools	
generator.jim.translator.command.impl	
generator.jim.translator.command.interfac	
generator.jim.translator.factory.impl	
generator.jim.translator.factory.interfaces	
generator.jim.translator.visitor.interfaces	
ncl.b1037041.bpmn.choreographies	
ncl.b1037041.dao	
ncl.b1037041.db.tool	
ncl.b1037041.exception	
🔻 🔠 ncl.b1037041.gui	
▶ 🔊 BPMN2PROMELAWindow_ori.java	
► 🔊 BPMN2PROMELAWindow.java	
ncl.b1037041.gui.file	
ncl.b1037041.gui.help	
ncl.b1037041.gui.ltl	
ncl.b1037041.gui.parts	
ncl.b1037041.gui.tools	🔝 Problems 🏾 @ Javadoc 🔯 Declaration 🖾 Console 🛛 📝 Search 📚 Forge Console
ncl.b1037041.LTL.entites	<terminated> BPMN2PROMELAWindow [Java Application] /System/Library/Java/JavaVirtualMachin</terminated>
ncl.b1037041.manager	
simulator.jim.choreographies.entites	
Image: Simulator.jim.choreographies.interfaces	
simulator.jim.path	
Simulator.jim.tools	
iii config.properties	X.

Fig. 12. Configuration and start files of the BPMN verifier.

- buffer.size: This option defines the size of channel buffer. Its value should be greater or equals than 0.
- message.type: This option defines the type of predefined messages which can be stored in channels. There are five basic types in PROMELA: bit, bool, byte, short, int. The value should be chosen from one of them.
- *file.location:* This option specifies where to store these automatically generated files. e.g. D://promelaFile.
- file.suffix: It specifies the suffix of these PROMELA files.
- comment.*: Some comment in the content of PROMELA file.
- tranlator.type: This option specifies the form of generated PROMELA code. Currently, it supports "decentralized" and "centralized" forms. We will discuss it in detail in next later.
- BPMNName: There is a block which named Examples in the file. It lists all choreography diagrams in BPMN folder. All there diagrams are created by SAVARA tool and they are the source of the project. We can choose which diagram will be used by simply marking out it in the configure file.
- *init.type:* Let us choose the method of initialization.
- generation.model: The PROMELA model can be generated optional.
- generation.sequences: The sequences can be generated optional.
- *generation.sequences.format:* We can choose either plain text or XML format of the sequences.

- generation.LTL: The never block can be generated optional.
- generation.LTL.formula: The formula we used to verify a PROMELA model.

7 Database Configuration

Once the MySQL Server 5.X. is deployed you need to create a database for the template repository of Fig. 1 and initialise it with five tables. The simplest way of doing is by executing the *bpmn2promela20130109.sql* script, which is provided with the BPMN verifier bundle. Alternatively, you can manually create the database and initialise the five tables as follows:

1. **b2p_users:** maintains the users' (administrator and ordinary users) information. You need to create this table and initialise it shown in Fig. 13.

ld 🔺	usemame	password	user_type
1	admin	admin	0
2	carlos	carlos	1
3	ellis	ellis	1

Fig. 13. Users table.

The fields are defined as follows:

- Id a unique identification number of a user (for example, a choreography designer) of the BPMN verifier.
- **Username** The name of a user name (for example, the name of a choreography designer), such as *admin*, *carlos* and *ellis*.
- **Password** user's password. For simplicity, we do not use encrypted password in this version of the BPMN verifier; consequently, the password fields are shown in clear text in Fig. 13.
- **User_type** the type of user, which is 0 (for user with administration privileges) or 1 (for ordinary users).
- 2. **bpmn_choreography table:** This table stores the information about each BPMN file. The fields, which are populated by the BPMN verifier, at run time, are defined as follows:

Id Unique identification that identifies a BPMN choreography file.

file_path The path of the BPMN choreography file.

 $\mathbf{image_path}$ The path of the image file of the choreography.

3. **bpmn_choreography_message_info** stores information about messages (for example, *BuyReq, BuyPay*, etc.) involved in each choreography. The fields, which are populated by the BPMN verifier, at run time, are defined as follows:

Id Unique identification

choreography_id The choreography that the message belongs to.
message The message (like BuyReq) that is delivered in the execution of a
task.

ltl_symbol a character used for representing a message in an ltl formula.bool_message a Boolean variable, like *BuyReqRecv*, involved in an LTL formula.

4. **ltl_formula_definition:** This table maintains the templates of LTL formulae. The fields, which are populated by the BPMN verifier, at run time, are defined as follows:

Id Unique identification.

 $\label{eq:Description} \begin{array}{l} \text{Description} \ \text{The statement that describe the formula in natural language.} \\ \text{For example, always $@V0@ is eventually followed by either $@V1@ or $@V2@ but not both.} \end{array}$

Formula: the formula that written in LTL notation. For example, !([](@V0@ -><>((@V1@ && !@V2@) || (@V2@ && !@V1@)))).

Nickname A short name for the formula.

5. **ltl_formula_instance:** This table stores the information about LTL formulae related to choreographies. The fields, which are populated by the BPMN verifier, at run time, are defined as follows:

Id Unique identification

- **choreography_id** The BPMN choreography, for example *BuyerStore.bpmn*, that the LTL formula is related to.
- definition_id the LTL template used for producing the LTL formula.
- **specific_description** A description of an LTL template in natural language.

specific_formula An LTL template in LTL syntax.

is_setup The parameterised LTL. For example, *iča*.

7.1 Database Connection

The current implementation of the GUI of BPMN verifer, does not offer means for configuring the connection to the database, consequently, you need to do in directly in the Java class.

- 1. The actual connection to the database takes place in *line 11r* of the *ncl.b1037041.db.tool.-DataBaseUtil.java* file.
- 2. You need to customise the username and password.

jdbc:mysql://127.0.0.1/bpmn2promela?user=root&password= &useUnicode=true&characterEncoding=utf-8";

¹² Carlos Molina–Jimenez and Wenzhong Sun (Jim)

8 Launch of the BPMN Verifier

Once the choreography designer is satisfied about the configuration options, he can launch the BPMN verifier by running the *BPMN2PROMELAWindow.java* located within the *src* (see Fig. 12).

The launch procedure is as follows: Right click on the file; next click on *Run* As menu of eclipse and select Java Application. See Fig. 14

Run As	• 🗊	1 Java Application
Debug As	+	Run Configurations
Profile As	•	

Fig. 14. Run the BPMN verifier as a Java application.

The GUI menu of the BPMN verifier appears on the screen as shown in Fig. 15.



Fig. 15. Launch of the BPMN verifier.

9 Running Example

The BPMN verifier is now ready for use. We will demonstrate by means of an example. Let us assume that the choreography designer is presented with the following contract example and requested to produce a choreography model.

- 14 Carlos Molina–Jimenez and Wenzhong Sun (Jim)
- 1. The buyer can place a **buy request** with the store to buy an item.
- 2. The store is obliged to respond with either **buy confirmation** or **buy rejection** within 3 days of receiving the buy request.
 - (a) No response from the store within 3 days will be treated as a buy rejection.
- 3. The buyer can either **pay** or **cancel** the buy request within 7 days of receiving a confirmation.
 - (a) No response from the buyer within 7 days will be treated as a cancellation.

As a result, he produces the choreography diagram shown in Fig. 16.



Fig. 16. Choreography between a buyer and store.

10 Validation of BPMN Choreographies

Imagine that to be assured that his choreography model is logical sound, he decides to use the facilities offered by the BPMN verifier. The following steps explain the procedure.

10.1 Uploading of BPMN Diagram and Image

You need to upload the BPMN choreography file you wish to analyse to the BPMN verifier. Strictly speaking, the BPMN verifier only needs the BPMN choreography diagram (the xml file produced by your BPMN 2.0 compliant editor when you save your graphical choreography). However, for your own convinience, the BPMN verifier also asks you to upload an image (in gif, jpg, and jpeg formats). To perform this task follos these steps:

1. Click on the File and next on Upload ChoreFile. See Fig. 17.

i 😝 🙆 🔘			BPMN verifier
File LTL Mngmt	Tools	Help	
Upload ChoreFil Exit	e		

Fig. 17. Upload BPMN file and image.

2. Click on Select (right side of Upload...). See Fig. 18.

0	BPMN verifier
File LTL Mngmt Tools Help	
Upload Select Clear	
Upload ima Select Clear	Add

Fig. 18. Select BPMN file to upload.

- 3. Select the file that contains your BPMN diagram from your local disk, for example *BuyerSeller.bpmn*.
- 4. Click on Clear if you get the wrong file and click on Select again to repeat the procedure.
- 5. Click on Add to upload the selected file.
- 6. Click on Select (right side of Upload ima...). See Fig. 19.

00	BPMN verifier	
File LTL Mngmt Tools Help		
Upload Select Clear	orkspace/bpmn2promelagui/bpmnFile/BuyerStoreChore.bpmn	Add
Upload ima Select Clear		

Fig. 19. Select choreography image to upload.

- 7. Select the file that contains the image of your BPMN diagram from your local disk, for example *BuyerSeller.jpeg*.
- 8. Click on Clear if you get the wrong file and click on Select (right side of Upload ima...) again to repeat the procedure.
- 9. Click on Add to upload the selected file.

The name (BuyerStoreChore in this example) of the BPMN diagram will be shown on the left side of the pane.

- 16 Carlos Molina–Jimenez and Wenzhong Sun (Jim)
- 1. Click on rename if you wish to rename it and then on Confirm.
- 2. Click on Delete if you wish to delete both BPMN diagram and its image.
- 3. To see the image of your choreography, click on View Image. You can enlarge or scroll the frame. See Fig.20.



Fig. 20. A view of the BPMNchoreography image.

- 4. The N/M read-only parameter of the LTL(s): 0/0 provides information about the LTL formulae related to the current BPMN choreography. M (which equals 0 in this example) indicates the number of LTL templates that have been specified for the current BPMN diagram. N (which equals 0 in this example) indicates the number of LTL templates that have already been parameterised for the BPMN diagram (*BuyerStoreChore* in the example). For a newly uploaded BPMN diagram the desiger will always see LTL(s): 0/0.
- 5. Choreography and image storage: Once a BPMN diagram is loaded to the BPMN verifier, it remains stored in the verifier's data base and can be called for correctness verification against LTL formulae (explained later) several times until the designer deletes it.

10.2 Specification of LTL Templates

The BPMN choreography diagram can be validated againts a set of LTL properties composed from LTL templates available in the repository of LTL templates. LTL templates are LTL formulae that include abstract variables that follow the notation @Vi@ where *i* is an integer. Examples of valid, abstract variables are @V0@, @V1@, @V2@, @V3@, etc. Two examples of valid LTL templates are [] @V0@ and <> @V1@. Such LTL templates can be parameterised by choreography designers to convert them to specific LTL properties. LTL templates can be uploaded into the LTL template repository of BPMN verifier by some body with administrator privileges. The current version of the BPMN verifier does not check for administration privileges but we are planning to implement them so that only administrators of the LTL template repository can add and delete LTL templates.

1. Click on LTL Mngmt as seen in Fig. 21.

0 0	BPMN verifier	
File LTL Mngmt Tools	Help	
Uplc Add/Del LTLs Statistics	Clear	Add
Upload ima Select	Clear	Add
BuyerStoreChore	Rename View Image Delete	LTL(s): 0/0

Fig. 21. Specification of LTL templates

- 2. In the Description box, specify (keyboard type or copy and paste), in English or other human language, the LTL template you wish to upload into the data base. Variables in the LTL template are abstract and enclosed within a pair of @ characters.
- 3. In the Formula box specify the corresponding formal description of the template that includes the abstract variables enclosed in a pair of @ characters. You can type, copy and paste or use the the set of characters (!, &&, etc.) available just below the *formula* box). You also need to specify a name (in the *Nickna...* box) for the templete.
- 4. As an example, Fig. 22 shows the description in English and formal notation of a template that can be used for specifying eventually properties. It has been named *eventually* but other names chosen by the choreography designer can be used.

0 0	BPMN verifier
File LTL Mr	ngmt Tools Help
Descripti	eventually @V1@
Formula	▷@V1@ Nickna eventually 1 && II <>> U @V@

Fig. 22. Specification of the eventually LTL template.

5. Once the *Description*, the *Formula* and the *Nickna...* fields are completed, you need to press *Add LTL* to store the template in the repository. A view of the eventually template (together with other examples) after including it in the repository is shown in Fig. 23.

0	BPMN verifier	
File LTL M	igmt Tools Help	
Descripti		
Formula	Nickna 1 &&	Add LTL
Description	eventually @V1@	DELETE
Formula	<>@V1@	Nickna eventually
Description	always @V2@ is followed by either @V3@ or @V4@	DELETE
Formula	[](@V2@ -> <>(@V3@ @V4@))	Nickna implication
Description	always @V0@	DELETE
Formula	[@V0@	Nickna always

Fig. 23. Examples of LTL templates.

- 6. Click on the corresponding DELETE if you wish to delete an LTL template from the data base.
- 7. Statistics: The BPMN verifier offers a means for assessing the usage rate of the LTL stored in the data base. Click on LTL mng-> Statistics (see Fig. 24) to see a graphical view of LTL usage.

0	0		BPMN verifier	
File	LTL Mngmt	Tools	Help	
Des	Add/Del L Statistics	TLs		
F	ormula		Nickna I -> (I) (I) (II) (II) (III) (IIII) (IIII) (III) (III) (III) (III) (IIII) (IIII) (IIII) (IIII) (IIII) (IIII) (IIII) (IIII) (IIII) (IIIII) (IIIII) (IIIIIII) (IIIIIIIII) (IIIIIIIIIIII) (IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Add LTL

Fig. 24. Show statistics.

Currently, the this facility can only output bar charts. We are planning to include more output formats in the future.

10.3 PROMELA Model Generation

Click on Tools and then on ProModel Generation to gain access to the BPMN2PROMELA translator that is capable of producing PROMELA models from BPMN choreography diagrams. See Fig. 25.

0 🔴		BPMN verifier						
File LTL Mngmt	Tools Help							
Descripti	ProModel Generation LTL Vars Parameterisation ProModel Verification							
Formula		Nickna	Add LTL					
	! && -> <-> <> U @V@							

Fig. 25. Generation of PROMELA models.

1. On the left side you see all the BPMN choreography diagrams currently stored in the data base, **BuyerStore** in this example. See Fig. 26.

0 0	BPMN verifier	
File LTL Mngmt Tools	Help	
BuyerStoreChore	Generate View Image	LTL properties
		eventually @V1@

Fig. 26. Generation of PROMELA models.

- 2. Click on View Image if you wish to display a choreography image on the screen.
- 3. Click on Generate to generate the PROMELA model of the BPMN choreography diagram. The result of the generation is displayed on the eclipse console. As shown in Fig. 27, this PROMELA model does not include yet any LTL formula. In principle, this is a complete PROMELA model that can be presented to SPIN as it is for validation of safety and liveness properties that SPIN validates by dafault; unfortunatelly, this cannot be done with the current version of the BPMN verifier as it can validate only PROMELA models with LTL properties included. This is a programming flaw that needs correction.
- 4. On the right side of the screen you see (in read–only mode) all the LTL templates currently stored in the LTL repository. In this example, there is only one which is called *eventually*.
- 5. Use the drop down menu on the left side of the LTL template to select the number of instances of the LTL template you wish to associate to the PROMELA model (*BuyerStoreChore* in this example). In Fig. 27 this parameter is set to 2 which suggests that we are interested in validating two actual LTL properties derived from the *eventually* template, againts the PROMELA model. For instance the two instances of the *eventually* @V1@ LTL template might be parameterised with *BuyReq*, and *BuyPay*, respectively.
- 6. Click on Add. A message (2 instances have been added to the BuyerStore) will appear on the screen to indicate that two instances of the eventually LTL



Fig. 27. Generated PROMELA model.

property have been instantiated and associated to the PROMELA model. Notice that the generated PROMELA model displayed on the eclipse console is not altered after clicking on *Add*; the association between the LTL intances and the model takes place internally.

10.4 Parameterisation of LTL Templates

In is worth emphasising that the LTL properties created above are only templates that need instantiation:

1. Click on Tool and next on LTL Vars Parameterisation (see Fig. 28).



Fig. 28. Parameterisation of LTL templates.

- 2. On the left side, you are presented with a menu from where you can select a BPMN choreography diagram. Select one of them, for example, *BuyerStore*.
- 3. Click on Properties to see the LTL templates associated to this BPMN choreography diagram. See Fig. 29.

0 0	BPMN verifier	
File LTL Mngmt Tools	Help	
BuyerStoreChore	Properties View Image	
	eventually BuyConf	Add
		Delete
	eventually BuyConf 🛟	Add
		Delete

Fig. 29. Specification of actual LTL properties.

- 4. As one can see from Fig. 30, the LTL templates appear with means for instantiating their variables.
- 5. Select the variables you wish to include and click on Add.
- 6. Repeat the procedure to instantiate each LTL property.

0	0			BPMN verifier	
File	LTL Mngmt	Tools	Help		
Bu	verStoreChore		•	Properties View Image	
				eventually BuyReq :	Add
					Delete
					_
				eventually Very BuyPay	Add
				BuyRej BuyCanc BuyReg	Delete

Fig. 30. Parameterisation of eventually LTL instances.

7. In Fig. 30 the first LTL instance has been parameterised to *BuyReq*, whereas the second in being parameterised to *BuyPay*.

10.5 **PROMELA Model Verification**

To actually include each parameterised LTL instance as a *never claim* in the PROMELA model generated above, click on Tools (see Fig. 31).

0 0		BPMN verifier
File LTL Mngmt	Tools Help	
BuyerStoreChore	ProModel Generation LTL Vars Parameterisation	View Image (De)se Validate
	ProModel Verification	

Fig. 31. Verification of PROMELA model.

- 1. Click on *LTL* to see the LTL formulae associated to the model. The PROMELA model of this example is *BuyerStoreChore* that appears selected in the drop down menu. The drop down menu allows designer to select PROMELA models stored in the data base that have LTL formulae associated to them.
- 2. View Image allows you to see a graphical image of the BPMN choreography diagram.
- 3. Click on LTLs to see all the LTL formulae associated to the PROMELA model (see Fig. 32).
- 4. Tick the left side button to select (or deselect) each LTL property you wish to verify against the BPMN choreography diagram. Press the Details button (on right side) if you wish to see more information about the LTL property.

0	0	0							BPMN verifier			
File	e	LTL Mr	igmt	Tools	Help							
	Buy	erStore	Chore		\$	\subset	LTLs	\supset	View Image	🗌 (De)se	Validate	
(√	ventua	lly Buy	Req							Details Delete	Wait
(√	ventua	lly Buy	Pay							Details Delete	Wait

Fig. 32. Selection of LTL formulae.

- 5. Press Delete if you decide to leave it out of the validation. Alternatively, you can use the (De)se... button to select or deselect all the LTL properties in a single step.
- 6. Click on Validate to validate each LTL property against the PROMELA model. On the background, the BPMN verifier includes each LTL property into a copy of the PROMELA model generated before. Thus if you select two LTL properties for verification, the BPMN verifier will create two copies of the PROMELA model. Next it presents each PROMELA model with its corresponding LTL property included, to the SPIN model checker.
- 7. The results of the SPIN validation will be shown on the eclipse console and appear as a green or red message on the right side of each LTL formulae which indicate respectively, *satisfied* and *violated*. Fig. 33 shows the validation results produced from an LTL formula that satisfied the PROMELA model.

0	0					BPMN verifier			
File	LTL Mngmt	Tools	Help						
Bu	verStoreChore		:	(5	View Image	🗌 (De)se	Validate	
	eventually Buy	rReq						Details Delete	Passed

Fig. 33. Validation results.

8. A counter example (trail file) is created in the working folder when an LTL formulae is violated.

10.6 Help

This button is meant to take you to the documentation of the BPMN verifier. It is not fully functional yet. But is you would like to see the names of the people involved in this project, click on About BPMN verifier.

11 Licence

The BPMN verifier is released under the Apache License, Version 2.0[12], which is available from Apache's web pages. Also, you can find a *txt* copy from our home page [10].

12 Implementation History

Version	Date Contributors K		Key features
$\begin{array}{c} 1.1 \\ 1.0 \end{array}$	Dec 2012 Oct 2012	Wenzhong Sun (Jim) Wenzhong Sun (Jim)	Automatic LTL inclusion. Automatic BPMN to PROMELA translation.

Table 1. BPMN verifier-implementation history.

References

- Sun, W.: Design and implementation of a bpmn to promela translator. http://homepages.cs.ncl.ac.uk/carlos.molina/home.formal (visited in Nov 2012 2012) MSc Dissertation Project, Aug 2012.
- 2. OMG: Documents associated with business process model and notation (bpmn) version 2.0. http://www.omg.org/spec/BPMN/2.0 (Jan 2011)
- Pnueli, A.: The temporal logic of programs. In: Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS 1977). (1977) 46–57
- 4. Holzmann, G.J.: Design and Validation of Computer Protocols. Prentice Hall (1991)
- 5. SPIN: On-the-fly, ltl model checking with spin. http://spinroot.com (visited in Jul 2012 2012)
- 6. Foundation, T.E.: Eclipse. http://www.eclipse.org (2012)
- 7. Foundation, T.E.: Bpmn2 modeler. http://eclipse.org/bpmn2-modeler (2012)
- 8. Jboss: Savara and testable architecture. http://www.jboss.org/savara (2012)
- 9. Corporation, O.: Mysql data base. http://www.mysql.com (2012)
- 10. Molina-Jimenez, C.: Carlos molina-jimenez home page. http://homepages.cs.ncl.ac.uk/carlos.molina (2012)
- 11. Inc., G.: Github distributed version control system. https://github.com (2012)
- 12. Foundation, T.A.S.: Apache license version 2.0, january. http://www.apache.org/licenses (2004)