

# HiWorD: A Petri Net-Based Hierarchical Workflow Designer\*

Boualem Benatallah<sup>1</sup>, Piotr Chrzastowski-Wachtel<sup>3,4</sup>, Rachid Hamadi<sup>1</sup>, Milton O'Dell<sup>2</sup>, and Adi Susanto<sup>1</sup>

<sup>1</sup>School of Computer Science and Engineering  
The University of New South Wales  
Sydney NSW 2052, Australia  
{boualem,rhamadi,adis}@cse.unsw.edu.au

<sup>2</sup>Justwin Technologies Pty Ltd  
7-9 West Street Suite I.20, Level 1  
North Sydney NSW 2060, Australia  
modell@justwin.com

<sup>3</sup>Institute of Informatics, Warsaw University  
Banacha 2, PL 02-097 Warszawa, Poland  
pch@mimuw.edu.pl

<sup>4</sup>Polish-Japanese Institute of Information Technology  
Koszykowa 86, PL 02-008 Warszawa, Poland

## Abstract

*Much work is being conducted in the area of business process modeling using workflow technology. HiWorD is a hierarchical workflow modeling prototype with simulation capability. It models business processes using Petri nets in a hierarchical manner and implements recovery transitions as a technique to recover from exceptions. The workflow hierarchy is created by refining places and transitions using predefined patterns. By using these patterns, it is proven that the resulting workflow will be sound.*

## 1. Introduction

*HiWorD* (Hierarchical WORKflow Designer) was created to support workflow designers to model business processes and perform business process simulation (see Figure 1). *HiWorD* uses Petri nets to model workflows firstly because they are a well established means to describe concurrent systems, and secondly because of their solid and proven theoretical foundation.

*HiWorD* is unique compared to other Petri net editors that already exist for workflow modeling. *HiWorD* supports not only the creation of a standard Petri net workflow, but emphasizes other new ideas, such as building a hierarchical Petri net using predefined and safe refinement patterns, and recovery transitions for exception handling. The latter allows the designer to specify where to put tokens in the workflow net when recovering from a particular exception.

\*This work is partially supported by an ARC SPIRT grant "Managing Changes in Dynamic Workflow Environments" between UNSW, QUT, and Justwin Technologies, and by an internal research grant No. BW/ALG/01/2002 of PIJWSTK financially supported by KBN in Poland.

*HiWorD* is built along with a simulator. The simulator enables designers to view the workflow execution graphically and generates logs that will be stored in a relational database for further processing. Currently, the editor and the simulator are two separate independent programs. Both the editor and the simulator understand a common XML schema which provides a framework for communicating between them.

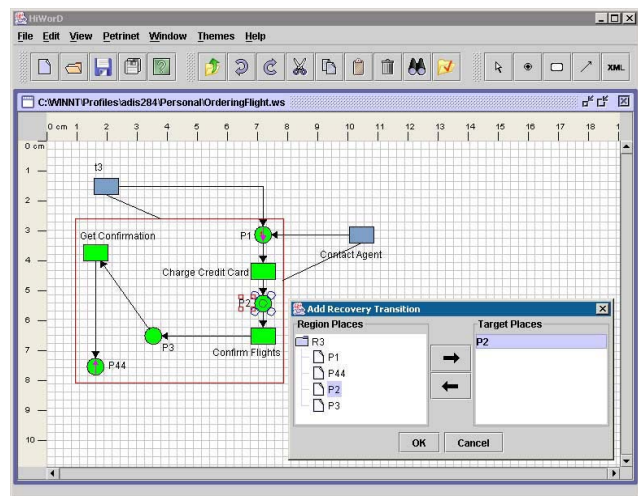


Figure 1. HiWorD Modeling Tool

*HiWorD* is written in Java and will run on any machine where a Java virtual machine is available. This includes OS/2, Apple Macintosh, several flavours of Unix such as Solaris or Linux, Windows 95/98/NT/2K, and many others. In the sequel, we will describe the main features supported by *HiWorD* in helping workflow designers to model and simulate business processes.

## 2. Features

### 2.1. Refinement Patterns

*HiWorD* allows the designer to design sound workflows by refining places and transitions using predefined patterns. By using these patterns, it is guaranteed that the produced workflow will be sound [2]. There are five main refinement patterns implanted in the tool itself, shown in Figure 2. Note that when a pattern is selected for a particular place or transition, it is possible to choose whether to move one level down or stay at the current level.

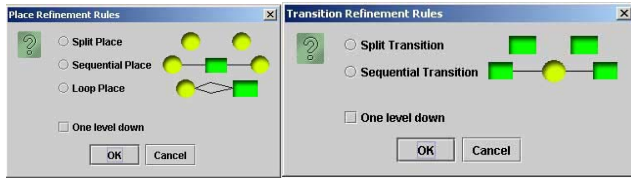


Figure 2. Refinement Patterns

### 2.2. Object Attributes Definition

A Petri net model consists of places, transitions, and arcs. *HiWorD* allows designers to add different attributes to these objects. For instance, every Petri net object has ID (non-editable identifier), name, and description attributes, regardless whether it is a place, a transition, or an arc (see Figure 3). Furthermore, a place can have zero or more tokens. A transition can have attributes such as lists of input data, output data, and resources needed to perform the corresponding task. One can also add weight and condition to an arc.

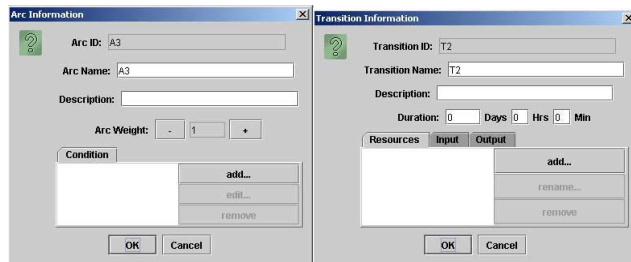


Figure 3. Transition and Arc Attributes

### 2.3. Recovery Transitions

Recovery transitions [1, 2] are a new concept introduced in *HiWorD*. The idea behind is that, by using recovery transitions, one can reset the location of tokens when an exception occurs during workflow execution. The exception is handled by putting tokens in some places and resuming the execution (see Figure 1).

*HiWorD* supports the addition of recovery transitions to regions. A region is constructed by refining a place using

refinement patterns defined previously. A recovery transition can only be added to a specified region (i.e., a place that has been refined). Each recovery transition has a condition attribute which specifies the event condition of the corresponding exception.

### 2.4. Hierarchical Browsing Capability

*HiWorD* builds workflows hierarchically. It makes sense to have browsing capability implemented in the tool. Browsing hierarchical workflows in *HiWorD* is easy. One can either use the refinement tree to browse through the structure of the hierarchy, or double click on the refined place or transition on the canvas to go down one level and press a specific button on the toolbar to go back up one level.

### 2.5. Simulation

Figure 4 depicts a screenshot of the simulator, which is integrated with *HiWorD*. The Petri net diagram drawn using *HiWorD* can be exported to an XML file. The simulator takes the XML generated by the editor as input in order to perform the simulation of workflow executions. It connects to the database to record the execution details.

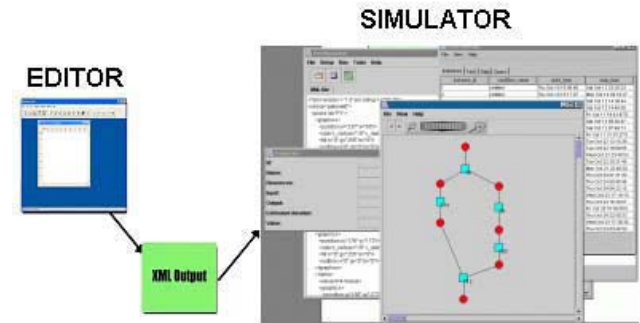


Figure 4. HiWorD Simulator

The simulator is able to run many simulations at once given various input parameters. It is also built using Java. The GUI support implemented in the simulator allows observers to view graphically the current state of the workflow instance execution.

## References

- [1] P. Chrzastowski-Wachtel. Recovery Nets: Model for Dynamic Workflows. In *Proceedings of the 13th Workshop on Concurrency, Specification, and Programming, Humboldt University Report*, Berlin, Germany, 2002.
- [2] P. Chrzastowski-Wachtel, B. Benatallah, R. Hamadi, M. O'Dell, and A. Susanto. A Top-Down Petri Net-based Approach for Dynamic Workflow Modeling. In *Proceedings of the International Conference on Business Process Management (BPM'03)*, Eindhoven, The Netherlands, June 2003. Springer Verlag.