Application of Petri-Nets in Object-Oriented Environment

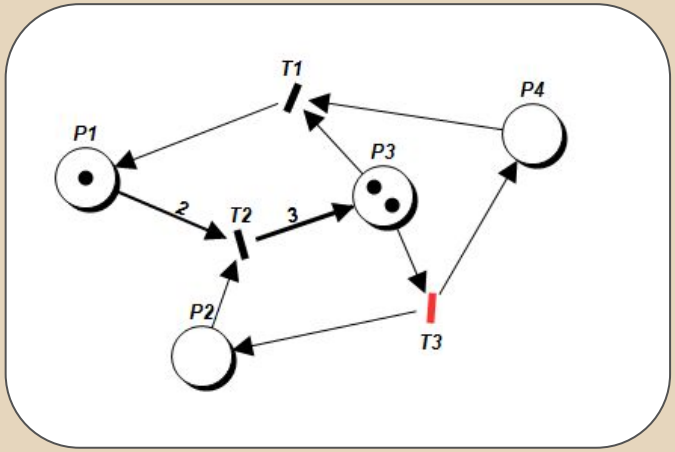
17th International Symposium on Computational Intelligence and Informatics November 17-19, 2016 Budapest, Hungary

Dávid Bedők
Óbuda University
John von Neumann Faculty of Informatics
bedok.david@nik.uni-obuda.hu

Abstract

Petri nets

- Effective modeling tool
  - graphical and mathematical description at the same time
  - quickly review complex systems
- Intuitive characteristics
  - nondeterministic simulations



This kind of behavior is **hard to be implemented in an imperative language**.

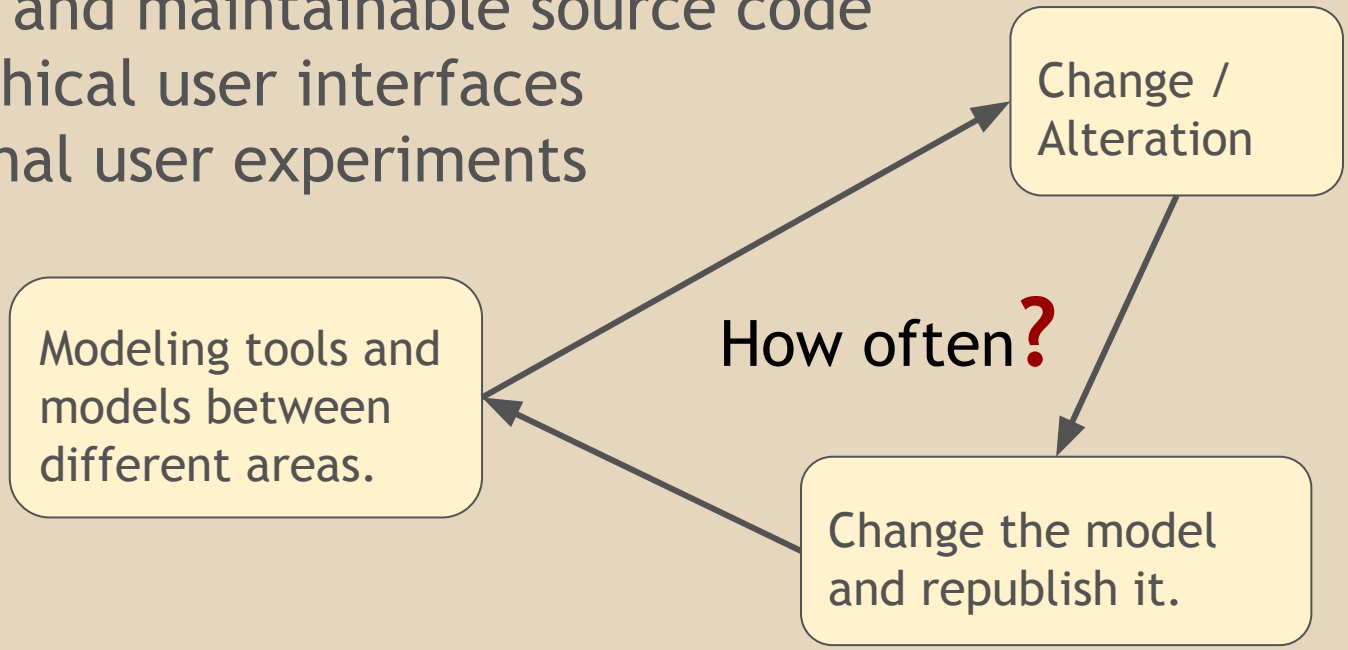
Objective: embed Petri-nets into an imperative high-level language and use its beneficial properties.

Why?

- Live visualization
- Live or generated documentations

Non or not-only technical skills

- Deliver quality
- Readable and maintainable source code
- Nice graphical user interfaces
- Professional user experiments



Petri nets or Place/Transition nets

- Mathematical modeling language for describing distributed and parallel systems
- Directed and weighted bipartite graph
  - two types of nodes: **position** and **transition**
  - weighted directed edges between positions and transitions
- Each position can contain any number of **tokens**
  - the distribution of the tokens determines the actual state of the Petri-net

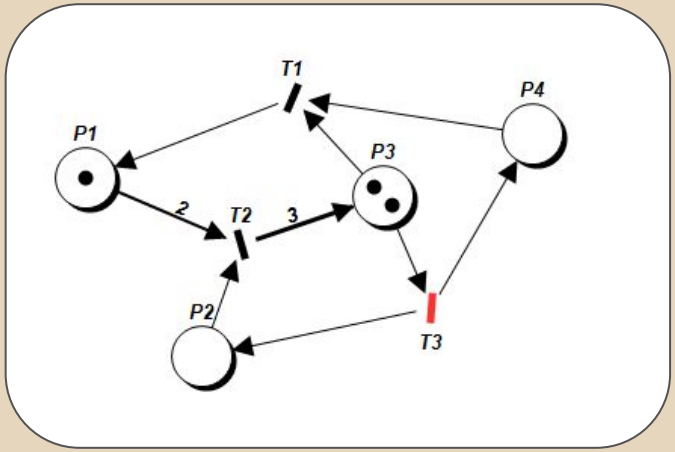
Carl Adam Petri

(12 July 1926 – 2 July 2010)

- The firing order is fully **nondeterministic**

Petri nets or Place/Transition nets

- Mathematical modeling language for describing distributed and parallel systems
- Directed and weighted bipartite graph
  - two types of nodes: **position** and **transition**
  - weighted directed edges between positions and transitions
- Each position can contain any number of **tokens**
  - the distribution of the tokens determines the actual state of the Petri-net



- The firing order is fully **nondeterministic**

Extensions

Problem: simple but real system → quite big graph → this reduces the transparency

Solution: Petri-net extensions / High-Level PNs nets

- Reset arc
- Prioritised Petri-nets
- Coloured Petri-nets
- Algebraic Petri-nets
- Hierarchical Petri-nets
- Object Petri-nets
- etc.

Representation

Petri Net Markup Language (PNML)

- output format of the ***Petri Net Kernel*** application
- standard XML based description
  - Original Petri-nets (Place/Transition nets)
  - High-Level Petri-nets (e.g.: Coloured Petri nets)
  - Symmetric nets
- openness, expandable
  - standalone **Petri Net Type Definition** (PNTD) file

Related works

Renew

- Petri-nets → object-oriented **classes** (Reference net)
- edge and guard expressions
  - like in a Coloured Petri-net
  - simplified Java syntax
- the Reference nets can be loaded in a Java application
  - the simulation can be played via the API

All operations, methods, initializations **have to be defined in advance** inside the source code of the Reference net.

System design

Create a new Petri net extension where **Petri events** are able to be set inside the net.

With that events connections will be created between the simulation and a real application whatever program language is used.

The following extensions will be used in the new model:

- reset arcs
- capacity limits
- inhibitor arcs
- prioritized Petri-nets
- **Petri events** (new)

Preparations

- a **well-formed description**
- a **new extension** to model the object-oriented events
- identify the **Petri events**
  - global events (e.g. cycle or deadlock situation)
  - moments of the transitions' activation (before/after a transition fires)
  - token player movement (before/after a position gets/loses token players)
  - change of the token distribution (before/after a specified state is activated)

Portrayal of the network

- **XML document**
- well-defined hierarchy and a cross-platform behavior
- clear and detailed **XSD schema files**

PetriNetwork

```
<?xml version="1.0" encoding="utf-8"?>
<pn:PetriNetwork xmlns:pn="http://petrinetwork.hu">
  <pn:NetworkSettings>
    [..]
  </pn:NetworkSettings>
  <pn:Network>
    <neit:NetworkItems xmlns:neit="http://networkitem.petrinetwork.hu">
      [..]
    </neit:NetworkItems>
    <neit:Edges xmlns:neit="http://networkitem.petrinetwork.hu">
      [..]
    </neit:Edges>
  </pn:Network>
</pn:PetriNetwork>
```

Portrayal of the events

Position

```
<i:Position bi:name="P1" bi:unid="1" bi:showannotation="True"
i:radius="20" pos:capacitylimit="0">
    <pf:TopLeftPoint pf:x="92" pf:y="188" />
    <pf:Origo pf:x="112" pf:y="208" />
    <pf:LabelOffset pf:x="0" pf:y="0" />
    <sf:Size sf:width="40" sf:height="40" />
    <pos:Tokens>
      <tok:Token bi:name="1" bi:unid="26" bi:showannotation="True">
        <c:TokenColor c:red="0" c:green="0"c:blue="0"c:alpha="5" />
      </tok:Token>
      [..]
    </pos:Tokens>
    <i:Events>
      <pe:ItemEvent pe:name="test" pe:type="PREACTIVATE"
xmlns:pe="http://event.petrinetwork.hu" />
      [..]
    </i:Events>
</i:Position>
```

Events

- **Global events**
  - DEADLOCK
  - CYCLE
  - TICK
- **Topology items** (Position/Transition)
  - Before activation (PREACTIVATE)
  - After activation (POSTACTIVATE)
- **State hierarchy**
  - Before activation (PREACTIVATE)
  - After activation (POSTACTIVATE)

Only the type and the name identifies an event, and the name is not unique intentionally.

Application in OO environment

Steps

- The data of the network's topology have to be **loaded into the memory**
- The visualization properties of the network can be omitted
- The **firing process** of the opened network has to be **played in the application**

All of these features are part of an API which is a key part of the new Petri-net extension.

API of Petri-events

- the color of a token player
- the weight, type and junction points of an edge
- the text of any annotation (comment) and its owner
- the capacity and the list of tokens of a position
- the priority and the type of a transition
- the entire token distribution of a state vector
- **the EventTrunk of any entity and the network**
- the non-visual properties of the network
- **list of all state's names in the network**
- **all unique (!) event's names in the network**

Token gameplay

Open

```
using PetriNetworkLibrary.Model.NetworkItem;
[..]
Random rand = new Random();
PetriNetwork network = PetriNetwork.openFromXml(rand,
@"networks\Demo.pn.xml");
```

In order to build connections between the API and the application the developer has to bind the event handlers via the `PetriHandler` delegate.

Delegate

```
public delegate void PetriHandler(AbstractEventDrivenItem
item, EventType eventType);
```

Token gameplay

Event Handler

```
using PetriNetworkLibrary.Model.NetworkItem;
[..]
private static void eventHandler(AbstractEventDrivenItem item, EventType
eventType)
{
    StringBuilder sb = new StringBuilder(100);
    sb.Append("eventHandler(item: " + item.Name + ", eventType:
"+eventType+")");
    if (item is Position) {
        sb.Append(" token count: " + ((Position)item).TokenCount);
    }
    System.Console.WriteLine(sb.ToString());
}
```

Bind Petri Events

```
using PetriNetworkLibrary.Model.NetworkItem;
[..]
List<String> listOfEvents = network.EventsName;
foreach (String item in listOfEvents)
{
    network.bindPetriEvent(item,  new PetriHandler(eventHandler));
}
```

Token gameplay

Token Gameplay

```
using PetriNetworkLibrary.Model.NetworkItem;
[..]
FireEvent fireEvent = FireEvent.INITFIRE;
FireReturn current = null;
while (!FireEvent.DEADLOCK.Equals(fireEvent))
{
   current = network.fire();
   System.Console.WriteLine(current);
   fireEvent = current.FireEvent;
}
```

Comparison and objectives

Another High-Level Petri-net (to model complex algorithms) but in object-oriented environment like the Renew? **NO**

A simpler model will be used inside an object-oriented application for instance as part of a state machine.

In a Petri-net simulator

- Play the token gameplay
- Set the Petri events

In a production code

- Use the non-deterministic behavior
- Part of the business logic

Results

When somebody wants to implement a state machine which is able to handle the transitions between the application's states, a Petri-net model creation may be needed to help the understanding of the task. In that case the model should not be only kind of static document of the product.

With this new Petri-net extension it can be highlight connections between the model and the real source code events.

If somebody changes the model it will take effect "immediately" in the product.

Q & A

Thank you