EZPetri: A Petri net interchange framework for Eclipse based on PNML

Adilson Arcoverde, Jr.[1], Gabriel Alves Jr.[1], Ricardo Lima[1], Paulo Maciel[2], Meuse Oliveira Jr[2], and Raimundo Barreto[2]

[1] {aoaj,gabriel_alves}@ezpetri.com, ricardo@upe.poli.br
Departamento de Sistemas Computacionais – Universidade de Pernambuco
Recife, PE, Brazil
[2] {prmm, mnoj, rsb}@cin.ufpe.br
Centro de Informática – Universidade Federal de Pernambuco
Recife, PE, Brazil

Abstract. Petri net community has suffered with the lack of a standard format to represent Petri net models. This situation led to an undesirable tool incompatibility. In order to solve this drawback, the PNML has been proposed. PNML is an interchange file format for Petri nets based on XML. This paper presents a framework, called EZPetri, based on PNML. The EZPetri framework is a *perspective* of the Eclipse platform. The union of Eclipse and PNML has demonstrated to be an effective instrument for integrating Petri net tools and applications. The paper discusses the principles of the EZPetri framework, and presents three applications integrated into the EZPetri framework: *software power estimation*; *A SystemC model for Petri nets*; and *hard real-time software synthesis*. Such applications have been developed with no knowledge about EZPetri. This is a demonstration of the integration facilities provided by EZPetri. The framework is a fertile ground for combining existing of Petri net tools and applications into a single environment, offering Petri net community a new perspective of integration.

1 Introduction

Petri nets is a powerful specification language useful for modelling concurrent, asynchronous, distributed, parallel, non deterministic, and/or stochastic systems. They are also a formal specification technique with powerful methods for qualitative and quantitative analysis [mur89, rei85]. Since their introduction by C. A. Petri in 1960, Petri nets has been widely applied in many fields of science and industry.

There are numerous Petri net tools with support for specific type of net (high-level, low-level, timed, stochastic, etc.). The lack of a standard forced Petri net tools designers to create their own file format. Hereby, a model created through a Petri net tool cannot be read by other tools. This assertion is true even for tools supporting the same Petri net type.

This situation has motivated Petri net community towards creating a Petri net interchange format. In order to define a single XML-based file for any Petri net type, many proposals were presented during the International Conference on Application and Theory of Petri nets 2000. The most prominent of them was the Petri Net Markup Language (PNML)[jun00].

The development of a tool supporting PNML would be the next step. Such tool is supposed to provide facilities for integrating existing Petri net tools. For example, through importing/exporting functions. Not necessarily PNML has to be its internal format, but it is a reasonable idea (otherwise, another format would be created). Other desirable requirement of such kind of tool is the ability to be easily extended with new features.

This paper presents a new Petri net PNML-based tool, called EZPetri, which fulfills these requirements. EZPetri takes advantage of the plug-in technology offered by the Eclipse platform[ecl01].

The paper is organized as follows: this introduction; some aspects of the PNK tool are discussed in Section 2; Section 3 describes PNML; Section 4 presents the Eclipse platform; the main features of the EZPetri project are discussed in Section 5; Section 6 highlights the EZPetri perspective; Section 7 provide an outline of the EZPetri modelling environment; Section 8 discusses the EZPetri integration facilities, and describes plug-ins developed for connecting two existent Petri net tools with the EZPetri environment; Section 9 introduces three applications integrated into EZPetri; Eventually, the conclusion remarks are given in Section 10.

2 Related works

Petri Net Kernel (PNK) [kin01] is a infrastructure for building Petri net tools based on PNML. In the matter of fact, its description language lately became the PNML proposal.

PNK provides an interface which consists of several functions to access, to manipulate, and to visualize Petri net elements. Hence, it relieves programmers of Petri net tools from implementing standard functions on Petri nets, such as read/write nets, modifying net structure, as well as building a graphical user interface. Therefore, programmers can be focused on building new algorithms for analysis, simulation, or verification of Petri nets. PNK was designed for implementing new Petri net application in the Java (Python in its firsts versions) language and for integrating these applications into a single Petri net tool. Authors claim in [kin98] that PNK can be used to integrate existing Petri net tools. To demonstrate this claim, they integrated INA tool into PNK. Unfortunately, the developer must deeply understand the whole implementation of this platform, which requires a lot of effort and project time.

Besides PNK, there are many tools (i.e. PEP, INA, TimeNet, CPN/Tools, etc.) that have been developed in various parts of the world [wik96]. There are also some tools (i.e. Jarp, Netlab, P3, etc.) which provides support for PNML. These won't be discussed here, since their main proposal isn't the extensibility feature.

3 Petri Net Markup Language

The number of Petri net types and tools has significantly increased over last four decades. Such diversity represents an advance for the Petri net community. But, due to the use of specific file format, these tools are generally incompatible. The problem occurs even for tools supporting the same type of Petri nets. The lack of integration among these tools imposes serious limitations on Petri nets users productivity. Therefore, functions for importing/exporting Petri nets from/to other tools are an important requirement nowadays.

An important step was taken during the International Conference on Application and Theory of Petri nets 2000, when a number of XML-based interchange formats for Petri nets were presented. The Petri Net Markup Language (PNML) [web02] was one of these proposals. The first ISO/IEC 15909-2 working draft on PNML was released in March 2003. Figure 1 and Figure 2 depicts a Petri net example and its description in PNML format, respectively.

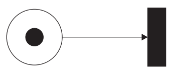

Fig. 1. Petri net example

```
<pnml>
    <net id="0">
        <place id="1">
            <graphics>
                <position x="115" y="95"/>
            </graphics>
            <marking><value>1</value></marking>
        </place>
        <transition id="2">
            <graphics>
                <position x="218" y="94"/>
            </graphics>
        </transition>
        <arc id="3" source="1" target="2"/>
    </net>
</pnml>
```

Fig. 2. PNML description

EZPetri environment provides functions for importing/exporting nets from/to other tools. Imported nets are translated to a PNML file. On the other hand, exporting functions read a PNML file and translate it to a specific file format. Indeed, all functionalities provided by EZPetri manipulate PNML files.

4 Eclipse Platform

Eclipse [ecl01] is an open source tool integration platform launched in 2001 by the IBM Corporation and other companies.

Eclipse can run on a variety of operating systems. It may be considered an open source community comprising corporate professionals, researchers, academy members and individual developers. The Eclipse users are free (and encouraged) to include new functionalities and tools.

Eclipse has a file-based approach that provides integration with other external tools and makes easy the management of different types of programming artifacts, like images, documents and source codes. Its window interface has some panes focused in the resources (projects, folders and files) that are stored in the *workspace*.

The Graphical User Interface (GUI) of Eclipse is based on *perspectives*. A perspective defines a set of *views* and *editors* arranged to fulfill the requirements of a particular task. Views are useful to navigate through resources, to provide information, or even to change values of a particular resource. Editors are used to create, edit or simply show the resources contents. Designer might develop from simple text editors to complex graphical editors, or even editors used to work with a particular kind of file.

Eclipse has been developed thinking in extensibility. The result reached by this approach is an Integrated Development Environment (IDE) which can be easily extended by the addition of new *plug-ins*. Figure 3 depicts the basic architecture of Eclipse distribution.

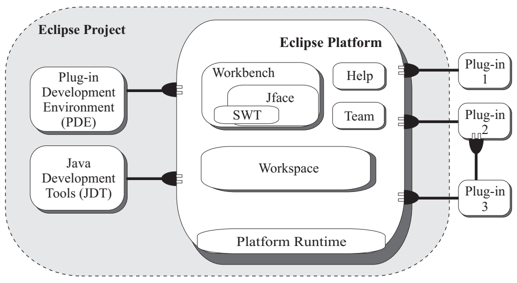

Fig. 3. The Eclipse architecture

The architecture gives Eclipse a great extensibility. Moreover, it allows the development process modularization since isn't necessary to know the implementations of all plugged components.

5 The EZPetri Project

EZPetri is an extensible Eclipse-based tool suite that supports editing Petri nets, as well as importing exporting Petri nets from/to different Petri net tools. It takes advantage of plug-in technology of Eclipse to couple existing Petri net tools and to implement new functionalities.

For instance, one may decide to build a new analysis method for Time Petri nets. Instead of implementing a new graphical interface, the developer may reuse all features already defined, such as editors, compilers, etc, and maintain the focus in what really matters: the new analysis method. Eventually, with no knowledge of EZPetri graphical implementation, the new method is integrated.

PNML forms the kernel of EZPetri. It means that any Petri net types may be represented through the PNML format in EZPetri environment. Therefore, it glues together the integration facilities provided by Eclipse with the PNML interchange format.

The project contributes to reduce the gap between members of Petri net community which uses different Petri net types, tools and file formats. Moreover, EZPetri improves productivity in the development of new products by offering several functionalities in a single development platform.

Currently, there are a number of tools that take advantage of EZPetri under development (see Section 9 for more details). Figure 4 depicts current state of the EZPetri project. It also depicts the adopted design strategy. Each plug-in accesses a single and shared PNML file. For example, the Eppc box, in this same figure, is responsible to compile export/import the PEP format to/from PNML format.

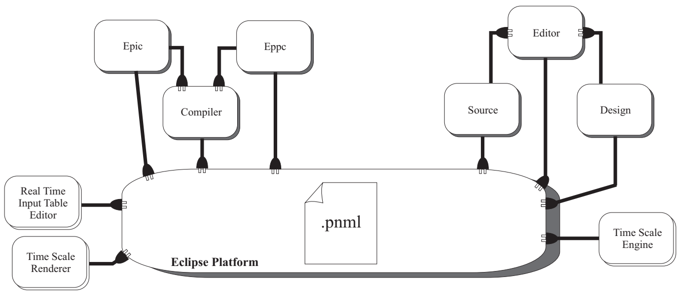

Fig. 4. EZPetri Architecture. Each plug-in accesses a single and shared PNML file.

6 The EZPetri Perspective

The design of the EZPetri perspective took into account an Eclipse problem known as *loss of context*. The problem occurs when a user doesn't know where they are in the *User Interface* (UI) or where to go to complete a task. A frequent cause of the problem is the inclusion of an excessive number of *views* and *editors* in the perspective. For instance, an object action may differ between two distinct *views* or menu items may vary with *views*. Therefore, EZPetri perspective reduced the number of *views* and *editor* as much as possible. This decision yielded an intuitive (consistent) platform. As can be seen in Figure 5, EZPetri perspective is composed of three *views*: *navigator view*, *properties view*, and *outline view*. There is also the *editors area*, the *menu bar* and *tool bar*.

Menu bar includes functionalities found in many commercial tools: create new files, help, arrange views and so on. The *tool bar* works as a shortcut for functionalities most used of the menu bar. Some options of these bars are context-sensitive and are available only when a specific view or editor is focused.

Navigator view enables users to show the workspace area. It is useful to manage projects, folders and files. In addition to ordinary tasks, such as create

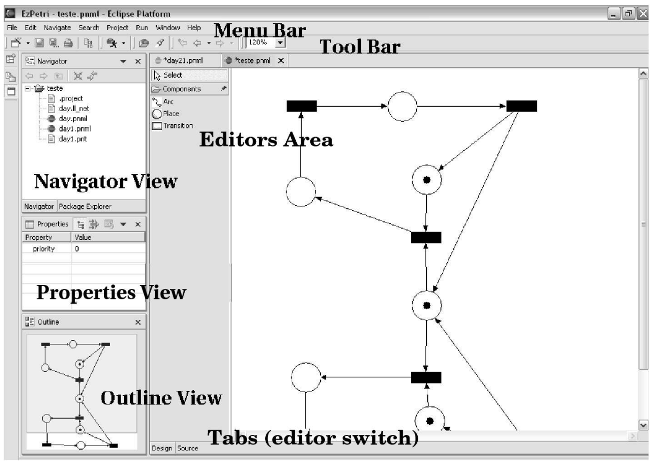

Fig. 5. EZPetri Views and Editors

a new folder or invoking an editor to modify a file, specific functionalities are available. For example, user may right-click a PNML file, to choose the *Compile To* option, and to translate PNML into the file format of a specific Petri net tool.

Property view is useful to show and modify attributes of selected objects. For instance, one may select a place and edit its name, marking, priority, etc.

Outline view displays attributes of specific files. A tree structure is usually adopted in the outline view to represent attributes. A non-conventional usage of outline view is provided by the EZPetri editor. It presents an overview of the whole Petri net model, as seen in Figure 5, when the EZPetri graphical editor is activated. Such view is useful for large models. The user may choose the part of the specification to be presented in the *editors area* by clicking in the corresponding point of the outline view.

Editors area is a large blank box where specific file formats are manipulated. EZPetri provides a *multipage editor* which is the parent of several editors. It uses tabs to switch between different children editors, i.e., source (PNML), graphical (Section 7). For instance, when user presses over *Design* tab in the parent multipage editor, the graphical editor will be displayed.

All editors in the *multipage editor* are supposed to implement a synchronization interface. Such interface defines methods to translate the editor contents into PNML, and vice-versa. When one selects an editor through the *multipage editor* tab, the parent editor executes the method in the corresponding interface to synchronize the contents of the currently opened editor and the requested editor. In order to not compromise performance, synchronization is performed only when either users switch between editors, or the work is saved.

7 Modelling Environment

EZPetri contains a plug-in for graphical edition of Petri nets. It provides functionalities found in many Petri net graphical editors: drawing by select-and-click, drag and drop, resize, undo, redo, zoom in, zoom out, select all, select all of the same type, etc. The editor gives flexibility for changing the source/target of an arc by dragging it to another valid source/target. Some functionalities are enabled in specific situations. For instance, *select all of the same type* is enabled only when one transition or place, but never both at same time, is selected.

The editor includes an *overview* of the Petri net model in the *outline* view. The overview is a small view with a picture of the whole Petri net model. The corresponding area clicked in the overview becomes visible in the *editors area.* This innovative functionality is useful for large projects. It allows users to rapidly move to a specific point of the model by clicking in the corresponding area (see gray area in Figure 5) of the *outline* view. It also made possible to take off the *page* concept used by PNK. Besides nice to modularize large models, the pages make the net edition confusing to users, since they have to manage many windows to edit a single net.

Alignment functionalities are useful to organize objects in the model. Selecting two or more objects and right-clicking over them, shows a popup menu that provides a number of alignment functionalities. For instance, *Align Left* aligns the left side of all selected objects with the left side of the last selected object. This includes object labels. The last selected object is identified by a black border. Figure 6 exemplifies the usage of the alignment functionality.

Similarly, *same size* functionality applies the same width and height to all selected object. The width and height used will be that of last object selected.

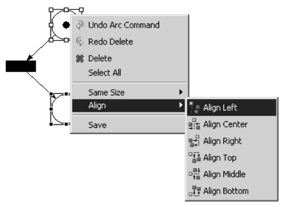

Fig. 6. Aligning left two places of the net

The graphical editor was developed using the Eclipse plug-in *Graphical Editing Framework*(GEF)[hud03]. GEF assists the task of building several GUIs. It is based on the widespread *Model-View-Controller* (MVC) pattern, which recommends the definition of different classes for the implementation of *modelling*, *viewing* and *controlling* functionalities. The Figure 7 represents an MVC architecture overview.

Model classes encapsulate data used to represent Petri net elements (place, transition, arc, etc). *View classes* are responsible for painting graphical elements

Model
Controller
View

Fig. 7. The MVC architecture

in the editors' area. *Control classes* is the bridge between the *model* and the *view*. So, they are responsible to propagate changes in the view to the model and vice-versa.

8 Integration of Petri Net Tools

Currently, two well-known Petri net tools are integrated to EZPetri. The Integrated Net Analyzer (INA) [sta99] and the Programming Environment based on Petri nets (PEP) [bes96]. INA provides a rich set of analysis techniques. PEP is a comprehensive set of modelling, compilation, simulation and verification components, linked together within a Tcl/Tk graphical user interface.

The *EZPetri PNML INA Compiler* (Epic) and *EZPetri PNML PEP Compiler* (Eppc) converts, respectively, INA and PEP formats into PNML, and vice-versa. Subsection 8.2 describe the implementation of Epic and Eppc.

8.1 Tool integration principles

The implementation of EZPetri compilers has followed two principles. First, the compiler is supposed to translate from a specific file format into PNML, and vice-versa. The second principle is important to improve the compiler modularity. The core of the compiler (parser, helper, etc) is part of a single plug-in. The GUI should be kept in a different plug-in. For instance, Eppc compiler was implemented through the *Eppc core* and *Eppc UI* plug-ins (see Figure 4).

8.2 Epic and Eppc

This subsection describes the implementation of the *EZPetri PNML INA Compiler - Epic* and the *EZPetri PNML PEP Compiler - Eppc* compilers. Currently, both Epic and Eppc support Place/Transition Petri nets. Additionally, Epic supports Time Petri nets.

INA provides a simple textual representation for Petri nets. This fact yields a simple implementation of the INA parser. Design decisions regarding the implementation of PNML parser recommended the usage of the Document Object Model (DOM) [w3c02]. DOM is a platform which allows accessing and updating the content of XML files. DOM was used to parse and also to write PNML files.

Both INA and PNML parsers generate a similar object structure which represents the net information.

PNML defines more information than INA requires. For instance, INA file format does not contain graphical information. This extra information in PNML is discarded during generation of the INA file format. On the other hand, INA supports transitions with time, but the current version of basic PNML does not. This information was included in PNML using the *toolspecific* tag[3].

The implementation of Eppc employed the same strategy and technology adopted by Epic. Parse trees of both compilers are similar.

9 EZPetri applications

EZPetri is a fertile ground for applications based on Petri nets. This section describes three applications developed with no knowledge about EZPetri and their integration into the framework.

The integration of these applications are a demonstration of the integration capability of EZPetri.

9.1 Software power estimation

This subsection describes the ongoing work for integrating the software power estimation framework presented in [meu04] with the EZPetri environment. That framework was conceived to be applied to embedded systems design in which energy consumption is the main concern. In particular, in order to capture power consumption and its distribution along the code so that to implement code optimization and software-hardware migration [sti02].

The software behavior is modelled in Colored Petri Net (CPN). The model captures the behavior of individual instructions computing the energy consumption of each instruction (instruction base-cost) and the inter-instruction consumption. Both parameters comes from a previous instruction power-model. Initially, an instruction level model for the 8051 architecture was proposed, but it is also possible to adapt the model for different architectures at either higher or lower abstraction level. A compiler translates the binary-code to CPN-Model. The model can be extended to deal with hardware power models under the trace-driven approach [giv01], the CPN engine would be able to deal with parallel instructions allowing modelling complex hardware cores and processor architectures. The integration on the EZPetri environment resolves the *lost of context problem* present when handling the power model under a general purpose CPN tool. The EZPetri offer a front-end to tackle with specifics analysis function from the CPN-Model. As back-end is used a widespread CPN engine, the CP-NTools. The CPNTools is integrated to EZPetri via a TCP/IP channel [gal01] allowing the construction of a *Grid*. The CPN-Model would be dispatched for each engine instance (spread over a net) , allowing to perform parallel analysis, hugely improving the analysis time cost. In a first approach, an analysis tool with a single engine will be developed.

[3] Readers are reported to [web02] for more information about *toolspecific* tag.

Using a plugged compiler, the EZPetri takes the machine-code program and the library with CPN models of 8051 ISA[4] in order to generates a CPN model representing the 8051 program. Both the library and the resulting program model are represented in PNML. The library is transparent for EZPetri users. During the machine-code importation, the compiler automatically consults the specific library to generate the CPN model. At this point, the model can be processed by a internal engine or sent to a remote engine generating a specific output format such as the CPNTools and Design/CPN format. Through the EZPetri interface the user can request analysis that will be performed by the engine under control of EZPetri environment. The results is shown by the EZPetri user-interface and by reporting files. The Figure 8 depicts this mechanism.

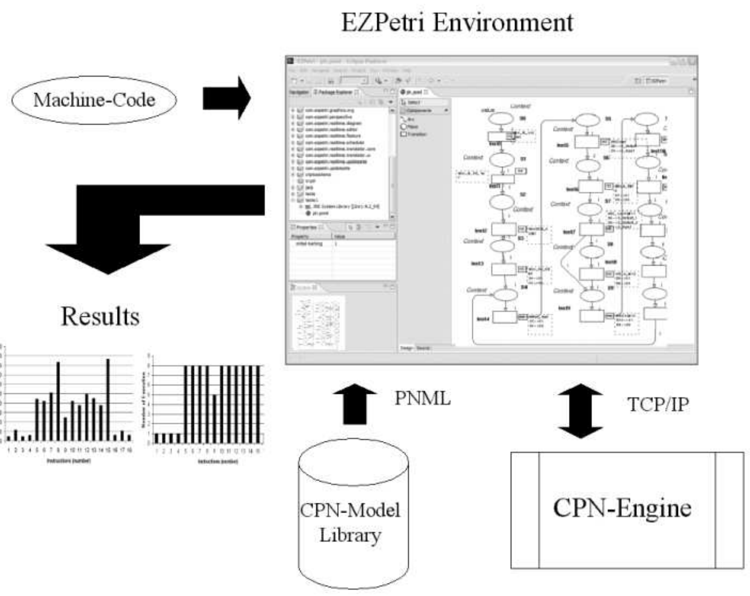

Fig. 8. Power estimation environment

9.2 A Petri net model for SystemC

The Open SystemC Initiative (OSCI) is a collaborative effort among a broad range of companies to support and advance SystemC as a *de facto* standard system-level design. SystemC specifications are essentially C++ programs. Its class library includes constructors for modelling concurrency, reactive behavior and hardware timing. SystemC is a modelling platform consisting of C++ class

[4] Instruction Set Architecture

libraries and a simulation kernel for System on a Chip [syc03] (SOC) design at the system-behavioral and register-transfer levels, therefore supporting embedded system design methodology.

The design methodology adopted takes into account a Petri net model as an intermediate model allowing qualitative analysis, verification and metrics estimations [mac99].

This work proposes the first formal model for SystemC. The model is based on time Petri net. The proposed translation methodology considers two consistent Petri models for SystemC, each one more suitable for a given kind of analysis. The first model is based on High Level Petri Nets (HLPN) [jen94]. This model represents the system's behavior at a high level of abstraction and provides more succinct and manageable descriptions than low-level nets and still offers a wide range of analysis methods, especially those based on state space analysis. The second model is based on time Petri nets (low-level Petri nets). Although low-level nets tend to be very large for even medium size systems, the analytical methods developed for such models are better understood, especially those based on structural analysis, than analytical methods for HLPN. Besides, some specific analysis, such as dependency analysis (data-dependency and control-dependency analysis), are more directly performed if a low-level model is available. The proposed low-level model is represented by two interconnected sub-models that describe both control and data-dependencies.

A compiler for translating SystemC system design into *Petri Net Markup Language (PNML)* was developed. The compiler is another plug-in of the EZPetri framework.

9.3 Hard Real-Time Scheduling Synthesis

Embedded hard real-time systems are dedicated computer applications having to satisfy stringent timing constraints. In other words, systems must guarantee that all tasks complete before their deadlines. A failure to meet deadlines may have serious consequences such as resources damage or even loss of human life. Examples of embedded hard real-time systems include systems for military applications, flight mission control, traffic control, production control, robotics, and so on. In order to meet timing requirements, scheduling performs an important role.

This section aims to present a solution, based on time Petri net formalism, for the problem of finding feasible schedules considering time-critical systems. However, this problem is known to be NP-Hard in its general form. Nevertheless, differently from other works that have the same objective, the proposed approach describes a method for modelling the system, not just the scheduling problem. This feature allows not only the computation of a feasible scheduling, but also it makes possible to extend system's constraints (such as memory and energy requirements) and automatic code generation.

The input specification is composed by a set of periodic preemptive tasks that run on a set of pre-allocated processors. Each task has discrete and bounded timing constraints (release time, worst-case execution time, deadline and period).

The specification also provides arbitrary inter-task relations, such as precedence and exclusion relations.

The system is modelled by time Petri nets through building blocks. There are specific blocks for modelling task arrival, task structure (release, computation, processor granting and releasing), deadline checking, exclusion relations, precedence relations, processors, inter-processor communication, etc.

In order to guarantee that timing constraints are satisfied, the proposed scheduling method is *pre-runtime*, where the schedule is computed entirely off-line, can achieve 100% processor utilization, reduces context switching, its execution is predictable, and excludes the need for complex operating systems.

In order to find a feasible pre-runtime schedule, this work uses *state space exploration* on a timed labelled transition system (TLTS) derived from a time Petri net (TPN) model. In spite of the fact that a scheduling can be found using this strategy, it may be limited by the excessive size of its state space. This problem comes up due to the analysis based on the interleaving of concurrent activities. This exponential growth is known as the *state explosion problem* [val98]. The proposed method applies minimization techniques (partial-order reduction and undesirable states elimination) on the state space in order to maintain the state space size under control. Furthermore, the proposed algorithm is a depth-first search method on a TLTS. Thus, the TLTS is partially generated, according to the need. When successful, the result of the proposed algorithm is a sequence of transition firings that represents a feasible schedule found.

Table 1. Simple Task Timing Specification

Task	release	WCET	Deadline	Period
τ_1	0	2	7	8
τ_2	2	3	6	6

As an example, suppose we have the task timing specification depicted in Table 1. This specification is entered into the EZPetri framework by using a *task/message editor plug-in*, that is a graphical editor for specifying timing information of both tasks and inter-tasks message passing, as well as, inter-task relations. After that, the time Petri net model is automatically generated by a *Petri net model generator plug-in* and this model is used for finding a feasible schedule. When successful, the EZPetri framework shows a timing diagram representing the feasible schedule found by using a *schedule renderer plug-in*. Figure 9 shows the timing diagram representing a feasible schedule found for the task timing specification presented in Table 1.

Using the EZPetri framework is easier for integrating several tools. In this specific situation, we specify the system and have as result the timing diagram that represents a feasible schedule found. Moreover, all formal activities, from the specification up to the final result, are hidden from the final user.

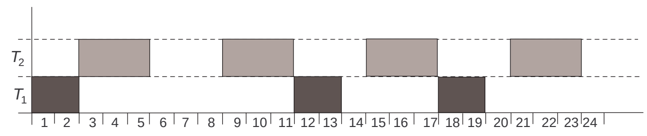

Fig. 9. Timing Diagram for Example in Table 1

10 Conclusions

The paper described a framework for the development of Petri net applications and integrating existent Petri net tools. The framework joins together the PNML interchange file format with extensions facilities provided by the Eclipse platform. The combination of these technologies demonstrated to be an effective instrument for integrating Petri net tools and applications.

A description of the EZPetri architecture and its implementation were provided. The paper detailed the EZPetri perspective, including their *editors* and *views*. The EZPetri modelling environment was also briefly discussed.

The successful and quick integration of two existent Petri net tools, namely INA[sta99] and PEP[bes96], into EZPetri demonstrated the potential of the framework for importing/exporting Petri net file formats of other Petri net tools. It must be highlighted that the PEP and INA plug-ins required less than sixteen hours (each) to be implemented.

We demonstrated that EZPetri can be extended to incorporate Petri net applications. In particular, we described three Petri net based applications developed with no knowledge about EZPetri and their integration into the framework. The *software power estimation* is under development, but the *SystemC model for Petri nets*, and the *hard real-time software synthesis* are already integrated into the framework.

We believe that EZPetri framework offers more than another tool for Petri net community. It offers them a perspective of real integration. Through collaboration, corporate professionals, researches, members of academia, and individual developers can further the goal of producing interoperable Petri net based products and offerings.

References

[bal1] Balbo, G. Introduction to Stochastic Petri Nets. LNCS 2090, Springer-Verlag, Lectures on Formal Methods and Performance Analysis, 2001.

[bes96] Best, B. and Grahlmann, B. PEP - more than a Petri net tool. LNCS 1055, Springer-Verlag, p397-401, 1996.

[cpnt] CPN Group (2002): CPN Tools. www.daimi.au.dk/CPNtools.

[dcpn] Design/CPN Online. www.daimi.au.dk/designCPN.

[ecl01] Eclipse Platform Technical Overview. Object Technology International Inc., July 2001.

[gal01] Gallasch, G. and Kristensen, L. M. Comms/CPN: A Communication Infrastructure for External Communication with Design/CPN. 3rd Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01) / Kurt Jensen(Ed.). DAIMI PB-554, Aarhus University. Aug. 2001

[giv01] Givargis T. and Vahid, F. and Henkel, J. Trace-driven System-level Power Evaluation of System-on-a-chip Peripheral Cores. Asia South-Pacific Design Automation Conference. Jan. 2001.

[hud03] Hudson, R. How to get started with the GEF. http://www.eclipse.org/gef. 2003

[jen94] Jensen, K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Uses. EACTS Monographs on Theoretical Computer Science, Springer-Verlag, 1994.

[jun00] Jüngel, M. and Kindler, E. and Weber, M. Towards a Generic Interchange Format for Petri Nets, i 21st International Conference on Application and Theory of Petri Nets Aarhus, Denmark, June 26-30, 2000.

[kin98] Kindler, E. and Oschmann, F., The Petri Net Kernel: An INA-Pilot. In J. Desel, P. Kemper, E. Kindler, A. Oberweis: Workshop Algorithmen und Werkzeuge für Petrinetze, Oct. 1998.

[kin01] Kindler, E. and Weber, M., The Petri Net Kernel: An Infrastructure for Building Petri Net Tools. Software Tools For Technology Transfer; DOI 10.1007/s100090100055, Springer Verlag Online First, 2001.

[kin01b] Kindler, E. and Weber, M., A Universal Module Concept for Petri Nets. An Implementation-Oriented Approach. Informatik-Bericht Nr. 150, Humboldt-Universität zu Berlin, April 2001.

[mac99] Maciel, P. and Barros, E. and Rosenstiel, W. A Petri Net Model for Hardware/Software Codesign. Design Automation for Embedded Systems Journal, Kluwer Academic Publishers, $n^0 4$, Vol 4, October, 1999.

[mar85] Marsan, A. M. and Balbo, Bobbio, A. and Chiola, G. and Conte, G. and Cumani A. On the Petri Nets with Stochastic Timing. International Workshop on Timed Petri Nets. IEEE press. Torino, Italy, 1985.

[mer76] Merlin, P. M. and Faber, D. J. Recoverability of Communication Protocol Implications of Theoretical Study. IEEE Transaction Communication, vol COM-24, September, 1976.

[meu04] Oliveira, M. and Maciel,P. and Barreto, R. and Carvalho, F. Towards A Software Power Cost Analysis Framework Using Colored Petri Net. PATMOS 2004. LNCS Kluwer Academic Pubishers. Sep. 2004.

[mol81] Moloy, M. K. On the Integration of Delay and Throughput Measures in Distributed Processing Models. PhD thesis. UCLA, USA, 1981.

[mur89] Murata, T. Petri Nets: Properties, Analysis and Applications. Proceeding of The IEEE, 1989.

[pet81] Peterson, L. J. Petri Net Theory and the Modelling of Systems. Prentice-Hall, Englewood Cli s, NJ, USA, 1981.

[pet62] Petri, C. A. Kommunikation mit Automaten. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962, Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65-377, v.1, 1966, Pages: Suppl. 1, English translation.

[ram73] Ramchandani, C.: Analysis of Asynchronous Concurrent Systems by Timed Petri Net. Technical Report n 120, Laboratory of Computer Science, MIT, Cambridge, MA, USA. 1973.

[rei85] Reisig, W. Petri Nets. An Introduction, Volume 4 of Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[roz98a] Rozemberg. G. and Reisig. W., Informal Introduction to Petri Nets, Lecture Notes on Petri Nets I: Basic Models. Springer Verlag. 1998.

[roz98b] Rozemberg, G. and Engelfriet, J. Elementary Net Systems. Lecture Notes on Petri Nets I: Basic Models. Springer Verlag. 1998.

[su02] Su, S. and Hsiung, A. Extended quase-static scheduling for formal synthesis and code generation of embedded software. *In CODES*, May 2002.

[sta99] Starke, P. H. and Roch, S. INA - Integrated Net Analyzer - Version 2.2. Humbolt Universität zu Berlin - Institut für Informatik. 1999.

[sti02] Stitt, G. and Vahid, F. Hardware/software partitioning of software binaries. Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design. San Jose, California. ACM Press. 2002.

[syc03] SystemC Reference Manual 1.0. Synopsys Inc., http://www.systemc.org.

[val98] Valmari, A. The state explosion problem. *LNCS: Lectures on Petri Nets I: Basic Models*, 1491:429–528, June 1998.

[web02] Weber M., Kindler, E. The Petri Net Markup Language. Petri net Technology Communication Systems. Advances in Petri Nets 2002.

[wik96] Wikarski, D. Petri Net Tools: a Comparative Study. ISST-Bericht Nr. 39. Fraunhofer ISST. Berlin, 1996.

[w3c00] World Wide Web Consortium (W3C) (ed.). Extensible Markup Language(XML). http://www.w3.org/XML/.

[w3c02] World Wide Web Consortium (W3C) (ed.). Document Object Model. http://www.w3.org/DOM.

[zub91] Zuberek, W.M. Timed Petri Nets Definitions, Properties and Applications. Microelectronic and Reliability, vol. 31, no.4, pp 627-644, 1991.