Yet Another Smart Process EditoR

Kees van Hee     Reinier Post     Lou Somers

Architecture of Information Systems
Department of Mathematics and Computer Science
TU Eindhoven

ESM 2005, Porto
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion

Kees van Hee, Reinier Post, Lou Somers
Our main interests:

- business process modelling
- software systems modelling

An adequate modelling technique is

- clear
- powerful
- exact
- well-supported
Process modelling

Process modelling must express concurrency / cooperation.

Petri nets are adequate, but need better support.

Most processes are workflow nets: with fixed start and end points.
Many Petri net tools exist, mostly in the academic world. Our past contribution: the ExSpecT coloured Petri net tool.

Reasons to create another tool, Yasper:

- make workflows easy to simulate
- make Petri nets more palatable (for industry)
- Microsoft integration via .NET (for industry)
- integrate with other tools
Use cases for Yasper

- **Project: Deloitte Industry Prints:**
  - "best/standard practices" business process models
  - used by Deloitte consultants
  - without a good modelling technique
  - ⇒ many ambiguities and plain errors

- **Project: OGO 2.2:**
  - business software modelling/prototyping project
  - part of computer science curriculum
  - previously done with ExSpecT

- **Many other uses**
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
## Modelling features: overview

### basic Petri net elements

<table>
<thead>
<tr>
<th>element</th>
<th>notation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>place</td>
<td>○</td>
<td>condition or resource</td>
</tr>
<tr>
<td>transition</td>
<td>[        ]</td>
<td>event or action</td>
</tr>
<tr>
<td>arc</td>
<td>→</td>
<td>process flow</td>
</tr>
<tr>
<td>token</td>
<td>♦</td>
<td>(object in a) condition</td>
</tr>
</tbody>
</table>
## Modelling features: overview
### general purpose extensions

<table>
<thead>
<tr>
<th>element</th>
<th>notation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet</td>
<td><img src="image" alt="subnet" /></td>
<td>spread across multiple pages</td>
</tr>
<tr>
<td>xor</td>
<td><img src="image" alt="xor" /></td>
<td>choice (split / join)</td>
</tr>
<tr>
<td>role</td>
<td>(-)</td>
<td>executer / resource</td>
</tr>
<tr>
<td>store</td>
<td><img src="image" alt="store" /></td>
<td>data involved</td>
</tr>
<tr>
<td>inhibitor</td>
<td><img src="image" alt="inhibitor" /></td>
<td>negative condition (no tokens)</td>
</tr>
<tr>
<td>reset</td>
<td><img src="image" alt="reset" /></td>
<td>clear condition (clear tokens)</td>
</tr>
</tbody>
</table>
### Modelling features: overview
extensions for automatic simulation

<table>
<thead>
<tr>
<th>element</th>
<th>notation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td><img src="image" alt="time" /></td>
<td>processing time</td>
</tr>
<tr>
<td>cost</td>
<td>(-)</td>
<td>processing cost</td>
</tr>
<tr>
<td>case</td>
<td><img src="image" alt="case" /></td>
<td>preserves workflow case</td>
</tr>
<tr>
<td>emitor</td>
<td><img src="image" alt="emitor" /></td>
<td>generates workflow case</td>
</tr>
<tr>
<td>collector</td>
<td><img src="image" alt="collector" /></td>
<td>terminates workflow case</td>
</tr>
</tbody>
</table>
A basic Petri net
places, transitions, arcs, tokens

Getting fuel at a petrol station

Kees van Hee, Reinier Post, Lou Somers

TU/e

Yet Another Smart Process Editor
The net extended with Yasper features exhibiting most of them

Getting fuel, with more accuracy
Extension: subnets
spreading content over multiple pages
Extension: subnets
spreading content over multiple pages

The interface: references to places outside
Extension: choice
as known from flowcharts, UML activity diagrams

Choice elements indicate alternatives
Extension: choice
as known from flowcharts, UML activity diagrams

A basic Petri net equivalent
Extension: stores
indicate (not simulate) data manipulation

Stores: data involved in transitions
Transitions can be executed by roles
Extension: transition time and cost
fixed or stochastic

Time and cost assignments (for automatic simulation)
Extension: workflow cases

Case sensitive places, emitors, collectors

Emitors and collectors mark start and end of workflow
Extension: workflow cases

Case sensitive vs. case insensitive places

Kees van Hee, Reinier Post, Lou Somers
Extension: workflow cases

Transitions match cases on input places
Extension: special arc types

An example with more complex flow logic
Manual simulation in Yasper

Playing the token game in the diagram
Automatic simulation in Yasper

Starting an automatic run
Automatic simulation in in Yasper

The report

Kees van Hee, Reinier Post, Lou Somers

Yet Another Smart Process Editor
Yasper makes simulation work

In Yasper, simulations are

- based on exact execution semantics
- very easy to set up and run
Yasper makes simulation work

In Yasper, simulations are

- based on exact execution semantics
- very easy to set up and run
Yasper makes simulation work

In Yasper, simulations are
- based on exact execution semantics
- very easy to set up and run
Automatic simulation demonstrates errors and with alarming frequency, too

An error repaired
In Yasper, simulations

- immediately pinpoint most modelling errors
- pinpoint deadlocks / bottlenecks in the process itself
- can estimate overall throughput and efficiency
Simulation makes Yasper work

In Yasper, simulations

- immediately pinpoint most modelling errors
- pinpoint deadlocks / bottlenecks in the process itself
- can estimate overall throughput and efficiency
Simulation makes Yasper work

In Yasper, simulations

- immediately pinpoint most modelling errors
- pinpoint deadlocks / bottlenecks in the process itself
- can estimate overall throughput and efficiency
Simulation makes Yasper work

In Yasper, simulations
- immediately pinpoint most modelling errors
- pinpoint deadlocks / bottlenecks in the process itself
- can estimate overall throughput and efficiency
Many Petri net tools exist; few are for .NET. .NET = Microsoft’s Java equivalent:

- (good languages, libraries, and IDE)
- Microsoft integration (SQL Server, Office, GUI)
- more acceptable to industry (Deloitte)

Drawback:

- less portability (no Yasper on Linux)
Architecture
Yasper’s library dependencies
Availability

Yasper is partly free:

- Yasper program completely free to use (but don’t sue us when it breaks)
- code not free (but talk to us when you want it)

Get Yasper from www.yasper.org
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Development strategy: integration

Yasper’s design philosophy:

- do not compete with other tools trying to duplicate their features
- work with them instead by translation / calling
Areas of integration

- **the use of data ("color")**: interface Yasper with a "data manager"
- *Petri net model checking / verification*: interface Yasper with analysis tools
- *non-Petri net modelling techniques*: supply translations from/to Yasper
Areas of integration

- the use of data ("color"): interface Yasper with a "data manager"
- Petri net model checking / verification: interface Yasper with analysis tools
- non-Petri net modelling techniques: supply translations from/to Yasper
Areas of integration

- *the use of data ("color"):*
  interface Yasper with a "data manager"

- *Petri net model checking / verification:*
  interface Yasper with analysis tools

- *non-Petri net modelling techniques:*
  supply translations from/to Yasper
Methods of integration

- **reuse of code libraries**: see architecture diagram above
- **common file formats**
  PNML, extended PNML
Examples of integration

- analysis tools, e.g. Woflan, INA
- workflow engine (Yasper/InfoPath)
- simulation-only (with BPMN modeller)
- process model translations (e.g. UML activity diagrams, ProVision, BPMN, BPEL, ARIS, μCRL)
Outline of this talk

1. The need for Yasper
   - Petri nets for process modelling
   - Why create Yasper?

2. A closer look at Yasper
   - Yasper’s modelling features
   - Simulation in Yasper
   - Implementation notes

3. Integration
   - Yasper and other tools

4. Conclusion
Findings

- Yasper simplifies Petri-net based modelling and simulation
- Yasper’s simulation is of great benefit in modelling
- Integrating tools is hard, but pays off
Ongoing/future work on Yasper itself

Provide more convenient editing:
- larger nets
- transformations
- consistency checks
Ongoing/future work on integration

- continue with workflow engine (Yasper/InfoPath)
- more process model translations
- better feedback from analysis tools
- Petri net transformation and generation
- process model repository
- etc.
Thank you

- **Maarten Leurs**
  - lots of Yasper programming
  - applying Yasper at Deloitte

- **Andries van Dijk**
  - support at Deloitte

- **Olivia Oanea, Ivo Raedts, Jan Martijn van der Werf, a.o.**
  - using Yasper, making suggestions, bug reports
  - writing related software
  - help with this presentation

- **Till Tantau**
  - the \texttt{\LaTeX} beamer package

- **this audience**
  - any feedback you have