Appendix C

Electrical vehicle installation guide

C.1. System requirements

The electrical vehicle simulation can be supported for any operating system; the only condition is that the necessary programs for its execution are correctly installed. The required programs are:

Java
The java programming language was selected for the development of the electrical vehicle simulation because it is well suited for the distributed context that we are using, for the high development level that the object oriented programming provide to us, and for the Fractal component model implementation (Julia). The java used version was the J2SE 1.3.1.

Apache ant
It is a Java-based build tool, which at difference from other building tools like make, gnumake, or jam, use XML files for its configuration. These files are based on a tree schema in order to execute several tasks. Apache ant is commonly used with Julia. The Apache ant used version was the 1.6.5.

Fractal component model
It is a modular and extensible component model that can be used with various programming languages to design, implement, deploy and reconfigure various systems and applications, from operating systems to middleware platforms and to graphical user interfaces. The Fractal used version was the 2.3.1., it includes the Julia version 2.1.3.

ASM
It is a Java bytecode manipulation framework. It offers similar functionalities as BCEL or SERP, but is much smaller and faster than these tools. The ASM used version was the 2.0.

C.2. Input

The electrical vehicle simulation requires as input two text files: TaskList.txt and NodeTaskRelation.txt. Both files are located in the same directory where the system is placed (by default: ERTS_executor). The TaskList file has entire set of task that can be present on the system (Task name, start time, duration), and the NodeTaskRelation has the relation of which nodes can execute associated tasks (Node, associated task). Next is presented an example of both files:

TaskList:
A 5 12
B 7 4
C 3 8
D 9 11
A 11 12
B 2 4
C 24 8
D 20 11

NodeTaskRelation:
1 A
2 B
3 C
4 D

C.3. Steps for execution

It is supposed that the distribution of the electrical vehicle simulation is placed together in an only directory, by default called ERTS_executor. The electrical vehicle simulation is executed with the aid of the Apache ant, but before executing the ant command from a terminal the build.xml that comes with the distribution must be edited. Next are presented the content of the build.xml file:

```
<project name="ERTS" default="execute">

 <!-- ================================== -->
 <!-- ======== PROPERTY DEFINITION ======= -->
 <!-- ================================== -->

 <property file="build.config"/>
 <property file="build.properties"/>

 <!-- ================================== -->
 <!-- ========  INITIALIZATION  ======= -->
 <!-- ================================== -->

 <target name="init">
  <path id="classpath">
   <pathelement path="C:\hetfield\fractal_2.3.1\julia\output\dist\lib\julia-asm.jar"/>
   <pathelement path="C:\hetfield\fractal_2.3.1\julia\output\dist\lib\julia-mixins.jar"/>
   <pathelement path="C:\hetfield\fractal_2.3.1\julia\output\dist\lib\julia-runtime.jar"/>
   <pathelement path="C:\hetfield\asm\asm-1.5.3.jar"/>
   <pathelement path="C:\hetfield\fractal_2.3.1\fractal\output\dist\lib\fractal.jar"/>
   <pathelement path="C:\hetfield\fractal_2.3.1\fractal\output\dist\lib\naming.jar"/>
  </path>
 </target>

 <!-- ================================== -->
 <!--         EXECUTE          -->
 <!-- ================================== -->

 <target name="execute">
  <java classname="Console" classpathref="classpath"
      fork="yes"
      failonerror="yes">
    <classpath>
        <pathelement path="C:\hetfield\fractal_2.3.1\julia\examples\ERTS\output"/>
    </classpath>
    <jvmarg value="-Dfractal.provider=org.objectweb.fractal.julia.Julia"/>
    <jvmarg value="-Djulia.loader=org.objectweb.fractal.julia.loader.DynamicLoader"/>
    <jvmarg value="-Djulia.config=etc/julia.cfg,etc/julia-tutorial.cfg"/>
    <!-- arg line="${run.parameters}"/ -->
  </java>
 </target>

</project>
```

The file section that must be edited is the <-- Initialization --> with the appropriated paths where the julia-asm.jar, julia-mixins.jar, julia-runtime.jar, asm-1.5.3.jar, fractal.jar, naming.jar files are located on the computer. After this is done we can finally (!) launch the application using the command “ant” on the distribution directory.

C.4. Example execution

The execution of the electrical vehicle simulation with the input text files (TaskList.txt and NodeTaskRelation.txt) presented before crates a window frame like the illustrated in figure C1:

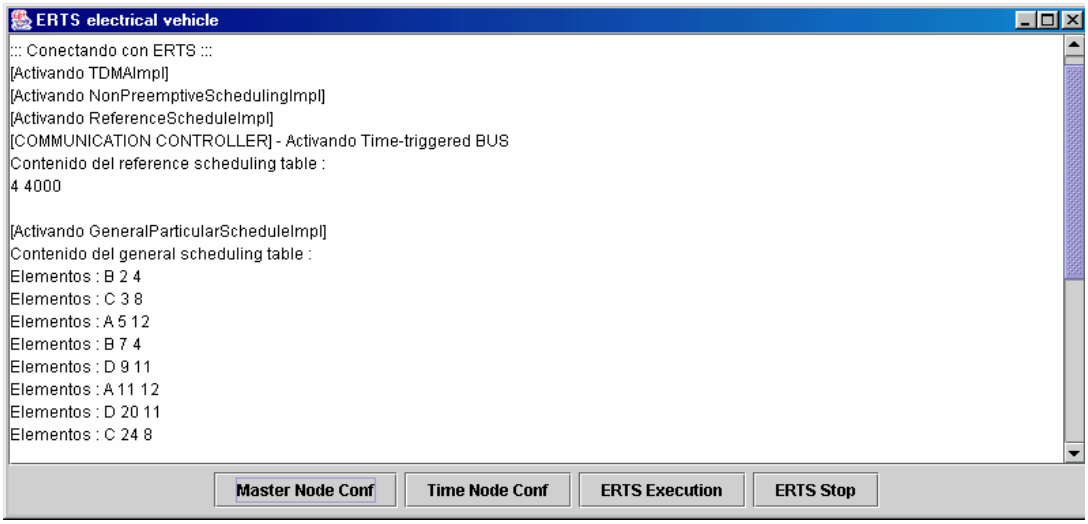

Figure C1 Electrical vehicle simulation execution frame

The figure C1 illustrates a window frame that is show when the command ant is executed. The frame contains four buttons: Master Node Conf, Time Node Conf, ERTS Execution, and ERTS Stop.

- Master Node Conf: It generates the electrical vehicle composite component, starts the master node component, defines the network time unit, generates the reference schedule table, and generates the general and the particular schedules tables.

::: Conectando con ERTS :::

[Activando TDMAImpl]
[Activando NonPreemptiveSchedulingImpl]
[Activando ReferenceScheduleImpl]
[COMMUNICATION CONTROLLER] - Activando Time-triggered BUS
Contenido del reference scheduling table :
4 4000

[Activando GeneralParticularScheduleImpl]
Contenido del general scheduling table :
Elementos : B 2 4
Elementos : C 3 8
Elementos : A 5 12
Elementos : B 7 4
Elementos : D 9 11
Elementos : A 11 12
Elementos : D 20 11
Elementos : C 24 8

Contenido del particular scheduling table :
Elementos : A 5 12
Elementos : A 11 12

Elementos : B 2 4
Elementos : B 7 4

Elementos : C 3 8
Elementos : C 24 8

Elementos : D 9 11

```
Elementos : D 20 11
```

- Time Node Conf: It starts the time node component, present the content of the reference schedule, present the content of the particular schedule that corresponds to it, and tries the communication with the time-triggered bus.

```
::: Activando TimeNode :::

[Time node] - schedule : 4 TU : 4000
[Activando ApplicativeImpl]
[Normal node] - schedule : 4 TASK : D 9 11
[Normal node] - schedule : 4 TASK : D 20 11

[NODE] - Enviando mensaje Time-triggered bus : TIME UNIT KO-A-LA!!!
[NODE] - Enviando mensaje Time-triggered bus : NORMAL UNIT KO-A-LA!!!

[MASTER NODE] - Mensaje recibido: NORMAL UNIT KO-A-LA!!!
[MASTER NODE] - Mensaje recibido: TIME UNIT KO-A-LA!!!
```

- ERTS Execution: It starts the normal node components, presents the content of the particular schedules that correspond to each normal node component, and tries the communication of every normal node components with the time-triggered bus.

```
::: Activando NormalNode :::

[Activando ApplicativeImpl]
[Normal node] - schedule : 1 TASK : A 5 12
[Normal node] - schedule : 1 TASK : A 11 12
[NODE] - Enviando mensaje Time-triggered bus : NORMAL UNIT KO-A-LA!!!
[MASTER NODE] - Mensaje recibido: NORMAL UNIT KO-A-LA!!!

[Activando ApplicativeImpl]
[Normal node] - schedule : 2 TASK : B 2 4
[Normal node] - schedule : 2 TASK : B 7 4
[NODE] - Enviando mensaje Time-triggered bus : NORMAL UNIT KO-A-LA!!!
[MASTER NODE] - Mensaje recibido: NORMAL UNIT KO-A-LA!!!

[Activando ApplicativeImpl]
[Normal node] - schedule : 3 TASK : C 3 8
[Normal node] - schedule : 3 TASK : C 24 8
[NODE] - Enviando mensaje Time-triggered bus : NORMAL UNIT KO-A-LA!!!
[MASTER NODE] - Mensaje recibido: NORMAL UNIT KO-A-LA!!!
```

ERTS Stop: it stops the time-triggered bus and finalize the electrical vehicle composite component execution.

```
::: Desactivando MasterNode :::

[Desactivando TDMAImpl]
[Desctivando NonPreemptiveSchedulingImpl]
[Deactivando ReferenceScheduleImpl]
[COMMUNICATION CONTROLLER] - Desactivando Time-Triggered Bus
```