

UML2oWFN

Translating Models of the IBM WebSphere Business Modeler to Petri Nets
Version 2.10, 24 November 2009

Dirk Fahland

About this document:

This manual is for UML2oWFN, version 2.10, a tool to translate models of the IBM WebSphere Business Modeler to Petri nets, last updated 24 November 2009.

Copyright © 2008-2009 Martin Znamirovski, Dirk Fahland, and Niels Lohmann

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover Texts being “A GNU Manual,” and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License.”

(a) The FSF’s Back-Cover Text is: “You are free to copy and modify this GNU Manual. Buying copies from GNU Press supports the FSF in developing GNU and promoting software freedom.”

Table of Contents

1	Overview	1
1.1	Usage	1
1.2	Setup and Installation	1
1.3	Copyright	1
2	Invoking UML2oWFN	2
2.1	Command Line Options	2
3	Translating UML2 activity diagrams/IBM WebSphere Business Modeler process models to Petri nets	4
3.1	Generating Petri nets	4
3.2	Filtering Processes Based on Structural Properties	4
4	Soundness Analysis with UML2oWFN	5
4.1	Setup LoLA as Analysis Tool	5
4.2	Use Case: Standard Soundness Analysis	5
4.3	Generating Additional Information on the Processes	6
4.4	Making the Analysis More Efficient	6
4.5	Use case: weaker soundness analysis with dataflow-invariant process termination....	7
4.6	Use Case: Choosing Termination Semantics for Your Analysis	7
4.7	Case Study: Soundness Analysis of Industrial Business Process Models	8
4.8	Use Case: Anonymize the Processes	8
4.9	Use Case: Automated Translation, Analysis and Compilation of Results	8
4.10	Use Case: Verify Soundness via CTL properties	9
4.11	Use Case: Soundness Analysis Using Tools for Workflow-Nets	10
4.12	Use Case: Translate, Analyze, and Compile Reults of Several Large Process Suites	10
5	Version History	11
6	Anonymizing	15
6.1	Usage	15
6.2	Processing	15
6.2.1	Whitelists	15
6.2.1.1	Whitelist for Attributes	15
6.2.1.2	Whitelist for Attribute Values	16
6.2.1.3	Whitelist for Element Values	17
6.2.2	Standard Values	17
6.2.2.1	Standard Values for Element Values	18
6.2.3	Anonymization	18
6.2.3.1	Element Names Inside Attribute Values	18
Appendix A	GNU Free Documentation License	20

1 Overview

UML2oWFN translates the XML export of the IBM WebSphere Business Modeler into standard Petri net file formats. The translation can be targeted to a specific analysis like checking soundness, or for deriving contracts from business process specifications.

1.1 Usage

To translate a process library ‘input.xml’ into Petri nets and print render these nets as graphs, call

```
uml2owfn -i input.xml -f dot -o
```

The translation creates one output file ‘input.<catalog-name>__<process-name>.png’ per process of the library.

1.2 Setup and Installation

1. Go to <http://service-technology.org/files/uml2owfn> and download the latest version of UML2oWFN, say ‘uml2owfn-2.10.tar.gz’. To setup and compile UML2oWFN, change into your download directory and type

```
tar xzf uml2owfn-2.10.tar.gz
cd uml2owfn-2.10
./configure
make
```

After compilation, two binaries ‘src/uml2owfn’ and ‘src/anon/bom-anon’ are generated.¹ If you experience any compiler warnings, don’t panic: UML2oWFN contains some generated or third party code. If an error occurs, please send the output to dirk.fahland@service-technology.org.

2. To install the binary and the documentation, type

```
make install
```

You might need superuser permissions to do so.

If you need any further information, see file ‘INSTALL’ for detailed instructions.

1.3 Copyright

UML2oWFN was written by Martin Znamirovski (znamirov@informatik.hu-berlin.de), Dirk Fahland (dirk.fahland@service-technology.org), and Niels Lohmann (niels.lohmann@service-technology.org).

¹ On Microsoft Windows, these files will be called ‘uml2owfn.exe’ and ‘bom-anon.exe’.

2 Invoking UML2oWFN

2.1 Command Line Options

For an overview of the command line options, type ‘uml2owfn -h’ to see the following help screen:

```
UML2oWFN -- Translating UML Processes into Petri Net Models

Usage: ../src/uml2owfn [OPTION]

Options:
  -p, --parameter=PARAM      modify processing with given parameter
  -i, --input=FILE           read a BPEL process from FILE
  -o, --output[=NAME]       write output to file (NAME sets filename)
  -f, --format=FORMAT       create output of the given format
  -d, --debug=NUMBER        set a debug level (NUMBER=0..4 or "flex" or "bison")
  -r, --reduce=NUMBER       apply structural reduction level (NUMBER=0..5)
  -R, --rolecut             cuts away all swimlanes in a process that
                           do not contain a startnode
  -e, --roleexclusive=ROLE  cuts away all swimlanes in a process that
                           do not contain a startnode or don't have
                           exactly the exclusive role
  -c, --rolecontains=ROLE  cuts away all swimlanes in a process that
                           do not contain a startnode or are one of the
                           contained roles
  -s, --skip=PROCESS-PROP  skip processes in the translation that have the given
                           structural property
  -h, --help                print this help list and exit
  -v, --version             print program version and exit

PARAMETER is one of the following (multiple parameters permitted):
  filter                   filter out infeasible processes from the library
                           equivalent to: '-s empty -s overlappingPins'
  keeppins                keep unconnected pins
  log                     write a log file for the translation
  taskfile                write analysis task to a separate file
  anon                    anonymize the process output
  ctl                     generate CTL model checking properties for analysis
                           (if applicable)

FORMAT is one of the following (multiple formats permitted):
  lola, owfn, dot, tpn

TASK is one of the following (multiple parameters permitted):
  soundness               analyze for soundness
  deadlocks               check for deadlocks (except in the final state),
                           requires -a soundness
  safe                    analyze for safeness
  stop                    distinguish stop nodes from end nodes

the following TASK parameters determine the process termination semantics
that is used for the analysis, they are used mutually exclusive, all
parameters require '-a soundness'
  noData                  ignore state of data flow upon process termination,
  wfNet                   attempt to translate net into a workflow net,
  orJoin                  analyze net by assuming an implicit OR-join,

PROCESS-PROP is one of the following (multiple parameters permitted):
  empty                   the process contains no nodes
  multi                   the process contains edges with token production
                           or consumption different than 1
  multiNonMatching        the process contains edges where token production and
                           consumption do not match, e.g. produce 1, consume 2
  overlappingPins         the process has overlapping pinsets
```

trivialInterface the process has a trivial interface (at most one input place or output place), effective only with '-R'

Examples:

```
uml2owfn -i library.xml -f dot -o
uml2owfn -i library.xml -f lola -a soundness -o
```

3 Translating UML2 activity diagrams/IBM WebSphere Business Modeler process models to Petri nets

3.1 Generating Petri nets

Invoke the following command to translate a library 'lib.xml' to Petri nets

```
uml2owfn -i lib.xml -f FORMAT -o
```

This creates a set of Petri net files, 'lib.XXX_YYY.<ext>', where 'YYY' is the name of a process, and 'XXX' its containing catalog. The format of the file is specified the '-f FORMAT' option. If you have 'dot' installed, try '-f dot' to generate a graphical representation of each net in '.png' format.

3.2 Filtering Processes Based on Structural Properties

UML2oWFN allows to skip some processes in the translation based on structural properties of the process. The switch '-s PROP' controls which processes shall not be translated. 'PROP' can take the following values

- 'empty' - Do not translate empty processes.
- 'multi' - Do not translate processes that have edges with pin multiplicities different from min:1 and max:1. Please note that UML2oWFN only checks the presence of these values, but that the multiplicities are not translated to the Petri net level in the current version.
- 'multiNonMatching' - Do not translate processes that have edges with non-matching pin multiplicities. For instance an edge where the source pin may produce 2 tokens, but the target pin can only consume 1 token.
- 'overlapping' - Do not translate processes that have overlapping pinsets. UML2oWFN can handle overlapping pinsets; the resulting Petri net is not free-choice.

4 Soundness Analysis with UML2oWFN

You may use UML2oWFN together with the Petri net model checker LoLA for analyzing the soundness of business process models created in the IBM WebSphere Business Modeler. This chapter describes the available parameters of UML2oWFN for this use-case and how to set up LoLA for the analysis.

4.1 Setup LoLA as Analysis Tool

To prepare the soundness analysis you need three different binaries of the Low-Level Net Analyzer (LoLA, <http://www.service-technolorg.org/lola/>, source-code releases available at <http://service-technology.org/files/lola>) to be accessible from the command-line.

- ‘lola-bpm-liveprop1’ - LoLA compiled with LIVEPROP,
- ‘lola-bpm-state-predicate1’ - LoLA compiled with STATEPREDICATE, and
- optionally ‘lola-bpm-modelchecking1’ - LoLA compiled with MODELCHECKING;

each with CAPACITY 1 and CHECKCAPACITY. The analysis script files rely on these model checkers.

In ‘<uml2owfn>/lola/’, we provide the corresponding LoLA configuration files for these LoLA variants with this release. You can generate these LoLA executables as follows.

1. download ‘<http://service-technology.org/files/lola/lola.tar.gz>’
2. untar ‘lola.tar.gz’, configure
3. call ‘make lola-bpm-liveprop1’
4. copy the binary ‘src/lola-liveprop1’ to a directory of your path (e.g., ‘/usr/local/bin’) or call ‘make install’
5. repeat steps 3 and 4 with ‘make lola-bpm-statepredicate1’ and optionally ‘make lola-bpm-modelchecking1’

4.2 Use Case: Standard Soundness Analysis

This section describes how to achieve standard soundness analysis. Translate the library ‘lib.xml’ you want to verify with the following command-line arguments

```
uml2owfn -i lib.xml -f lola -a soundness -a safe -p taskfile -o
```

This creates a set of Petri net files, ‘lib.XXX__YYY.lola’, where ‘YYY’ is the name of a process, and ‘XXX’ its containing catalog. Each ‘lib.XXX__YYY.lola’ gets two process-specific analysis task files:

- The file ‘lib.XXX__YYY.lola.safe.task’ contains a state predicate that specifies that process ‘YYY’ is not safe (has a lock of synchronization).
- The file ‘lib.XXX__YYY.lola.fin.task’ contains a state predicate that specifies the valid final states of the process.

UML2oWFN also generates a script ‘check_lib.sh’ containing the appropriate LoLA command calls for each file.

```
lola-bpm-statepredicate1 lib.XXX__YYY.lola -a lib.XXX__YYY.lola.safe.task -P
lola-bpm-liveprop1 lib.XXX__YYY.lola -a lib.XXX__YYY.lola.fin.task -P
```

If the first call returns ‘predicate is not satisfiable’, then the process is safe. Otherwise LoLA returns a place which can hold more than one token. If the second call returns ‘predicate is live!’, then the process is deadlock free and livelock free and always reaches a terminal state. Calling

```
sh ./check_lib.sh
```

will make these checks for each process of the library.

4.3 Generating Additional Information on the Processes

The following options can be used to generate additional information about the processes.

- ‘-p log’ write a log ‘uml2owfn_lib.log’ in CSV (comma separated file) format containing information about each process. The column format is

```
file name;correct syntax;structural properties;
number of nodes;number of edges;
average number of incoming edges;average number of outgoing edges;
maximum number of incoming edges;maximum number of outgoing edges;
Petri net properties
```

The column ‘correct syntax’ holds ‘true’ if the input process had correct static semantics, and ‘false’ otherwise. The column ‘structural properties’ contains a list of characteristic properties of the process, namely, whether the process

1. is empty,
2. contains edges with pin multiplicities,
3. contains edges with non-matching pin multiplicities (produce 2 token, consume 1 token), and
4. whether the process contains overlapping pinsets.

The column ‘Petri net properties’ contains a list of structural properties of the Petri net, namely, whether the net

1. is not connected,
 2. has an insufficient interface,
 3. has an incorrect structure according to UML2 Activity Diagrams,
 4. is a free-choice Petri net,
 5. has workflow-net structure, and
 6. whether the translation did no preserve soundness of the process.
- ‘-d 2’ prints further information of the translation process to standard output, and statistical data on the size of each generated net to standard error. To store this data to a file during the translation, redirect the output to a file, e.g.

```
uml2owfn -i lib.xml [your-options-here] -d 2 > translate.log 2> translate-err.log
```

This option is particularly useful for debugging.

4.4 Making the Analysis More Efficient

UML2oWFN includes structural Petri net reduction techniques that can be applied to reduce the size of the Petri net while preserving the (un)soundness of the net. Use option ‘-r LEVEL’ to include net reduction in the translation. UML2oWFN applies the following reduction rules up to ‘LEVEL’ until no more rule can be applied:

1. merge series of places (in a variant that preserves the k-boundedness of the net)
2. –
3. merge parallel places
4. –
5. –
6. –

This may drastically reduce the size of the nets and hence improve model-checking efficiency in terms of memory consumption and runtime.

4.5 Use case: weaker soundness analysis with dataflow-invariant process termination

The soundness analysis invoked by parameters ‘`-a soundness -a safe`’ is rather strict as it requires that all data is available at the process output upon process termination. While a violation of this property is still considered an error, it might prevent a detection of more severe control-flow errors in the process. To this extend, the soundness analysis can be changed to ignore the dataflow at the process output. Use option ‘`-a noData`’ when translating the library to create slightly changed ‘`.lola`’ files with corresponding ‘`.task`’ files. The analysis itself is performed as described above.

```
uml2owfn -i lib.xml -f lola -a soundness -a safe -a noData -p taskfile -o
```

4.6 Use Case: Choosing Termination Semantics for Your Analysis

UML2oWFN provides four kinds of termination semantics that can be implemented upon translation a process library to Petri nets. You can choose the termination semantics by the "right" selection of parameters. We briefly explain each termination semantics and the corresponding parameter combinations:

1. The "standard" UML2 termination semantics synchronizes control-flow and data-flow upon termination by clearing the control-flow tokens from end nodes as soon as data-flow has terminated via one output pinset. This semantics prioritizes data-flow over control-flow. Choose

```
-a soundness
```

to use this termination semantics

2. The "workflow net" semantics synchronizes control-flow and data-flow in order to detect proper termination locally at a unique "omega" sink place of the net as in workflow Petri nets. The termination semantics assumes that the control-flow and data-flow terminate via an implicit OR-join of several terminal nodes. It computes an extension of the multi-terminal workflow net to a single terminal workflow net. This extension equivalently preserves soundness and unsoundness of the net. Choose

```
-a soundness -a wfNet
```

to use this termination semantics.

3. The "control-flow only" semantics ignores data-flow upon termination (by removing the corresponding parts from the process) and lets the control-flow terminate via an implicit OR-join. Choose

```
-a soundness -a noData
```

to use this termination semantics.

4. A variant of the UML2 termination semantics which assumes that control-flow and data-flow terminate via an implicit OR-join. Choose

```
-a soundness -a orJoin
```

to use this termination semantics.

These termination semantics should not be combined (i.e. it will produce unpredictable results). The analysis for the safeness of a process is independent of the chosen termination semantics and can always be included by adding

```
-a safe
```

to the parameters.

4.7 Case Study: Soundness Analysis of Industrial Business Process Models

We conducted an experimental case study with UML2oWFN and LoLA in which we checked soundness of over 700 industrial business processes. The original process models are available in XML format from <http://service-technology.org/soundness>. For this case study, we translated each original process library ‘lib.xml’ into Petri nets, by calling

```
uml2owfn -i lib.xml -f lola -a soundness -a safe -a orJoin -p taskfile -p filter -o
```

which creates one Petri net per process of the library assuming an OR-join termination of each process. Then we called

```
lola-bpm-liveprop1 lib.XXX__YYY.lola -a lib.XXX__YYY.lola.fin.task -P
lola-bpm-statepredicate1 lib.XXX__YYY.lola -a lib.XXX__YYY.lola.safe.task -P
```

for each generated Petri net to check soundness. See *Standard Soundness Analysis*, for details. The results of the case study have been presented at the Conference on Business Process Management 2009 in Ulm, Germany: Dirk Fahland, Cdric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Vlzer, and Karsten Wolf. *Instantaneous Soundness Checking of Industrial Business Process Models*. In BPM 2009, Ulm, Germany, volume 5701 of Lecture Notes in Computer Science, September 2009. Springer-Verlag.

4.8 Use Case: Anonymize the Processes

In case you want to generated anonymized processes models which can not be related back to the originals due to self-explaining names, use the option ‘-p anon’. This will enumerate the places and transitions of each process individually, starting with *p1* and *t1*. Any related files like the ‘.task’ files are consistent with the anonymized output.

4.9 Use Case: Automated Translation, Analysis and Compilation of Results

UML2oWFN comes with a makefile that eases analysis and compilation of the analysis results for process librares. The ‘scripts/Makefile’ the following make targets are available

- ‘translate’ translate library
- ‘translateweak’ translate library with weaker termination semantics
- ‘translatewf’ translate library into classical workflow nets
- ‘translateor’ translate library nets with OR-join on the controlflow
- ‘soundness’ soundness check (AGEF omega)
- ‘safeness’ safeness check
- ‘table’ collect results in table
- ‘clean’ remove analysis files
- ‘veryclean’ remove Petri nets

Copy ‘scripts/Makefile’ to the directory that contains your library ‘lib.xml’ and change the value of ‘LIBRARY’ in this ‘Makefile’ to ‘lib’ (stripping the extension ‘.xml’). Also make sure that ‘SED’ holds the file name of your sed (stream editor) binary, e.g. ‘sed’ or ‘gsed’.

The standard workflow for the first use case using the makefile would be:

```
make translate
make table
```

To run the second use case, execute ‘`make translateweak`’ instead of ‘`make translate`’.

The target ‘`table`’ prepares two different representation of the analysis results in ‘`results.csv`’ and ‘`results-comparison.csv`’. The target depends on ‘`soundness`’ and ‘`safeness`’, thus the table will always contain the analysis results of the most recent translation.

You can use ‘`REDUCTION-LEVEL`’ to set the level Petri net reduction rules that are applied upon translating the library. The calls

```
make translate LIBRARY=myLib REDUCTION-LEVEL=3
make table LIBRARY=myLib SED=sed
```

will translate library ‘`myLib.xml`’, apply reduction level 3, and create the result tables using the program ‘`sed`’.

The Makefile variable ‘`OUTPUT-FORMAT`’ allows you to set the output format of the translation of UML2oWFN, default value is ‘`lola`’. The targets ‘`full`’, ‘`table`’, ‘`safeness`’, ‘`soundness`’, ‘`clean`’, and ‘`veryclean`’ require the ‘`lola`’-format.

Use the parameter ‘`UML2OWFN-PARAM`’ for refining the translation with further command-line parameters of UML2oWFN if you need.

You can immediately try this Makefile with the following setup. Create the necessary LoLA binaries as described above and copy them to ‘`./lola/`’ of your UML2oWFN directory. Then execute the following commands in the root directory of UML2oWFN.

```
cd ./scripts/
make translateor LOLA_DIR=./lola/ UML2OWFN_DIR=./src/ \
  LIBRARY=./tests/soundness/TestSuite_soundness
make table LOLA_DIR=./lola/ UML2OWFN_DIR=./src/ \
  LIBRARY=./tests/soundness/TestSuite_soundness
```

This will translate and analyze the processes from our soundness test library in ‘`./tests/soundness/TestSuite_soundness.xml`’. The parameters ‘`LOLA_DIR`’ and ‘`UML2OWFN_DIR`’ point the Makefile to the just generated UML2oWFN and LoLA binaries.

4.10 Use Case: Verify Soundness via CTL properties

The standard translation of UML2oWFN generates a state-predicate that describes all valid terminal states. LoLA provides a dedicated algorithm to check whether this state is always reachable. This property can equivalently be transformed into a CTL property. Then standard CTL-model checking techniques are applicable.

To invoke the generation of CTL properties and corresponding analysis script files, use the parameter ‘`-p ctl`’. To verify whether the generated CTL formula holds for a given process, use LoLA with the model checking algorithms, e.g. call

```
lola-bpm-modelchecking1 lib.XXX_YYY.lola -a lib.XXX_YYY.lola.fin.task -P
```

This line is automatically included in the verification script ‘`check_lib.sh`’. To use the Using the ‘`./scripts/Makefile`’ to automate the compilation of results in this setting, you have to set two parameters:

```
make translate UML2OWFN-PARAM="-p ctl"
make table LOLA-LP=lola-bpm-modelchecking1
```

The first line causes the translation to generate CTL properties, the second line switches the model-checker for the soundness analysis to the LoLA variant with model checking algorithms (that has to be compiled as described above).

4.11 Use Case: Soundness Analysis Using Tools for Workflow-Nets

Aalst et al. have established the notion of a *workflow net* to model workflows. Workflow nets are a special class of Petri nets that have a unique start place ‘alpha’ that is initially marked, and a unique end place ‘omega’ that is marked when the workflow has finished. Further, every place and transition lies on a path from ‘alpha’ to ‘omega’. There are efficient soundness analysis algorithms for workflow nets.

In order to analyse processes of the IBM WebSphere Business Modeler with such tools, the generated nets have to provide a unique ‘alpha’ place, and a unique ‘omega’ place. An equivalent translation of an arbitrary IBM WebSphere Business Modeler process into a workflow net requires knowledge about the reachable states of the process. However, for a specific class of processes, this transformation can be done on the structure of the process alone.

The parameter ‘-a wfNet’ triggers the use of this structural translation, it must be used together with ‘-a soundness’ to become effective. Call

```
uml2owfn -i lib.xml -f tpn -a soundness -a wfNet -o
```

to translate each process of the library ‘lib.xml’ into a Woflan ‘.tpn’ file with workflow-net structure. Each file can be loaded into Woflan and analyzed for soundness.

The workflow-net translation introduces a unique ‘alpha’ place that is initially marked, like in all other soundness analysis use cases. It computes a completion logic of transitions and places that cover the reachable final markings of the net and the produce on a new unique ‘omega’ place that has no outgoing arcs.

The completion preserves soundness (and unsoundness) for free-choice Petri nets. Cases of non-free-choice nets can be identified in the translation log (‘-p log’).

Using the ‘./scripts/Makefile’, you can call

```
make translatewf
```

to automate the translation. The Makefile variable ‘OUTPUT-FORMAT’ allows you to set the output format of the translation of UML2oWFN, default value is ‘lo1a’.

4.12 Use Case: Translate, Analyze, and Compile Results of Several Large Process Suites

Invoking and compiling analysis results for several large process suites can be done automatically with the ‘./scripts/repository/Makefile’ and the corresponding script files. To use the file, create a "process repository" as follows. Create a new directory, e.g. ‘<analysis>’. In ‘<analysis>’ create a new sub-directory for each process library, e.g. ‘<analysis>/lib1’, ‘<analysis>/lib2’, ... and place the library files in the corresponding directories: ‘<analysis>/lib1/myLib1.xml’ etc. Then copy all files from ‘./scripts/repository/’ to ‘<analysis>’ and call

```
make table
```

which will automatically translate all process libraries for all corresponding analysis, perform the analysis and verification, collect the result of each library and analysis, and compiles result tables in ‘.html’ and ‘.tex’ format in ‘<analysis>/lib1’ etc. Call

```
make help
```

to learn about the other functionality provided by this script. Call

```
make cleanall
```

to clean all generated files, leaving only the original process libraries. The file ‘./scripts/repository/README’ provides further details.

5 Version History

The following list provides a detailed overview about the changes, features, and bugfixes to UML2oWFN as they have been included in this release.

Version 2.10

- Migrated UML2oWFN to the new Petri net API. The included PNAPI slightly deviates from the original API, see SVN repository for details.
- Dropped support for several file formats. The currently supported file formats are .lola, .owfn, .dot, and .tpn (Woflan)
- Changed the filenames of the LoLA binaries for soundness analysis to 'lola-bpm-liveprop1' and 'lola-bpm-statepredicate1'
- Role cutting functionality has been disabled.

Version 2.00

- Re-implementation of the compiler frontend.
- Local sub processes and loops are now treated as simple tasks in the parent process. Each subprocess and each loop is added as a process to the list of all output-processes.
- Processes can be cut according to the role information using new options
 - '--rolecut', cut away all roles that do not include a startnode
 - '--rolecontains=ROLE', additionally keep those roles
 - '--rolexclusively=ROLE', delete all roles, that are not covered by a startnode or by a containrole, that do not match the set of exclusive roles
- Added generation of FINALCONDITION for oWFN output; oWFN output now uses full node names instead of short node names.
- Bugfix: Updated generation of formula for safe states to consider post-places of transitions that lie on a cycle of the net as well.
- Modified semantics of workflow net termination semantics (parameter '-a wfNet') which now assumes that the net terminates with a global orJoin and extends the given net to a single terminal workflow net. The extension constructs a dead-path elimination logic from structural information. The extension preserves soundness if the net is free-choice.
- Made filtering of processes more precise according to the semantics implemented in the WebSphere tools, providing a dedicated filtering parameter for different process characteristics:
 - Translation may filter empty processes ('-s empty'),
 - processes with pin multiplicities ('-s multi'),
 - processes with non-matching pin multiplicities at edges (produce one token, consume two token, '-s multiNonMatching'),
 - processes with overlapping pinsets ('-s overlappingPins'), and
 - processes with trivial communication interfaces ('-s trivialInterface', only applicable if using '--rolecut').

The old parameter '-p filter' remains for backwards compatibility and implements '-s empty -s overlappingPins'.

- Extended the translation log file which now distinguishes properties of UML processes and of the Petri nets
- Added '--enable-debug' and '--disable-assert' to './configure' parameters to control the use of debug and assert code

- integrated BOM Anonymizer to UML2oWFN distribution (see directory ‘anon’)
- Added ‘./lola/Makefile’ to automatically create all LoLA binaries needed for soundness analysis. Prepared a sample translation and analysis scenario.

Version 1.11

- Added new ‘./scripts/repository/Makefile’ for an automated analysis and result compilation of several large process suites. See manual on usage.
- Fixed handling of paths for output files, output files can now be written to their own directories
- Changed the standard soundness verification procedures to the more efficient liveness property verification algorithm of LoLA. To use the old CTL model checking procedures use parameter ‘-p ctl’.
- Cleaned the autotools configuration files and added options to directly compile binaries for different platforms.

Version 1.10

- fixed a bug in handling filenames, UML2oWFN can now process files in other directories than the current working directory
- added parameter ‘-a orJoin’ to translate process for soundness analysis assuming an implicit OR-join on the control-flow
- added parameter ‘-a wfNet’ to translate processes into Aalst-workflow-nets, the translation assumes an AND-join on all end- and stop-nodes of the process
- added new analysis switch ‘-a removePinsets’ that relaxes the process termination semantics by requiring that only the control-flow terminates, but that data may still be pending at the process output, and added new ‘scripts/Makefile’ target ‘translateweak’ that uses the ‘-a removePinsets’ option,
- added parameter ‘-a safe’ to check for safeness of a net using LoLA with state predicates, this parameter can be used together with ‘-a soundness’
- added parameter ‘-p anon’ to anonymize the process output by enumerating places (p1, p2, ...) and transitions (t1, t2, ...) before writing to disk
- added log file output for translation, with parameter "-p log" a log file (name "uml2owfn_<inputfile>.log") with entries of the form "process name; translation successful;reason-for-fail;number of nodes, number of edges, average number of incoming edges, average number of outgoing edges" is written
- added checks for free-choice nets and for workflow-nets (each node is on a path from alpha to omega); this additional Petri net functionality is implemented in the class ExtendedWorkflowNet which inherits from PetriNet
- added check whether generated Petri net is empty, if this is the case, then the net is translated again without reduction rules, the name is extended with a ".unreduced" (if it is still empty, then the net is skipped: no net file, no analysis, not in script file)
- changed task file naming scheme to ‘net.lola.<tasktype>.task’
- worked on bug #12160: PNAPI: Petri net reduction rules do not preserve (un)-safeness, disallowing place/transition fusion if places/transitions are together in the preset/postset of another transition/place
- overrode the Petri net reduction method in the ExtendedWorkflowNet class, now implementing reduce_series_places (level 1) and reduce_identical_places (level 3)
- stored reduction level in the ‘scripts/Makefile’ in variable ‘REDUCTION-LEVEL’, so the reduction level can be set at the make call

- extended filtering criteria: empty processes are not translated
- fixed bug #12156: UML2oWFN: translation of overlapping pinsets fails, <https://gna.org/bugs/?12156>
- added documentation in ./doc/
- added tests to './tests/', run 'make check' to run the test suite

Version 1.02

- Changed output: The property for the soundness analysis is now appended to the net and written to the net file by default.
- Added parameter switch "-p taskfile" to re-enable the old functionality and to write a separate task file.
- The generated verification script considers all cases accordingly.
- Re-enabled the error routine of the parser. Parser errors are now printed to the console.
- Added null pointer checking in Block::linkInserts() and disabled filtering in the parser. Fixes Bug #12027, <https://gna.org/bugs/index.php?12027>
- Added wbim:humanTask to the lexer. Is parsed and represented as normal task.
- Added wbim:bulkResourceRequirement to the lexer. Is parsed as role requirement. This is feasible because role requirements are (currently) ignored in the process translation, fixes Bug #12031, <https://gna.org/bugs/?12031>
- Added wbim:individualResourceRequirement to the lexer. Is parsed as role requirement. This is feasible because role requirements are (currently) ignored in the process translation
- Extended parser to read atomic "callToProcess"-tags without children
- changed command line help output to provide correct and meaningful examples for command line parameters
- implemented Woflan file format (.tpn), see <https://gna.org/task/?6011>
- file format can be triggered with "-f tpn"
- Soundness checking now distinguishes checking for reachability of end states, checking for deadlocks, and checking for non-reachability of the final state (this will deprecate...)
- soundness analysis now appends "omega" places to each output pinset and transitions that remove tokens from endNodes and stopNodes once the "omega" place is marked
- reachability of end states:
 - adds live-locking transitions to each "omega" place
 - generates the formula $AG EF (\omega_1 > 1 \text{ OR } \dots \text{ OR } \omega_N > 0) \text{ AND } (p_1 = 0 \text{ AND } \dots \text{ AND } p_M = 0)$ with p_i being internal places of the process
 - corresponding task file and verification script is generated, verification requires a LoLA with MODELECHECKING, executable file name "lola"
 - command line call: `uml2owfn -i <lib.xml> -a soundness -f <format> -o`
- checking for deadlocks
 - adds live-locking transitions to each "omega" place
 - generated verification script requires a LoLA with DEADLOCKS, executable file name "lola_deadlock"
 - command line call: `uml2owfn -i <lib.xml> -a soundness -a deadlocks -f <format> -o`
- inversed parameter "pins" to "keeppins" (unconnected pins are now removed by default)
- All analysis tasks allow using Petri net reduction rules.

- merged functionality of rev.65-69 of the trunk, <http://svn.gna.org/viewcvs/service-tech?rev=69&view=rev>
<http://svn.gna.org/viewcvs/service-tech?rev=68&view=rev>
<http://svn.gna.org/viewcvs/service-tech?view=rev&rev=66>
<http://svn.gna.org/viewcvs/service-tech?view=rev&rev=65>
- dot-output colors nodes according to their origin in BOM
- translation from BOM to PN now implements the complete task pattern (bug in rev.69) and links process input pinsets and output pinsets
- added class BomProcess (bom-process.cc .h) as an internal representation of the process, the class currently only stores some sets of places and transitions. The sets are used when modifying the Petri net for soundness analysis
- changed parameters:
 - added analysis parameter [-a (soundness|stop)];
 - "-a soundness" is equivalent to -p soundness in earlier revisions,
 - by "-a stop", the translation for soundness distinguishes stop-nodes and end-nodes, with "-a stop" a state predicate is needed for checking soundness (corresponding lola files and scripts are generated), without "-a stop" a deadlock analysis is sufficient (corresponding lola files and scripts are generated)

The most recent change log is available at Rachel's website at <http://service-technology.org/files/uml2owfn/ChangeLog>.

6 Anonymizing

BOM-Anonymizer implements a complete whitelist-based stream filter that anonymizes attribute values, literal values, and documentation tags occurring in an XML input file without jeopardizing the syntactical correctness of the input.

6.1 Usage

To anonymize the file ‘input.xml’, call

```
bom-anon < input.xml > output.xml 2> mapping.txt
```

The anonymized version is written to ‘output.xml’. Furthermore, the mapping from explicit to anonymized strings (e.g., for debug purposes) is written to ‘mapping.txt’.

6.2 Processing

BOM-Anonymizer parses the input file and anonymizes any attribute values or literal values that are not listed in a whitelist. The syntax of the input file is not checked.

6.2.1 Whitelists

6.2.1.1 Whitelist for Attributes

The following attributes are not anonymized at all:

- a
- alpha
- atBeginning
- b
- beginingOn
- beta
- callSynchronously
- capacity
- counterIncrement
- currency
- dayOfWeek
- delta
- duration
- expMean
- finalCounter
- gamma
- initialCounter
- isConditionTestedFirst
- isConsumable
- isInclusive
- isInsert
- isObserveContinuously
- isOrdered

- `isReadOnly`
- `isRemove`
- `isSimpleDecision`
- `isStatic`
- `isUnique`
- `k`
- `lambda`
- `max`
- `maximum`
- `mean`
- `min`
- `minimum`
- `mode`
- `multipleInstancesSatisfyCondition`
- `noInstancesSatisfyCondition`
- `numberOfTimesToRepeat`
- `probability`
- `repetitionPeriod`
- `schemaVersion`
- `standardDeviation`
- `startTime`
- `time`
- `timeRequired`
- `timeUnit`
- `unit`
- `unit`
- `valueTypev`
- `weekNumber`
- `xi`
- `version`
- `encoding`
- `xmlns:wbim`

6.2.1.2 Whitelist for Attribute Values

The following values are not anonymized if they occur in any attribute—no matter whether the attribute is on the whitelist or not:

- `"Boolean"`
- `"Byte"`
- `"Date"`
- `"DateTime"`
- `"Double"`
- `"Duration"`
- `"Float"`

- "Integer"
- "Long"
- "Short"
- "String"
- "Time"
- "Communication Service"
- "Equipment"
- "Facility"
- "General Service"
- "Machine"
- "Tool"
- "Person"
- "Staff"
- "acres"
- "centimeters"
- "feet"
- "gallons (UK)"
- "gallons (US)"
- "grams"
- "hectares"
- "inches"
- "kilograms"
- "kilometers"
- "liters"
- "meters"
- "miles"
- "ounces"
- "pints (UK)"
- "pints (US)"
- "pounds"
- "units"
- "yards"
- "true"
- "false"

6.2.1.3 Whitelist for Element Values

The following elements are not anonymized:

- `<wbim:literalValue>` containing a value of type `xsd:duration`
- `<wbim:startTime>` containing a value of type `xsd:dateTime`
- `<wbim:probability>` (containing anything)

6.2.2 Standard Values

6.2.2.1 Standard Values for Element Values

- The `wbim:individualResourceRequirement` element and all its child elements are replaced by a minimal bogus entry.

```
<wbim:individualResourceRequirement individualResource="Person"
  name="Individual requirement:1" timeRequired="POYOMODTOHOMOS" />
```

- XML comments are replaced by `'...'`.

before anonymization:

```
<!-- This is an XML comment that might contain additional information. -->
```

after anonymization:

```
<!-- ... -->
```

- Values in the elements `<wbim:documentation>`, `<documentation>`, `<wbim:description>`, `<description>`, `<wbim:annotationText>`, or `<annotationText>` are replaced by `'...'`.
- Values in the elements `<wbim:literalValue>` or `<literalValue>` are replaced by `'0.0'`.
- Real numbers in any element are replaced by `'0.0'`.

before anonymization:

```
<someElement>42.0</someElement>
```

after anonymization:

```
<someElement>0.0</someElement>
```

6.2.3 Anonymization

Any other string that is not affected by a whitelist or predefined standard value is replaced by a consecutive number. The replacement is unique: all occurrences of a string are replaced by the same anonymized version. However, the original string cannot be reconstructed from the anonymized version.

Example before anonymization:

```
<wbim:catalog id="Prccss" name="Processes"/>
```

Example after anonymization:

```
<wbim:catalog id="s00000001" name="s00000002"/>
```

6.2.3.1 Element Names Inside Attribute Values

Attributes can contain element names. The XSD of BOM's input format states:

The name of the global element may be prefixed by the catalog ID followed by `'##'` to denote the name space for the element name.

Therefore, strings containing `'##'` are split, anonymized separately, and finally joined to make sure the anonymized file correctly addresses the element names.

Example before anonymization:

```
<wbim:catalog id="PrCSSS" name="Processes"/>  
...  
<wbim:process name="PrCSSS##ProcessSimple">
```

Example after anonymization:

```
<wbim:catalog id="s00000001" name="s00000002"/>  
...  
<wbim:process name="s00000001##s00000003">
```

Note that 'PrCSSS' is replaced by 's00000001' in the first occurrence in the 'id' attribute and in the substring of the 'name' attribute.

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible.

You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled

“Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified

version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License’’.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.