Towards a multi-lingual workflow system – a practical outlook

Bastin Tony Roy Savarimuthu, Maryam Purvis
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
{tonyr,tehrany}@infoscience.otago.ac.nz

Abstract

Due to rapid development in the global market, workflow systems are not limited to a single country. Electronic business like workflows span across countries and hence there arises the need for the understanding among the users of the system to operate/use them in their own local language. For a software such as workflow management systems it is highly imperative that it should be internationalized as some of the processes/sub-processes of the workflow might be in a different country and the people in that locality would express a strong need for localizing the software. In this paper we explain some aspects of internationalization such as user interfaces, feedback information and the text messages in various graphs depicting the progress of the workflow processes. We also describe the problems that we encountered and the challenges that still lie to be explored.

Keywords: Workflow, Internationalization, Localization, Simulation, Separation, Re-factoring

1 Introduction

Workflow management systems WfMS (Aalst and Hee 2002, Aalst, Hofstede, Kiepuszewski and Barros 2002, Meilin,Guangxin,Yong and Shangguang 1998) are widely used to manage business processes for their known benefits such as automation, co-ordination and collaboration between entities.

The earlier work described by Fleurke, Ehrler and Purvis (2003) deals with the framework of a distributed network of autonomous software agents that can adapt to the changing circumstances in a workflow management system. Normally businesses are spread across countries and that necessitates software to be written in many languages that fulfill the global needs of the people. Software for workflow that is typically global in outreach should cater to the multi-linguistic nature of the users.

Our work focuses on internationalizing JBees, our framework for workflow management systems. In this paper we focus on internationalizing the simulation, monitoring and controlling aspects of our framework. These activities of any workflow could take place at any part of the world and there is a strong need for internationalization.

The languages that are used for the localization purposes are Dutch and Hindi, representing the west and the east respectively. The reason for choosing Hindi, the national language of India, is that there is a need among the diamond processors around Gujarat region in India who need to localize the diamond processing workflow in their local language Hindi. Limited work has been done in localization of software in Hindi and the potential of the local language market (Frost and Sullivan, 2003) in India is huge as only 5% of the Indian population speak English. There is a need for Hindi software as 25% of the Indian population speak the language. Hindi uses the Devanagari script.

Though there has been some work (Purvis, Hwang, Purvis, Madhavji and Cranefield 2001, He, Bustard and Liu 2002, Deitsch and Czarnecki 2001) on internationalisation and localization of software, our work is unique as we deal with the internationalisation and localization of the agent based workflow system. To the best of our knowledge, it's one of the first workflow management systems with multi-agents that is to be internationalized.

The paper is organized as follows. The second section gives an overview of the background information such as our existing multi-agent framework. In the third section we discuss about our approach in the process of internationalization. In the fourth section we describe the issues that we encountered. Section five describes the challenges that persist in the process. Finally the conclusions and future directions of our work are described in section six.

2 Background

2.1 Existing multi-agent framework for workflow system – JBees

JBees consists of five distributed agents using the opal framework (Purvis, Cranefield and Nowostawski 2002), which supports agent implementation and communication. The overview of our existing framework is shown in Figure1.

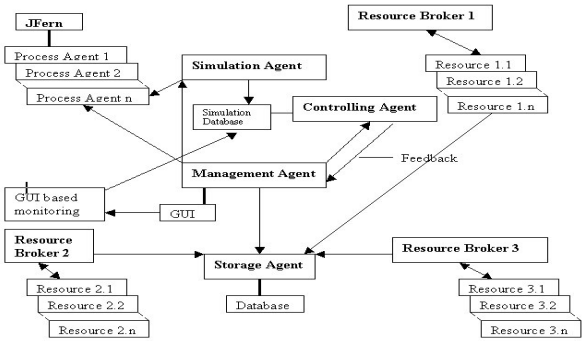

Figure 1: Architecture of the multi-agent based workflow system.

The manager agent provides all functionality the workflow manager needs such as creation and deletion of tasks, roles and process definitions, instantiation of new process instances and creation of resource agents. The process agent executes a process instance. Each resource in the system has its own resource agent. Every resource in the system gets registered to one of the broker agents that allocate the resources to the process. The storage agent manages the persistent data that is needed. The use of agent technology enables an open and distributed workflow system.

Our system uses coloured Petri Nets (Jensen, 1992), a graphical formalism to represent various process models. In our framework we have encoded process models in XML while transferring them on the wire between distributed agents. Coloured PetriNets models consist of places (circles), transitions (rectangles), arcs (arrows) and tokens (values). Figures 4 and 5 show the layout of a coloured Petri Net process model.

The system enables the users to manage workflows, simulate processes, monitor and control the process simulations. The sub-processes in a given process could be distributed and can be executed on different hosts that may be located in various regions. Our existing framework supports English as the default language for the simulation and execution of processes.

2.2 The need for internationalization

Workflows that model business processes associated with multi national companies are spread across countries. Most of the multinational companies operate in regions spanning many different languages and cultures and it is imperative for a workflow that spans across countries to have a localization option to suit to the needs of the local user's requirement.

3 Approach

3.1 Identification of region specific information across various modules

Our first objective was to identify the locale sensitive information/code that spread across different modules so that they could be put in a common place. This involved a careful introspection of the modules for the possible information that are to be internationalized.

3.2 Separation of relevant information

The separation of this relevant information and storing them into appropriate resource bundle constituted the second objective of our work. This information was stored in appropriate "properties" file that could be used by the resource bundle class of the java language. Figure 2 shows the sample properties file in Dutch.

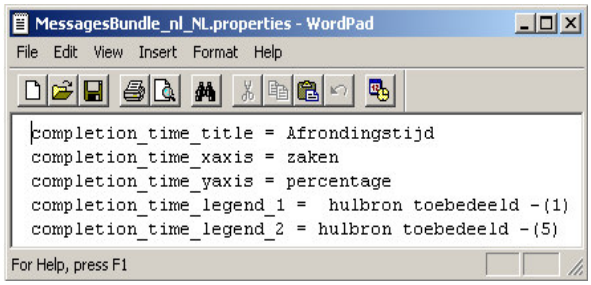

Figure 2: showing the sample properties file in Dutch.

The information that was separated includes UI labels, title and axes messages of the graphs, warning messages and the xml element values.

3.3 Code Re-factoring

The Separation of these text messages and strings led to the re-coding of the modules that originally contained these messages only in English. Figure 3 shows the overview of the information that was internationalized.

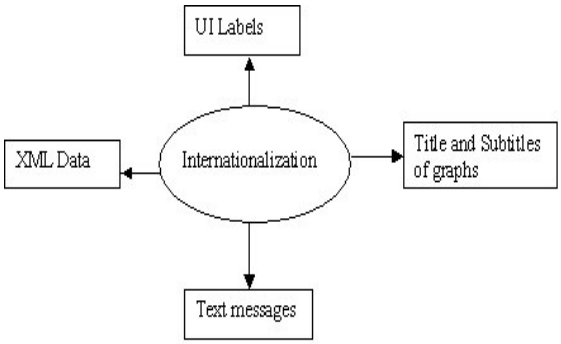

Figure 3: showing the categorization of the information that is to be internationalized.

The internationalization of user interface labels was carried out and the modules that involved these changes were recoded. Figures 4 and 5 show the user interfaces in English and Dutch respectively.

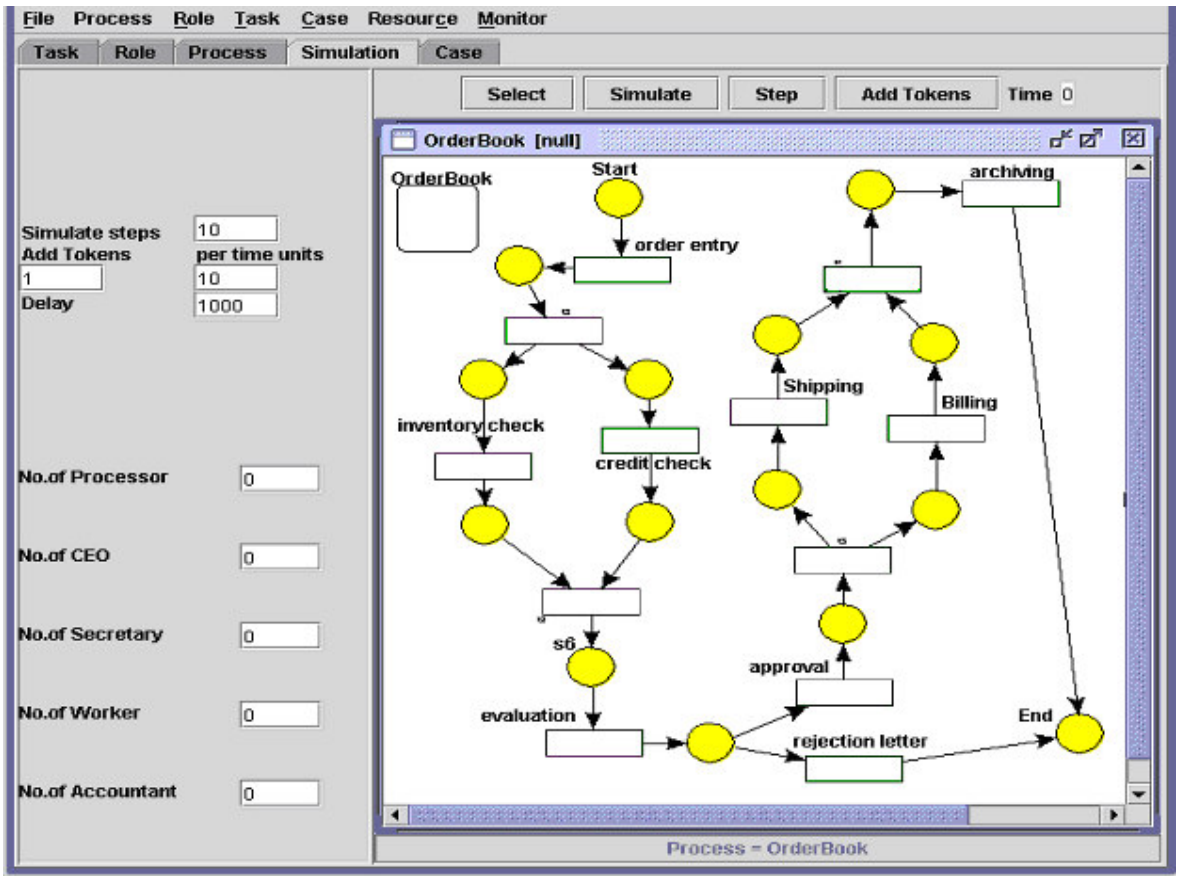

Figure 4: Showing simulation screen in English

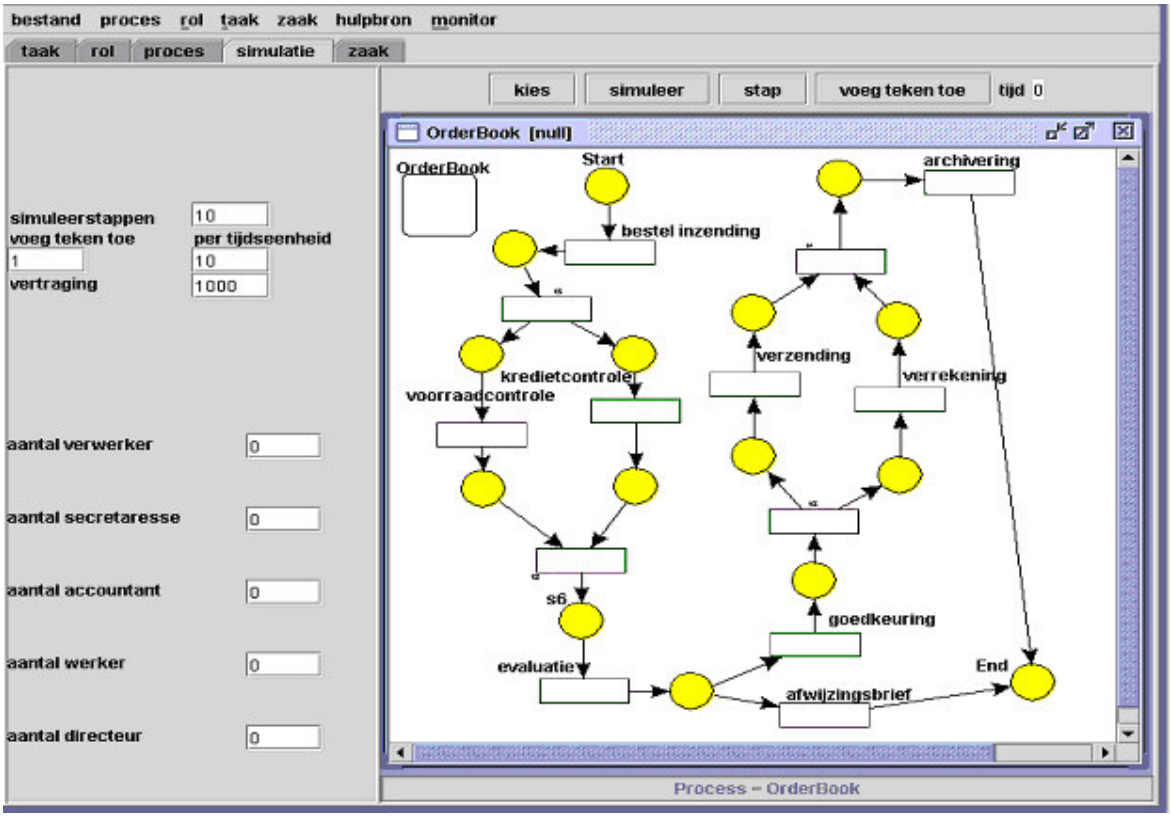

Figure 5: Showing simulation screen in Dutch

The titles and axes messages for the various graphs used for monitoring the simulation were translated into Dutch and Hindi and were stored in resource bundles and their corresponding modules were recoded. Figures 6, 7 and 8 give an overview of the graphs that contain title messages in English, Hindi and Dutch. These graphs show the total time taken by different cases to complete a process.

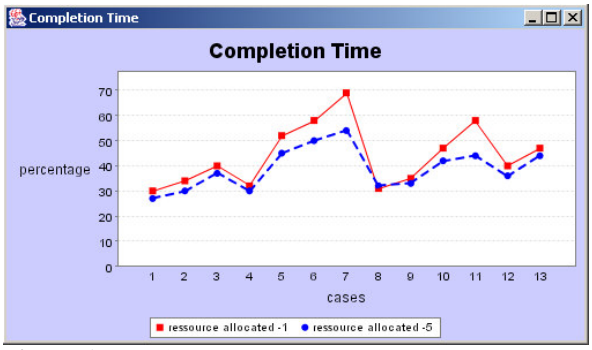

Figure 6: Showing the graph in English

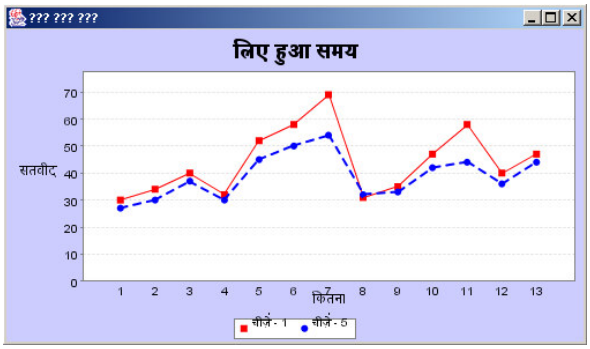

Figure 7: Showing the graph display in Hindi.

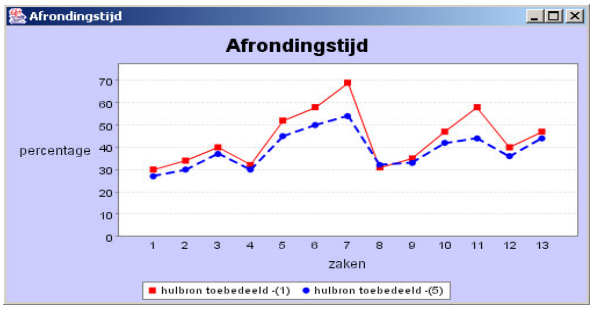

Figure 8: Showing the graph display in Dutch.

We then, isolated the warning text messages of our workflow system and translated them into appropriate messages in the chosen languages and stored in the appropriate resource bundle. Figures 9 and 10 show the warning messages in English, Dutch and Hindi respectively.

Figure 9: Showing the warning message in English

Figure 10: Showing warning message in Dutch.

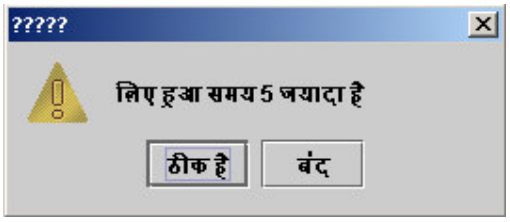

Figure 11: Showing warning message in Hindi.

The figures 4 and 5 also show the actual workflow sequence. These figures consist of text messages shown in English and Dutch respectively. The information for these was stored in the XML. So we needed appropriate changes to the code that read the XML file in the default language English and displayed these messages in appropriate local languages. In particular, the task names associated with transitions (rectangles) mentioned in the workflow in the figures 4 and 5 have been separated and have been stored in the appropriate resource bundle.

4 Issues

4.1 Issues related to distributed computing

JBees, our existing java framework that is being internationalized is distributed. The processes and sub-processes could span across countries. Distributed computing raises some interesting issues. For example while designing a workflow system in which the sub-processes span across various countries there appears the issues such as which language should be chosen for the process model. The distributed components of the workflow system might be running on different hosts that span across various countries. One has to make a decision about the language used by different components and how these components interact.

Our approach is that the components that produce the process model use the default language English to specify the process models and the display components such as the simulation results and feedback to the user will be represented in the local language. Process models associated with business processes are usually specified by a domain expert, which in turn can be used by the regular users.

4.2 Issues related to Re-factoring

Re-factoring of modules and interdependencies should not be postponed until the end of software development cycle. The need for internationalization must be foreseen from the requirements phase of software engineering and the tailoring of software right at the end of the life cycle to meet local needs poses lots of problems.

4.3 Issues related to the display of title messages in the frame

Title messages of figures 7 and 11 were not displayed properly when Hindi is used because the title messages use the operating system's fonts. The problem persisted unless the locale of the operating system was set to the language in which the message was written. We found this bug in the earlier versions of java and it has been fixed in the current version (jdk1.4.2). Figures 12 and 13 show the title message clearly in Hindi.

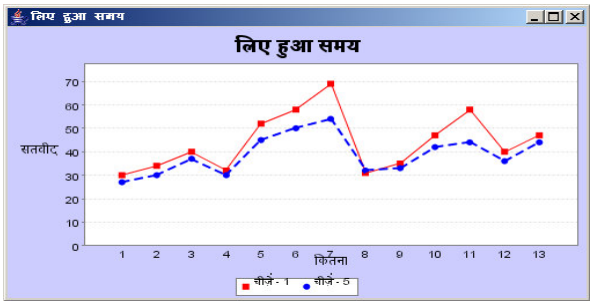

Figure 12: Showing a graph in Hindi.

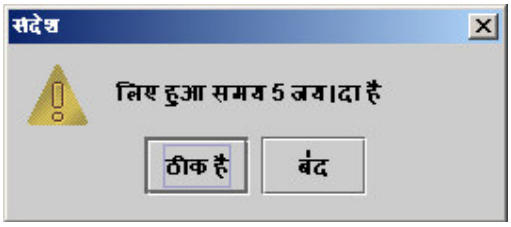

Figure 13: Showing a warning message in Hindi

4.4 Problems that arise when using external components

For drawing of graphs in our workflow systems we use an external component, JFreeChart, a free source Java API. Though it supports the western languages, when Hindi was used the characters looked raised (legend of the graph shown in Figure 7). This is attributed to the complex ligatures of Indic scripts. Figure 14 shows that the legend space has been increased to accommodate the Hindi characters. Also note that numbers along the x-axis has been converted into Hindi numerals.

In using already available components the problems that arise include a) Font problems b) Date, Time and Numeral problems c) UI alignment problems and d) Text Messaging problems. If the source code for those components is available then these problems could be solved. While deciding about using components provided by different vendors it is advisable to think about the levels of internationalisation support that they provide.

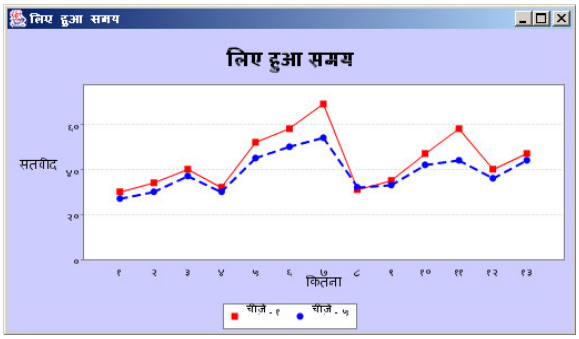

Figure 14: Showing a graph in Hindi

4.5 Development issues

When the user interfaces were designed using static layouts they always caused lots of problems, as they have to be redesigned with layouts that determine the size of the components during run time. As we had already finished implementing our software, we did have surprises when the Devanagari script for Hindi could not be accommodated in the label sizes originally designed in English. We had to resize them as well as the overall size of the graphs to incorporate the chosen languages.

We have also used separate resource bundles for the following categories such as UI Labels, Graphs, Messages and XML references so that possible confusions about message contexts could be avoided.

From the software engineering point of view, the modularization of the individual modules would have been better if the localisation needs had been thought before the requirement/design phase.

5 Challenges

5.1 Testing and Maintenance

Testing the application for the verification of the semantics of the language and its validity seemed to be problem as we had to go to the experts to perform these tasks and also the natural flow of the language used in the translation is quite questionable as it is always of aesthetic concern whether the translation has led to a better understanding in the localized language or not.

The improvements or changes to the workflow system now also need many linguists to verify the additions that are to be made to the product.

5.2 Static vs. Dynamic content

The issues that have been addressed so far deal only with the static content such as UI labels and messages. Our framework also incorporates facilities to insert a new process model dynamically.

The challenges that we encounter with dynamic contents are two folds. The first challenge is to modify the internal infrastructure of various components in our system to accommodate the input in local languages. The second is the decision of whether to use the local language or default language to store the data obtained from the user

such as the changed process model. The former approach introduces a set of new problems when communicating components uses different languages.

6 Conclusions

Our paper discusses some of the issues that are encountered during the internationalization of a complex and distributed system. We have also explained the approach that was taken in addressing some of these problems and also the challenges that are to be met.

As a part of our future work we would like explore the challenges that are present in internationalizing a dynamic and adaptable system where the user may wish to interact with the system by providing input such as inserting a new process model in order to reflect on some of the changes that might be present in the relevant processes and policies involved locally. We would also like to incorporate localization into Chinese that has a huge market potential and also that would give an insight into the orientation problems and the difficulties associated with a larger character set. We would also like to explore the possible ways of reducing the burden of the linguists by using online resources for possible message translation and look whether we could achieve satisfactory results.

7 Acknowledgements

The authors wish to thank Martin Fleurke for his help in the translation in Dutch and Prof. Martin Purvis for supporting this work.

Otago Research Grant has supported this work.

8 References

Fleurke, M., Ehrler, L. and Purvis, M. (2003): JBees - an adaptive and distributed framework for workflow systems. *Proc. IEEE/WIC International Conference on Intelligent Agent Technology*, Halifax, Canada.

Meilin, S., Guangxin, Y., Yong, X. and Shangguang, W. (1998): Workflow Management Systems: A survery. *In Proceedings of IEEE International Conferenece on Communication Technology*. Beijing, China.

Purvis, M.,Hwang P, Purvis, M., Madhavji,N and Cranefield,S (2001) A practical look at software internationalization. *Transactions of the SDPS* September 2001,Vol.5, No.3, pp 79-90

He,X., Bustard, D.W and Liu, X (2002): Software Internationalisation and Localisation: Practise and Evolution *Principles and Practise of Programming in Java 2002.*

JFreeChart: Java Free Chart, Object Refinery Limited, http://www.jfree.org/jfreechart/index.html Accessed 01 Sep 2003.

Green,D., Trial: Internationalisation. in *The Java Tutorial*.,http://java.sun.com/docs/books/tutorial/i18n/index.html

Frost and Sullivan (2003): *Local language information technology market in India*. Submitted to Ministry of Information Technology, India.

Deitsch, A and Czarnecki, D *Java Internationalization*. O'Reilly & Associates, Inc., 2001

Aalst, W.M.P.v.d., and Hee, K.v. (2002) *Workflow Management: Models, Methods and Systems*. London, MIT Press.

Aalst, W.M.P.v.d., Hofstede, A.H.M.t., Kiepuszewski, B. and Barros, A.P. (2002) *Workflow Patterns*, Queensland University of Technology, Brisbane, Australia.

Jensen, K. (1992): Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. I: BasicConcepts. EATCS Monographs on Theoretical Computer Science, Heidelberg, Berlin. 1-234.