HARDWARE SOFTWARE CODESIGN

User Manual of AADS-T

1.4

	Author(s)	Checked by	Approbation
Name	**Roberto Varona Gómez**	**Eugenio Villar Bonet**	
Company	**University of Cantabria**	**University of Cantabria**	
Department	**TEISA**	**TEISA**	
Date	**31-5-2011**	**31-5-2011**	
Visa			
Summary	**This document is the User Manual of the software AADS-T. AADS-T is a tool for simulating a subset of AADL including its Behavioural Annex. The source code produced by AADS-T is Ravenscar Computational Model (RCM) compliant.**		

Table of Contents

Index of Figures

1 Preface

1.1 Table of versions

Version	Date	Description & rationale of modifications	Sections modified
1.0	10/11/2010	First version	All
1.1	21/1/2011	AADS-T now runs under Linux. AADS-T launches SCoPE automatically. The synchronization operation of the protected objects has been modified to use a conditional variable instead of a clock_nanosleep. The .txt and .xml files will be generated from SCoPE's output. Grammatical corrections.	Cover, 1.1, 2.1, 2.3, 3, 4, 5, 6, 7, Annex I.
1.2	8/2/2011	Install AADS-T as a real plug-in.	Cover, 1.1, 4, 5, 6.
1.3	11/2/2011	Warning window if SCoPE un-installed. Rename AADS-T Console. Warning if no connections defined in AADL model.	Cover, 1.1, 5.
1.4	31/5/2011	Automatic generation of make files, modify copia_SCoPE and .bashrc	Cover, 1.1, 6

1.2 Table of references and applicable documents

Reference	Title & edition	Author or editor	Year
[1]	Architecture analysis & design language (AADL), AS5506, v1.0	SAE AS2C	2004
[2]	The Architecture Analysis & Design Language (AADL): An introduction.	P. Feiler, D. Gluch, J. Hudak	2006
[3]	POSIX de Tiempo Real.	Michael González Harbour	2004
[4]	An Extensible Open Source AADL Tool Environment (OSATE).	SEI	2006
[5]	R1-4 Evaluation of Compliance with the ASSERT Process	J. A. de la Puente J. Zamorano	2010
[6]	Annex Behaviour specification v 2.0	SAE AS5506	2007

1.3 Acronyms and glossary

Term	Description
AADL	Architecture and Analysis Design Language

Term	***Description***
AADS-T	AADL Simulator for TASTE
DMA	Direct Memory Access
ESTEC	European Space Research and Technology Centre
FTP	File Transfer Protocol
GNU	GNU is Not Unix
GPL	General Public License
MPSoC	Multi Processor System-on-Chip
NoC	Network on Chip
OSATE	Open Source AADL Tool Environment
POSIX	Portable Operating System Interface
RCM	Ravenscar Computational Model
RTOS	Real Time Operating System
SAX	Simple API for XML
SCoPE	System Co-simulation & Performance Estimation
TEISA	Electronics Technology, Systems and Automation Engineering Department
UTF	Unicode Transformation Format
W3C	World Wide Web Consortium
WIPO	World Intellectual Property Organization
XML	eXtensible Markup Language

2 Subject

2.1 Purpose of the document

The purpose of the document is to describe the User Manual of the software tool AADS-T. This tool will consistently provide, in accordance with a subset of the AADL standard and the Behavioral Annex, the capability to simulate an AADL model using the SCoPE tool. The source code produced by AADS-T is RCM-compliant. This document specifies the usage and the general characteristics of the AADS-T tool.

2.2 Editing particularities

2.2.1 Temporary editing

Special points are signalled like this:

. ***temporary***

. ***incomplete***

. ***to be defined***

. ***to be confirmed***

2.3 Application scope

The application scope of this document is the ESTEC 22810/09/NL/JK HW-SW CODESIGN project contracted to GMV Aerospace and Defense S.A.U. and partly outsourced to the University of Cantabria. More specifically this User Manual is an activity of Work Package 310 of the project, titled System-Level Performance Tool Implementation.

3 What does AADS-T do?

The AADS-T tool enables the modeling of a subset of AADL including the Behavioral Annex for purposes of implementation and simulation. The starting point of the simulator will be an AADL specification. This AADL specification must contain a minimum functionality described by means of some AADL properties in order to enable a proper simulation of the model. The AADL model will be parsed by AADS-T and a model defined with POSIX / C++ and XML will be obtained. This model will be simulated in order to check whether the AADL constraints are fulfilled. As the design process advances and the real functionalities are attached to the software components using the corresponding source code, the value of these properties will be refined. These refined properties will be added to the AADL model and a new model will be generated by AADS-T to check if the constraints are still fulfilled.

When the AADS-T tool is initiated it requests the name of two AADL XML files. One of these files is just the AADL model written in XML. The other is the result of an instantiation of a system implementation of a textual or object AADL model obtained with OSATE, a plug-in of the Eclipse platform used to process AADL models (see Figure 1). These files are written in XML as they are easier to analyze using AADS-T because of the use of SAX.

Files containing the actual source code of the subprograms of the AADL model can be supplied to AADS-T, although it is not mandatory. If they are supplied, the simulation and performance analysis done with SCoPE will be more realistic, as this source code is embedded in the POSIX / C++ files generated by AADS-T.

Once the XML files have been parsed by AADS-T, files written in C++ with the extensions .h and .cpp and one XML file are created. The number and names of the files created depend on the AADL model parsed. The C++ files use POSIX functions and the XML file must be as specified to be used by the SCoPE tool. The source code in C++ produced by AADS-T is compatible with the RCM.

AADS-T offers the possibility to the user of launching the SCoPE tool automatically. If the user launches SCoPE from AADS-T, AADS-T generates one .txt file and one .xml file from the output of SCoPE, containing the most important data about the simulation and performance analyses such as use of CPU, core energy consumed, number of instructions executed and others.

Moreover, a file is generated with the SystemC description of the AADL model.

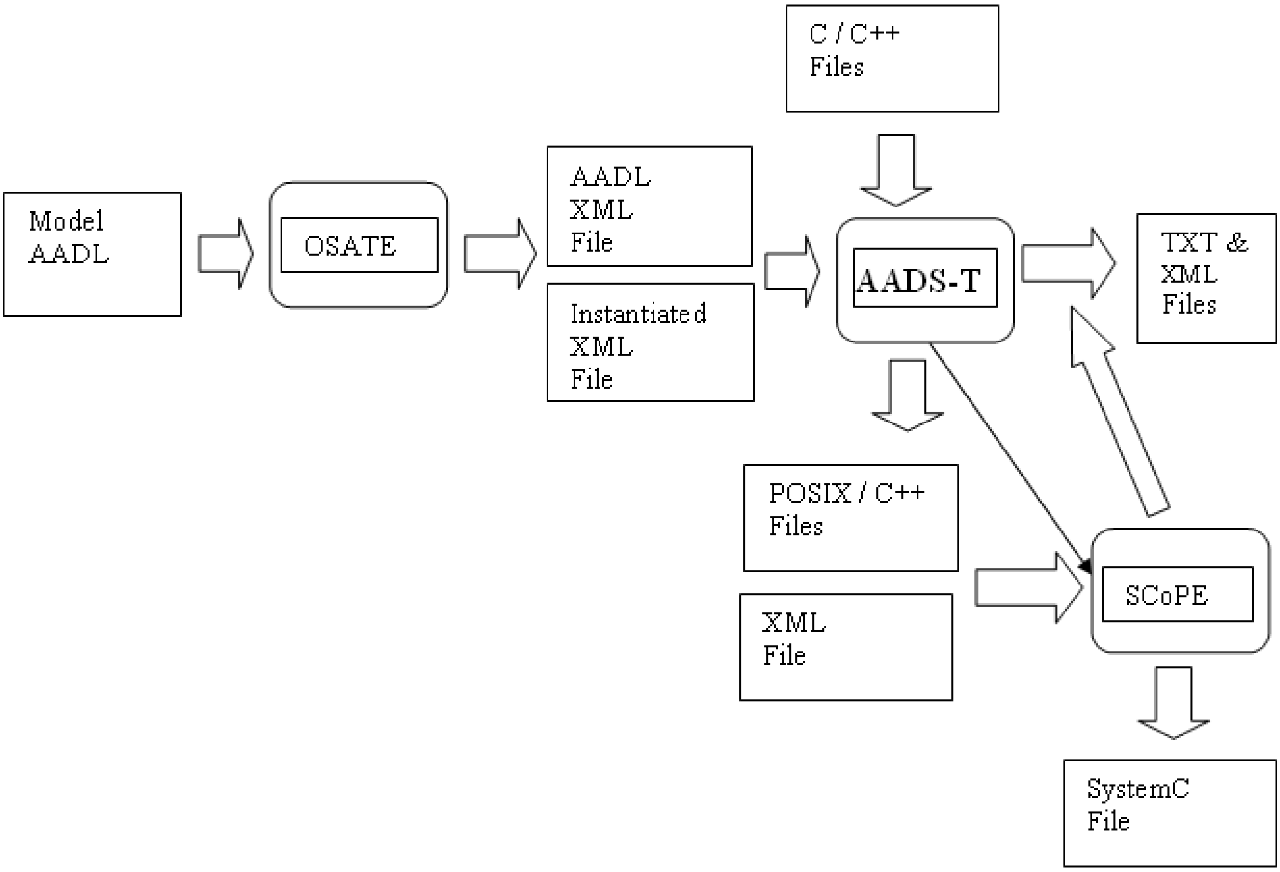


Figure 1. Relationship among OSATE, AADS-T and SCoPE.

4 Installation of AADS-T

The AADS-T tool will be delivered as a plug-in of the Eclipse platform (see more about Eclipse at www.eclipse.org). This means that it will be necessary to install OSATE (a plug-in of the Eclipse platform) to run AADS-T as a button in the toolbar. Now AADS-T runs under Linux, so the latest stable version of OSATE for Linux must be downloaded from www.aadl.info to install AADS-T.

You must download the file “uc.hwswco.sw.1.1-updatesite.zip” delivered by the TEISA department of the University of Cantabria through the Web www.teisa.unican.es/AADS.

Before installing AADS-T, you must run OSATE on the computer. Then you must choose on OSATE “Help”, then “Software Updates”, then “Find and Install…”, then check “Search for new features to install”, then “Next >”, then “New Archived Site…” and then you choose the file “uc.hwswco.sw.1.1-updatesite.zip” downloaded. You must check the recently created site and choose “Next >”, Then accept the AADS-T plug-in license agreement and select “Next >”. Select the “Finish” option. The AADS-T tool is ready to be installed, so select “Install All”. The plug-in will be installed and then you will be prompted to restart OSATE. When OSATE is restarted the plug-in will have been installed correctly.

5 Use of AADS-T

First of all OSATE must be initiated. It contains a button of AADS and an entry in the menu bar for AADS (see Figure 2, the window is cut off in this document for better legibility):

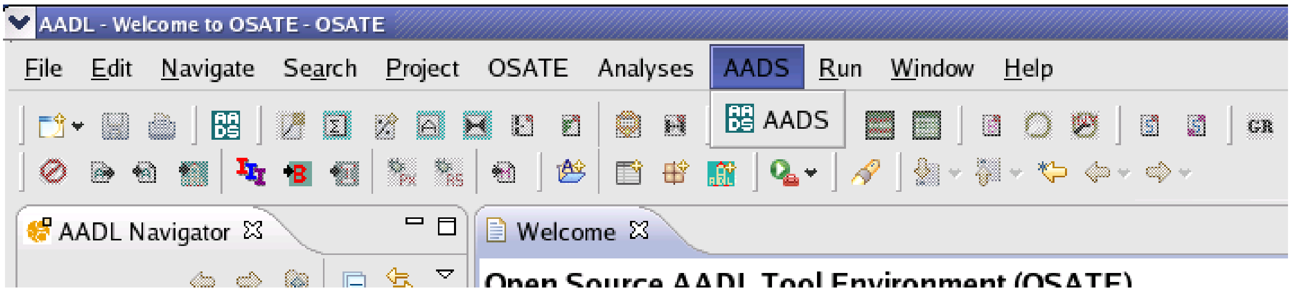


Figure 2. Menu bar of OSATE with button and entry for AADS.

When you click on this button or on the menu bar, the tool AADS-T starts. A new window opens showing a message about the version, web, author, warranty and so on (see Figure 3, split into two in this document for better legibility):

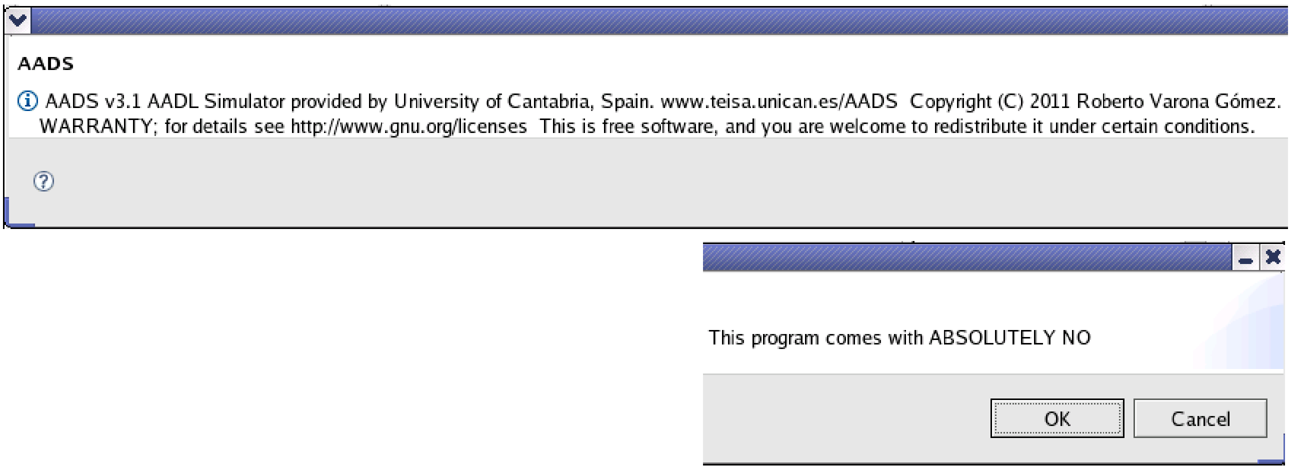
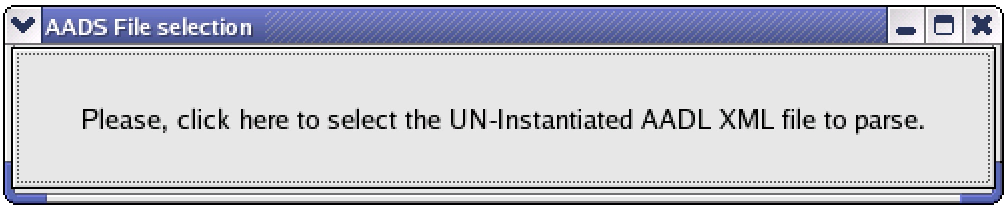


Figure 3. Initial information window of AADS-T.

If the user clicks the Cancel button, AADS-T returns to the previous state. If the user clicks the OK button, two windows appear successively asking for the name of the un-instantiated AADL file written in XML to be parsed such as for example cruise_control.aaxl (see Figure 4).



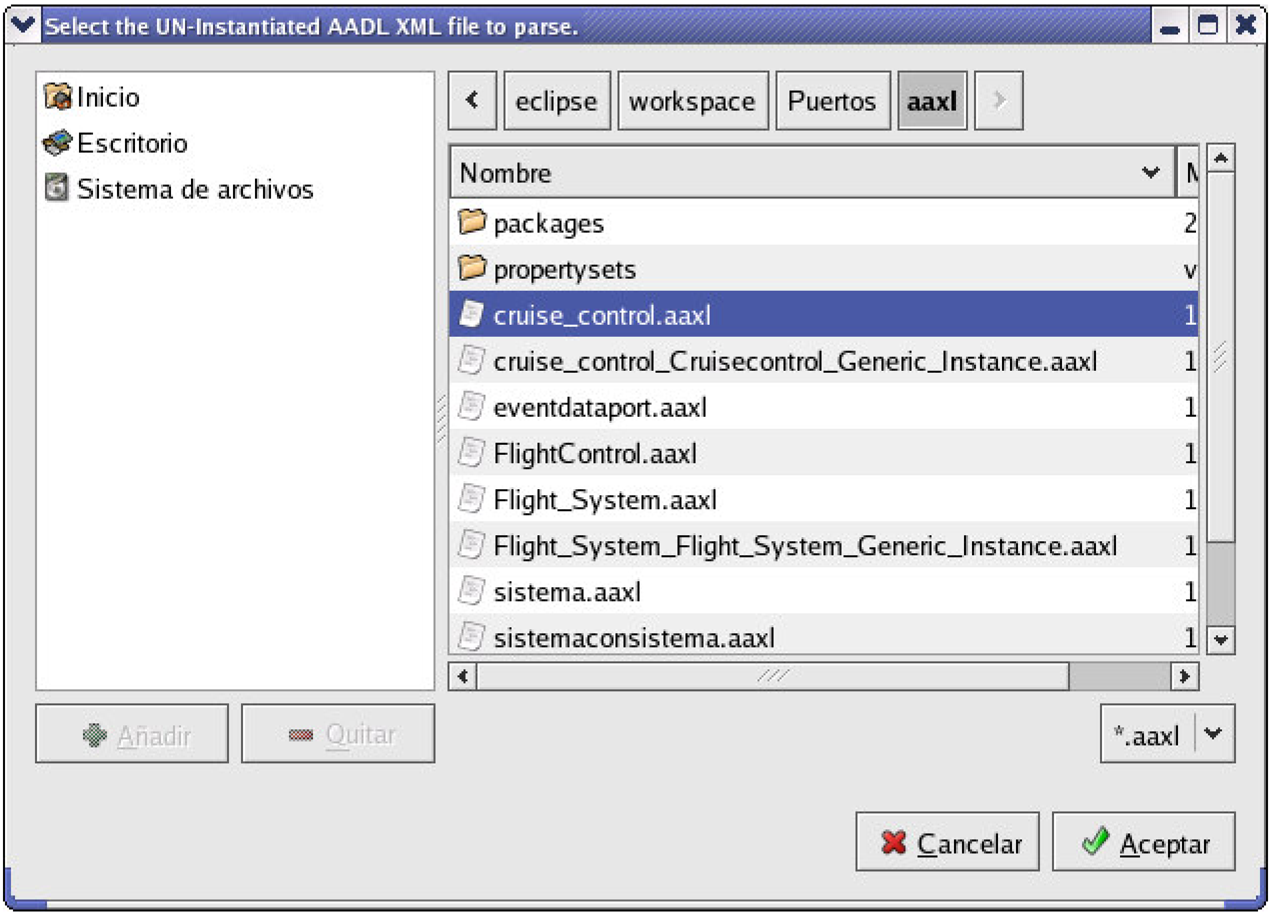
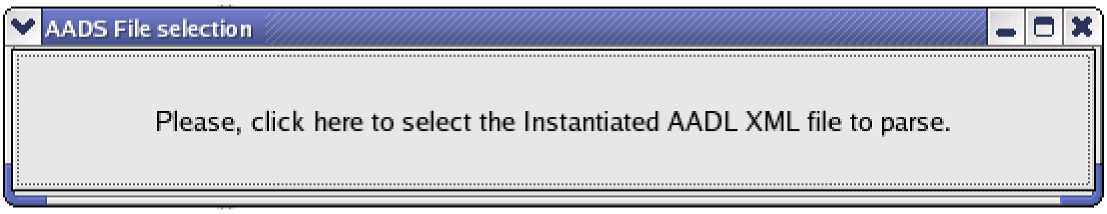


Figure 4. Selection of the un-instantiated AADL XML file.

The user can choose an .aaxl file and click on the Aceptar (Accept) button or click on the Cancelar (Cancel) button. In this last case AADS-T terminates and shows in the console of OSATE the message:

```
Exception1 org.xml.sax.SAXParseException: File "" not found.
```

If a correct file has been selected, AADS-T parses the file and two windows appear successively asking for the name of the instantiated AADL file written in XML to parse, for example cruise_control_Cruisecontrol_Generic_Instance.aaxl (see Figure 5).



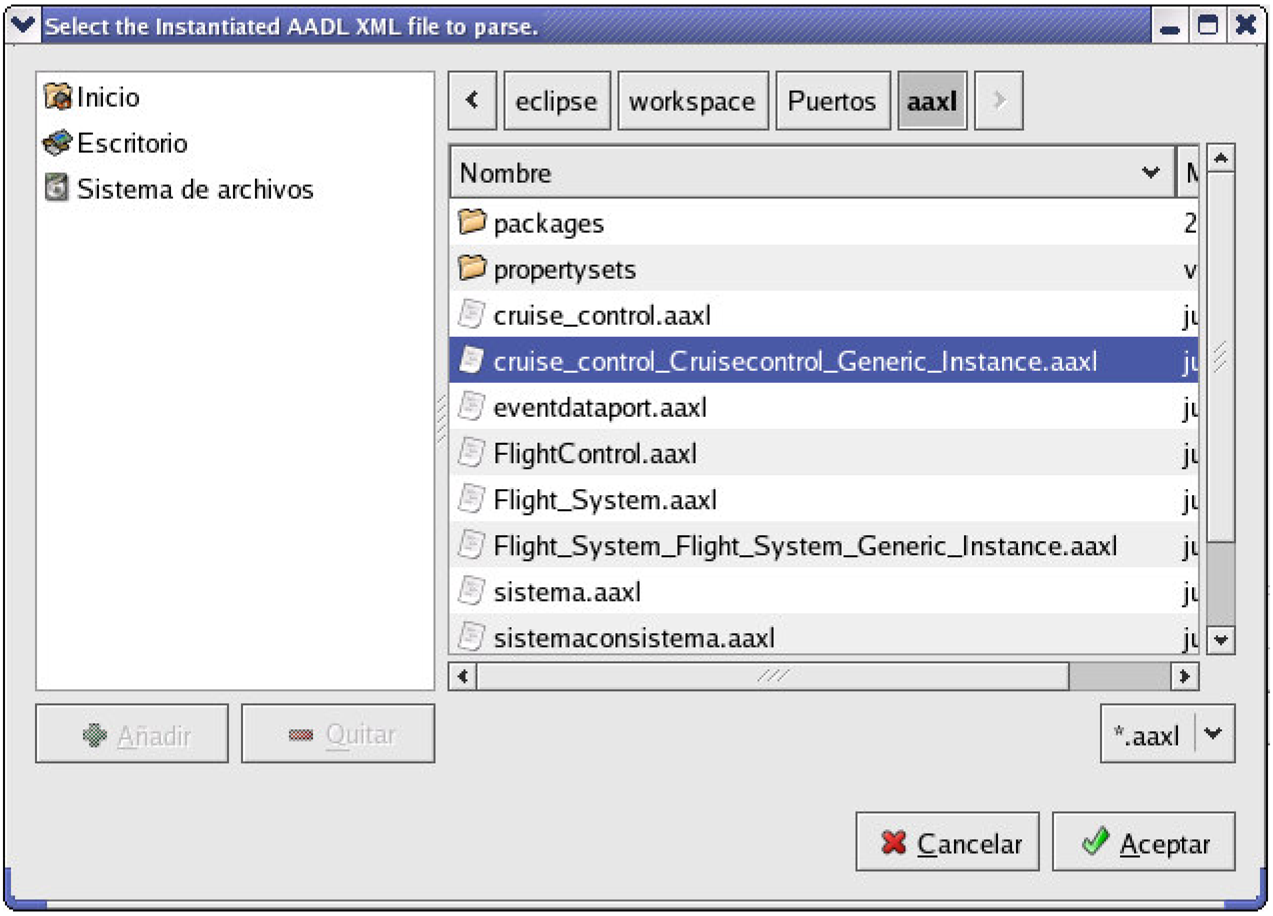


Figure 5. Selection of the instantiated AADL XML file**.**

The user can choose an .aaxl file and click on the Aceptar (Accept) button or click on the Cancelar (Cancel) button. In the latter case AADS-T terminates and shows in the console of OSATE the message:

```
Exception1 org.xml.sax.SAXParseException: File "" not found.
```

If a correct file has been selected, AADS-T parses the file and produces some files written in C++ (files with extension .cpp and .h) complying with POSIX standard, compatible with the RCM, and an XML file. These files are in the working directory and can be used with the SCoPE tool.

The console of OSATE shows some information about the process if this is successful (see Figure 6, the window is cut in this document for better legibility):

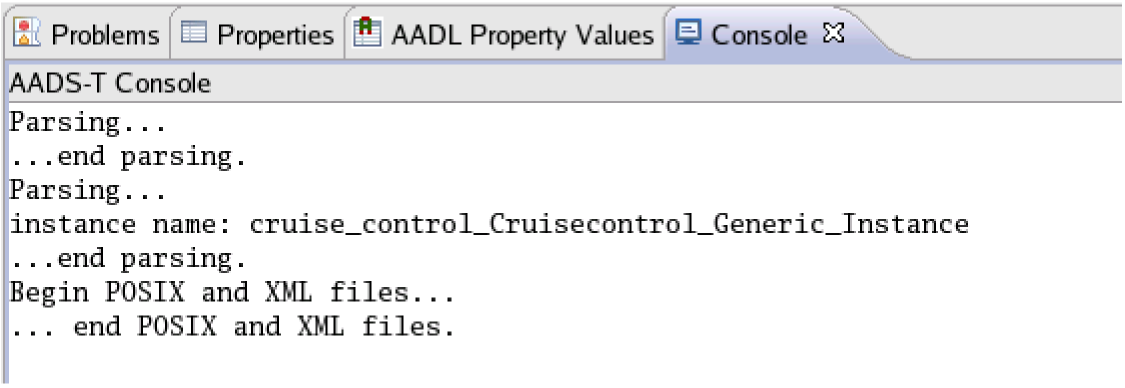


Figure 6. Information in console of OSATE.

If a file to be parsed is not in the proper format, AADS-T will show the following error messages in the console of OSATE (it is an example, it depends on the file) and will terminate:

```
Exception1 org.xml.sax.SAXParseException: The root element is required in a
well-formed document.
```

(...)

```
Exception1 org.xml.sax.SAXParseException: The markup in the document preceding
the root element must be well-formed.


java.lang.NullPointerException
	at parser.EscrituraFichero.HWComponent(EscrituraFichero.java:275)
	at parser.EscrituraFichero.GeneraXML(EscrituraFichero.java:212)
	at parser.EscrituraFichero.stringToFile(EscrituraFichero.java:107)
	at parser.Parseador.endDocument(Parseador.java:88)
	at org.apache.xerces.parsers.SAXParser.endDocument(SAXParser.java:1230)
	at
org.apache.xerces.validators.common.XMLValidator.callEndDocument(XMLValidator.ja
va:1146)
	at
org.apache.xerces.framework.XMLDocumentScanner$EndOfInputDispatcher.dispatch(XML
DocumentScanner.java:1499)
	at
org.apache.xerces.framework.XMLDocumentScanner.parseSome(XMLDocumentScanner.java
:381)
	at org.apache.xerces.framework.XMLParser.parse(XMLParser.java:1098)
	at org.apache.xerces.framework.XMLParser.parse(XMLParser.java:1139)
	at parser.Index.ParsearDocumento(Index.java:40)
	at parser.Index.main(Index.java:61)
```

If the AADL model translated by AADS-T has no connections defined, the message "Warning: There are no connections defined in the AADL model." is shown in the AADS-T Console.

If the generation of the POSIX / C++ and XML files has been correct, AADS-T will show the following window to offer the user the possibility of launching SCoPE automatically to simulate and analyze the performance of the model (see Figure 7, split into two in this document for better legibility):

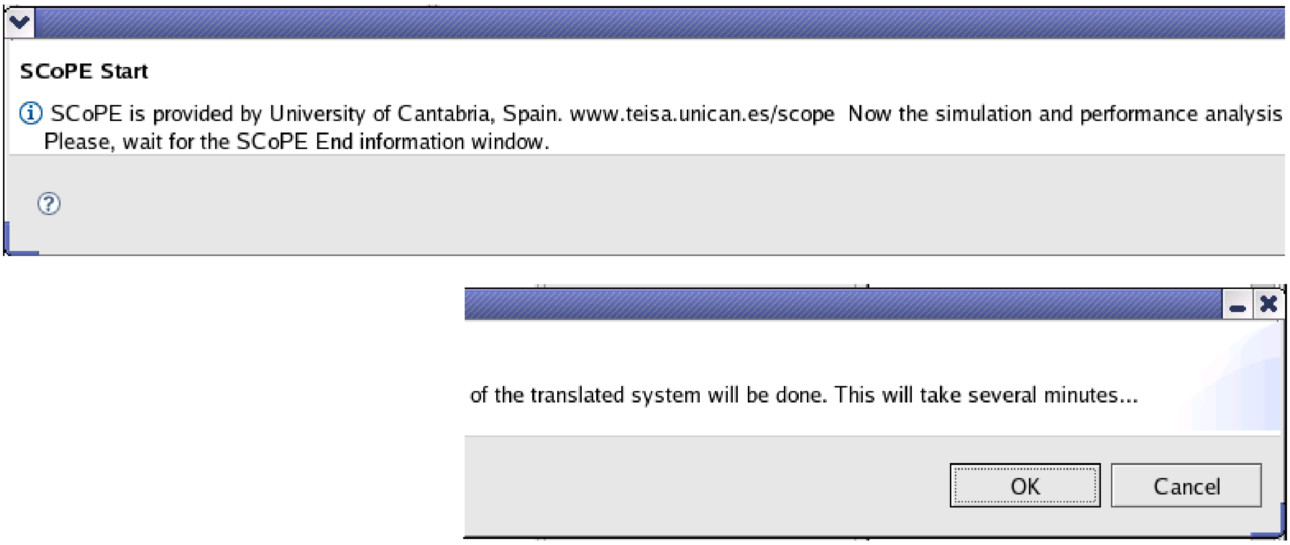


Figure 7. Start information window of SCoPE.

If the user clicks on the Cancel button SCoPE is not launched and AADS-T returns to the initial state. If the user clicks on the OK button, the files generated by AADS-T are compiled and SCoPE is launched to simulate the model. This takes some minutes.

If SCoPE is not correctly installed, AADS-T shows a warning window (see Figure 8) and returns to the initial state without launching SCoPE.

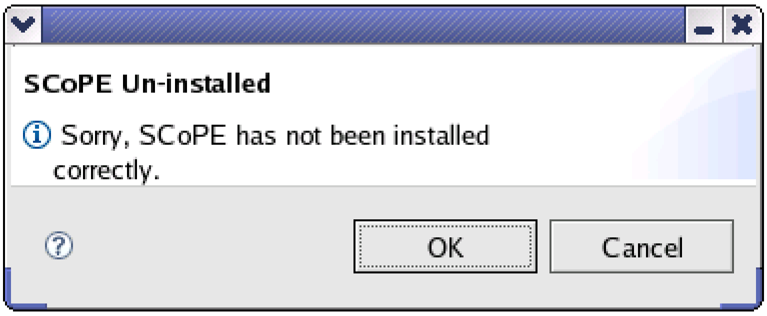


Figure 8 Warning window SCoPE un-installed.

If SCoPE is correctly installed, when the simulation has been made AADS-T shows a final window (see Figure 9) and generates a .txt file and an .xml file containing the more relevant results of the simulation done to permit the performance analysis.

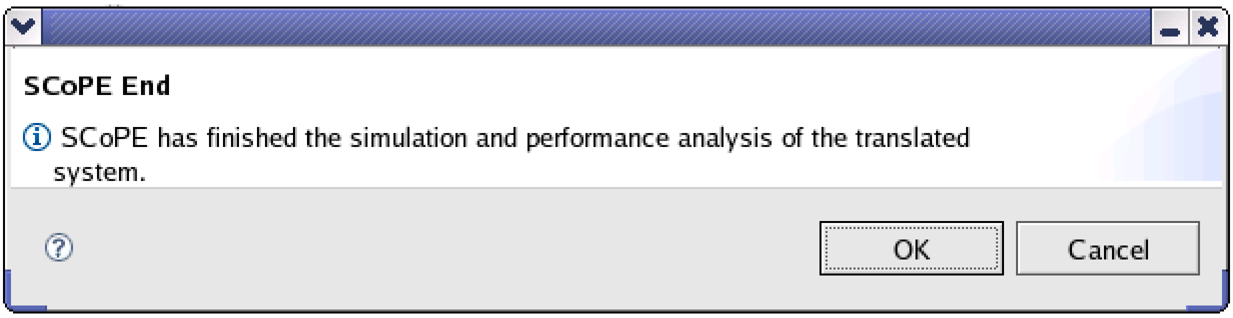


Figure 9. End information window of SCoPE.

If the simulation has been done correctly, AADS-T shows in the console of OSATE the output of SCoPE. There is part of an example in Figure 10:

Problems | Properties | AADL Property Values | Console

AADS-T Console

```
        Use of cpu: 54.3057%
        Instructions executed: 41827564
        Instruction cache misses: 54303
        Data cache hits: 0
        Data cache misses: 0
        Data cache write backs: 0
        Core Energy: 1.2816e+09 nJ
        Core Power: 256.319 mW
        Instruction Cache Energy: 1.95567e+09 nJ
        Data Cache Energy: 0 nJ
        Instruction Cache Power: 391.134 mW
        Data Cache Power: 0 mW
        Bus access time:  8681120 ns
        Idle time:  2273267400 ns
        Number of interrupts:  8819

        Total instruction miss transfers: 638

        Total data miss transfers: 0
processor_1_rtos_0
        Number of thread switches:  12102
        Number of context switches: 0
        Running time: 2055788040 ns
        Use of cpu: 41.1158%
        Instructions executed: 31666202
        Instruction cache misses: 38682
        Data cache hits: 0
        Data cache misses: 0
        Data cache write backs: 0
        Core Energy: 9.70252e+08 nJ
        Core Power: 194.05 mW
        Instruction Cache Energy: 1.47908e+09 nJ
        Data Cache Energy: 0 nJ
        Instruction Cache Power: 295.817 mW
        Data Cache Power: 0 mW
        Bus access time:  6205120 ns
        Idle time:  2935233780 ns
        Number of interrupts:  14799

        Total instruction miss transfers: 463

        Total data miss transfers: 0
Bus Lan_0
         Bytes transferred: 2977960
```

Figure 10. Part of the SCoPE output in the console of OSATE.

6 Relation with SCoPE

The AADS-T tool creates files written in C++ with the extensions .h and .cpp, compatible with the RCM, and one file written in XML. The number and names of the files created depends on the model AADL parsed. These files are used by SCoPE as we can see in Figure 1 to simulate the model. Therefore, the structure that these files have and functions supported by SCoPE must be known by AADS-T. AADS-T produces files to be used with the SCoPE tool, so the relationship between AADS-T and SCoPE is dependence of the former on the latter.

The XML file generated by AADS-T to be used by SCoPE as a configuration file follows the 1.0 standard of W3C and uses UTF-8 encoding. The hardware architecture is structured through the XML file generated by AADS-T. It is used as part of the configuration parameters of SCoPE and is divided into: HW_Platform, SW_Platform and Application.

- HW_Platform. Any AADL implementation of a processor, memory, bus or device must be specified with its category and name in the HW_Components subsection of HW_Platform. The AADL property Assign_Byte_Time is used to set the frequency parameter in the XML file. For memories we use the properties Access_Time or Read_Time and Write_Time. These properties have their values in time units (ns, ms and so on) and they must be transformed into MHz. To know the mem_size of a memory, both Word_Count and Word_Size AADL properties are required. Finally the mem_type of a memory is derived from Memory_Protocol in the AADL model. If the component is a processor, proc_type must be specified.

  The HW_Architecture and Computing_groups subsections of HW_Platform are next in the XML file. To know the start_addr of a memory we take the AADL property Base_Address. To know the start_addr of a device we take the AADL property UC::Base_Address_Devices. The component and name are inferred from the AADL model. Hardware components are grouped by buses as they are connected to them in AADL through the connections bus access and the features requires bus access.
- SW_Platform. This section has two subsections: SW_Components and SW_Architecture. This section takes into account the buses that are defined to make the equivalent nodes. In this section the operating systems are specified.
- Application. This section has two subsections: Functionality and Allocation. Filling the Functionality section is straightforward from the AADL model using the property of a thread Activate_Entrypoint for the function and Source_Text for the file. The name is the same as the one of the thread. For the Allocation section we need to know the property of a thread Actual_Processor_Binding, and find out which bus the processor is bound to and then find out which node that bus corresponds to. The AADL name of the thread is used for the name and the component.

Before using SCoPE, it must be installed, compiled and linked on the Linux host where AADS-T is installed. For more information about SCoPE you can visit www.teisa.unican.es/scope.

Two make files are generated automatically by AADS-T to compile and link the files created by AADS-T with the ones of SCoPE. One file is on the working directory of the xml plug-in of SCoPE. AADS-T uses the file Makefile_Template_Prueba that must be in the directory named in AADS_WORKSPACE (see later), as a basis to generate the make file. AADS-T replaces the label XML_XML_XML to use the corresponding configuration XML file of the specific model. Here is this file Makefile_Template_Prueba:

```
all:
	$(SCOPE_XML_PLUGIN)/build/scope_tool.x -xml XML_XML_XML.xml -sc
	cp sc_main.cpp scope/sc_main.cpp
	make all -C scope

sc_main:
	$(SCOPE_XML_PLUGIN)/build/scope_tool.x -xml XML_XML_XML.xml -sc

clean:
	rm -f *.o *.so *.x *.ii *.tmp
	make clean -C scope
	rm -f scope/sc_main.cpp
	rm -f sc_main.cpp

run: all
	make run -C scope

distclean:
	rm -f *.x
```

The other make file is on the scope directory that is on the working directory of the xml plug-in of SCoPE. AADS-T uses the file Makefile_Template_Scope that must be in the directory named in AADS_WORKSPACE (see later), as a basis to generate the make file. AADS-T replaces the label MAIN_MAIN_MAIN to use the corresponding main C++ file of the specific model. AADS-T replaces the label GNAT_GNAT_GNAT with the directory named in GNAT_FOR_LEON (see later). AADS-T adds the corresponding lines to compile and link the different C++ files. Here is this file Makefile_Template_Scope:

```
# SCoPE options:
SCOPE_CXX = scope-g++

SPARC_COMPILER = "sparc-elf-gcc -isystem GNAT_GNAT_GNAT/bin/../lib/gcc/sparc-
elf/4.1.3/include/ -isystem GNAT_GNAT_GNAT/sparc-elf/include/ -isystem
GNAT_GNAT_GNAT/sparc-elf/include/sys -isystem /usr/include -D_POSIX_THREADS -
D_POSIX_THREAD_PRIORITY_SCHEDULING -D_POSIX_THREAD_PRIO_PROTECT -D_POSIX_TIMERS
-D_LEON_2 -D'sched_setaffinity(...)' -D'sched_getaffinity(...)'"

SCOPE_FLAGS = --scope-method=asm-opcodes --scope-crosscompiler=$(SPARC_COMPILER)
--scope-cpu=LEON2 --scope-nodcache --scope-preserve-files #--scope-verbose -
scope-method=op-cost --scope-preserve-files --scope-crosscompiler=arm-linux-gcc
--scope-method=asm-sentences --scope-language=c
SCOPE_FLAGS_2 = --scope-method=asm-opcodes --scope-
crosscompiler=$(SPARC_COMPILER) --scope-cpu=FPGA --scope-nodcache --scope-
preserve-files #--scope-verbose -scope-method=op-cost --scope-preserve-files --
scope-crosscompiler=arm-linux-gcc --scope-method=asm-sentences --scope-
language=c
SCOPE_INC_DIR =-I$(SCOPE_HOME)/scope \
                              -I$(SYSTEMC)/include \
                              -I$(SCOPE_HOME)/TLM2/include/tlm \
                              -I$(SCOPE_HOME)/tinyxml \
```



```
                -I$(SCOPE_HOME)/scope \
                -I$(SCOPE_HOME)/scope/hal                   \
                -I$(SCOPE_HOME)/scope/rtos/api/posix        \
                -I$(SCOPE_HOME)/scope/rtos/api/ucos         \
                -I$(SCOPE_HOME)/scope/rtos/drivers          \
                -I$(SCOPE_HOME)/scope/rtos/kernel           \
                -I$(SCOPE_HOME)/scope/rtos/low_level        \
                -I$(SCOPE_HOME)/scope/rtos/qt_interface     \
                -I$(SCOPE_HOME)/scope/rtos/utils                \
                 -I$(SCOPE_HOME)/scope/sicosys/SC_Simul         \
                 -I$(SYSTEMC)/include                           \
                 -I$(SYSTEMC)/src/sysc/qt               \
                 -I$(SCOPE_HOME)/TLM2/include/tlm \
                -I$(SCOPE_SMPSIM_PLUGIN)/include

# Compiler options:
CXX = g++
DEBUG = -g
OPT = -O0
CFLAGS = $(DEBUG) $(OPT) -c
INC_DIR = $(addprefix -I,$(LOCAL_INC_DIRS)) $(SCOPE_INC_DIR)
LIB_DIR = -L$(SYSTEMC)/lib-linux -L$(SCOPE_HOME)/scope -
L$(SCOPE_HOME)/scope/sicosys/SC_Simul -L$(SCOPE_HOME)/tinyxml -
L$(SCOPE_SMPSIM_PLUGIN)/lib -L.
LIB =  -rdynamic -lscope -ltinyxml -lsystemc -lpthread -lrt -latcs -ldl

SRCS_CPP = MAIN_MAIN_MAIN.cpp devices.cpp
OBJS_CPP = $(SRCS_CPP:.cpp=.o)

OUT = MAIN_MAIN_MAIN.x

.PHONY:all $(OUT)

all:$(OUT)
# Link:
$(OUT): $(OBJS_CPP) sc_main.o
        $(CXX) $(LIB_DIR) $(OBJS_CPP) sc_main.o -o $@ $(LIB)


# Parse and compile software application files:
MAIN_MAIN_MAIN.o: MAIN_MAIN_MAIN.cpp
        $(SCOPE_CXX) $(SCOPE_FLAGS) $(CFLAGS) $(INC_DIR) $(SCOPE_INC_DIR) $^ -o
$@
devices.o: devices.cpp
        $(SCOPE_CXX) $(SCOPE_FLAGS) $(CFLAGS) $(INC_DIR) $(SCOPE_INC_DIR) $^ -o
$@


# Compile sc_main.cpp with standar g++
sc_main.o : sc_main.cpp
        $(CXX) $(CFLAGS) $(SCOPE_INC_DIR) $^ -o $@

# Clean:
.PHONY: clean distclean run
run: $(OUT)
        ./MAIN_MAIN_MAIN.x
clean:
```



```
        rm -rf $(OBJS_CPP) sc_main.o
        rm -f *.so
        rm -f *~
        rm -rf *.ii rm*.s rm table_* prsd_* asm_* uc_*

distclean: clean
        rm -rf $(OUT)
```

AADS-T uses three files that are in the working directory of AADS-T, when the user confirms the launching of SCoPE: copia_SCoPE, compila_SCoPE and run_SCoPE. The first is made to copy the .cpp, .h and .xml files generated by AADS-T to the working directory of the xml plug-in of SCoPE, and to generate the sc_main.cpp file (a SystemC file with the description of the model). Here is an example (some parts such as the name of the .xml file may have to be changed for the specific model):

```
cd $AADS_WORKSPACE
cp *.cpp $SCOPE_XML_PLUGIN/examples/prueba
cp *.cpp $SCOPE_XML_PLUGIN/examples/prueba/scope
cp *.h $SCOPE_XML_PLUGIN/examples/prueba
cp *.h $SCOPE_XML_PLUGIN/examples/prueba/scope
cp *.xml $SCOPE_XML_PLUGIN/examples/prueba
cp *.xml $SCOPE_XML_PLUGIN/examples/prueba/scope
cd $SCOPE_XML_PLUGIN/examples/prueba
$SCOPE_XML_PLUGIN/build/scope_tool.x -xml
cruise_control_Cruisecontrol_Generic_Instance.xml -sc
cp sc_main.cpp scope/sc_main.cpp
```

The second file, compila_SCoPE, is made to compile all the files generated by AADS-T and generate the .x file (the executable file):

```
cd $SCOPE_XML_PLUGIN/examples/prueba
make all -C scope
```

The third file, run_SCoPE, is made to run SCoPE and to generate the file with the output of SCoPE (it shows the number of thread and context switches, use of cpu, running time, etc.) to generate the .txt and .xml files later:

```
cd $SCOPE_XML_PLUGIN/examples/prueba
make run -C scope > borame.txt
cp borame.txt $AADS_WORKSPACE/borame.txt
```

You must be aware of the SCOPE_XML_PLUGIN, AADS_WORKSPACE, GNAT_FOR_LEON, etc. variables defined in the .bashrc file of your computer. For example they could be:

```
export CXX=g++
export SYSTEMC=/home/roberto/systemc
export TLM2=/home/roberto/TLM
export DS2_CXX=g++
export SCOPE_HOME=/home/roberto/scope_repo
export PATH=$PATH:$SCOPE_HOME/bin
export
SCOPE_XML_PLUGIN=/home/roberto/scope_xml_plugin_repo/branches/scope_xml_systemc/
scope_xml_plugin
```



```
export SCOPE_PROJECT=cr.scope
export SCOPE_CPU=LEON2
export PATH=$PATH:/home/roberto/gnatforleon-2.1.0/bin
export PATH=$PATH:/home/roberto/jre1.6.0_23/bin
export PATH=$PATH:/home/roberto/osate-topcased-1.5.8.201006182029PRD-
linux.gtk.x86/eclipse
export AADS_WORKSPACE=/home/roberto/osate-topcased-1.5.8.201006182029PRD-
linux.gtk.x86
export GNAT_FOR_LEON=/home/roberto/gnatforleon-2.1.0
```

The SCoPE tool provides the technology to perform MPSoC HW/SW co-simulation with NoC (Network on Chip). It obtains results for exploring the design space to choose the right processors and HW/SW partition for embedded systems. It also allows the simulation of different nodes connected through a NoC in order to analyse the behaviour of large systems. Commonly, these tools are based on slow ISSs. The differentiating feature of this technique is that SCoPE obtains the performance estimations at source code level. This level of abstraction allows the simulation time to be reduced significantly while maintaining good accuracy.

SCoPE is a C++ library that extends, without modification, the standard language SystemC to perform the co-simulation. On the one hand, it simulates C/C++ software code based on two different operating system interfaces (POSIX and MicroC/OS). On the other hand, it co-simulates these pieces of code with hardware described in SystemC.

An engineer with this tool can simulate specific software on a custom platform and obtain estimations of: Number of thread and context switches, running time and use of CPU, instructions executed and cache misses, energy and power (of core and instruction cache).

This library models the detailed behaviour of the RTOS including concurrency (among tasks in the same processor), parallelism (among tasks in different processors), scheduling and synchronization. Although the SystemC kernel executes processes following a non pre-emptive scheduling policy without priorities, SCoPE models pre-emption under different scheduling policies based on priorities.

SCoPE integrates a POSIX-based API that enables the execution of many software applications that fulfil this standard. POSIX is the main operating system interface nowadays, but it is not the only one. Thus, SCoPE has been improved to support extensions for other types of interfaces. An example is the integration with the MicroC/OS interface. This is a demonstration of the scalability of the tool, in terms of software support.

The design of embedded systems requires not only software handling but also hardware communication. For this reason SCoPE includes a set of more than a hundred driver facilities to implement this communication. One of the most extensively used operating systems in this sector is Linux, thus this driver facilities are based on the Linux kernel version 2.6. Furthermore, SCoPE is able to simulate the loading of kernel modules and the handling of hardware interruptions and their corresponding scheduling.

SystemC is the language used for the modelling of the hardware platform due to the easiness of implementation (C++ extension) and its simulation kernel. For the purpose of simulating different platforms, SCoPE incorporates some generic hardware modules: a bus based on TLM2 used for the communication with peripherals and the transmission of hardware interruptions, a DMA for copying large amounts of data, simple memory for the simulation of cache and DMA traffic, a hardware interface for simple custom hardware connection, a network interface that works as a net card for the NoC and an external network simulator to implement the NoC connected to SCoPE.

System simulation comprises Multicomputation and Modular structure. Multicomputation: One of the advantages of this tool is the possibility of interconnection among independent nodes and simulating the interaction among them. Modular structure: Each RTOS component is an independent object that does not share any data with the others. Furthermore, each process is isolated from the rest of the system, thus a process with global variables can be replicated in many nodes without data collision problems. That is, each process has a separate memory space.

7 Ravenscar Computational Model compliant

The real-time behavior specification of ASSERT models is based on the RCM, a model of concurrency for high-integrity systems that enables formal analysis of the temporal properties of a system using response-time analysis techniques. The model includes a static set of concurrent execution threads, communicating by means of shared protected data with mutually exclusive read and write access, and a restricted form of conditional synchronization. The model is simple enough to be implemented by a simple, small-size real-time kernel, thus easing the way to the eventual certification of real-time systems based on it.

Twelve properties must be fulfilled to be RCM-compliant; the source code generated by AADS-T fulfils all of them. These properties are stated in an internal document of the project entitled R1-4 Evaluation of Compliance with the ASSERT Process, written by Juan Antonio de la Puente and Juan Zamorano.

7.1 Basic elements

There are two main elements in the RCM: threads and protected objects (PO). A thread is the basic unit of execution, which can be executed concurrently with other threads on a single processor. POs are an abstraction of shared data, synchronization, and interrupt handling.

There are a static number of threads and POs. Therefore, threads and POs can only be created at system initialization time.

RCM 1 A real-time system consists of: a static set of N threads, $T=\{\tau_i\}$, $i \in 1..N$; and a static set of M POs, $O=\{\theta_i\}$, $i \in 1..M$. The set O may be empty (M = 0), in which case the system is said to have only independent threads.

In the source code generated by AADS-T, all the threads and POs are created calling *pthread_create* and as objects of the corresponding classes respectively at system initialization time.

7.2 Properties of threads

A thread is a concurrent unit of execution with the following properties:

RCM 2 Threads are non-terminating. They exhibit an endless repetitive behavior, alternating between the following states: Suspended (a suspended thread is not eligible for execution) and Ready (a ready task can be executed when the processor is allocated to it).

RCM 3 Threads have a single activation point. An activation point is a point in the executable code of a thread at which its state changes from Suspended to Ready. When activated, a thread becomes ready and then executes a piece of sequential code (thread activity), after which it becomes suspended awaiting the next activation.

Threads of the source code generated by AADS-T use *while(true)* to be non-terminating. They are suspended after executing the sequential code in a *clock_nanosleep* and when sleeping time has passed they become ready at their single activation point.

RCM 4 The activity of a thread is a sequence of code with a bounded and known worstcase execution time (WCET). The WCET of thread τ_i is C_i.

AADS-T utilizes the AADL property *Compute_Execution_Time* to know the WCET of a thread. The source code generated checks that this WCET is not exceeded.

This property implies that a thread does not execute any operation that could result in its becoming suspended other than the suspension immediately before the activation point. The threads created by AADS-T behave similarly.

RCM 5 A thread can be activated only by one of the following two kinds of events. One is by a timing event which is issued periodically by the environment. In this case the thread τ_i is said to be periodic or time-driven with period T_i.

The other is a synchronization event issued when the barrier of a synchronization PO is opened (see RCM 8 below). In this case, the thread τ_i is said to be sporadic. The synchronization event must have a minimum inter-arrival time associated to it, i.e. a minimum elapsed time interval between two consecutive occurrences of the event, T_i.

AADS-T uses the AADL properties *Period* and *Device_Dispatch_Protocol* to know the period and the type of a thread respectively. It accepts only *periodic* and *sporadic* threads and not *aperiodic* or *background* threads. The difference between the codes generated is that a sporadic thread waits for an event from an *event* or *eventdata port connection* after invoking a synchronization operation in the activation point. In both cases *clock_nanosleep* waits a time T_i.

7.3 Properties of protected objects

A PO is an object which encapsulates a set of data and a set of associated operations (protected operations). The value of the data makes up the state of the object. The state can only be read or changed by invoking one of the operations of the PO. If θ is a PO: θ.S denotes its state, θ.S $\in$ **S**, where **S** is an appropriate data domain; $\theta . P_k$ denotes the k-th operation of θ. Notice that a PO must have at least one operation; otherwise its state is inaccessible. The notation $\tau \rightarrow \theta$ will be used to denote that τ invokes one or more operations of θ. Similarly, $\tau \rightarrow \theta . P$ means that τ calls the operation $\theta . P$.

AADS-T generates an object of the corresponding class which is a PO in the source code for each AADL *data, event and eventdata port connection*. Classes generated by AADS-T have the appropriate data members to achieve the communication of data and/or events between threads. Each class has a constructor and member functions read and write to initialize and access data members.

POs have the following properties:

RCM 6 Only one thread can execute an operation of a given PO at any given time, i.e. protected operations are mutually exclusive. Consequently, if a thread invokes a protected operation at a time when another thread is already executing an operation of the same object, it has to wait. When the protected operation that was being executed is completed, the waiting thread is allowed to execute the operation it had invoked. Notice that a thread that is waiting to begin a protected operation is not considered to be suspended. In consequence, a thread activity can invoke protected operations without violating RCM 4.

Each class produced by AADS-T defines a *mutex* that is locked when a member function is called and unlocked when it ends, ensuring compliance with mutual exclusion.

RCM 7 All protected operations have a bounded and known WCET. The WCET of the protected operation $\theta_i . P_k$ is $C_{i,k}$. Again, this property implies that no operations that could result in a thread being suspended can be invoked from a protected operation.

AADS-T uses the ad hoc defined AADL properties *PO_read_WCET* and *PO_write_WCET* for each *port connection* to know the WCET of each member function. The source code generated checks if these WCETs are exceeded. Moreover, no member function calls any suspending operation.

RCM 8 A PO can have at most one synchronization operation that has an associated barrier, which is a Boolean variable that is part of the object state. When the value of the barrier is true, the barrier is said to be open, and otherwise it is said to be closed.

The behaviour associated with synchronized operations is as follows: When a thread invokes a synchronization operation, if the barrier is open the execution proceeds as with an ordinary protected operation; but if the barrier is closed, the thread is suspended. At most one thread can be suspended at a barrier at any given time. A thread that is suspended at a barrier is resumed whenever the barrier becomes true (as the result of the execution of another protected operation by some other thread).

Invoking a synchronization operation is a potentially suspending operation, and thus cannot be done within a thread activity; this can only be used to implement the activation events of sporadic threads.

In the source code produced by AADS-T only the objects corresponding to *event* and *eventdata port connections* have a synchronization member function because a sporadic thread is dispatched by an event as stated above. Only sporadic threads invoke the synchronization. The classes corresponding to *event* and *eventdata port connections* have a POSIX condition variable as a datum member that is initialized at system initialization time. The barrier is initialized as false in the constructor, then set to true in the write member function (besides signalling the condition variable to unblock the sporadic thread), then checked to see whether it is false in the synchronization to block the sporadic thread on the condition variable, and finally set to false after unblocking it.

7.4 Scheduling

The RCM is associated with an instance of the fixed-priority pre-emptive scheduling (FPPS) method, together with the immediate ceiling priority inheritance protocol (ICPP). The scheduling model is defined by the following properties:

RCM 9 Each thread τ_i has a basic priority, $P_i \in \mathbf{P} \subset Z$, where Z is the set of the integer numbers. The basic priority of a thread is fixed, i.e. it is never changed.

AADS-T uses the ad hoc AADL property *Priority* to create a thread at system initialization time with *sched_priority* at that priority, which is never changed.

RCM 10 Each PO θ_i has a ceiling priority CP_i which is the maximum of the basic priorities of all the threads invoking any of its operations: $CP_i = \max P_j, \tau_j \to \theta_i$. As basic priorities of all threads are fixed so too are the ceiling priorities of all POs.

RCM 11 At every instant of time, each thread has an active priority. The active priority of a thread is the maximum of the basic priority of the thread and the ceiling priority of all POs that contain an operation that is currently being executed by the thread. Therefore, whenever a thread invokes a protected operation, it immediately inherits the ceiling priority of the enclosing PO.

In the source code generated by AADS-T the function *pthread_mutexattr_setprotocol* is used with the value *PTHREAD_PRIO_PROTECT* and the function *pthread_mutexattr_setprioceiling* with the maximum of the priorities of the two threads communicating through a *port connection*. This is done when initializing the *mutex* of the object corresponding to that *connection* at system initialization time guaranteeing the fulfillment of RCM 10 and RCM 11.

RCM 12 Ready threads are conceptually grouped into ready queues. There is a ready queue for each priority level in **P**. Threads are added to and removed from priority queues according to the following rules: When a suspended thread becomes ready, it is added at the tail of the priority queue for its active priority. When the processor is idle, the thread which is at the head of the non-empty ready queue with the highest priority is dispatched for execution and removed from the queue. Whenever there is a non-empty ready queue with a higher priority than the priority of the currently running thread, the thread is pre-empted from the processor and it is added at the head of the ready queue for its active priority. Notice that according to the previous rule, the thread at the head of the ready queue that caused the pre-emption is dispatched for execution immediately afterwards.

AADS-T admits only *SCHED_FIFO* for the ad hoc AADL property *POSIX_Scheduling_Policy* of a thread, to set so *sched_policy* in the source code.

The above model specifies a concurrent system with a predictable, analyzable temporal behaviour. Since the execution time of threads is bounded (RCM 4, RCM 7) and the scheduling method is FPPS with ICPP, well-known response-time analysis techniques can be applied to statically guarantee that the system satisfies its temporal requirements.

Annex I: Subset of AADL and Behavioral Annex.

I.1 AADL

The following lists alphabetically the subset of AADL implemented by AADS-T:

Bus,

Composite data,

Data,

Device,

Memory,

Ports connections:

- Data port,
- Event data port,
- Event port.

Process,

Processor,

Properties:

- Actual_Subprogram_Call,
- ASSERT_Properties::Access_Time,
- Assign_Byte_Time,
- Base_Address,
- Compute_Entrypoint,
- Compute_Execution_Time,
- Device_Dispatch_Protocol,
- Dispatch_Protocol,
- Finalize_Execution_Time,
- Finalize_Entrypoint,
- Initialize_ Execution_Time,
- Initialize _Entrypoint,
- Memory_Protocol,
- Period,
- Read_Time,
- Source_Code_Size,
- Source_Data_Size,

Source_Stack_Size,

Source_Text,

UC::Base_Address_Devices,

UC::PO_read_WCET,

UC::PO_write_WCET,

UC::POSIX_Scheduling_Policy,

UC::Priority,

Word_Count,

Word_Size,

Write_Time.

Subprogram:

Subprogram calls,

Subprogram parameters.

System,

Thread:

Periodic thread,

Sporadic thread.

I.2 Behavioral Annex

The following lists alphabetically the subset of the AADL Behavioral Annex implemented by AADS-T:

Arrays,

Behavior_Properties::Abstract,

Computation,

‘Count,

Delay (no longer supported because of RCM-compliant),

Enumerated types,

For,

Function call depending on the value of the input parameter of a subprogram,

Function cout call,

If else endif,

Initially,

Modification of the input parameter of a subprogram and sending as output parameter, Behavior_Properties::Multiplicity,

On part (--> symbol) of Boolean condition of guards,

Passing parameters to subprogram,

States,

State variables,

Subprogram call through event port,

Transitions,

Transitions names,

Transitions priorities,

What is received by an event port, send it by another event port,

What is received by an eventdata port, send it by another eventdata port,

When part of Boolean condition of guards.

Annex II: License.

AADS-T is distributed under license GNU GPL which is related in this section.

II.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

II.1.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

II.1.2 TERMS AND CONDITIONS

II.1.2.0 Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

II.1.2.1 Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

II.1.2.2 Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

II.1.2.3 Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

II.1.2.4 Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

II.1.2.5 Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

II.1.2.6 Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.

The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

II.1.2.7 Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

II.1.2.8 Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

II.1.2.9 Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance.

However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

II.1.2.10 Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

II.1.2.11 Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

II.1.2.12 No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

II.1.2.13 Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

II.1.2.14 Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

II.1.2.15 Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

II.1.2.16 Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

II.1.2.17 Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

