SOFTWARE FOR FAULT DIAGNOSIS USING KNOWLEDGE MODELS IN PETRI NETS

SOFTWARE PARA EL DIAGNÓSTICO DE FALLAS MEDIANTE MODELOS DE CONOCIMIENTO EN REDES DE PETRI

ADRIAN ARBOLEDA
M.Sc, National University of Colombia, Medellin Campus, asarbolec@unal.edu.co

GERMAN ZAPATA
Ph.D, Universidad Nacional de Colombia, Medellin Campus, gdzapata@unal.edu.co

JOSÉ VELÁSQUEZ
Electrical Engineer, EMGESA S.A.E.S.P., Bogotá, Colombia, jvelazqu@emgesa.com.co

LUIS MARÍN
Electrical Engineer, EMGESA S.A.E.S.P., Bogotá, Colombia, lmarin@emgesa.com.co

Received for review: September 01th, 2011, accepted: January 27th, 2012, final version: March, 23th, 2012

RESUMEN: Los sistemas de diagnóstico de fallas en empresas asociadas al sector eléctrico requieren propiedades de precisión y flexibilidad cuando surgen eventos de falla. Actualmente existen sistemas que pretenden mejorar el proceso de diagnóstico mediante varios métodos y técnicas computacionales, reduciendo el tiempo de respuesta a perturbaciones. Sin embargo, son pocas las propuestas que unifican modelos gráficos de conocimiento con las señales de un proceso que pueden ofrecer dispositivos como controladores lógicos programables (PLCs). Este artículo propone un software novedoso guiado por modelos basados en redes de Petri e integrado con señales del proceso, para el diagnóstico de falla en centrales de generación eléctrica. Un caso de estudio demuestra la flexibilidad y adaptabilidad del software cuando nuevas nociones en los modelos de conocimiento cambian, sin realizar procedimientos de reingeniería al software.

PALABRAS CLAVE: diagnóstico de fallas, redes de Petri, desarrollo de software, modelos de conocimiento, sistemas de energía, automatización.

ABSTRACT: Fault diagnosis systems in electric sector companies require precision and flexibility properties in case of events. Currently, there are systems aimed at improving the diagnosis process through various methods and techniques, reducing response time to disturbances. However, few proposals unify graphical models of knowledge with process signals. These signals can be provided by devices such as programmable logic controllers (PLCs). This article proposes novel model-driven software based on Petri nets and integrated with process signals for fault diagnosis in power plants. A case study demonstrates the flexibility and adaptability of the software when new concepts change in the knowledge models, without requiring reengineering procedures to be performed on the software.

KEYWORDS: fault diagnosis, Petri nets, software development, models of knowledge, power systems, automation.

1. INTRODUCTION

Communication and diagnosis systems in energy companies use constantly evolving technologies due to the constant increase of variables and alarms that can be monitored. This trend forces such systems to be highly complex, especially with regard to fault diagnosis systems. According to [1], the development of software for fault analysis must be of a hybrid nature. This means combining computational techniques for precise and reliable diagnosis, based on continuous and discrete control components of the energy company.

Studies focused on fault diagnosis methods such as neural networks and genetic algorithms [2], have produced interesting results. However, when companies need to expand their business models, change control devices, or update infrastructure, these methods lack flexibility and adaptability characteristics, and require a process of re-engineering. Other alternatives such as Petri nets and methods based on models indicate their capacity for adaptability in the area of fault diagnosis in energy systems [3–5].

Currently, software aimed at use in power plants requires the basic characteristic of integrating its

control signals to diagnosis processes [1]. Based on this characteristic, this paper presents flexible software for fault diagnosis guided by Petri net models. The advantages of Petri nets are explored and a graphics editor is developed based on a framework of Eclipse Graphical Modelling (GMF) [6].

The software developed aims at improving supervision, operational efficiency, fault detection accuracy, and diagnosis speed. This article proposes a special form of structuring the functions in a software engine, and presents a generic architecture which explains the software modules of the system and their general functioning. The final product is an application which is easily adaptable to a power plant.

This article is structured as follows: Part two analyzes the studies undertaken in the field of fault diagnosis, Part three presents the definition of Petri nets and the construction method of a graphics editor, Part four presents the software›s architecture and functioning, and finally Part five presents conclusions and future work.

2. PREVIOUS WORKS

Modern energy systems tend to be interconnected, complex, and highly scalable systems. According to these strengths, different types of software for fault detection and diagnosis, based on computational science techniques have become more popular on the market. Techniques applied in the current literature include neural nets [7], knowledge-based methods [8], genetic algorithms [2], and expert systems [24]. However, these techniques lack flexibility and adaptability for when companies grow, or for when market conditions change, and high costs in the processes of re-engineering and re-adaptation get out of hand.

There are other approaches such as Petri nets and knowledge based methods which maximize flexibility and simplicity properties in computational systems. [9] indicates the advantages of applying Petri nets in business processes and their capacity to model, in a unified manner, the processes of control and other knowledge schemes proper to each company, also observed in [10]. Petri nets have been broadly used in the field of energy systems in order to model the configuration of power plants [3], producing important results in terms of reducing diagnosis time and improving diagnosis accuracy.

Additionally, studies such as [1] and [11] explain the importance of fault diagnosis automation and of its integration into information systems through model-based methods. These studies also propose the development of hybrid systems for complex system analysis and control, since today's problems require integrated mathematic and empiric solutions guided by company experience.

Other proposals such as [4], [5], and [12] attempt to unify model-based methods with Petri nets. These produce interesting results in terms of effectiveness, analysis quality, and the interaction with planned signals and process planning. However, these proposals omit the possibility of integration with software engines which, based on process events and model knowledge, allow users to interact and diagnose in an easy way. Since the early 90s, generic software engines have been developed at low costs in the different areas of science, resulting in great benefits in time and performance [13]. This type of software functions with specific models created by experts in a specific domain. [14] presents important aspects to be considered when it is required to unify business models in a software prototype, highlighting the tools' potentials when requirements change in the environment.

The papers just mentioned lack important aspects such as the combination of natural language, cost reduction in software development, and the capacity for integration between software engines, real control signals, and models. These are important aspects for developing software aimed at energy generation companies, specifically in the field of fault diagnosis.

3. MODELS IN PETRI NETS FOR FAULT DIAGNOSIS

Petri nets (PNs) are a formal modelling tool for dynamic systems that may become quite complex. They are characterized by an easy-to-understand graphical component, and by their mathematical theory foundations. Petri nets are currently applied in different fields of science, highlighting fault diagnosis [15], state analysis [16], work flows [17], and knowledge representation [18].

3.1. Definition of Petri Net

According to [19], a Petri net may be formally considered as a set of 5-tuples, $PN = (P, T, F, W, M_0)$, where:

$P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places,

$T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,

$W: F \rightarrow \{1,2,3,\ldots\}$ is a function of weight,

$M_0: P \rightarrow \{0,1,2,3,\ldots\}$ is the initial marking,

$P \cap T = \Phi$ and $P \cup T \neq \Phi$.

The evolution of a state S of a Petri net, from a state *k* to another *k+1,* can be expressed as

$$\mathbf{q}_s[k+1] = \mathbf{q}_s[k] + \mathbf{B}\mathbf{x}[k] \quad (1)$$

Where $\mathbf{B} = [b_{ij}^{+} - b_{ij}^{-}]$ is the incidence matrix of the Petri net(b_{ij}^{-}denotes the weight of the arc from place p_i to the transition t_j , and b_{ij}^{+} denotes the weight of the arc from transition t_j to the place p_i). $\mathbf{q}_s[k]$ is the state of the Petri net in time k. $\mathbf{x}[k]$ in the entrance to the Petri net. If $\mathbf{x}[k] = [0, \ldots,1,\ldots,0]^T$ (a 1 is in the j-th position), the transition t_j is triggered (j inside {1,2, ..., m}).

3.2. Petri Nets editor

To develop a software engine that interprets a binary Petri net, it is necessary to build a graphical editor of Petri nets that includes the basic components of modelling, and additional components that enable defining key expressions understandable only by the engine. These additional components are called *labels* and they define the functionalities to be programmed in the software.

To supply for the need to promptly build a robust and reliable Petri nets editor, the *framework* for Eclipse Graphical Modelling (GMF) [6] is used. Eclipse Graphical Modelling is a *plug-in* which provides the infrastructure and visual tools to develop modelling language editors in the Eclipse platform. Eclipse Graphical Modelling is used to build models of specific domains from a metamodel and following a methodology. In order to develop the editor, programmers need to build an intermediate number of models specifying the visual syntax of the required language. These intermediate models are combined and, through a set of transformations, an Eclipse *plug-in* is developed, which in turn is what really implements the editor. Figure 1 presents a summary of the process to be followed with GMF. For readers interested in the proper development of a graphical editor, [20] and [21] present assistance guides, while [25] presents a paper, analogue to this one, revealing the steps for building an editor for an specific domain model using GMF.

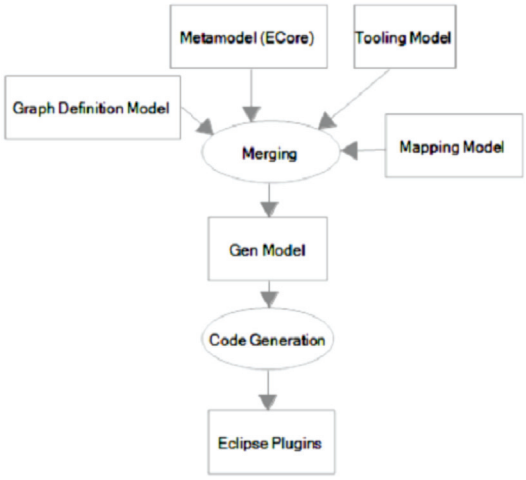

Figure 1. Process of modelling with GMF taken from [22]

The main component for building the editor of Petri nets is the metamodel. Figure 2 shows the Petri nets metamodel formed by 6classes in which the 3basic elements of the net (place, transition, and arc) and the label class, which as previously mentioned has 7 attributes which turn into functions, are highlighted. Note that the label class is an attribute of the place class, meaning that each place created in the model may receive a specific functionality. These functionalities are aimed at the domain of fault diagnosis in hydroelectric power plants and described as follows:

Indicate: A key word which indicates the publication of information to the software and to wait for an event from the operator.

Fault: A key word which indicates to the application the finding of a fault in the net.

Conclude: A key word which indicates to the application the finding of an important conclusion or event in the net.

Inform: A key word referring to the transmission, via text messages, of determined information to determined telephone numbers.

Silent: A key word which indicates the finding of relevant information to be saved in the data base, continuing immediately with the net's evolution (i.e., executing the next step in the network).

None: A key word referring to the omission of a specific function.

Signal: A key word integrating the value of a measuring device to the Petri net.

Button: A key word for indicating the appearance of buttons in the software interface, which has the information update in the associated PN.

It must be highlighted that when a company begins signal automation processes, the places of the models with "button" labels, can be easily replaced by the "signal" label. Thus, the software will not have to do an information presentation stop in the interface, reducing operator and software interventions.

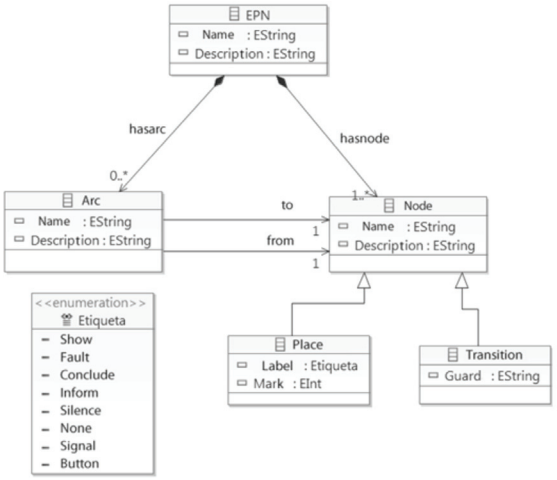

Figure 2. Petri nets metamodel

In order to have a fully automatic fault diagnosis and to chronograph analysis and results presentation time in seconds, the software proposed in this article aims at having no places in the models with the "button" label. This approach for the *Etiqueta* class enables programming by modules, indicating what the software must do when the engine goes through each one of the places.

When the unification of intermediate models is finally accomplished in Eclipse, the tool automatically executes code generation, obtaining the graphical editor of Petri nets. Then, the next procedure is to create Petri nets containing the knowledge for fault diagnosis through the association of places with descriptions in natural language. Figure 3 shows the editor's graphic interface indicating the symbols palette, the properties panel of each component, and the central panel of graphical edition. It highlights that it is, in general terms, an intuitive tool to model.

The creation of PN files linked to the new application is XML based. Therefore, it is necessary to develop in the software a module that interprets the XML components of the Petri net.

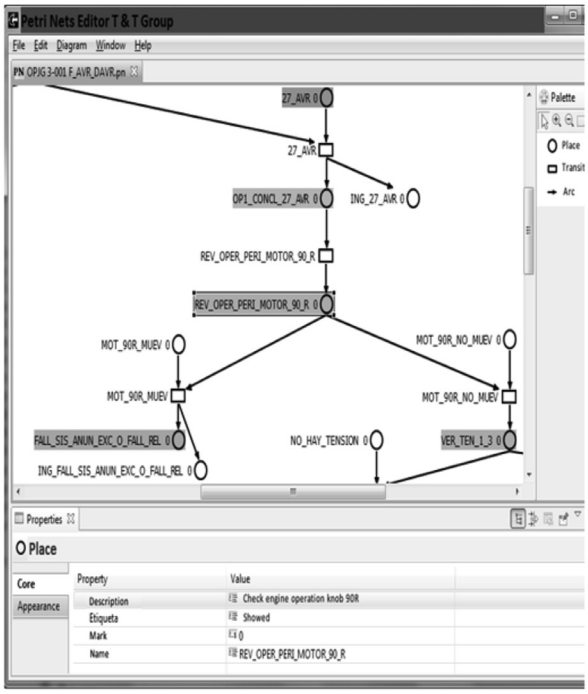

Figure 3. Petri nets editor

4. FAULT DIAGNOSIS SOFTWARE ARCHITECTURE

Process modelling becomes a complex task due to the constant requirement changes in the environments where the processes take place [14]. Diagnosis tools, in the settings of energy systems, tend to combine precision methods such as mathematics with empiric methods resulting from company knowledge and experience. Therefore, the reasons for developing a fault diagnosis system guided by Petri nets are avoiding the overhead costs of software engineering processes, converting the models into knowledge schemes that are reusable and flexible, and unifying measurement methods with natural language to have diagnoses that are more accurate and reliable.

The diagnosis system built under the Java platform is formed by 4 software modules, which are the following:

Engine: This contains the mechanisms for interpreting the elements of the Petri net which constitute the XML files produced by the graphical editor. In addition,

based on the Petri nets theory presented in Chapter 3, it includes the algorithms that determine the new states of the net, meaning the vector $\mathbf{q}_s[k+1]$. These algorithms were developed in previous research conducted by the *Universidad Nacional* and financed by Colciencias, which constitute the nucleus of the engine [26]. Besides this, the module has a high interaction with GUI and supervision modules, which constantly send data for updating places on the net, and are executed at determined times.

Data Base: This stores knowledge models developed in the editor, and contains the information associated with the results of previous diagnoses.

Supervision: This contains the algorithms for the supervision and monitoring of the signals previously defined by each diagnosis model. It is important to mention the use of the open-code library for communicating with industrial devices *UTGARD* [23], which is of vital importance for the integration of control signals. This module is responsible for alerting operators about alarm events and data updates from a PLC or control device to the linked Petri net. Figure 4 illustrates the PLC-PN update mechanism where, in summary, information is replaced in the places bearing a "signal" label.

GUI: This is the module assigned for unfolding and controlling the graphic user interfaces. It has a high level of interaction with the other system modules.

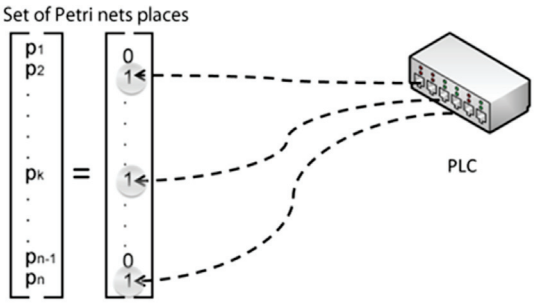

Figure 4. Signal updating from a PLC to the Petri net

4.1. Software functioning

The general functioning of the fault diagnosis system is illustrated in Fig. 5. First, after the development of the models in Petri nets, the person acting as administrator must enter the models into the system and perform the respective configuration of control signals; meaning, the association between the places with the "signal" label and the PLC variables, as shown in Fig. 6. Note that there is an option called "initiator". This refers to the signals at the start of a model, which turn into trigger signals, and therefore are references always examined by the supervision module in real time. When this configuration concludes, the model repository is updated and the supervision module activates the constant inspection of the registered signals. When the value of an "initiator" signal is 1, the fault detection process becomes active, loading the specific model linked to the signal and alerting, on screen, about the event of a signal pending diagnosis. At that point, it is important to remark that each signal detected by the software becomes a completely independent diagnosis, which may go through different states. Five states can be summarized in total. These are: Pending, in execution, ignored, completed, and cancelled.

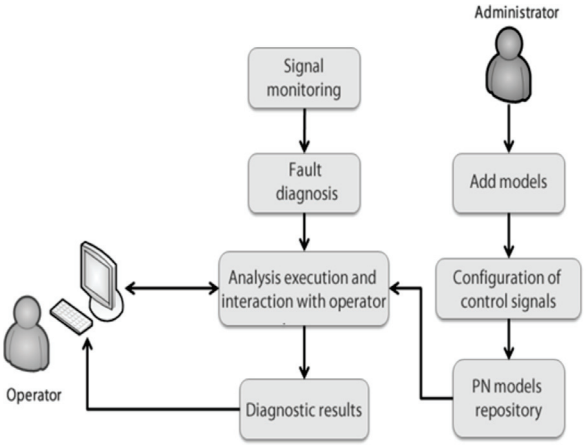

Figure 5. General functioning of the diagnosis system

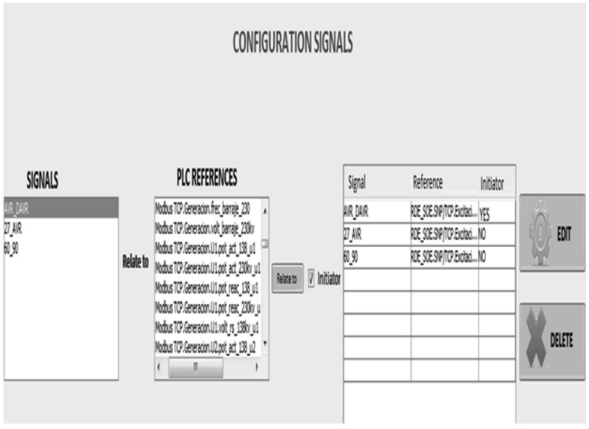

Figure 6. Signal control configuration in fault diagnosis system

When the software presents pending signals, the operator initiates interaction with the software to try to obtain results, according to the analysis of the model. This interface is presented in Fig. 7, showing three alert signals. Two signals on pending state and one signal in ignored state, which means that the operator decided to omit this last signal, as it is not important. Upon execution of the diagnosis model, the engine module assumes the main role in the execution of the software, sending information to the GUI module for presentation, and receiving information from the control signals.

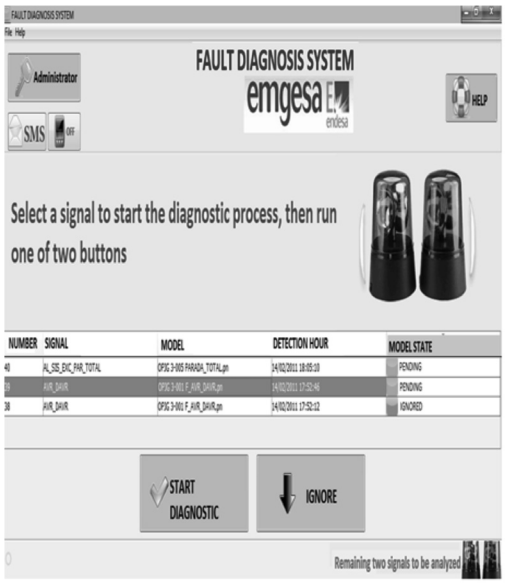

Figure 7. Fault detection in the diagnosis system

Figure 8 illustrates the case of Fig. 3, which presents the case of a manual revision of a knob. There is a software stop and an operator decision to select an option about the movement, or lack of, of engine 90R. The model, for this interface to unfold, interpreted the two places of the established net with the "button" label and, in this manner, updates the net in accordance with the executed option.

According to this example, Fig. 9 presents diagnosis results which can be divided into faults or conclusion. This means that models can have at least one type of result or both. A fault in the fuses is determined in this case, and it is recommended to continue monitoring the machine and adjusting the reactives. When the process is finalized, the system saves the decision route, and the relevant events leading to the results, in order to consult them at any time.

Figure 8. Decision making of the diagnosis system with the intervention of the operator

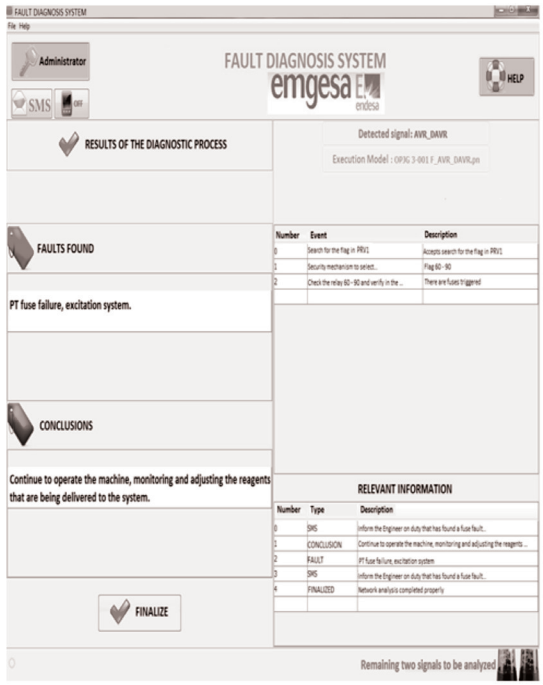

Figure 9. Results of the diagnosis system according to the AVR_DAVR model

5. CONCLUSIONS AND FUTURE WORK

There are diverse techniques and procedures covering fault diagnosis in companies linked to the electric

sector. They present a common disadvantage which is the lack of flexibility when a diagnosis process changes, or when part of the infrastructure is renewed. The previous article presents a software prototype for fault diagnosis systems, especially for power plants. The software is guided by models in Petri nets, and its main features are flexibility and ease of integration between control signals and models, aiming at diagnosis accuracy and speed. The software modules are explained, highlighting the engine module which supports the execution of Petri nets and the updating of data through the PLC. When the need for updating diagnosis models surges, an expert can easily edit them through the also-developed Petri nets graphical editor.

Planned as future work is the unification of the files currently in XML to the standard format of Petri nets called *pnml*. Also contemplated is the addition of functions linked to the analysis Petri nets such a vivacity, cyclicity, limitation, and conflictiveness, which may enable one to determine interesting phenomena or errors in modelling.

ACKNOWLEDGEMENTS

The authors thank the company EMGESA S.A.E.S.P. and the Administrative Department of Science, Technology, and Innovation of Colombia COLCIENCIAS, for their help with this research, as part of the framework of the project entitled “Automatic Fault Diagnosis in Hydraulic Power Plants of the System EMGESA - COLCIENCIAS - CODENSA S.A. E.S.P. - EMGESA S.A. E.S.P.”

REFERENCES

Wenping, W., Xiaomin, B., Wei, Z., Jian, D., and Zhu, F. Hybrid Power System Model and the Method for Fault Diagnosis, ISBN 0-7803-9114-4. IEEE/PES Transmission and Distribution Conference and Exhibition: Asia and Pacific,Dalian, China, pp.1-5, 2005.

[1] Fritzen, P., Cardoso, G., Zauk, J., Morais., Bezerra, U., and Beck, J. Alarm processing and fault diagnosis in power systems using Artificial Neural Networks and Genetic Algorithms, ISBN 978-1-4244-5695-6. IEEE International Conference on Industrial Technology (ICIT), pp.891-896, 2010.

[2] Ashouri, A., Jalilvand, A., and Noroozian, R. Fault diagnosis modeling of power systems using Petri Nets, ISBN 978-1-4244-7128-7. 4th International Power Engineering and Optimization Conference (PEOCO), pp.313-317, 2010.

[3] Rueppel, U., Meissner, U., and Greb, S. A Petri Net Based Method for Distributed Process Modelling in Structural Engineering, ISBN 3-86068-213-X. Proceedings of the 10th International Conference on Computing in Civil and Building Engineering ICCCBE, 2002.

[4] Rong, Z., Shengfang, F., and Jian, C. Hybrid modeling techniques for power electronics based on Petri Net, ISBN 978-1-4244-3826-6. International Conference on Electrical Machines and Systems ICEMS, pp.3850-3853, 2008.

[5] The Eclipse Graphical Modeling Project (GMP), Official Web-Site. http://www.eclipse.org/modeling/gmp/. March 2010.

[6] Yuan, H., and Chen, Z. Design and Implementation of Intelligent Fault Diagnosis System, ISBN 978-0-7695-3887-7. 1st International Conference on Information Science and Engineering (ICISE), pp.2296-2299, 2009.

[7] Chang, C., Tian, L., Wen, F., Han, Z., Shi, J., and Zhang, H. Development and Implementation Knowledge-Based System for On-Line Fault Diagnosis of Power Systems. Electric Power Components and Systems, ISSN 1532-5008, 29 (10), pp.897-913, 2001.

[8] Salimifard, K., and Wright, M. Petri net-based modelling of workflow systems: An overview. European Journal of Operational Research, ISSN 0377-2217, 134 (3), pp. 664-676, 2001.

[9] Zha, X., Lim, S., and S. C. Fok. Integrated knowledge-based Petri net intelligent flexible assembly planning. Journal of Intelligent Manufacturing, ISSN 0956-5515, 9 (3), pp. 235-250, 1998.

[10] Zoeteweij, P., Pietersma, J., Abreu, R., Feldman, A., and Van Gemund, A. Automated Fault Diagnosis in Embedded Systems, ISBN 978-0-7695-3266-0. Second International Conference on Secure System Integration and Reliability Improvement SSIRI, pp.103-110, 2008.

[11] Huang, H., and Zu, X. Hierarchical Timed Colored Petri Nets Based Product Development Process Modeling. In Computer Supported Cooperative Work in Design. ISBN 978-3-540-29400-9, Springer Berlin / Heidelberg 3168, pp. 378-387, 2005.

[12] Kurien. J., and Moreno, M. Costs and Benefits of Model-based Diagnosis, ISBN 1095-323X. IEEE Aerospace Conference, pp.1-14, 2008.

[13] Chen, A., and Buchs, D. Towards Service-Based Business Process Modeling, Prototyping and Integration. In Rapid Integration of Software Engineering Techniques. ISBN 978-3-540-34063-8, Springer Berlin / Heidelberg 3943, pp. 218-233, 2006.

[14] Genc, S., and Lafortune, S. Distributed Diagnosis of Place-Bordered Petri Nets. IEEE Transactions on Automation Science and Engineering, ISSN 1545-5955, 4 (2), pp. 206-219, 2007.

[15] Wu, Y., Xie, L., and Li, B. Software Design for Reliability Analysis Using Petri Nets, ISBN 978-0-7695-3962-1. International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), pp. 414-417, 2010.

[16] Hoheisel, A., and Alt,M . Petri Nets. In Workflows for e-Science Scientific Workflows for Grids, Part II. ISBN 978-1-84628-757-2, Springer London, pp.190-207, 2007.

[17] Ribaric, S. and Hrkac, T. A Knowledge Representation and Reasoning Based on Petri Nets with Spatio-Temporal Tokens, ISBN 978-1-4244-0813-9. The International Conference on Computer as a Tool EUROCON, pp.793-800, 2007.

[18] Hui, R., and Zengqiang, M. Power system fault diagnosis modeling techniques based on encoded Petri nets, ISBN 1-4244-0493-2. IEEE Power Engineering Society General Meeting, pp.1-6, 2006.

[19] The Eclipse Graphical Modeling Project (GMP), Online Documentation. http://help.eclipse.org/ganymede/index.jsp. March 2010.

[20] Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. EMF: Eclipse Modeling Framework 2.0. ISBN 0321331885, Addison-Wesley Professional, 2009.

[21] Kolovos, D., Rose, L., Paige, R., and Polack, F. Raising the level of abstraction in the development of GMF-based graphical model editors, ISBN 978-1-4244-3722-1. Workshop on Modeling in Software Engineering ICSE, pp.13-19, 2009.

[22] The OpenSource SCADA System, Official Web-Site. http://openscada.org/. March 2010.

[23] Gutierrez, S., and Branch, W. A comparison between expert systems and autonomic computing plus mobile agent approaches for fault management. Revista Dyna, ISSN 0012-735, 168, pp.173-180, 2011.

[24] Montenegro, C., Gaona, P., Cueva, J y O. Martínez. Aplicación de ingeniería dirigida por modelos (MDA), para la construcción de una herramienta de modelado de dominio específico (DSM) y la creación de módulos en sistemas de gestión de aprendizaje (LMS) independientes de la plataforma. Revista Dyna, ISSN 0012-7353, 169, pp. 43-52, 2011.

[25] Méndez, W. Diseño y Construcción de un Motor de Ejecución de Redes de Petri para el Análisis de Estados de una Unidad de Producción. Trabajo de grado. Escuela de Sistemas de la Universidad Nacional de Colombia, Medellín, Colombia, 2009.