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

Systems modelling

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.

Yasper

Why Yet Another Smart Process EditoR?

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

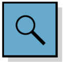
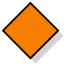
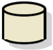
Modelling features: overview

basic Petri net elements

element	notation	description
place	○	condition or resource
transition	🟩	event or action
arc	→	process flow
token	⚄	(object in a) condition

Modelling features: overview

general purpose extensions

element	notation	description
subnet		spread across multiple pages
xor		choice (split / join)
role	(-)	executer / resource
store		data involved
inhibitor	———●	negative condition (no tokens)
reset	- - - - - -	clear condition (clear tokens)

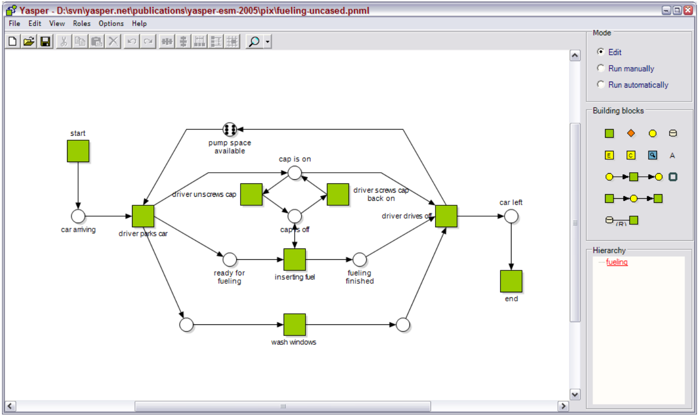
Modelling features: overview

extensions for automatic simulation

element	notation	description
time	◙	processing time
cost	(-)	processing cost
case	🟡	preserves workflow case
emitor	E	generates workflow case
collector	C	terminates workflow case

A basic Petri net

places, transitions, arcs, tokens



Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-uncased.pnml

Mode

● Edit

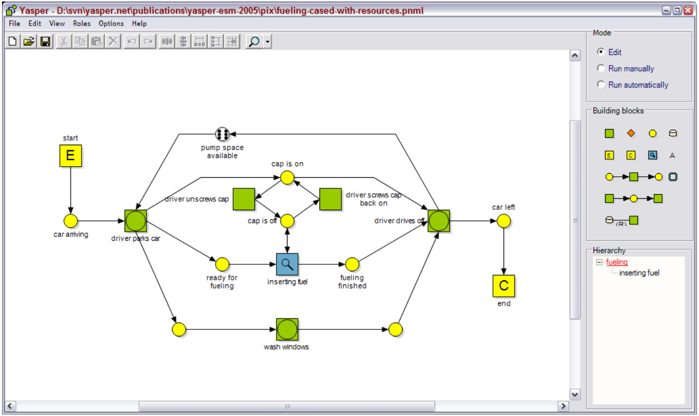
○ Run manually

○ Run automatically

Getting fuel at a petrol station

The net extended with Yasper features

exhibiting most of them



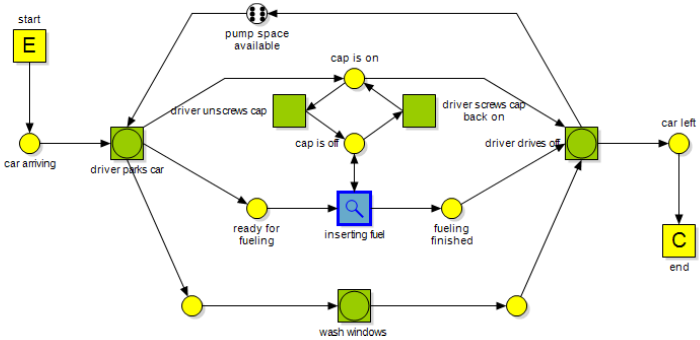
Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-cased-with-resources.pnml

Mode
● Edit
○ Run manually
○ Run automatically

Getting fuel, with more accuracy

Extension: subnets

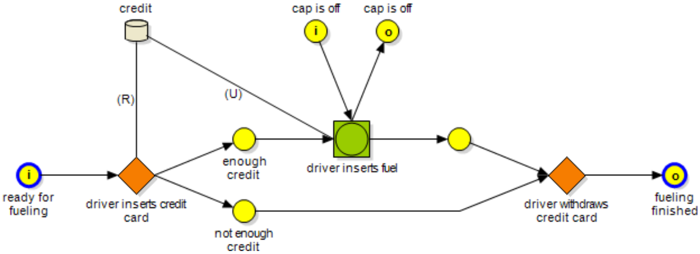
spreading content over multiple pages



A subnet element

Extension: subnets

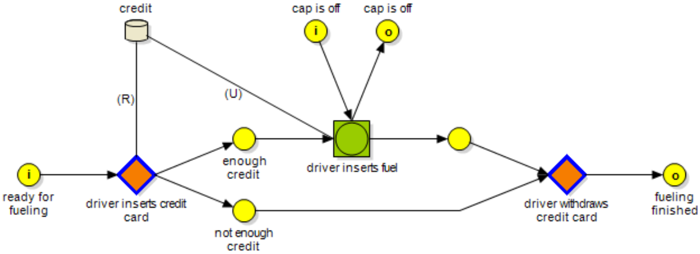
spreading content over multiple pages



The interface: references to places outside

Extension: choice

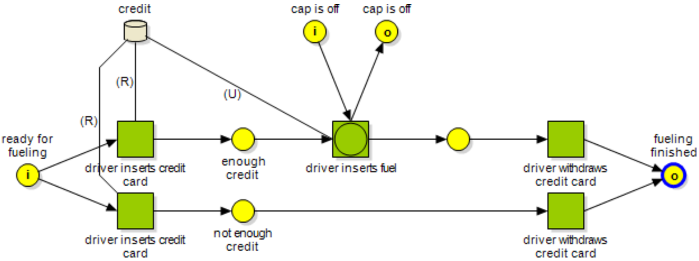
as known from flowcharts, UML activity diagrams



Choice elements indicate alternatives

Extension: choice

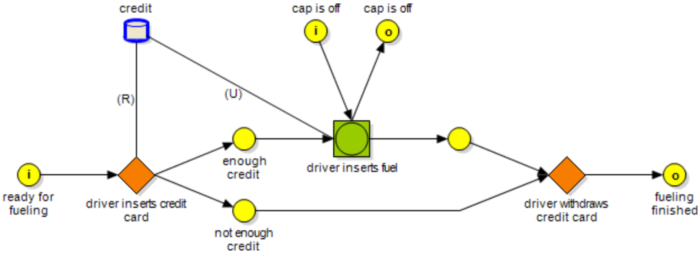
as known from flowcharts, UML activity diagrams



A basic Petri net equivalent

Extension: stores

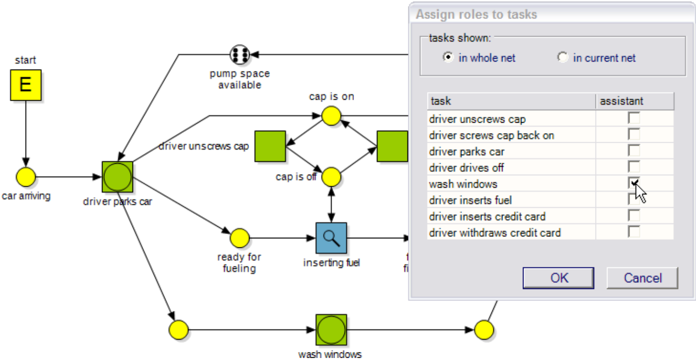
indicate (not simulate) data manipulation



Stores: data involved in transitions

Extension: roles

executers / resources



Assign roles to tasks

tasks shown:

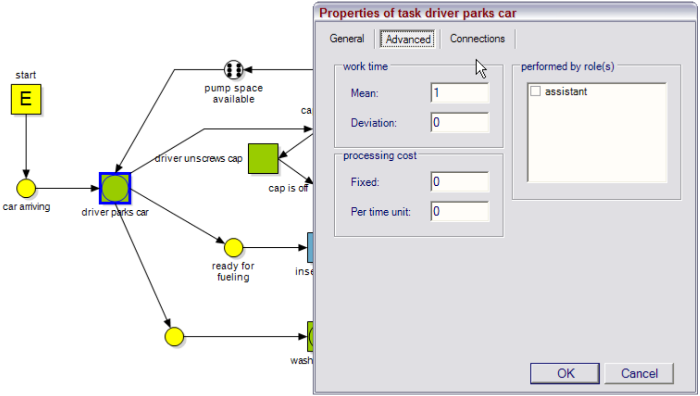
● in whole net ○ in current net

task	assistant
driver unscrews cap	□
driver screws cap back on	□
driver parks car	□
driver drives off	□
wash windows	☑
driver inserts fuel	□
driver inserts credit card	□
driver withdraws credit card	□

Transitions can be executed by roles

Extension: transition time and cost

fixed or stochastic



Properties of task driver parks car

work time

- Mean: 1
- Deviation: 0

processing cost

- Fixed: 0
- Per time unit: 0

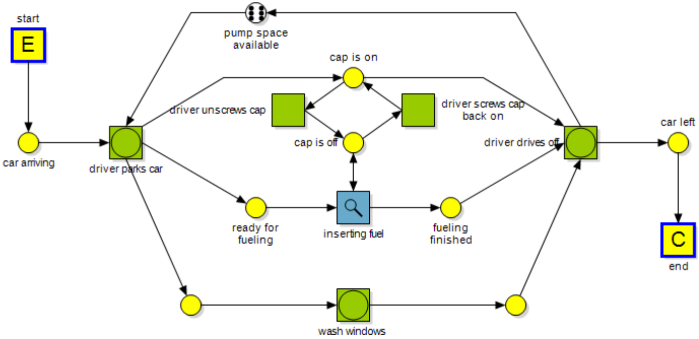
performed by role(s)

- □ assistant

Time and cost assignments (for automatic simulation)

Extension: workflow cases

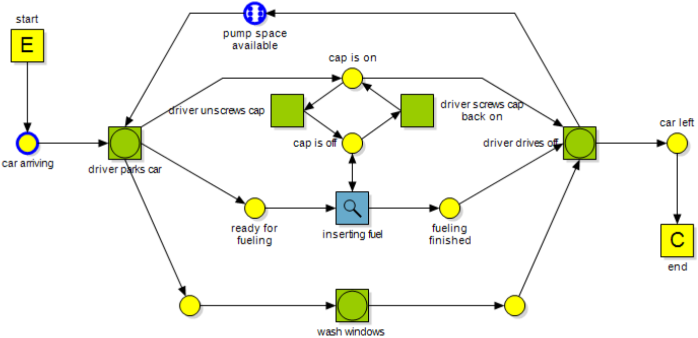
case sensitive places, emitors, collectors



Emitors and collectors mark start and end of workflow

Extension: workflow cases

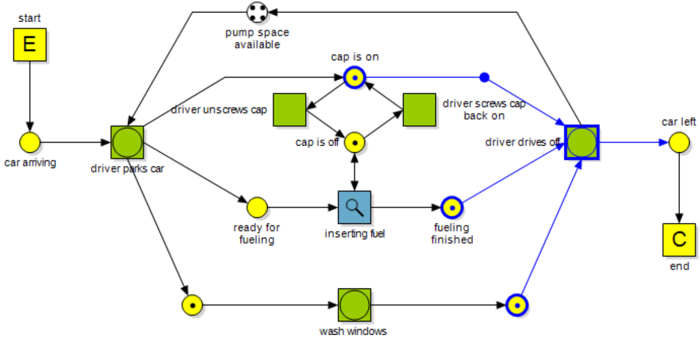
case sensitive places, emitors, collectors



Case sensitive vs. case insensitive places

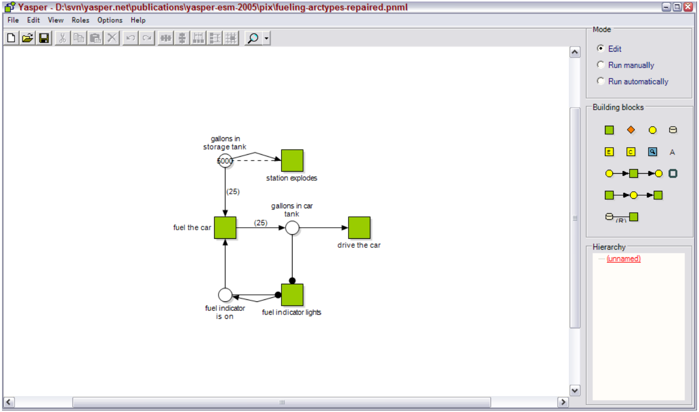
Extension: workflow cases

case sensitive places, emitors, collectors



Transitions match cases on input places

Extension: special arc types



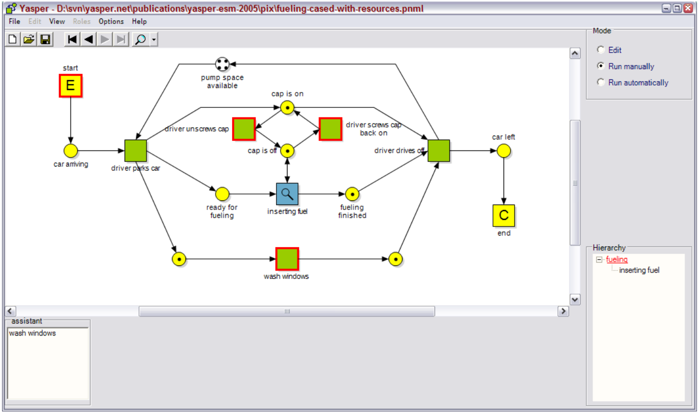
Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-arctypes-repaired.pnml

Mode

- ● Edit
- ○ Run manually
- ○ Run automatically

An example with more complex flow logic

Manual simulation in Yasper

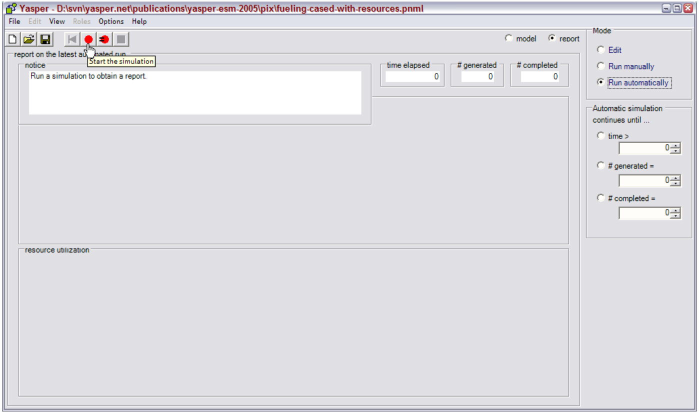


Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-cased-with-resources.pnml

Mode
○ Edit
● Run manually
○ Run automatically

Playing the token game in the diagram

Automatic simulation in Yasper



Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-cased-with-resources.pnml

○ model ● report

report on the latest automated run

notice

Run a simulation to obtain a report.

time elapsed	# generated	# completed
0	0	0

resource utilization

Mode

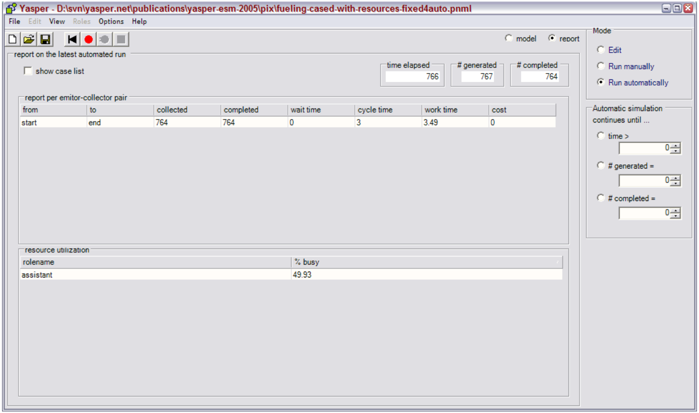
○ Edit
○ Run manually
● Run automatically

Automatic simulation continues until ...

○ time > 0
○ # generated = 0
○ # completed = 0

Starting an automatic run

Automatic simulation in Yasper


Yasper - D:\svn\yasper.net\publications\yasper-esm-2005\pix\fueling-cased-with-resources-fixed4auto.pnml
File
Edit
View
Roles
Options
Help

○ model ● report

Mode

○ Edit
○ Run manually
● Run automatically

report on the latest automated run

□ show case list

time elapsed: 766
# generated: 767
# completed: 764

report per emitor-collector pair

from	to	collected	completed	wait time	cycle time	work time	cost
start	end	764	764	0	3	3.49	0

Automatic simulation continues until ...

○ time > 0
○ # generated = 0
○ # completed = 0

resource utilization

rolename	% busy
assistant	49.93

The report

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

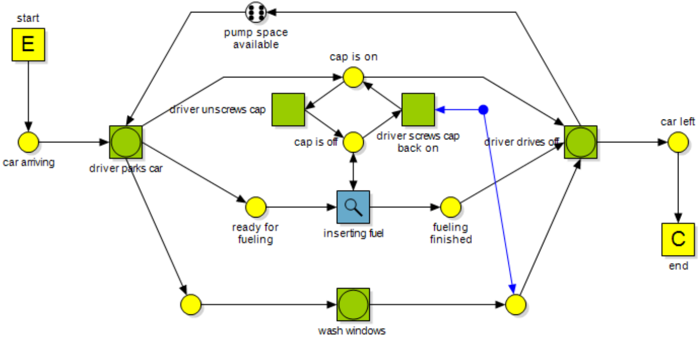
Yasper makes simulation work

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

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

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

Software platform

the choice for .NET

Many Petri net tools exist; few are for .NET.
.NET = Microsoft's Java equivalent:

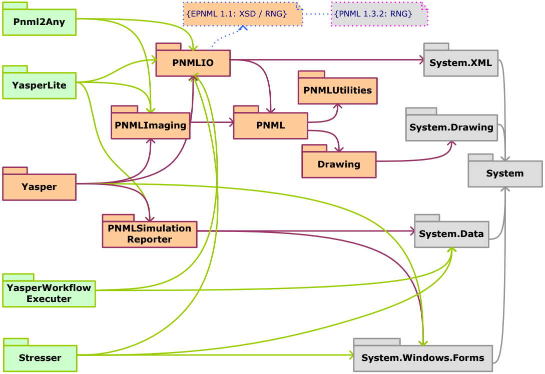
- (good languages, libraries, and IDE)
- Microsoft integration (SQL Server, Office, GUI)
- more acceptable to industry (Deloitte)

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 LaTeX beamer package
- *this audience*
  - any feedback you have