A software platform for the integration of discrete event systems tools

M.P. Cabasino(*), L. Contini(**), A. Giua(*), C. Seatzu(*), A. Solinas(**)

Abstract

This paper presents a software platform for the integration of discrete event systems tools that is being developed within the FP7 European project DISC "Distributed supervisory control of complex plants". The objective of this software platform is to integrate several tools dealing with Petri nets and automata. The purpose is twofold: first allow for a rigorous comparison of the methods and algorithms developed by the DISC project partners, second provide a packaged tool which would facilitate transfer of these techniques to the end users. The interchange format is compliant to the ISO standard Petri Net Markup Language PNML. The platform includes a series of plug-ins and adapters to manipulate/transform the different file formats supported by the platform.

Published as:

M.P. Cabasino, A. Giua, C. Seatzu, A. Solinas, L. Contini, "A software platform for the integration of discrete event systems tools," *CASE11: 7th IEEE Conference on Automation Science and Engineering* (Trieste, Italy), August 2011.

(*)M.P. Cabasino, A. Giua and C. Seatzu are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy. E-mail:{cabasino,giua,seatzu}@diee.unica.it; (**) L. Contini and A. Solinas are with Akhela s.r.l, Cagliari. E-mail:{luca.contini,antonio.solinas}@akhela.com.

This work has been partially supported by the European Community's Seventh Framework Programme under project DISC (Grant Agreement n. INFSO-ICT-224498).

I. INTRODUCTION

The many application areas in which Discrete Event Systems (DES) arise and the different aspects of behavior relevant in each area have led to the development of a variety of discrete event models. Thus it is not surprising that the majority of existing tools in the DES domain are not able to communicate with each other, or can do that only with heavy restrictions. This means that if a user, for example, wants to simulate a system for which two different models are available (e.g., a Petri net and a finite state automaton) using different tools pertaining to the two models, there is no way to convert the input file for one tool into an input file compatible with the other one in an automated way. This also means that the output of a tool cannot directly be used as an input for a different tool if the file format is not compatible. A similar situation also applies if one only considers tools developed for a single model (or family thereof) such as Petri nets (PNs).

The scope of the FP7 European project DISC "Distributed supervisory control of complex plants" [1] that will end on 31 December 2011, is the design of discrete event supervisors and fault detectors for large scale plants exploiting the concurrency and the modularity of the plant model. The project merges the effort of 2 industrial, 1 governmental and 7 academic partners and entails a strict collaboration with North-American researchers. It was expected that the same design techniques will be applied to heterogeneous models, such as timed or fluid PNs, finite state machines, timed automata, etc. Thus, an important part of the project is the development of a common software platform that will integrate, along with tools already existing, new algorithms developed by the partners during the course of the project. The reference model chosen for the platform was PNs, although some capabilities for other simpler models (such as automata) are also provided.

The software platform will give the user the ability to integrate into a unique place both tools from the DISC project and already existing tools. The main problem to be solved was the fact that each tool has its own file format, so the platform has a specific section in charge of converting one file format into a different one. The platform also allows the user to convert a model into a different one, in particular it allows the conversion between a Petri net and an automaton. These conversions are performed through a combination of plugins and adapters (small executables). The conversion between a file format and another one, thus the necessary

combination of plugins and adapters, is predefined but can be easily modified. The user never needs to know any detail about the file formats. For example, if the user creates a PN using PIPE3 [2] and wants to simulate it with HYPENS [3], using the File Conversion Section (see Section III) he/she can directly convert the PIPE3 file format into the HYPENS input file format without having to know anything about the two file formats. The fact that each plugin or adapter is capable of making only a single transformation will give the user the necessary flexibility of combining them in order to make multiple transformations using the scripting section of the platform. A particular case is when the user wants to convert a Petri net into an automaton. In this case the tool TINA is used in a completely invisible way to extract the reachability graph of the net and use it to generate the corresponding automaton. Since the PN model is predominant in the DISC project, the PNML (Petri Net Markup Language) has been chosen as a starting point to create a common language to describe PNs all over the platform and across all types of transformations. The new extended version of PNML has been called PNML_DISC and it is able to handle hybrid PNs and tools dealing with distributed systems.

Note that, the transformation process performed using plug-ins and/or adapters sometimes implies a model transformation. In this case the target model may require additional information, which the user must enter properly during or after the transformation phase. To this aim, the PNML_DISC has been enriched with respect to PNML with a number of tags in order to be able to describe all models supported by the platform. The software platform, according to the type of transformation, will ask the user to enter the missing information or will skip the superfluous information.

Tools currently supported by the platform are: PIPE3 [2], TINA [4], DESUMA [5], HYPENS [3], Petri toolbox [6], PN_DIAG [7], PN Toolbox [8], SimHPN [9].

The literature in this area is sparse. The DISC software platform has similar interchange objectives to those of the Compositional Interchange Format for Hybrid Systems (CIF) developed in the framework of other European research activities such as the network of excellence HYCON [10] and EU FP7 project MULTIFORM [11]: CIF supports inter-operability of a wide range of tools by means of model transformations to and from the CIF. However, the formats so far developed by CIF do not address the discrete event models. In fact, CIF does not support PNs while our software platform is mainly focused on discrete event systems. Moreover, CIF is too general for our goal and for this reason a tool integrating it would be too complicated.

II. OBJECTIVES

The software platform aims to integrate, along with other existing tools suggested by the DISC partners, the algorithms developed during the course of the project. Its purpose is twofold:

- providing a packaged tool which would facilitate transfer of these techniques to end-users;
- allowing the user to compare different methodologies and tools.

To ensure a broad dissemination of the software platform among academia and industry, the following features are guaranteed:

- releasing the software as open source;
- interfacing to standard DES modeling and analysis tools;
- allowing for import/export of files from/to different tools;
- providing the software platform with a uniform user interface.

To implement the features described in the third item above, an extension of the standard PNML has been created. As a result, a more complete language to describe PNs has been integrated into the platform. This language has been used as a central point for all the file transformations, thus allowing each tool supported by the platform to communicate to the other tools. The PNML format has many advantages, but it is only able to describe high level, P/T or symmetric PNs. Since the software platform supports different types of PNs, such as hybrid PNs, and tools dealing with distributed systems, we need to extend the standard PNML format. The result is a format called PNML_DISC described in detail in Section V. This format includes all the information needed to describe hybrid, labeled PNs, i.e., PNs where two or more transitions may share the same label, and other information that will be used by the tools dealing with distributed systems.

Note that, the fact that the platform will be released as an open source software does not imply that all the tools used by the platform are actually open source. As an example the platform uses and supports Matlab which is not open source, but many features of the platform itself can be used anyway without Matlab. By definition the platform is like a “container” connecting tools together and this container is actually open source.

III. FUNCTIONALITY

A platform is a crucial element in software development. It might be simply defined as “a place to launch software”. With this application the intent is to go even further. The idea is to

create a modular, open and easy application allowing the user to use all the tools available from the partners starting from a unique place, with a unique graphic user interface.

The platform will be "scalable", i.e., other plug-ins, adapters and tools can be added to the platform without the need to change the software. In order to support a new tool the user must provide a proper plugin/adapter converting the PNML_DISC into a proper tool input file format.

The main goals when designing the platform were:

1) The user must be able to interact with the platform through a unique graphical user interface. The interface must be flexible and allow the user to decide which tools to include into it.
2) The user must be able to launch the tools directly from the Graphical User Interface (PIPE3, TINA, and DESUMA) to design/simulate a PN or an automaton.
3) The user must be able to convert the output file format from a tool into a format compatible with a different tool or simulation software using specifically created plug-ins and adapters.
4) The user must be able to manually edit file formats.
5) The user must be able to call conversion tools from a command line.
6) Display, when possible, the Matlab output of the simulation tools running on Matlab.

The software platform is divided into 4 main areas (see Fig. 1):

- *Tools*: This area has been developed to transfer different techniques to end users.
- *File Conversion*: This area allows different tools to communicate each other.
- *PNML Analysis*: This area helps the user to verify the PN structure described through PNML.
- *Script Manager*: This area allows rigorous comparison of methods and algorithms.

The other three tabs (areas), namely *Conversion Matrix*, *Plugins/Adapers* and *File Formats*, are just user information tabs and will be described later.

— The Tools area allows the user to add/remove external tools to/from the software platform. This creates a unique place where the user can see and launch all the available tools. It is divided into 3 sub-areas: (a) Graphic Tools; (b) Matlab Tools; (c) Other Tools.

This division has no theoretical relevance and has only been introduced to separate the different tools in the platform from an usability point of view.

The Graphic Tools sub-area handles all the external tools which usually have a graphical user interface and then they can be simply used by launching the Graphical User Interface (GUI) and

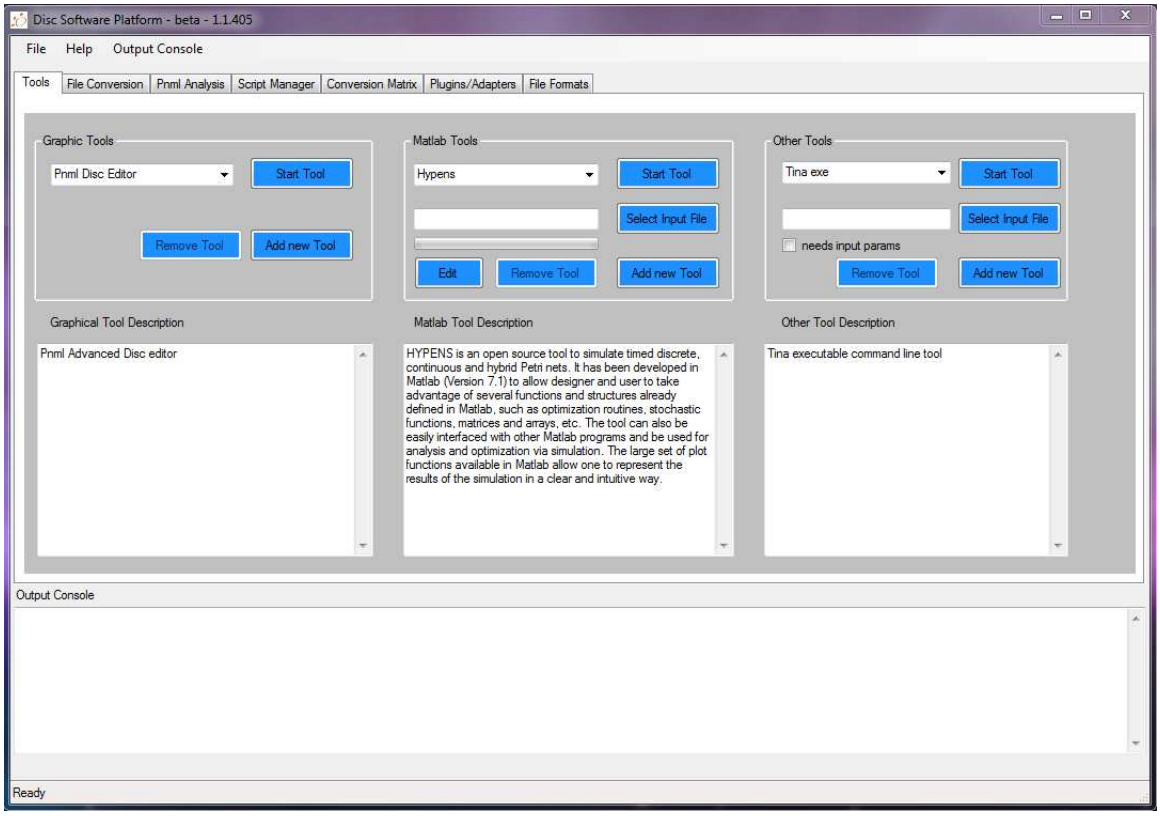

Fig. 1. Main application window with tools area selected.

interacting with that.

The Matlab Tools sub-area handles the Matlab-based tools and has an additional field for the input file. This is necessary because usually a Matlab tool needs an input file to work with, and that file can be selected by the user.

The Other Tools sub-area has been created in order to integrate tools not having a GUI, for example command line tools. This section has characteristics similar both to the graphic tools and the Matlab tools sub-areas. For example a command line tool can be added to the platform in this area and launched from there passing the parameters through an input file.

When adding a new tool to the platform there are two main scenarios:

1) the tool is already supported by the platform (see platform documentation in [12]);
2) the tool is not currently supported by the platform.

In the first case, the user must simply use one of the 3 sub-areas in the Tools section to add the tool. If the tool is an executable having a graphic user interface, then the GUI sub-

section can be used. If the tool is a command-line tool, the “Other Tools” section can be used. If the tool is a Matlab tool, then the “Matlab tools” section is the right choice. When adding a Matlab tool, one important thing has to be taken into account: the Matlab tools must have a single Matlab file calling all the necessary tool functions. This is because the platform is able the launch one Matlab file at a time. The input file, in this case, is a file containing the input matrices/vectors/variables for the above Matlab tool.

In the second case, the user must first create the necessary plugin(s) and/or adapter(s) and add them both to the platform and to the conversion matrix file in order to have the platform recognize them properly. Then the tool can be added using the first case procedure.

As a future work, two improvements can be implemented: the first one is making easier to include new plugins and adapters in the platform and simplify the conversion matrix modification with respect to the new plugins, adapters. The second improvement is the implementation of an automated file format conversion implemented in order to give a file as an input to a tool even if not directly compatible with it, having the file conversion done in an invisible way.

As it can be seen in Fig. 1 below each sub-area there is a box for the description of the corresponding tool. At the bottom of the window there is the output console. This area is common to all tabs in the application. It shows useful information to the user according to the current selected tool or area. For example if the user launches a Matlab tool from the Tools area, the result will be shown in the output console.

— The File Conversion area is the place where the user can select and convert a file produced by a tool into another file format compatible with a different tool. In this area the user can also edit both the source file and the target file. In order to convert a file the user can select the source file on the left side (the file format will be automatically detected by the platform in most cases, if not, the user must select the input file type manually before proceeding to the conversion) and then press the button “convert” and select the target file format. Pressing the button “OK” the file conversion automatically starts. The result of the conversion is presented in the output console at the bottom of the main window. The editor of the source and target files is also user-selectable.

This conversion is performed through a combination of plugins and adapters. This combination is written in a particular file called “conversion_matrix.txt” and can be modified by the user in order to add new file formats to the platform. Anyway, if not needed, the combination is

completely invisible to the user who has just to choose the starting file format and the target file format. If the platform does not support a particular file conversion a message is shown to the user.

— The PNML Analysis area is dedicated to the analysis of the properties of a PN described with the common interchange format called PNML. Opening a PNML file in this area starts two processes: the first one extracts the properties which are explicitly present in the PNML file such as the number of places and transitions of the PN, while the second one uses the tool TINA [4] in order to perform a more detailed analysis like the boundness of the net. Also in this case the result is presented in the output console at the bottom.

— The fourth area is the Script Manager. In this area the user can create and run scripting files (e.g. batch files, Matlab files, sh). This area is really useful when the user wants to take advantage of the modularity of the software platform. For example, when performing a file format conversion, the platform combines a number of plug-ins and/or adapters in order to go from the original file format to the target file format. This sequence is normally embedded into a conversion matrix (see also the fifth tab in Fig. 1). But if the user wants to combine the plug-ins and the adapters in a custom way, he/she can create a script to solve the problem. Furthermore, the user can combine conversion sequences and tool launching all in the same script, making possible the creation of repetitive and complex tasks.

Moreover, this area supports a drag&drop system in order to drag the plugins/adapters directly into the script. This action causes the plugin or adapter to be copied into the script folder thus simplifying the script readability itself. Also, an “add file” button has been added to allow the user to add to the script whatever type of file (both ascii, Matlab and executables) having the corresponding script line added automatically to the script.

Finally, the Script Manager area can also be useful to make a comparison between different tools. This comparison is currently done manually by the user using the output console.

— The remaining 3 areas are: Conversion Matrix, Plugins/Adapers and File Formats. They currently have only descriptive purposes. The Conversion Matrix shows the combination of plug-ins and adapters necessary to go from a file format to another. The file formats are described in the last tab. The Plugins/Adapers (P/A) tab contains a list of the available plug-ins and adapters and the directory path where the P/A are located.

IV. ARCHITECTURE

Most of the tools developed by the partners are Matlab based, while the tools developed by third partners run on Windows. To integrate these two environments, interaction mechanisms from the user-interface and Matlab/Windows tools have been created.

The software platform runs on Windows, and requires Matlab only if a Matlab based software tool is to be used.

A graphical user interface has been created in Windows and offers the following functionalities:

- Menus for adding, removing and interacting with all different tools;
- Transformation of PNs and automata models in different input/output formats corresponding to different models and tools.

The executable files that convert from (resp., to) the platform standards are called adapters (resp., plug-ins). If required, the user may also choose to interact with the software platform with a shell that accepts commands lines or by means of running batch files (scripting).

In order to satisfy the concepts described in Section III, the software architecture sketched in Fig. 2 has been designed and it can be considered as divided into 4 main blocks.

— Block 1 represents the input files to the software platform. These files can be created by the user in many ways or can be created by the tools supported by the platform. This feature can be found in all the 4 main areas described in Section III.

— Block 2 represents the way the platform handles the input files and transforms them to be input for the other tools. This block also shows that the main component in the transformation process is the PNML file format. Moreover it shows that the platform can convert an automaton into a bounded PN and vice versa. This feature is basically implemented in the File Conversion area described in the previous section.

— Block 3 represents the tools supported by the platform. As can be seen in Fig. 2, the input file for a tool can be obtained as a transformation of a different file format and this transformation is done by Block 2. The tools management is implemented in the tools area (see Section III).

— Block 4 represents the GUI of the platform, and, as can be seen in the diagram it basically interacts with all the other 3 blocks, since almost each aspect of the platform can be handled by the GUI.

Note that the output of Block 3 can be an input for Block 2, i.e., a file produced by a tool

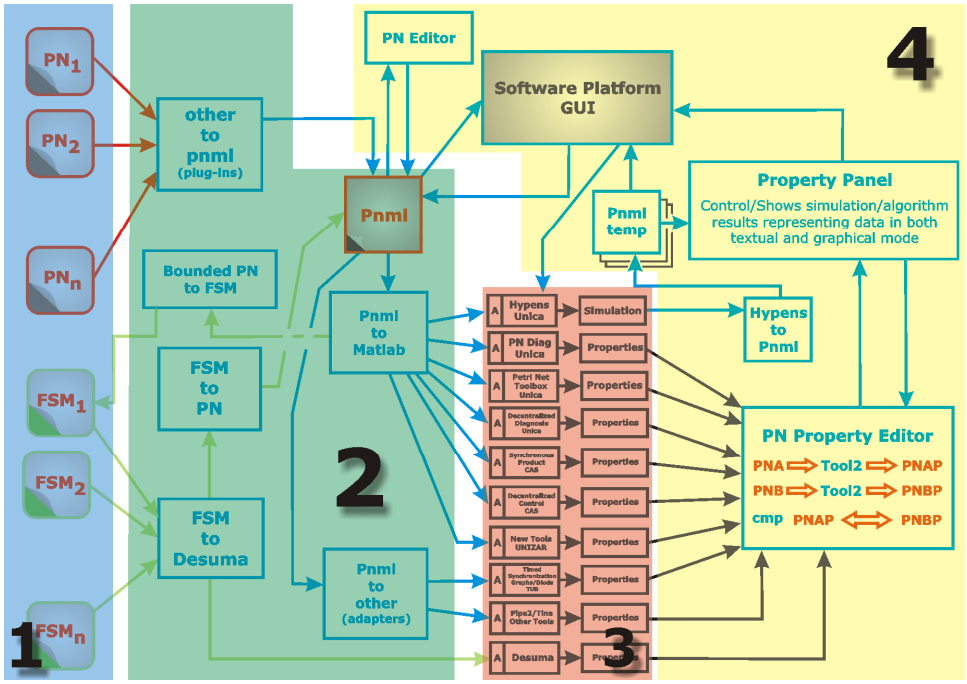

Fig. 2. Software architecture main blocks.

can be transformed into an input file for a different tool. Moreover Block 1 interacts with Block 4 since the GUI can directly handle the input files.

V. The PNML_DISC

The PNML format has many advantages, but is only able to describe high level, P/T or symmetric Petri nets. Since the software platform will support different types of Petri nets, such as hybrid Petri nets, and tools for the diagnosis of distributed systems, we need to extend the standard PNML format. The result is a format called PNML_DISC. In the following, we briefly describe each object composing a Petri net (places, transitions and arcs) specifying which elements are mandatory for the standard format (PNML) and which have been added in the extended format (PNML_DISC) in order to support the integrated tools. For more details we refer to [13].

A place in the PNML format is described by the following tags:

- *id*: identifier of the object place;
- *name*: contains the place text and its coordinates;

- *initial_marking*: initial marking of the place;
- *graphics*: coordinates of graphics description of the place.

The following tags have been added introducing the PNML_DISC:

- *type*: type of place (continuous or discrete). It is used for hybrid PNs;
- *observable*: defines the observability property of a place;
- *controllable*: defines the controllability property of a place;
- *site_observability*: the set of sites from which the place is observable. It is used for tools dealing with distributed systems;
- *site_controllability*: the set of sites from which the place is controllable. It is used for tools dealing with distributed systems.

A transition in the PNML format is described by the following tags:

- *id*: identifier of the object transition;
- *name*: contains the transition text and its coordinates;
- *graphics*: coordinates of graphics description of the place.

The following tags have been added introducing the PNML_DISC:

- *type*: type of transition (continuous or discrete). It is used for hybrid PNs;
- *observable*: defines the observability property of a transition;
- *controllable*: defines the controllability property of a transition;
- *site_observability*: the set of sites from which the transition is observable. It is used for tools dealing with distributed systems;
- *site_controllability*: the set of sites from which the transition is controllable. It is used for tools dealing with distributed systems.
- *site_membership*: the set of sites to which the transition belongs;
- *firing_speed_min*: minimum firing speed of the continuous transition;
- *firing_speed_max*: maximum firing speed of the continuous transition;
- *time_distribution*: specifies the timing structure of discrete transition;
- *delay*: discrete transition temporization parameter (temporization delay);
- *number_of_servers*: number of servers associated to discrete transition;
- *priority*: specifies the conflict resolution policy among discrete transitions;
- *fault*: specifies to which fault class the transition belongs;

- *label*: label associated to the transition;
- *memory*: used memory policy.

We had no need to extend the tags of the PNML format for the object arc. These tags are:

- *id*: identifier of the object arc;
- *source*: object id of the arc start point;
- *target*: object id of the arc end point;
- *text*: weight of the arc;
- *graphics*: coordinates of graphic description of the arrow.

VI. Tools of the Software Platform

In this section a list of the tools supported by the software platform is reported. The tools supported by the platform are those which the partners have used/use/will use during the DISC project.

- *PIPE3* [2]: it is a tool to model Place/Transition nets and generalized stochastic PNs. It has many analysis modules including advanced generalized stochastic PN analysis. This tool has not been developed by DISC partners and it is free, but not distributable. Thus it has to be downloaded by the user after installing the software platform.
- *TINA* [4]: it is an editor for graphically or textually described PNs, timed PNs and automata. It builds reachability graphs and perform structural analysis of PNs. This tool has not been developed by DISC partners and it is free, but not distributable. Thus it has to be downloaded by the user after installing the software platform.
- *DESUMA* [5]: it is an integration of the UMDES library with the graphical environment for visualizing discrete event systems. It allows manipulations of discrete event systems modeled by finite state automata related to model building, fault diagnosis, verification, control under full and partial observation, and decentralized control. This tool has not been developed by DISC partners and it is free, but not distributable. Thus it has to be downloaded by the user after installing the software platform.
- *HYPENS* [3]: it is a tool for analysis and simulation of timed discrete, continuous and hybrid PNs. The large set of plot functions available in Matlab allows one to represent the results of the simulation in a clear and intuitive way. This tool has been developed by a DISC partner, it is free and integrated in the platform.

- *Petri toolbox* [6]: it is a collection of MATLAB functions specifically designed to extract properties from a Petri net. These functions are divided in four parts to analyze a specific property of the net: general purpose, behavioral analysis, structural analysis and control with GMEC. This tool has been developed by a DISC partner, it is free and integrated in the platform.
- *PN_DIAG* [7]: it is a tool for the diagnosis and diagnosability of labeled Petri nets. This tool has been developed by a DISC partner, it is free and integrated in the platform.
- *PN_DIAG_DISC*: it is a tool for diagnosis of decentralized labeled Petri nets. It is being developed and it will be integrated by the end of the project.
- *PN Toolbox* [8]: it us a tool to draw PNs in a natural fashion, to store, retrieve and resize drawings. It allows simulation, analysis and design of PNs, by exploiting all the computational resources of the environment, via the global variables stored in the Matlab's workspace. This tool has been developed by a DISC partner. A demonstration version is free but not distributable. Thus it has to be downloaded by the user after installing the software platform.
- *SimHPN* [9]: it is a continuous and hybrid PNs simulator. This tool has been developed by a DISC partner. A demonstration version is free but not distributable. Thus it has to be downloaded by the user after installing the software platform.

Other tools, e.g. PN_DIAG_DISC and another one about decentralized supervisory control with coordination, will be integrated in the platform as soon as they will be developed by DISC partners.

Note that, the software platform does not require the Matlab installation to work. However, since most of the tools integrated are developed in Matlab it is strongly recommended to test its functionalities.

VII. USE CASES

In this section we present an example on how to use the functionalities of the software platform. In particular, we want to use the platform to compare two different diagnostic approaches, one based on automata [14] and the other one based on PNs [15], [16]. Both approaches allow one to perform diagnosis and know if the considered system is diagnosable. Solving a diagnosis problem means that we associate to each observed string of events a diagnosis state, such as "normal"

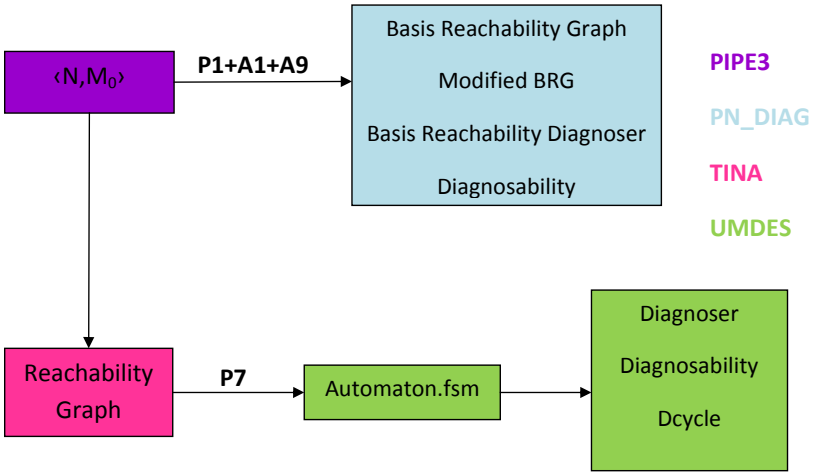

Fig. 3. Diagnosis process description.

or "faulty" or "uncertain". Solving a problem of diagnosability is equivalent to determine if the system is diagnosable, i.e., once a fault has occurred, the system can detect its occurrence in a finite number of steps. A diagram that summarizes the main steps to perform the above comparison is represented in Fig. 3. The symbols P_i and A_j in the arcs represent respectively the plugins and the adapters used for the conversions.

This comparison can be done using the software platform since both tools implementing the automata and the PN diagnosis approach are implemented in the software platform. They are respectively DESUMA [5] and PN_DIAG [7]. Since we need to compare two approaches: one based on PNs and the other one based on automata, the input that the user has to specify is a labeled bounded PN, i.e., a labeled PN whose state space is finite. Note that the automaton will not be specified by the user since given a bounded PN system $\langle N, M_0 \rangle$ the corresponding automaton can be obtained computing the reachability graph of the considered PN.

The Matlab based batch file executed in the *Script Manager* area is reported in Fig. 4.

The first line command "clear all" removes all variables, globals, functions and MEX links from the Matlab environment. The second line command uses the tool TINA to compute the reachability graph starting from a PNML file describing the input Petri net. Line command 3 converts the reachability graph created by TINA at the previous step into an automaton described in a fsm format using the plugin 7 (P7). Line command 4 starts a stopwatch timer. Line command 5 generates the diagnoser for the corresponding input file fsm and the input failure transition file.

```
1. clear all
2. !tina -R -PNML PN_Diag_PetriNet.xml
"Output_Files\PN_Diag_PetriNet_reach_graph.txt"
3. !pluginP7.exe "Output_Files\PN_Diag_PetriNet_reach_graph.txt"
"Output_Files\PN_Diag_PetriNet_reach_graph.fsm"
4. tic
5. !diag_UR.exe "Output_Files\PN_Diag_PetriNet_reach_graph.fsm"
failure.ft "Output_Files\diagnoser.diag" "Output_Files\diagnoser.fsm" 2
6. !dcycle.exe "Output_Files\PN_Diag_PetriNet_reach_graph.fsm"
failure.ft "Output_Files\inputFile.cycles"
7. t_UMDES = toc
8. !pluginP1.exe PN_Diag_PetriNet.xml
"Output_Files\PN_Diag_PetriNet.pnml"
9. !adapterA1.exe "Output_Files\PN_Diag_PetriNet.pnml"
"Output_Files\PN_Diag_PetriNet_Matlab_Disc.m
10. !adapterA9.exe "Output_Files\PN_Diag_PetriNet_Matlab_Disc.m"
"Output_Files\PN_Diag_PetriNet.m"
11. run('C:\Users\Luca\Desktop\Demo2 - Script
Manager\Output_Files\PN_Diag_PetriNet.m')
12. tic
13. run('D:\Progetti
Akhela\rseps\PAEH003\HMI\Windows\WP4_Software_Platform\WP4_S
oftware_Platform\bin\Debug\Tools\PN_DIAG_2010_07
(DISC)\main_PN_DIAG.m')
14. PN_DIAG = toc
```

Fig. 4. Matlab based batch file.

Line command 6 returns the diagnosability property of the system. Line command 7 saves the value of the stopwatch timer and saves it in a variable called t_UMDES. Line command 8 converts the XML PIPE3 format into a PNML file using the plugin 1 (P1). Line command 9 converts the PNML file into the corresponding MATLAB_DISC file using the adapter 1 (A1). Line command 10 converts the MATLAB_DISC file into a MATLAB based input file for PN_DIAG using the adapter 9 (A9). Line command 11 loads the input file for the PN_DIAG tool. Line command 12 starts a stopwatch timer. Line command 13 launches the PN_DIAG tool that computes the basis reachability graph, the modified basis reachability graph, the diagnoser and the diagnosability properties of the considered PN system. Line command 14 saves the value of the stopwatch timer and saves it in a variable called PN_DIAG.

This batch file shows that it is possible for the user to compare two different tools starting from the same file in an automated way. The necessary plug-ins and adapters are recalled by the user in a proper way according to the specific tools needs. This task could be performed also in the *Tools* area. However, in such a case this task cannot be automatized and the tools has to be launched one at a time manually. Moreover, the necessary file transformation must be

performed individually in the *File Conversion* area.

As a further Demo, the user can create a new Test script project and try to create a script similar to the one reported above, but using the drag&drop functionality for plugins and adapters and the "add file" button for the other types of files. For example, the line 3 of the above script can be created following this procedure:

1. Drag and drop the plugin P7 from the right side of the Script Manager section.

2. Press the "add file button" and select the file "PN Diag PetriNet reach graph.txt".

3. Copy and paste the line with the file name into the "input file" place holder created at point 1.

4. Change the "output file" place holder created at point 1 with "PN Diag PetriNet reach graph.fsm".

The whole script can be recreated using a similar procedure.

Note that, this case study requires the installation of the Matlab tool. Others use cases, also that do not require Matlab, are reported in [12].

VIII. INSTALLATION

The software platform is downloadable at the webpage *http://www.disc-project.eu/software_platform.html*. Here the user can also find detailed information on how to install and get started with the platform, a software platform manual and some case studies that can be used to test the functionalities of the platform.

IX. CONCLUSIONS

In this paper a software platform for the integration of discrete event systems tools has been presented. Its main goal is that of integrating several tools dealing with Petri nets and automata. An interchange format based on PNML has been defined. A series of plug-ins and adapters have been created to manipulate/transform the different file formats supported by the platform.

By the end of the project the platform will be tested with different use cases to prove the robustness of plugins and adapters. Moreover the part regarding that integration of new external tools will be improved and tested.

REFERENCES

[1] DISC website, "*http://www.disc-project.eu*."

[2] PIPE2 website, "*http://pipe2.sourceforge.net/*."

[3] HYPENS website, "*http://www.diee.unica.it/automatica/hypens/*."

[4] TINA website, "*http://homepages.laas.fr/bernard/tina*."

[5] DESUMA, "*http://www.eecs.umich.edu/umdes/toolboxes.html*."

[6] Petri net toolbox website, "*http://www.diee.unica.it/giua/ARP/*."

[7] PN_DIAG tool, "*http://www.diee.unica.it/giua/TESI/09_Marco.Pocci/*."

[8] M. Matcovschi, C. Mahulea, and O. Pastravanu, "Petri net toolbox for MATLAB," in *11th IEEE Mediterranean Conference on Control and Automation MED'03*, june 2003.

[9] J. Júlvez and C. Mahulea, "SimHPN: a MATLAB toolbox for continuous Petri nets," in *Proceedings of the* 10^{th} *Workshop on Discrete Event Systems*, Berlin, Germany, August 2010, pp. 24–29.

[10] HYCON Network of Excellence (2005), "*http://www.ist-hycon.org/*."

[11] MULTIFORM consortium (2008). Integrated multi-formalism tool support for the design of networked embedded control systems MULTIFORM., "*http://www.multiform.bci.tu-dortmund.de/*."

[12] Software Platform webpage, "*http://www.disc-project.eu/software_platform.html*."

[13] PNML_DISC format, "*http://www.disc-project.eu/PNML_DISC.pdf*."

[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Automatic Control*, vol. 40 (9), pp. 1555–1575, 1995.

[15] M. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[16] ——, "Diagnosability of bounded Petri nets," in *Proc. 48th IEEE Conf. on Decision and Control*, Shanghai, China, dec 2009.