AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

AGH

Formal methods for software development.
An overview

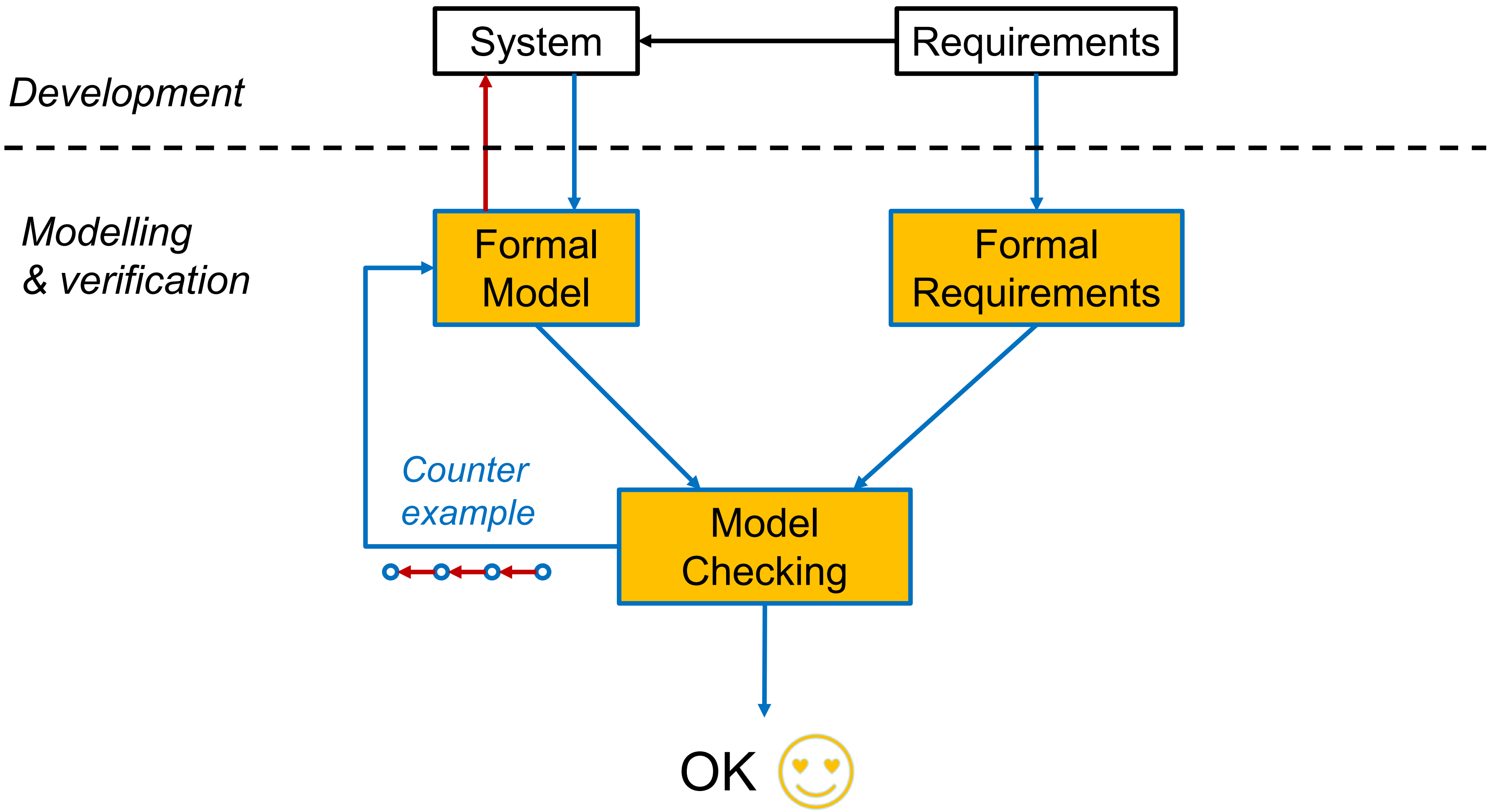
Tomasz Szmuc
AGH University of Science and Technology
Department of Applied Computer Science
tsz@agh.edu.pl

Budapest, November 4, 2019

Agenda

1. Qualitative and quantitative approach. Scope of the presentation
2. Direct use of formal description language for modelling
3. Automatic translation of software artefacts into formal models
4. Construction of integrated environment for software development with rigorous formal support – Alvis project
5. Implementation issues
6. Conclusions

Formal modelling & verification



AGH

Formal methods supporting modelling and verification. Qualitative approach

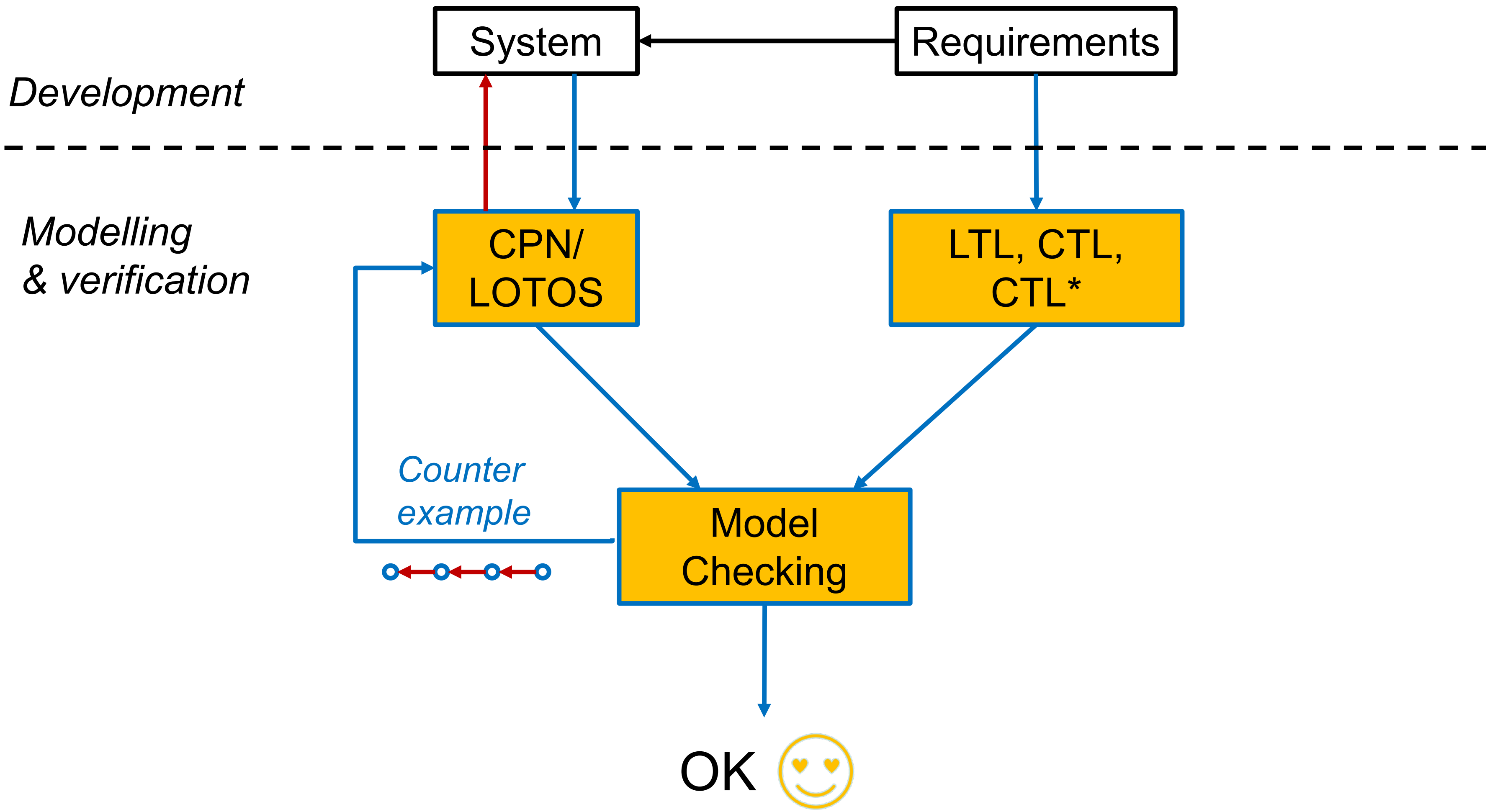
A. Modelling behaviour - generation mainly LTS

- (High Level) **Petri nets** (**CPN Tools**, UPAAL)
- **Timed automata, Hybrid Automata** (UPAAL)
- **Process Algebras** (LOTOS, CADP)

B. Logic based description of requirements
temporal logics: LTL, CTL*, CTL, …

C. Proving using **Model Checkers** or **SAT Solvers**.

Qualitative formal modelling & verification



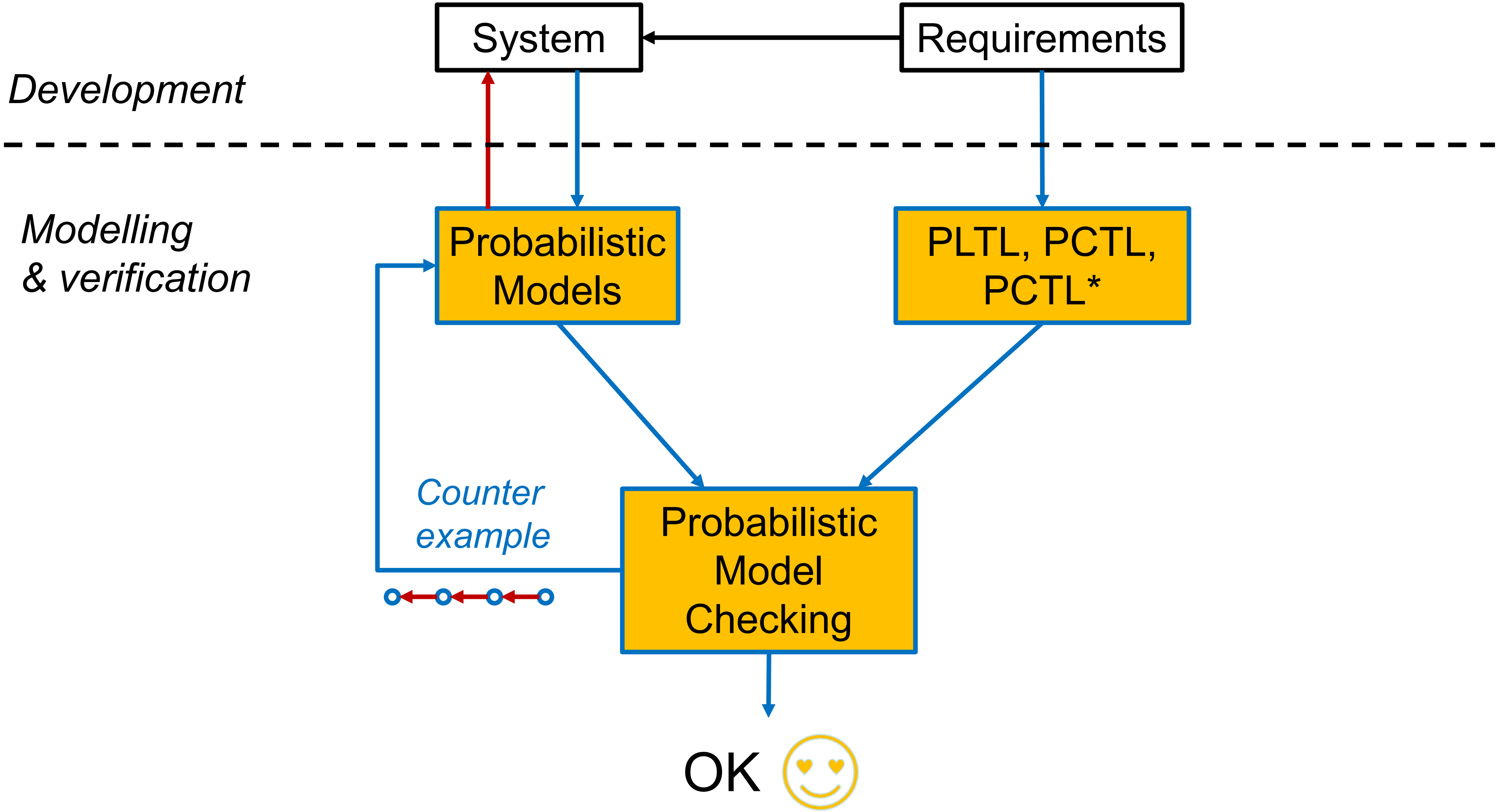
Formal methods supporting modelling and verification. Quantitative approach

A. **Modelling of processes** – Bayesian networks, Markov Chains, Markov Processes (Discrete, Continues), etc.

B. **Logic based description of requirements**
 probablilistic temporal logics: PLTL, PCTL*, PCTL, …

C. **Verification** – probabilistic model checking – PRISM
 https://www.prismmodelchecker.org/

Quantitaive formal modelling & verification.



PRISM – Probabilistic Model Checker 1/3

Probabilistic models described using PRISM language:

- discrete-time Markov chains (DTMCs)
- continuous-time Markov chains (CTMCs)
- Markov decision processes (MDPs)
- probabilistic automata (PAs)
- probabilistic timed automata (PTAs)
- Stochastic Petri Nets

Property specification language incorporates the well known temporal logics:

- PCTL (probabilistic computation tree logic),
- CSL (continuous stochastic logic),
- LTL (linear time logic),
- PCTL* (subsumes both PCTL and LTL).

https://www.prismmodelchecker.org/

PRISM – Probabilistic Model Checker 2/3

Case studies in many application domains

- Randomised distributed algorithms
- Communication, network and multimedia protocols
- Security related systems, contract signing and fair exchange protocols, anonymity, threads and attacks, quantum cryptography protocols, …
- Planning and synthesis
- Performance and reliability,
- Game theory
- Power management
- Biology
- …

https://www.prismmodelchecker.org/

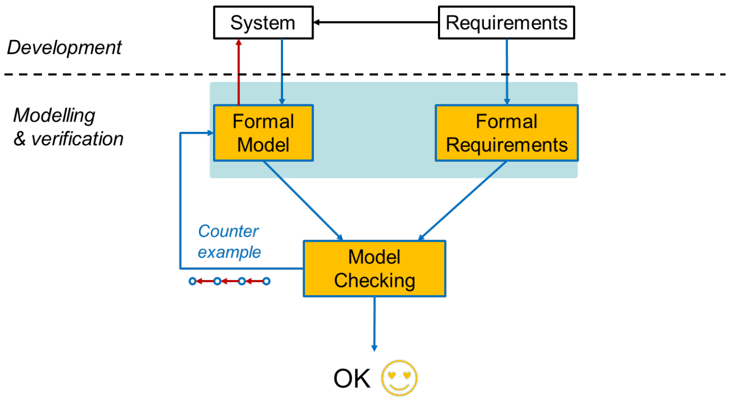
PRISM – Probabilistic Model Checker 3/3

Main publication:

Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, vol. 6806 of LNCS, pp. 585-591, Springer, 2011

https://www.prismmodelchecker.org/

Qualitative approach. Building formal models



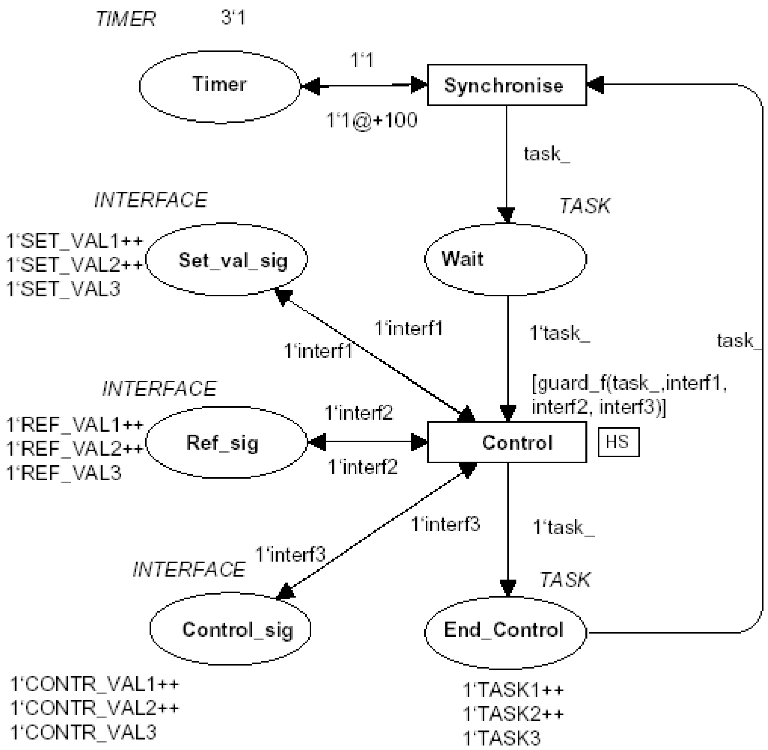
1. Direct approach using existing (modified) formal description language and the related tool (CPN, Automata, Process algebra, e.g. LOTOS)
2. Automatic translation of software models (UML, SysML, AADL) into formal description language
3. Development of environment supporting software design building formal models

AGH

Direct approach using existing (modified) formal description language and the related tool (CPN, Automata, Process algebra, e.g. LOTOS)

Hierarchical Timed Coloured Petri Nets (HTCPN) An overview

http://cpntools.org/

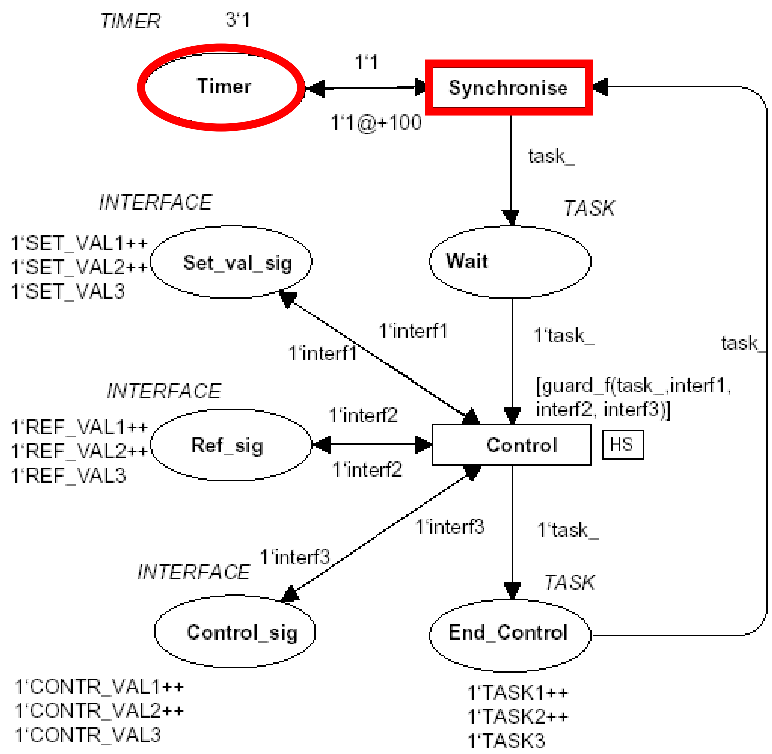



```
color TASK = with
 TASK1 | TASK2 | TASK3 timed;
color INTERFACE = with
 SET_VAL1 |  SET_VAL2 |  SET_VAL3 |
 REF_VAL1 | REF_VAL2 | REF_VAL3 |
 CONTR_VAL1 | CONTR_VAL2 |
 CONTR_VAL3;
color TIMER = int timed;

var task_  : TASK;
var interf, interf1, interf2, interf3 : INTERFACE;
var tim  : TIMER;
```

Samolej S., Szmuc T.: *HTCPNs–Based Tool for Web–Server Clusters Development* in Software Engineering Techniques, LNCS vol. 4890, 2011, pp. 131-142

Hierarchical Timed Coloured Petri Nets (HTCPN) Overview

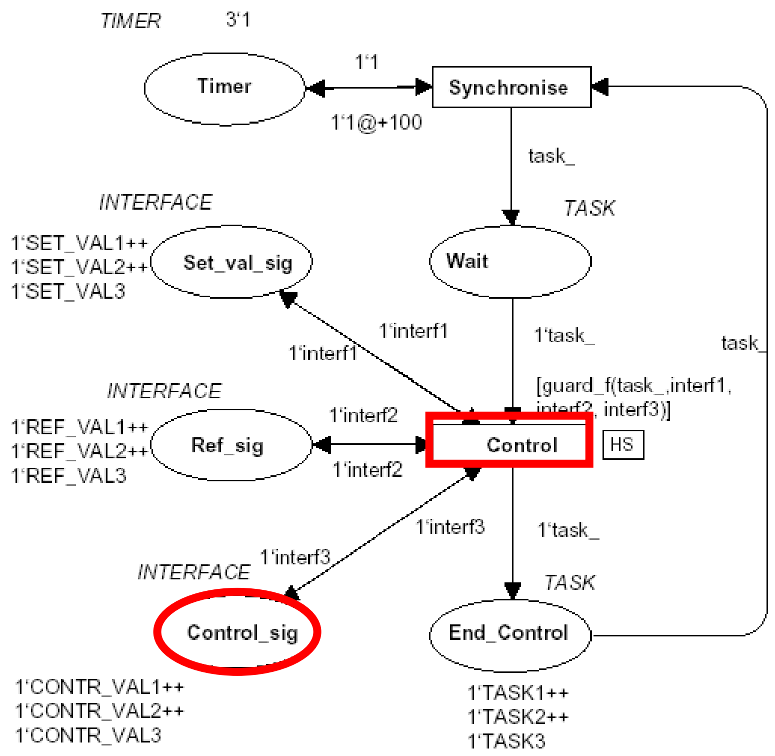



```
color TASK = with
 TASK1 | TASK2 | TASK3 timed;
color INTERFACE = with
 SET_VAL1 |  SET_VAL2 |  SET_VAL3 |
 REF_VAL1 | REF_VAL2 | REF_VAL3 |
 CONTR_VAL1 | CONTR_VAL2 |
 CONTR_VAL3;
color TIMER = int timed;

var task_  : TASK;
var interf, interf1, interf2, interf3 : INTERFACE;
var tim  : TIMER;
```

Samolej S., Szmuc T.: *HTCPNs–Based Tool for Web–Server Clusters Development* in Software Engineering Techniques, LNCS vol. 4890, 2011, pp. 131-142

Hierarchical Timed Coloured Petri Nets (HTCPN)
An overview



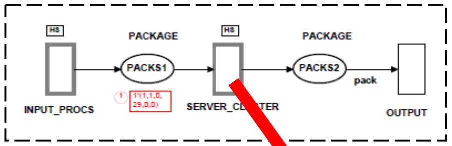

```
color TASK = with
 TASK1 | TASK2 | TASK3 timed;
color INTERFACE = with
 SET_VAL1 |  SET_VAL2 |  SET_VAL3 |
 REF_VAL1 | REF_VAL2 | REF_VAL3 |
 CONTR_VAL1 | CONTR_VAL2 |
 CONTR_VAL3;
color TIMER = int timed;

var task_  : TASK;
var interf, interf1, interf2, interf3 : INTERFACE;
var tim  : TIMER;
```

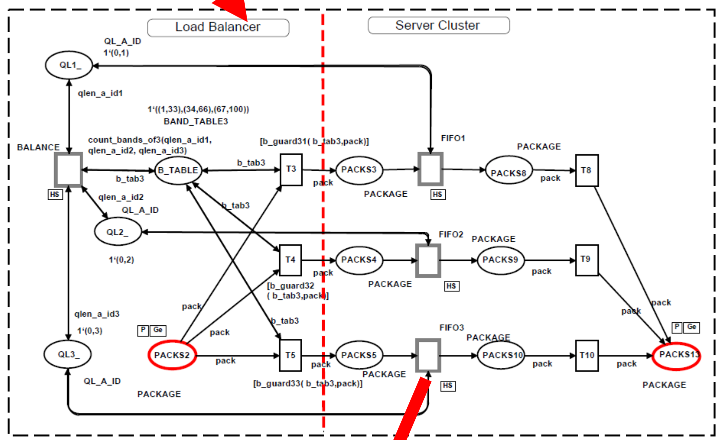
Samolej S., Szmuc T.: *HTCPNs–Based Tool for Web–Server Clusters Development* in Software Engineering Techniques, LNCS vol. 4890, 2011, pp. 131-142

Modelling queueing system

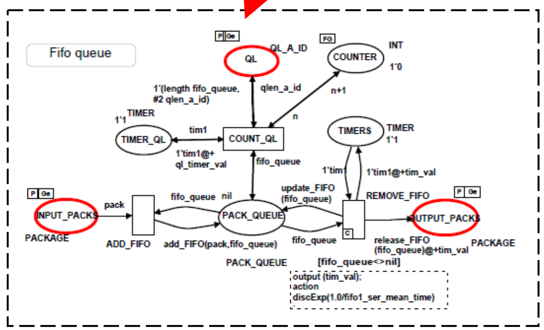
Top–level system model



Packet distribution patterns



Queueing systems patterns



Samolej S., Szmuc T.: *HTCPNs–Based Tool for Web–Server Clusters Development* in Software Engineering Techniques, LNCS vol. 4890, 2011, pp. 131-142

Direct validation possibilities

- Detection of the following system states
  - Balance or unbalance of the system under certain load
  - Average system parameters under balanced state
- New cluster structures and data flow rules may be tested
- Checking maximal length of queues, time requirements etc.
- Others offered by CPN Tools

Samolej S., Szmuc T.: *HTCPNs–Based Tool for Web–Server Clusters Development* in Software Engineering Techniques, LNCS vol. 4890, 2011, pp. 131-142

Modified HTCPN.
Decision Nets (D-Nets) and Real Time Coloured Petri Nets (RTCP)

Introduction of D-Nets (Decision Nets) modelling decision tables enabling checking consistency and completnes of the tables (requirements)

Modifications of HTCPN - RTCP

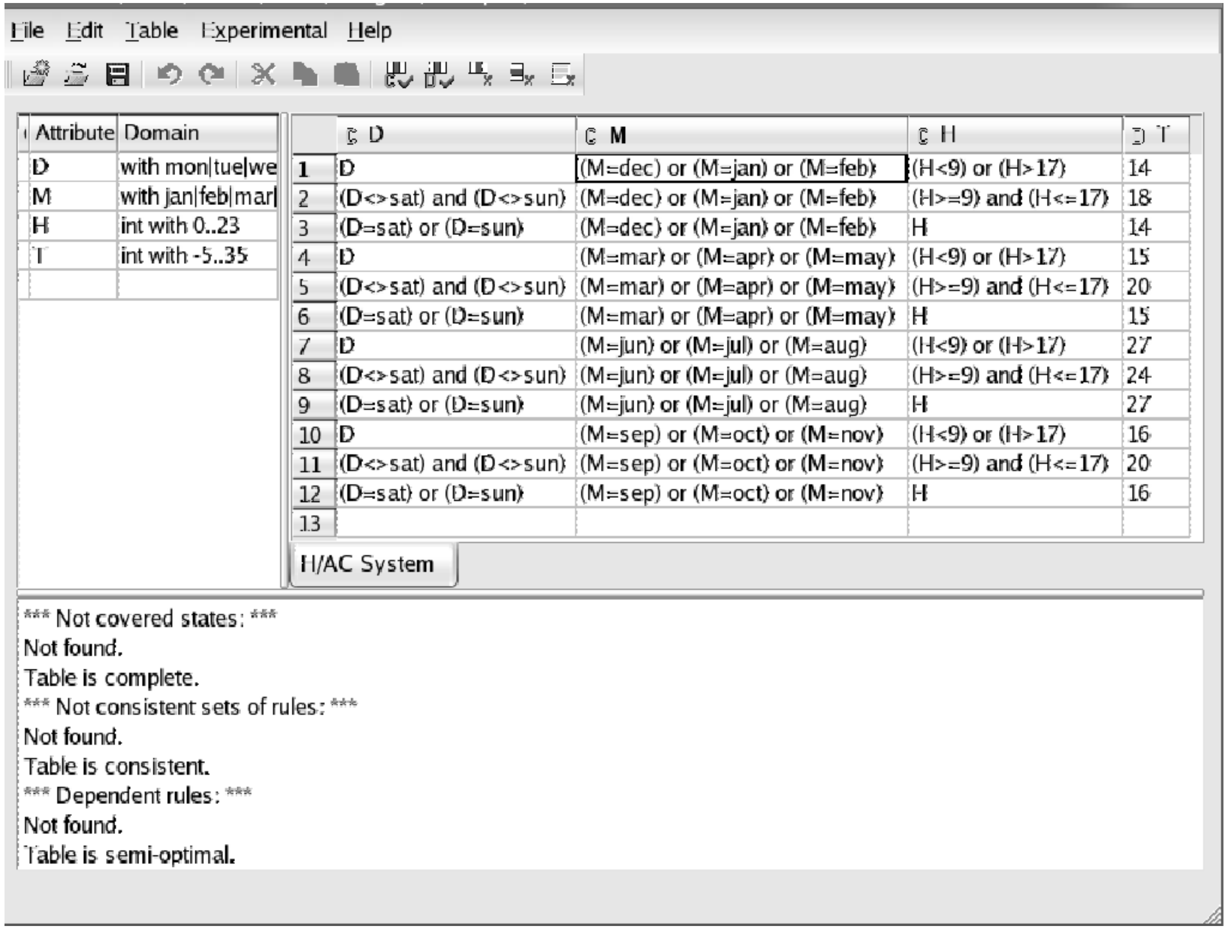
1. Priorities are assigned to transitions
2. Multiple arcs are not allowed
3. All colours are of time type
4. Time stamps are attached to places. Positive value of a stamp specifies minimal time before the token may be used. Negative value specifies the „age" of the token.

D-Nets & Adder tool

– *Completeness* – iff it exists at least one rule succeeding for any possible input state.

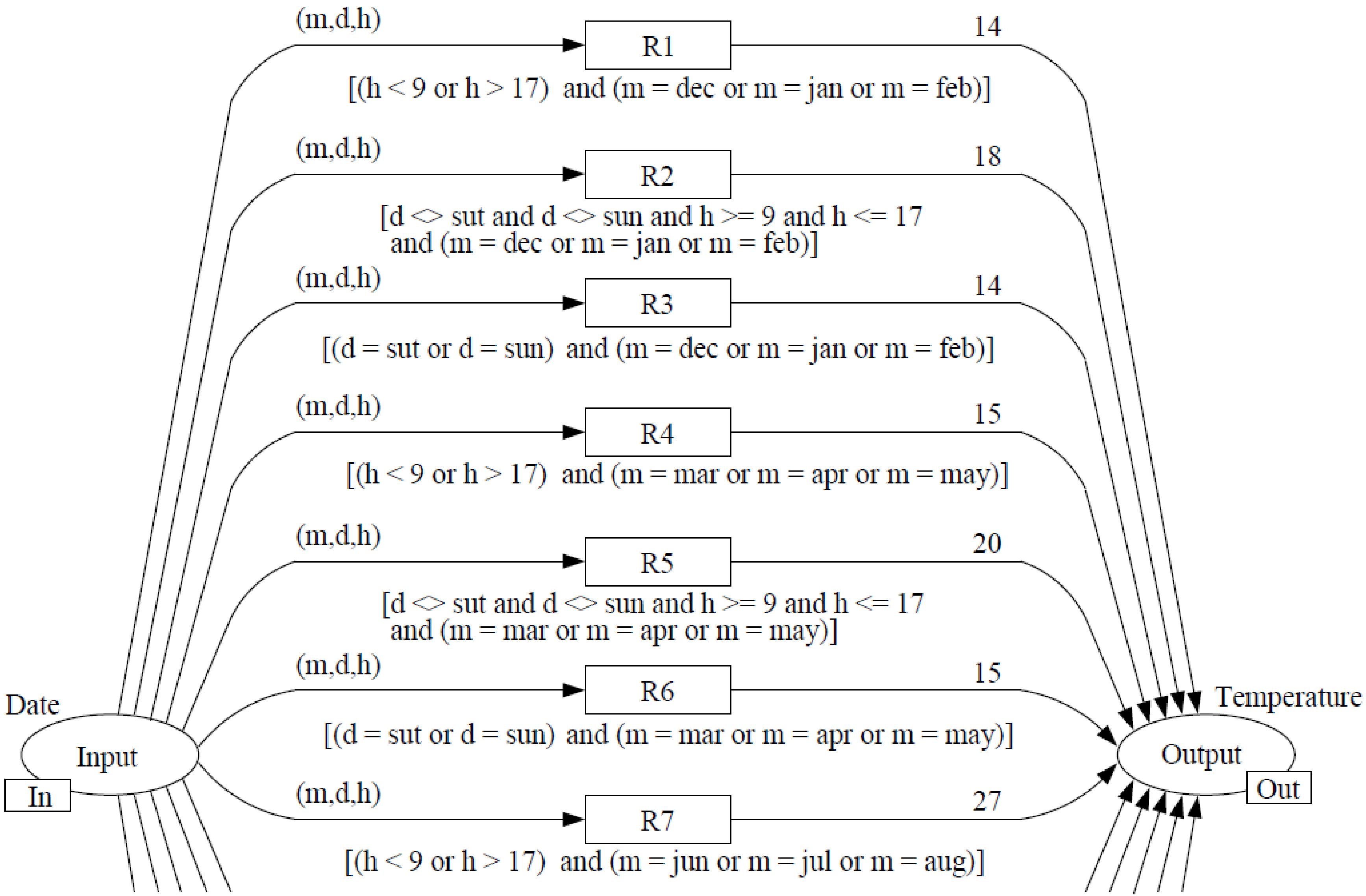
– *Consistency* (*determinism*) iff no two different rules can produce different results for the same input state

– *Optimality* (*redundancy*) – iff no redundant rules exist



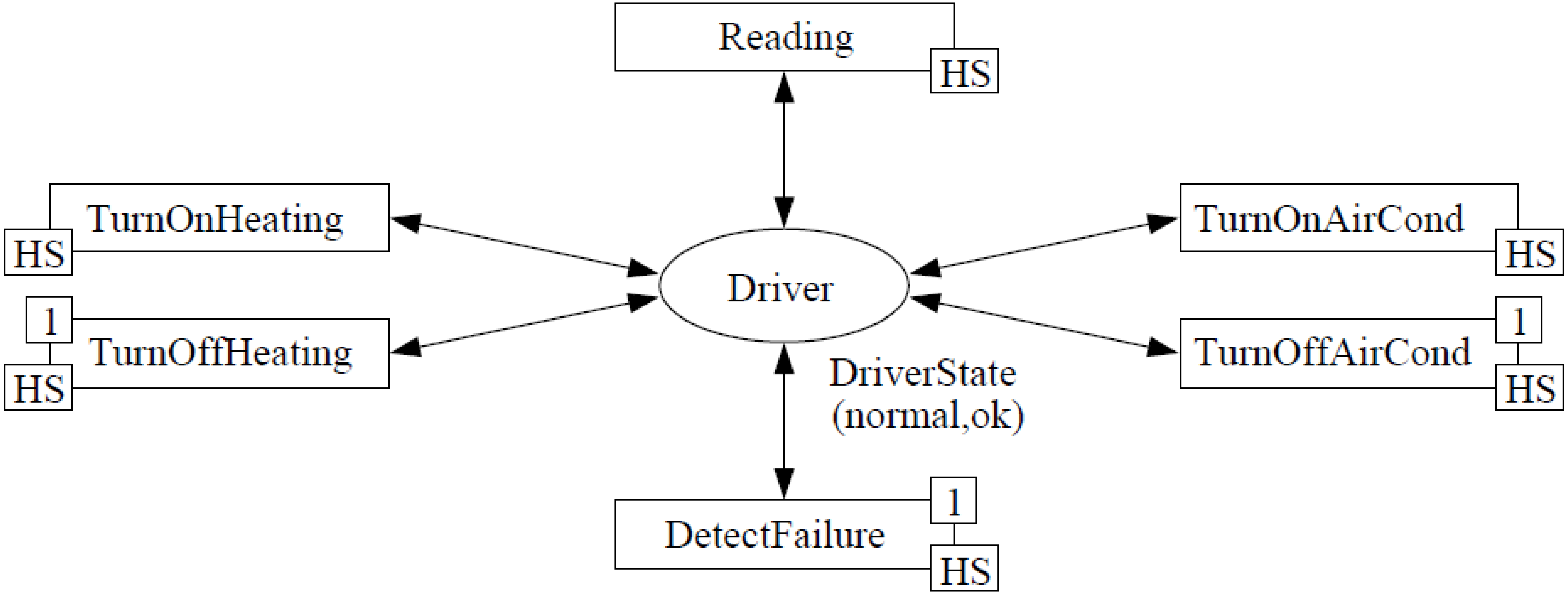
Szpyrka M., Szmuc T.: *Integrated Approach to Modelling and Analysis Using RTCP-Nets*. In: Software engineering techniques : design for quality (ed. Krzysztof Sacha). — New York, NY, USA: Springer

Generated D-Net



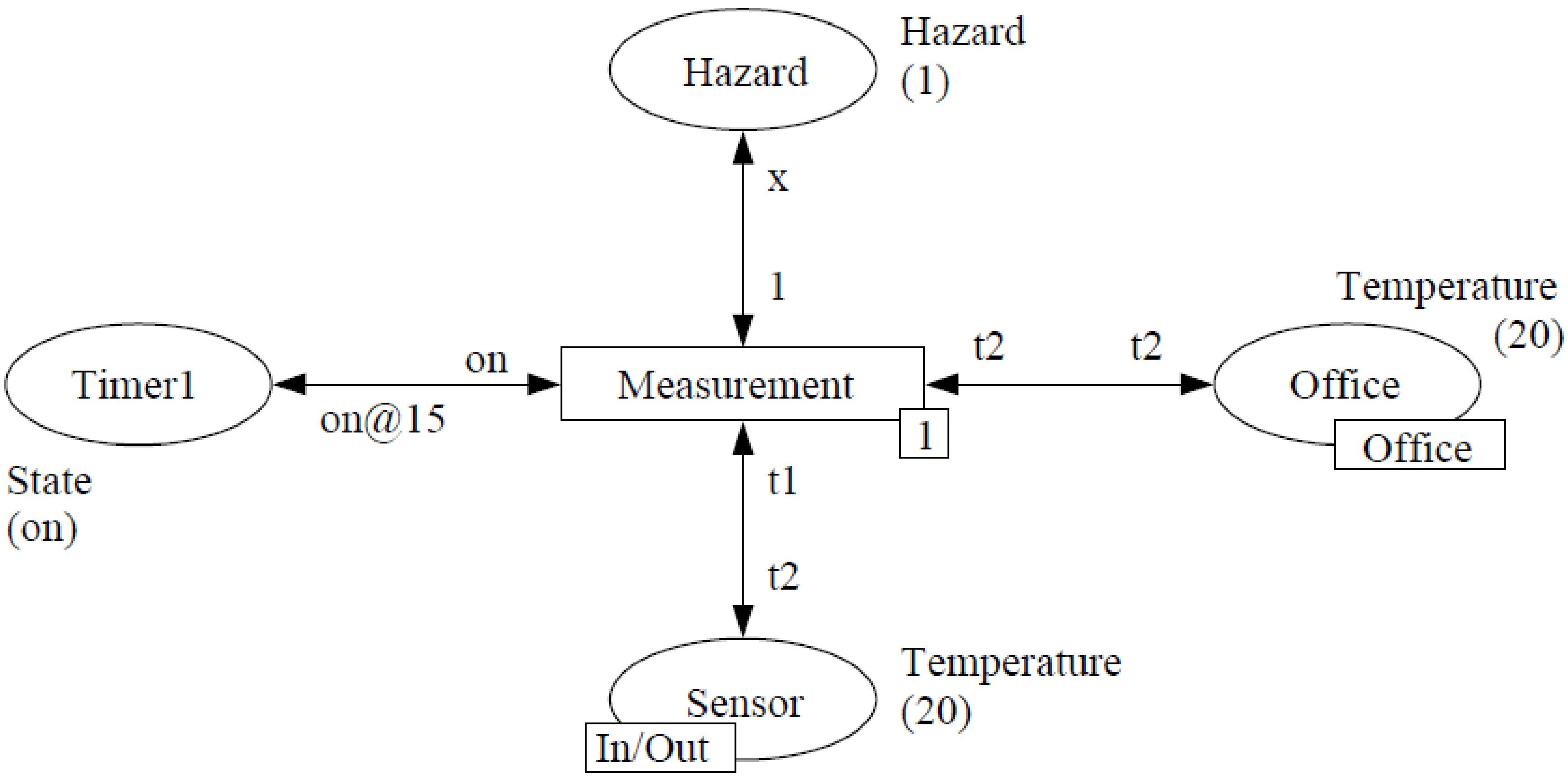
Szpyrka M., Szmuc T.: *Integrated Approach to Modelling and Analysis Using RTCP-Nets*. In: Software engineering techniques : design for quality (ed. Krzysztof Sacha). — New York, NY, USA: Springer

Driver page



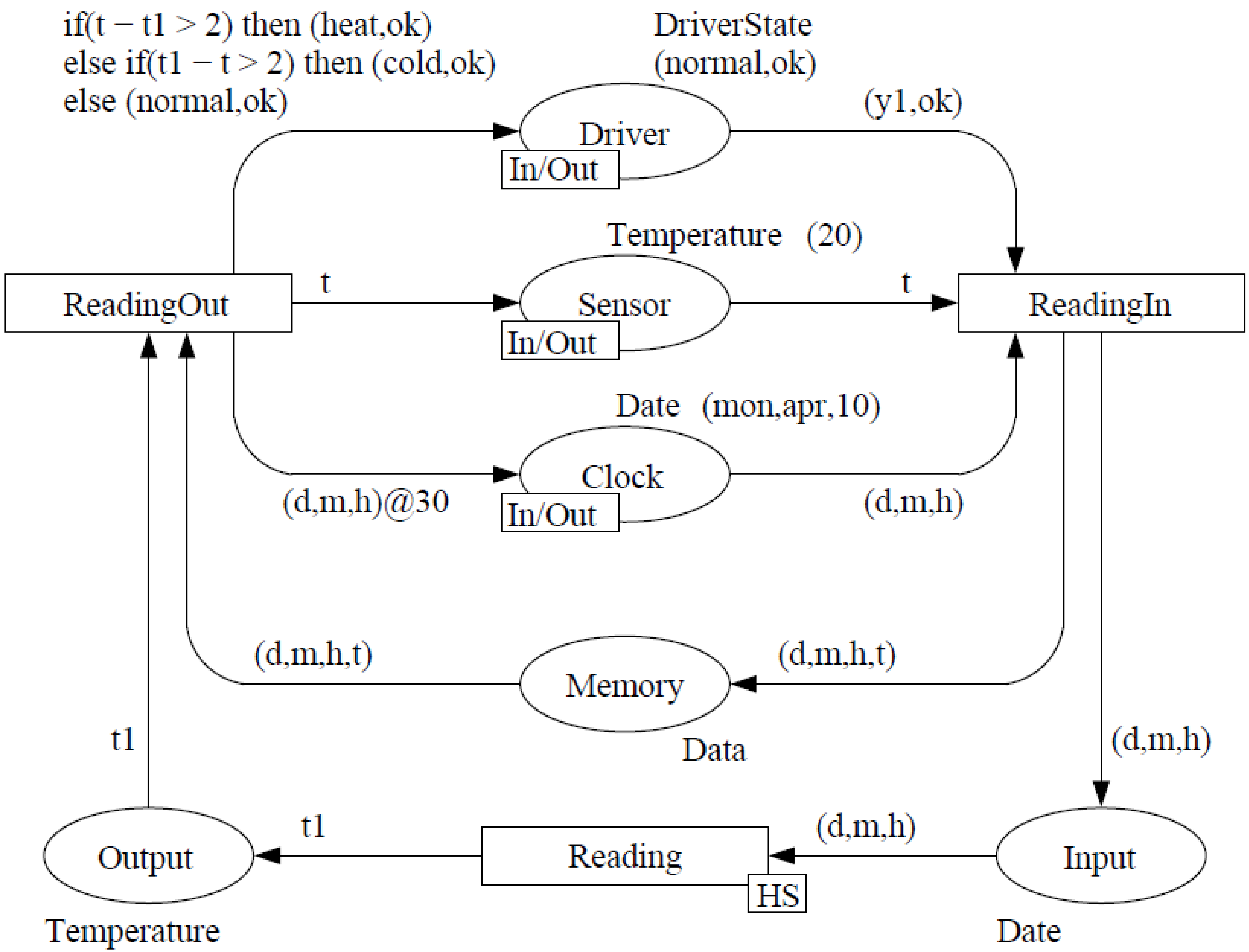
Szpyrka M., Szmuc T.: *Integrated Approach to Modelling and Analysis Using RTCP-Nets*. In: Software engineering techniques : design for quality (ed. Krzysztof Sacha). — New York, NY, USA: Springer

Measurement page



Szpyrka M., Szmuc T.: *Integrated Approach to Modelling and Analysis Using RTCP-Nets*. In: Software engineering techniques : design for quality (ed. Krzysztof Sacha). — New York, NY, USA: Springer

Reading – linking page



Szpyrka M., Szmuc T.: *Integrated Approach to Modelling and Analysis Using RTCP-Nets*. In: Software engineering techniques : design for quality (ed. Krzysztof Sacha). — New York, NY, USA: Springer, 2006

AGH

Automatic translation of software models (UML, **SysML**, AADL) into formal description language

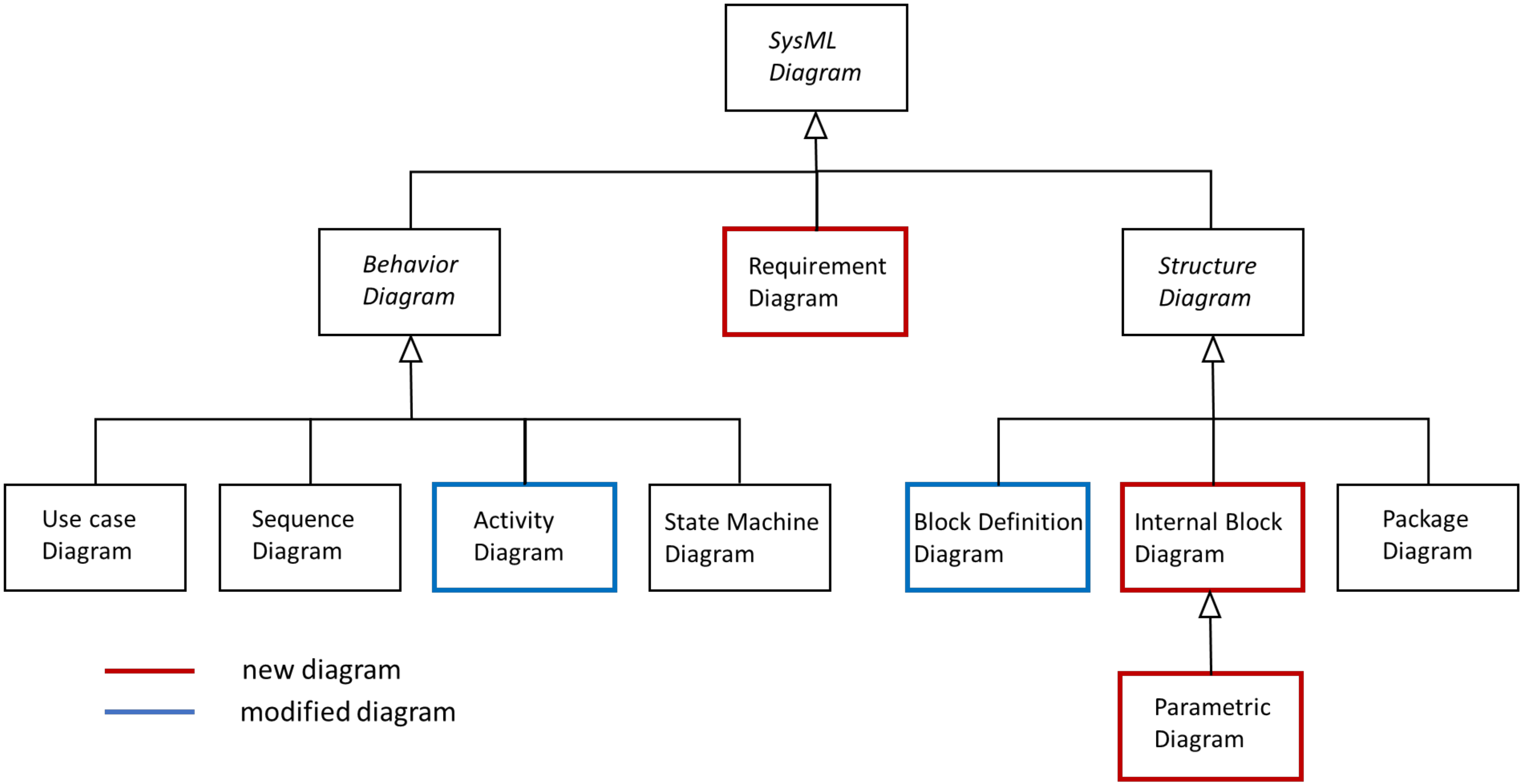
SysML → CPN

https://www.eclipse.org/papyrus/ http://cpntools.org/

SysML features

1. Simpler than UML (less diagrams)
2. Integrates hardware and software description
3. Possibility to integrate other models e.g. output from ControlShell

SysML overview 1/2

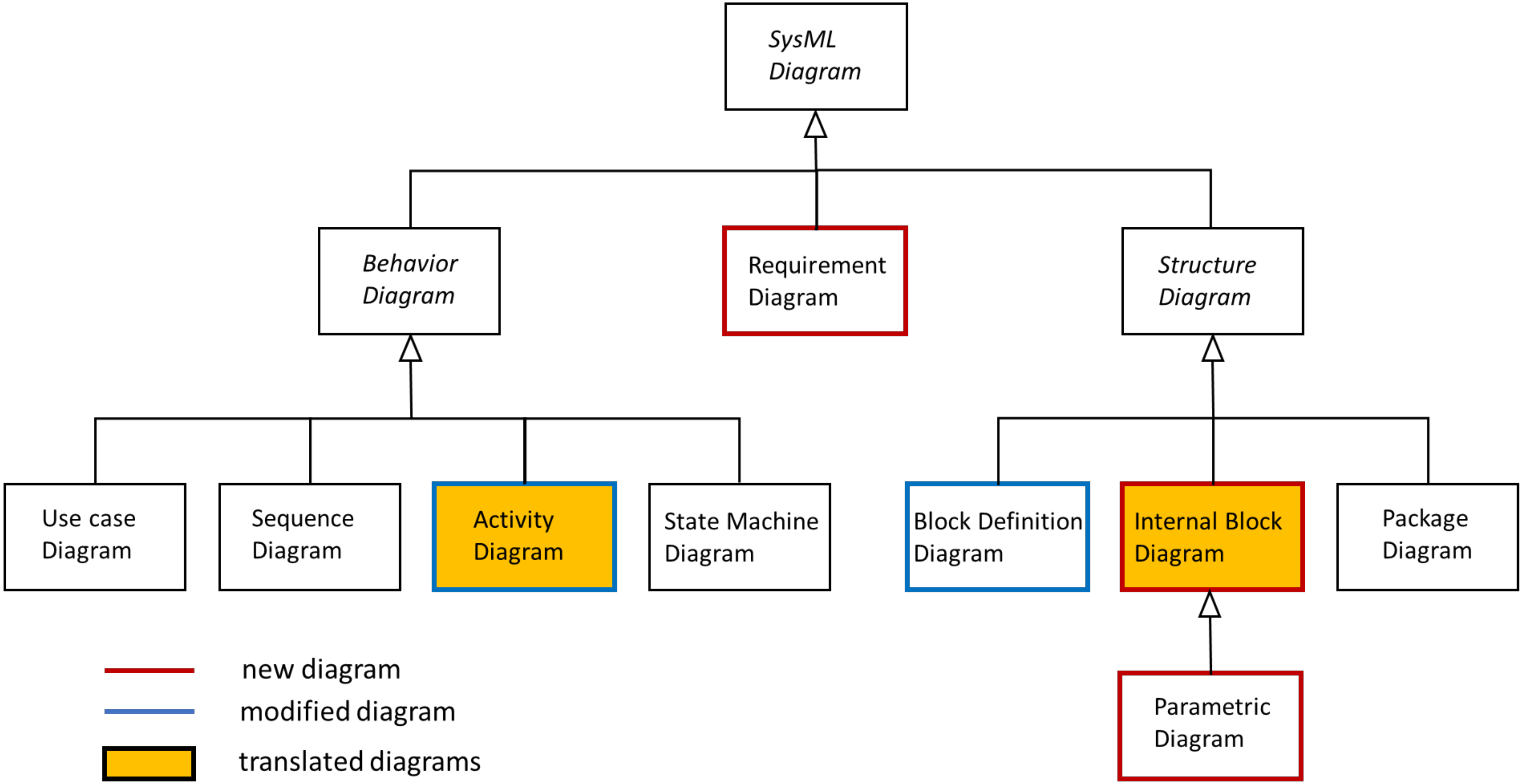


Sanford Friedenthal, Modelling with SysML – Tutorial at INCOSE 2010 Symposium, 2010

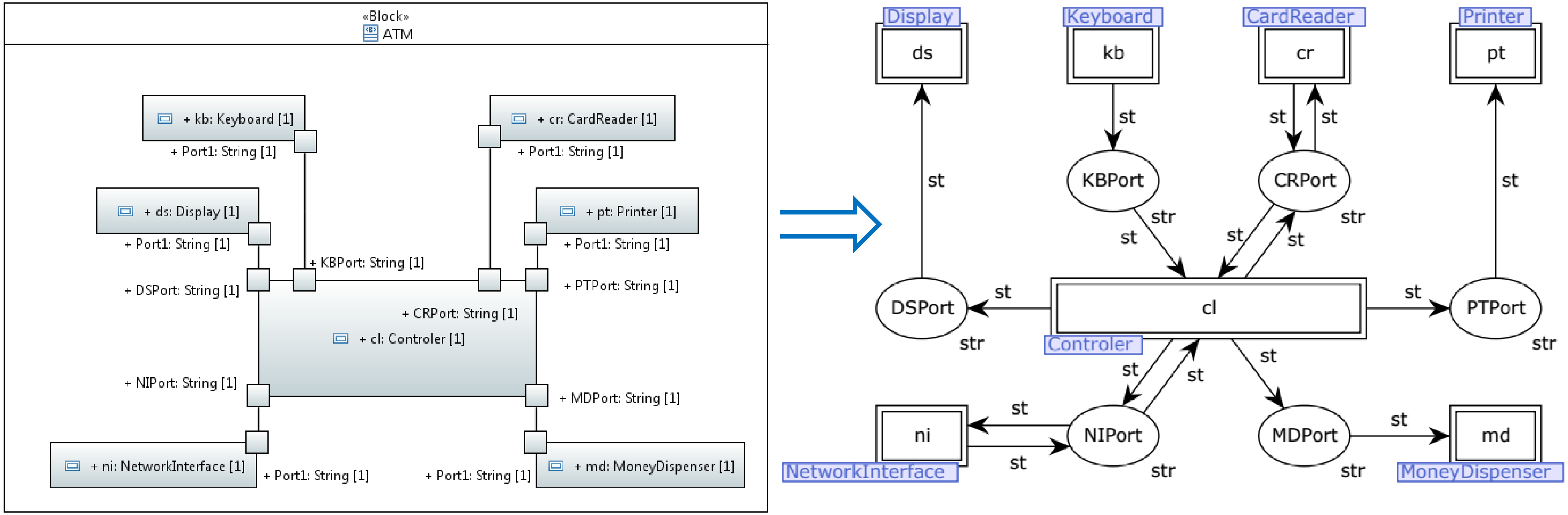
SysML overview. Additional diagrams 2/2

- **requirements diagrams** (RD) – **relationships between requirements and/or related use cases, blocks**, etc. They are used for structuring textual requirements using several dependency relations: containment, trace, derive requirement, refine, satisfy, and verify.
- **block definition diagrams** (BDD) – used to specify blocks, actors, value type, constraint blocks, flow specifications, and interfaces **form types for other elements appearing in other SysML diagrams**.
- **internal block diagrams** (IBD) – **internal structure of the related blocks**. Any IBD describes in which way parts of a block must be connected to create an instance of the block.
- **parametric diagrams** (PR) - specify **relationships between *blocks* and *constraint blocks***. Constraint blocks are used to close inside frame constraints, i.e. bindings between parameters expressed by equations and mathematical relationships.

Translation of selected diagrams



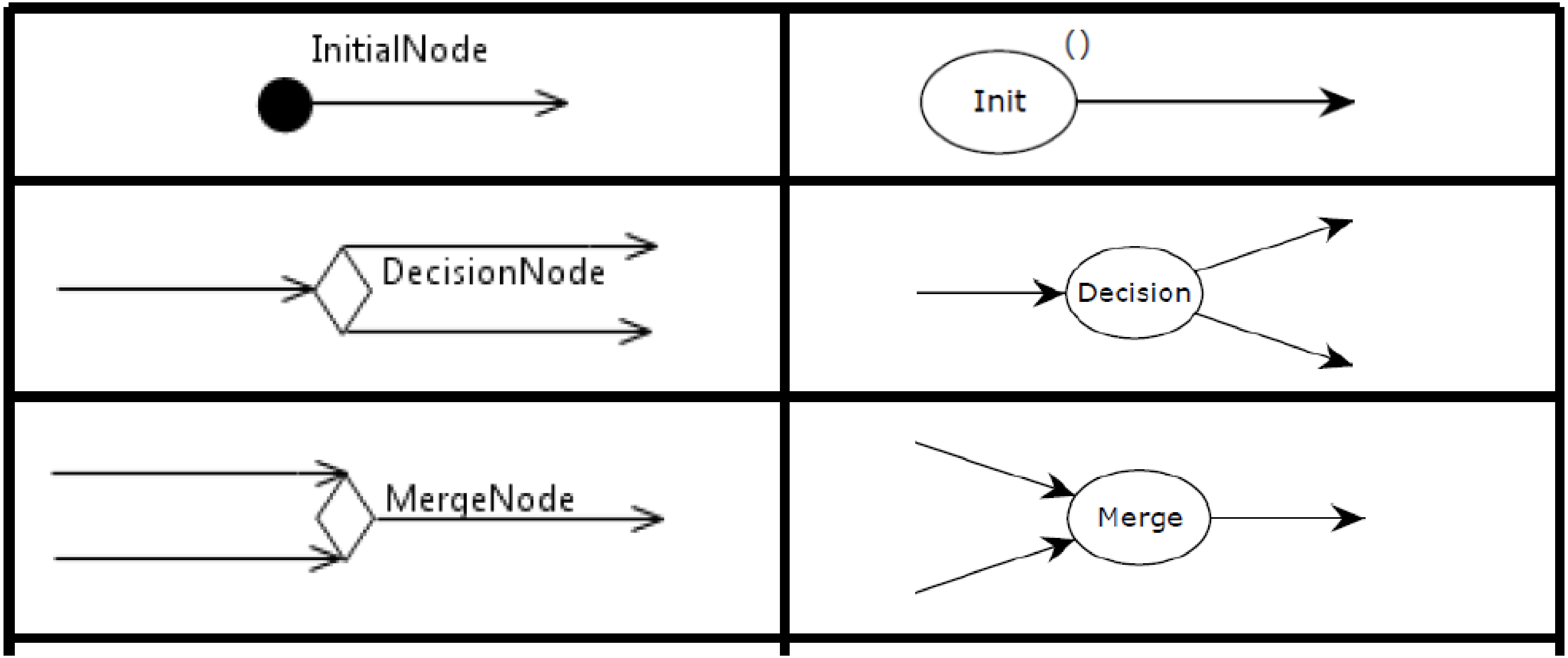
IBD → CPN



Structure of ATM

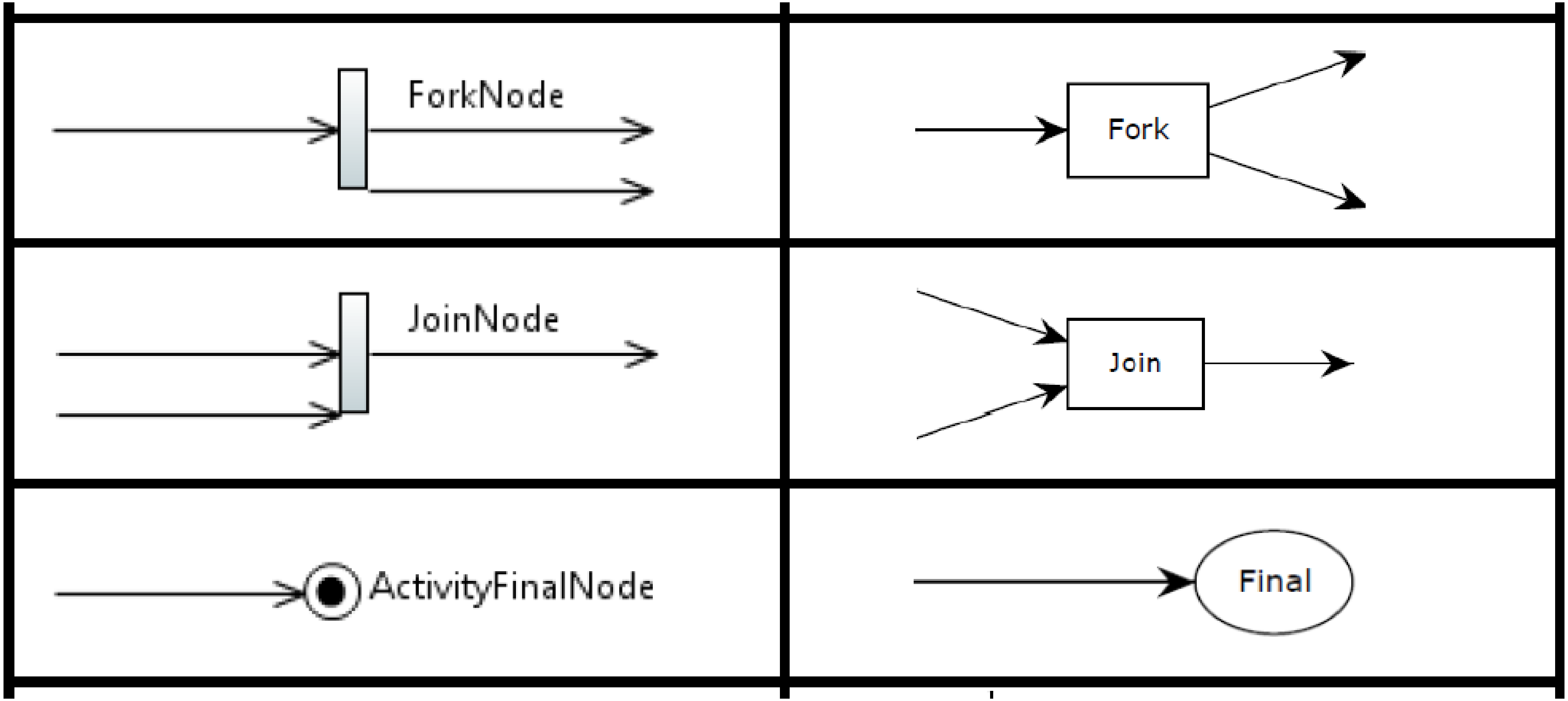
W. Szmuc, and T.Szmuc: *Towards Embedded Systems Formal Verification. Translation for SysML into Petri Nets*. In Proceedings of teh Internationa;l Conference Mixded Design of Integrated Cuircuits and Systems, 2018, pp. 420-423

Activity diagrams → CPN. Mapping of symbols 1/3



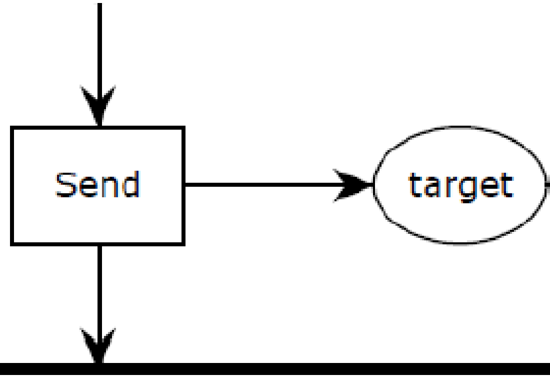
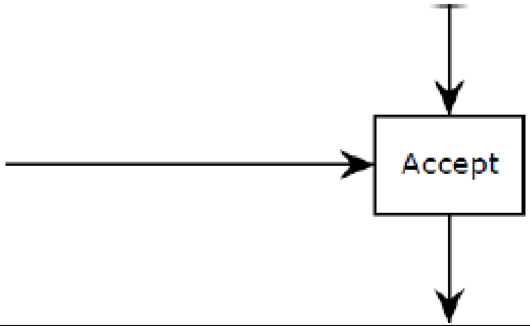
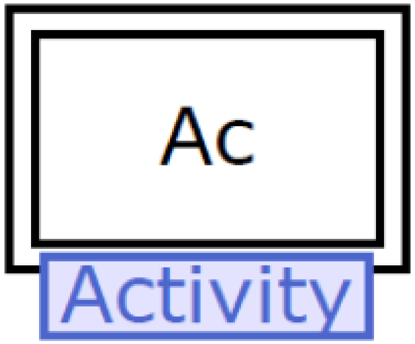
W. Szmuc, and T.Szmuc: *Towards Embedded Systems Formal Verification. Translation for SysML into Petri Nets*. In Proceedings of teh Internationa;l Conference Mixded Design of Integrated Cuircuits and Systems, 2018, pp. 420-423

Activity diagrams CPN. Mapping of symbols 2/3

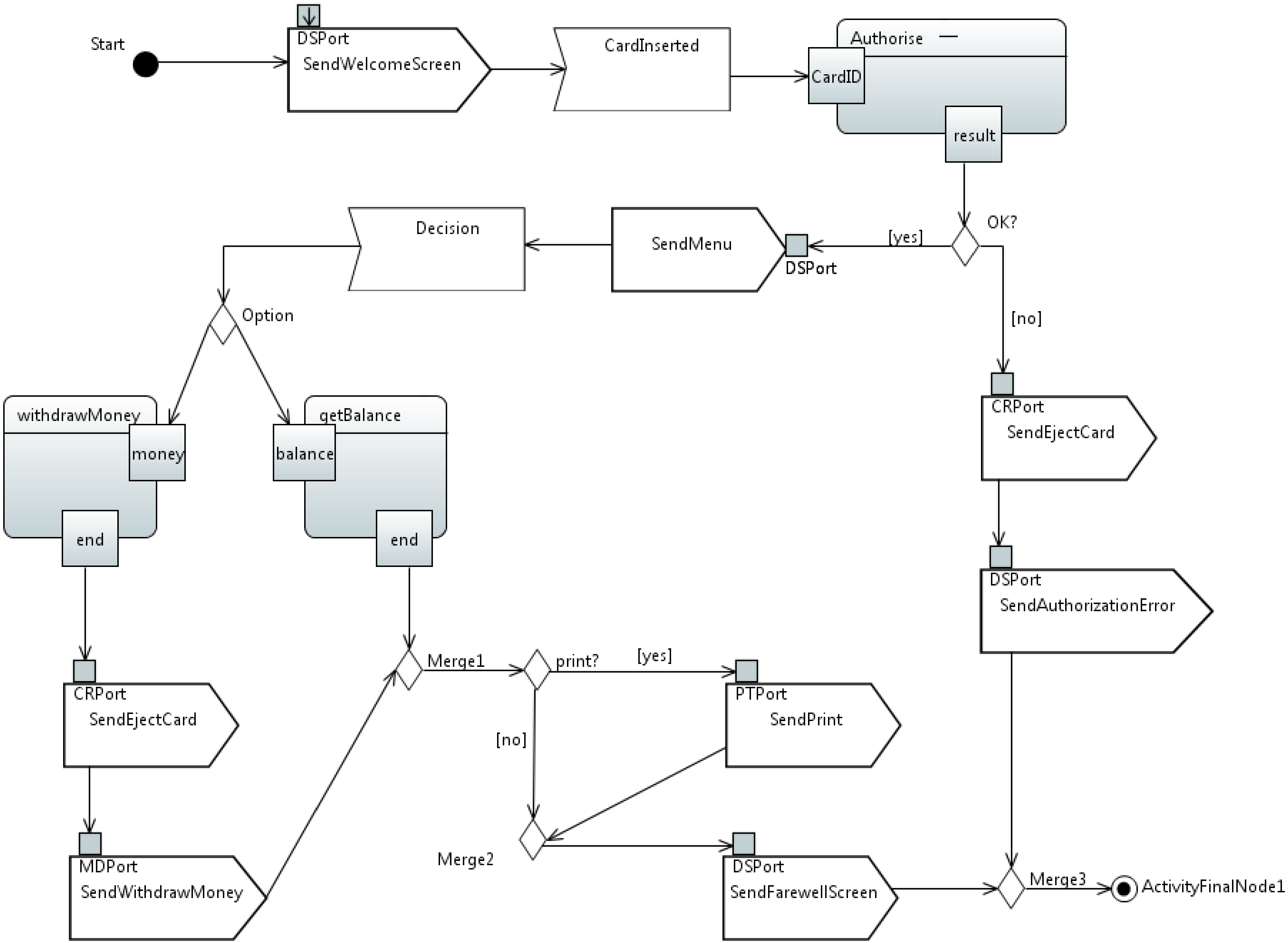


W. Szmuc, and T.Szmuc: *Towards Embedded Systems Formal Verification. Translation for SysML into Petri Nets*. In Proceedings of teh Internationa;l Conference Mixded Design of Integrated Cuircuits and Systems, 2018, pp. 420-423

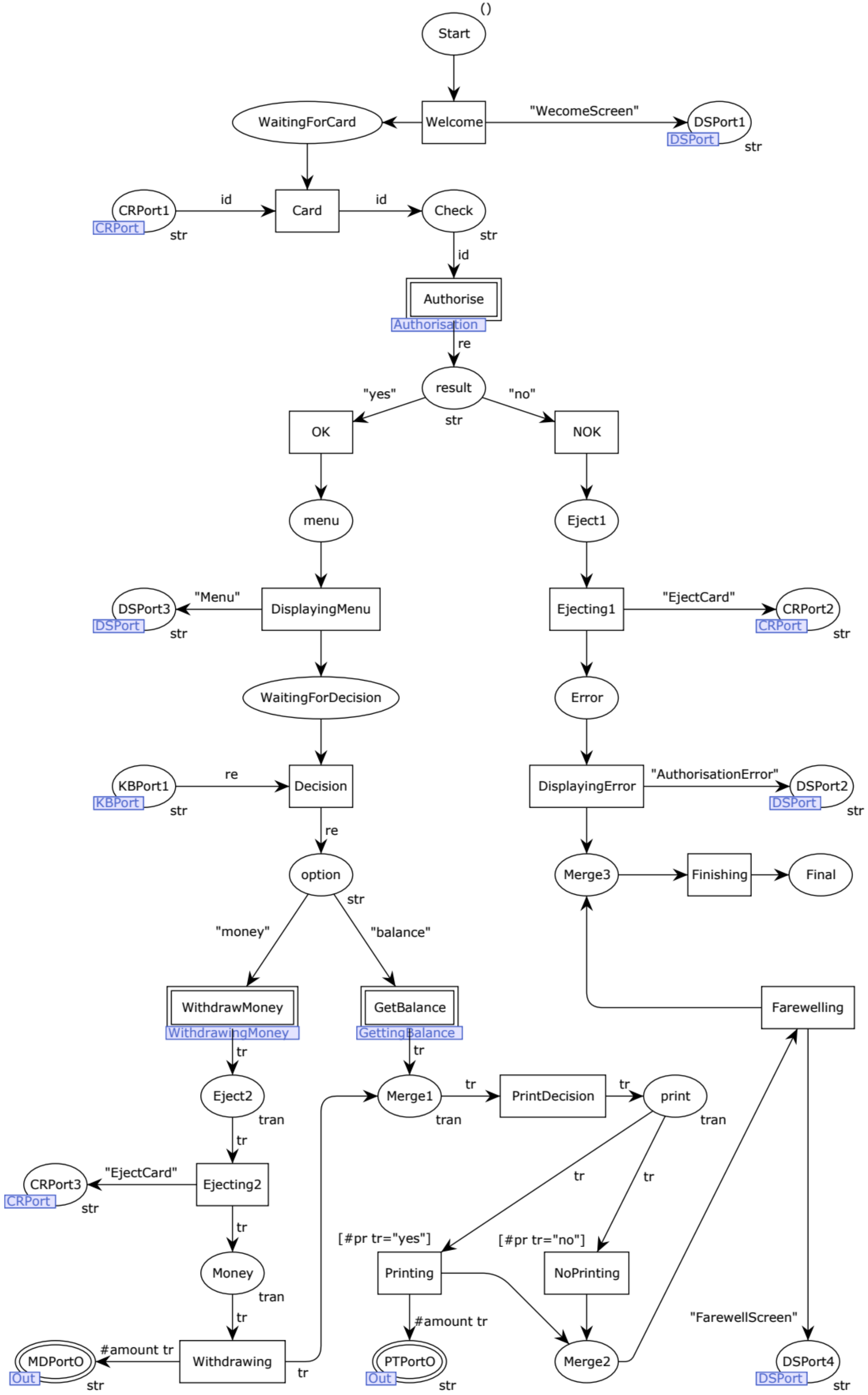
Activity diagrams → CPN. Mapping of symbols 3/3

Activity Diagram describing behaviour of ATM



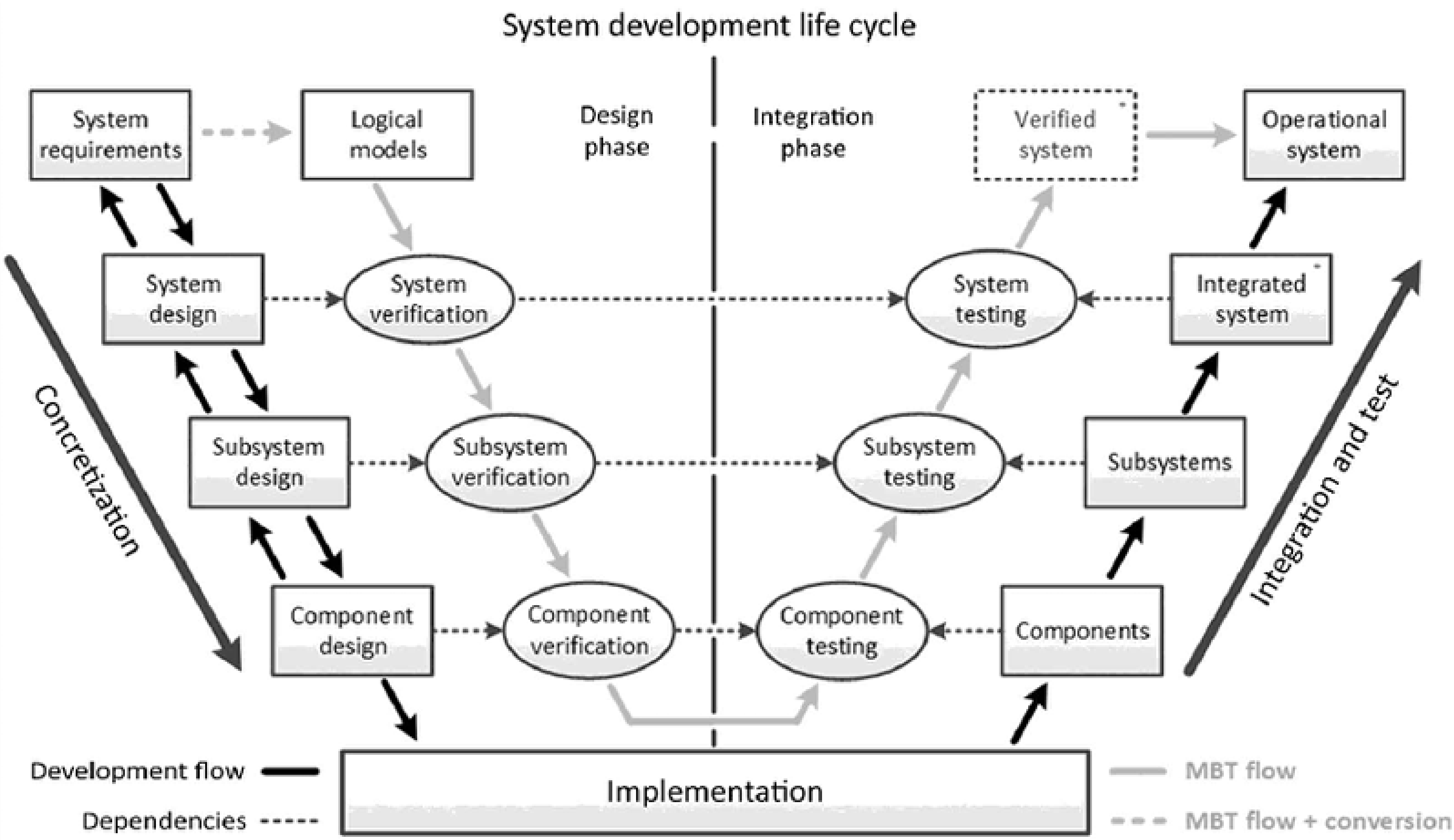
The translated CPN model



Implementation

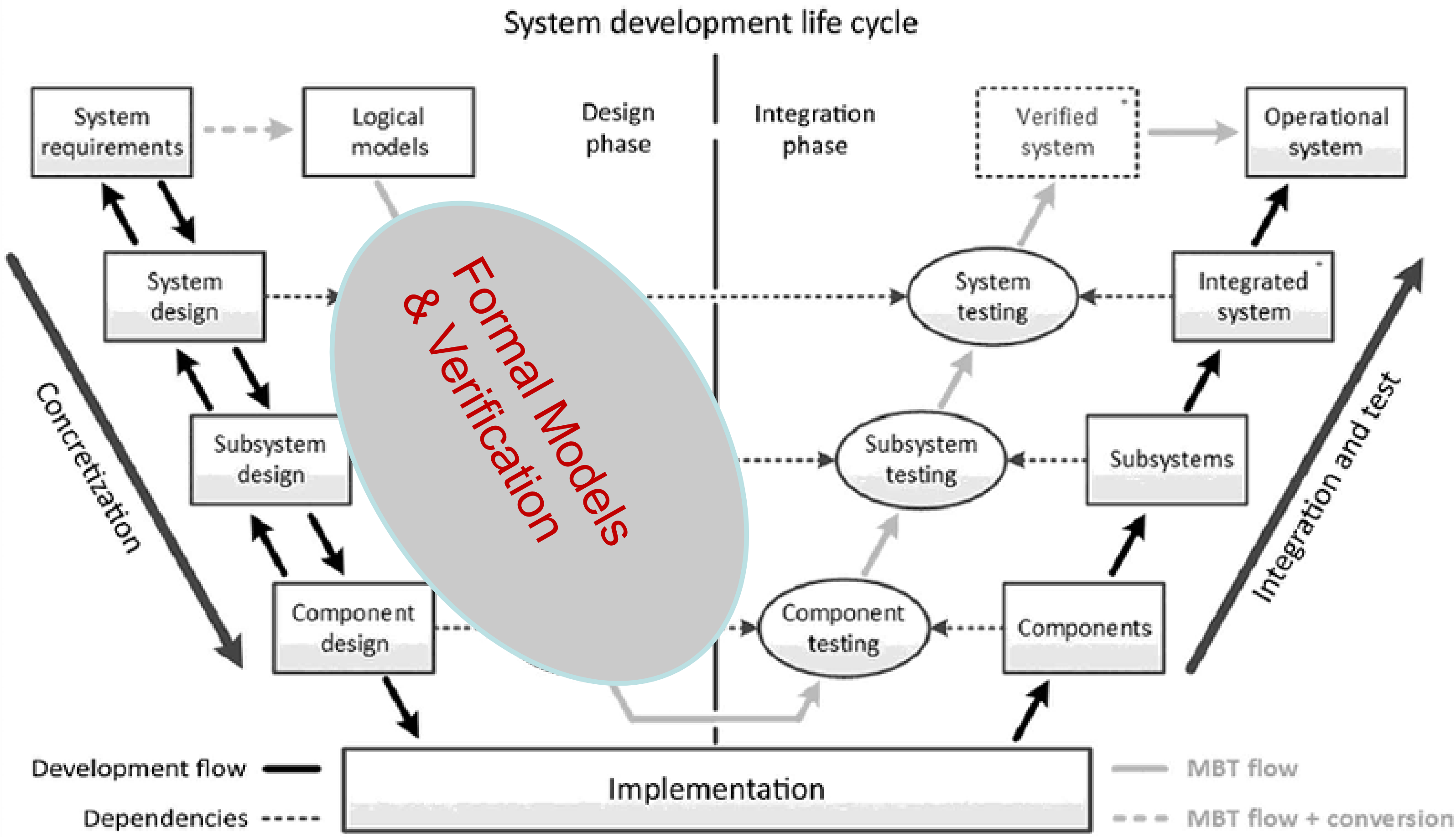
1. Papyrus (https://www.isis-papyrus.com/software) tool are used for modelling of SysML artefacts.
2. Papyrus output – XML specification of SysML model is converted into XML model of Coloured Petri Net being an input to CPN Tools (http://cpntools.org/).
3. Coloured Petri Nets are modelled and analysed using CPN Tools.
4. Beta version of the prototype

V-Development life-cycle



* Houbing Song, Danda B.Rawat, Sabina Jeschke, and Christian Brecher: Cyber-Physical Systems. Foundations, Principles, and Applications, Elsevier, Intelligent Data Centric Systems, 2017

Place in software development



Development of environment supporting software design building formal models

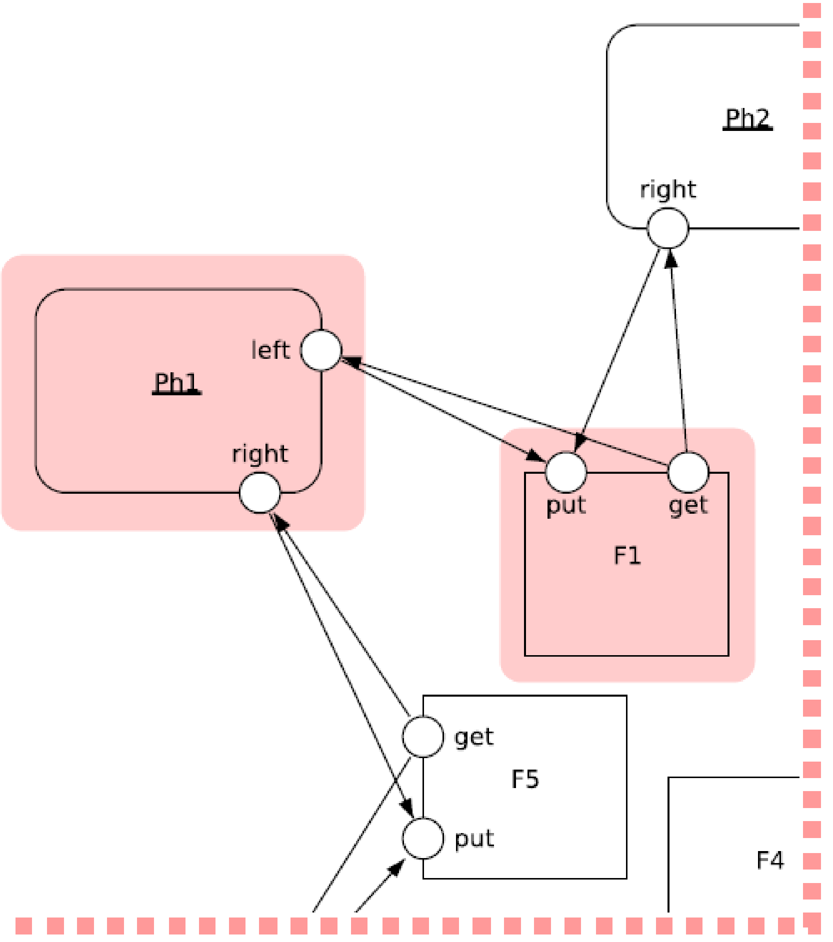
Alvis = **AL**gebra + **VIS**ualisation

Aim of the project

- easy to use for engineers formal modelling language
- visual language for models structure design
- high level programming language for models behaviour definition
- models verification with model checking techniques
- suitable for modelling concurrent, real-time, embedded and distributed systems

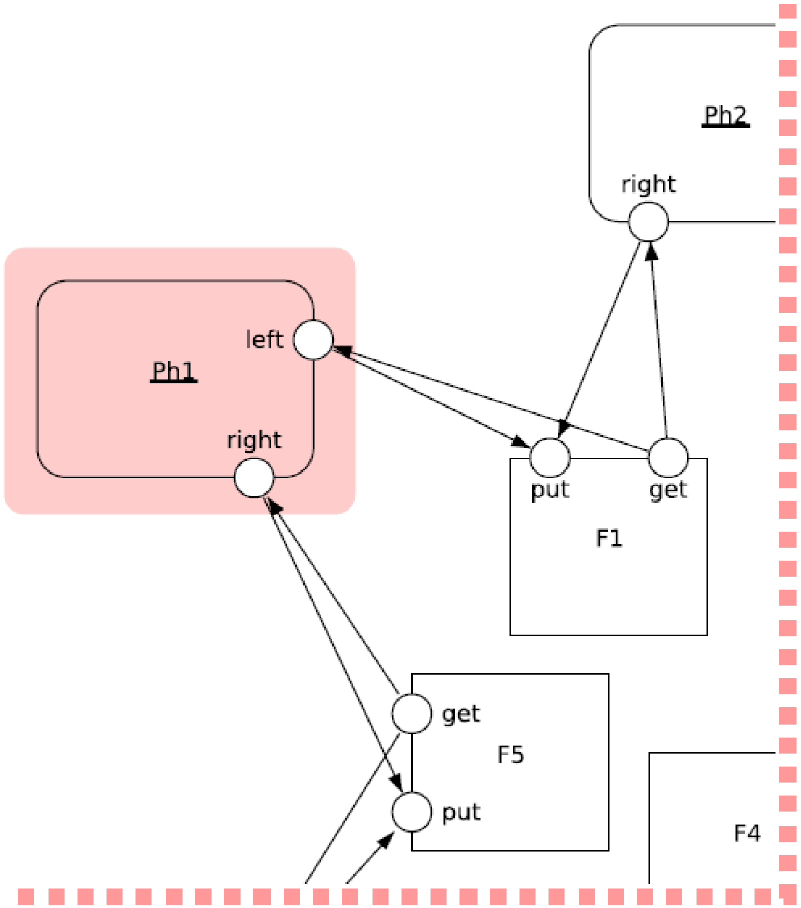
Key concepts

Agent – a distinguished part of the system under consideration with defined identity persisting in time;



Key concepts

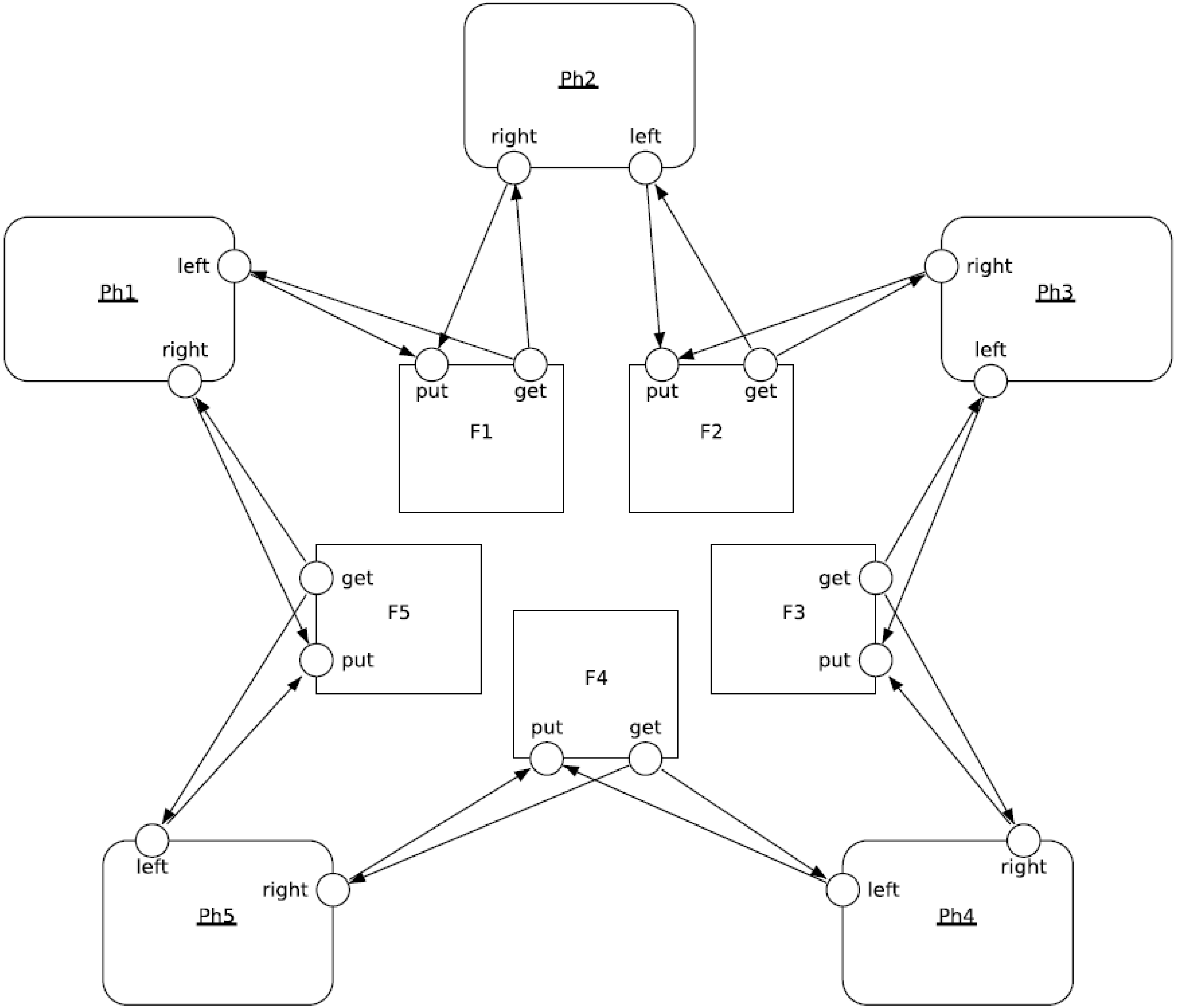
Active agent – performs some activities, similar to a task in Ada language;



Code statements

```
if (g) {...} else {...}
if (g1) {...} elseif (g2) {...} ... else {...}
loop (g) {...}
loop {...}
jump label;
null;
[exec] x = expression;
in p;
in p x;
out p;
out p x;
select { alt (g1) {...} alt (g2) {...} ...}
proc (g) p {...}
start A;
exit;
```

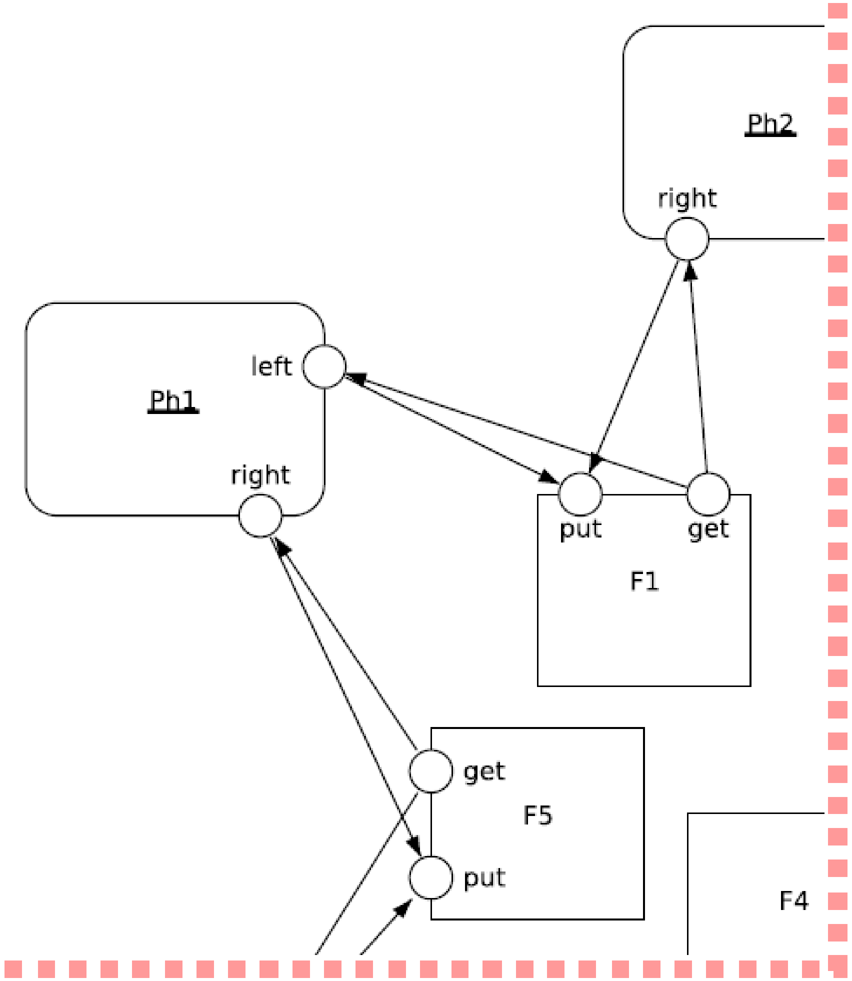
Communication diagram



Code layer

```
agent Ph1, Ph2, Ph3, Ph4, Ph5 {
  loop {
    in right;
    in left;
    out right;
    out left;
  }
}

agent F1, F2, F3, F4, F5 {
  taken :: Bool = False;
  proc (taken == False) get {
    taken = True;
    out get;
  }
  proc (taken == True)  put {
    taken = False;
    in put;
  }
}
```



Haskell filtering functions

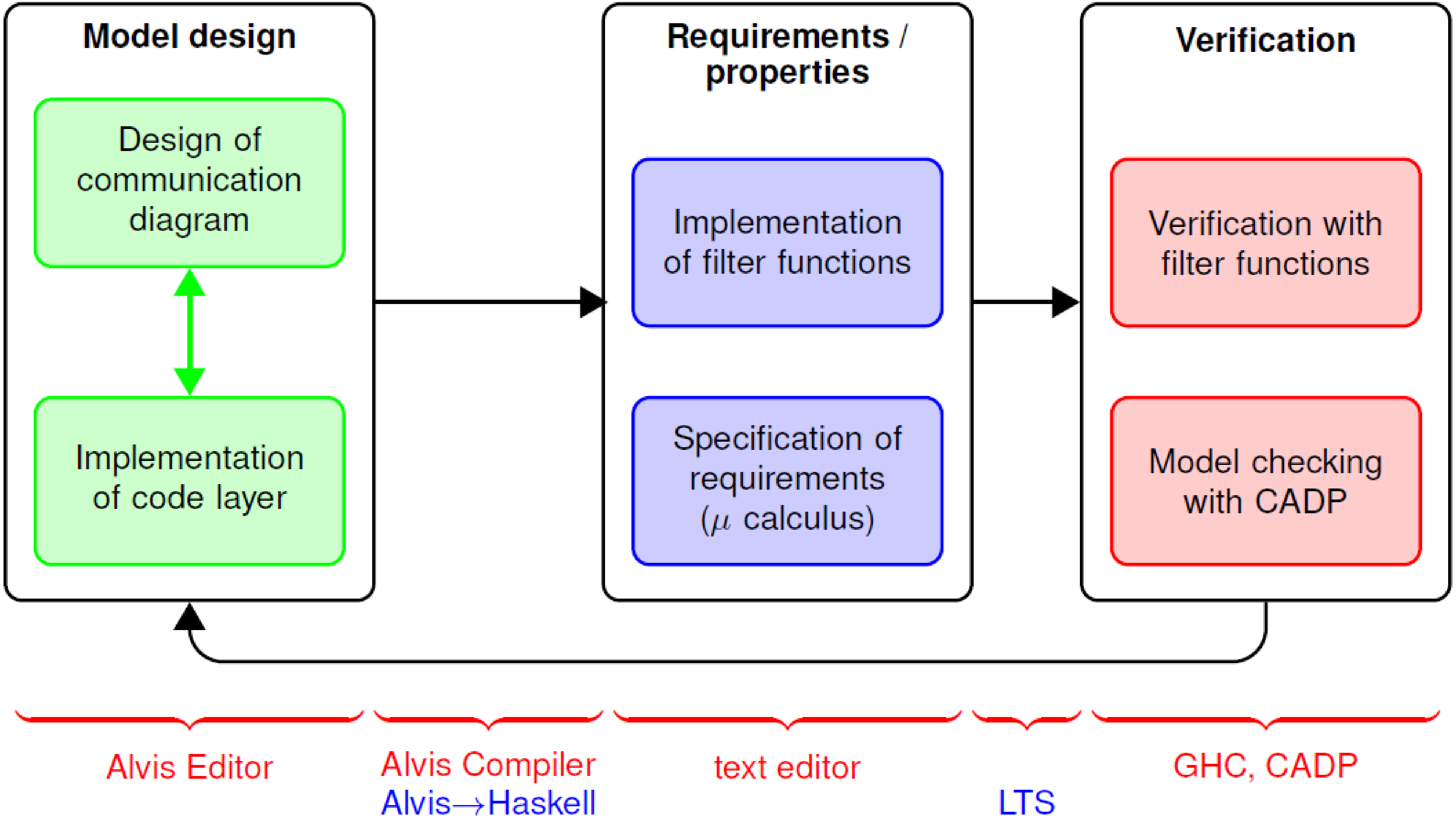
- Filtering functions search for parts of a given LTS graph stored as a Haskell list.
- Filtering functions can be both state and action oriented. CADP approach is action oriented.

Universal filtering functions

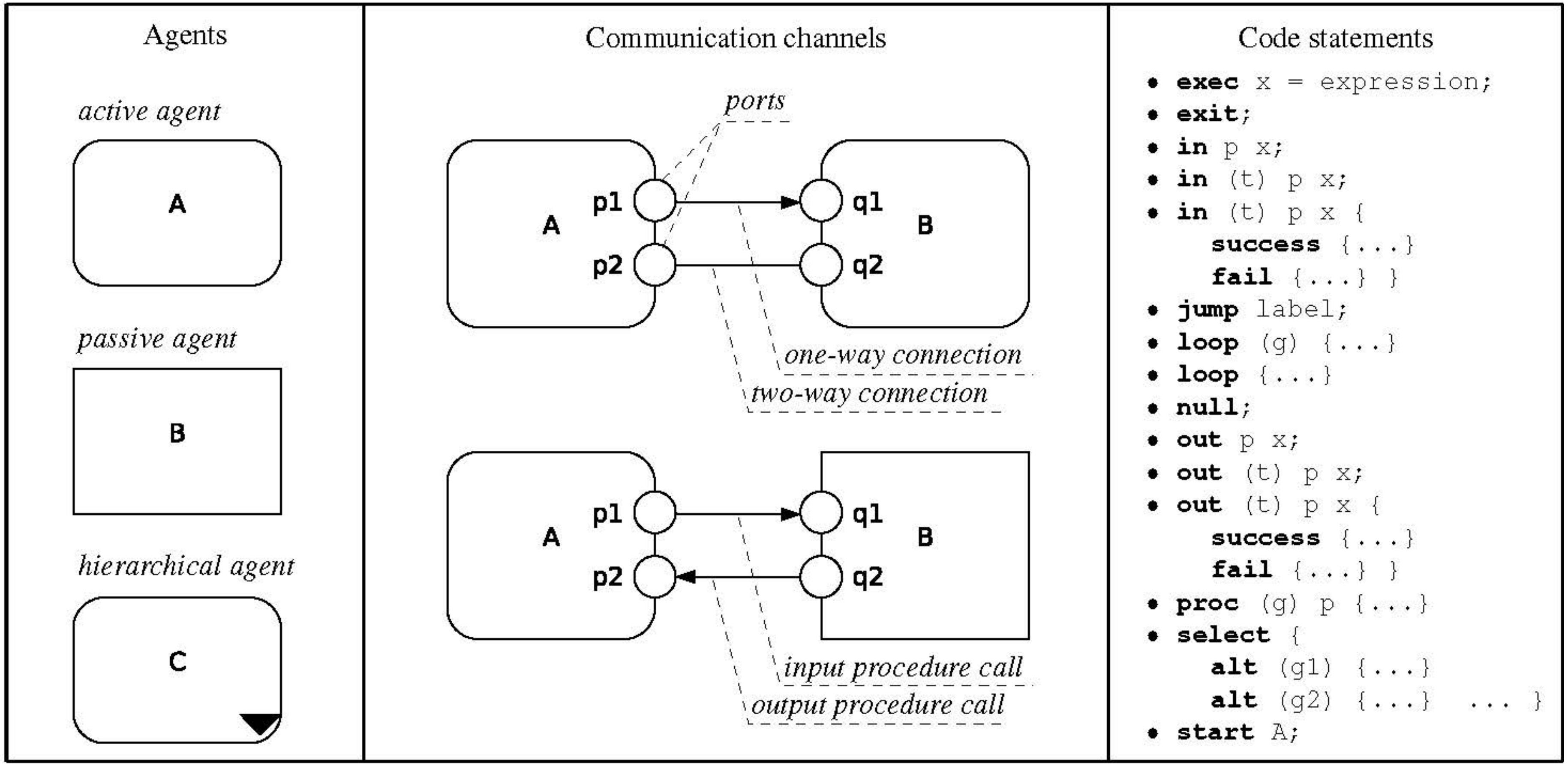
```
deadState :: Node -> Bool
deadState (n,s,ls) = ls == []
-- filter deadState lts

singleOutState :: Node -> Bool
singleOutState (n,s,ls) = (length ls) == 1
-- filter singleOutState lts
```

Development process

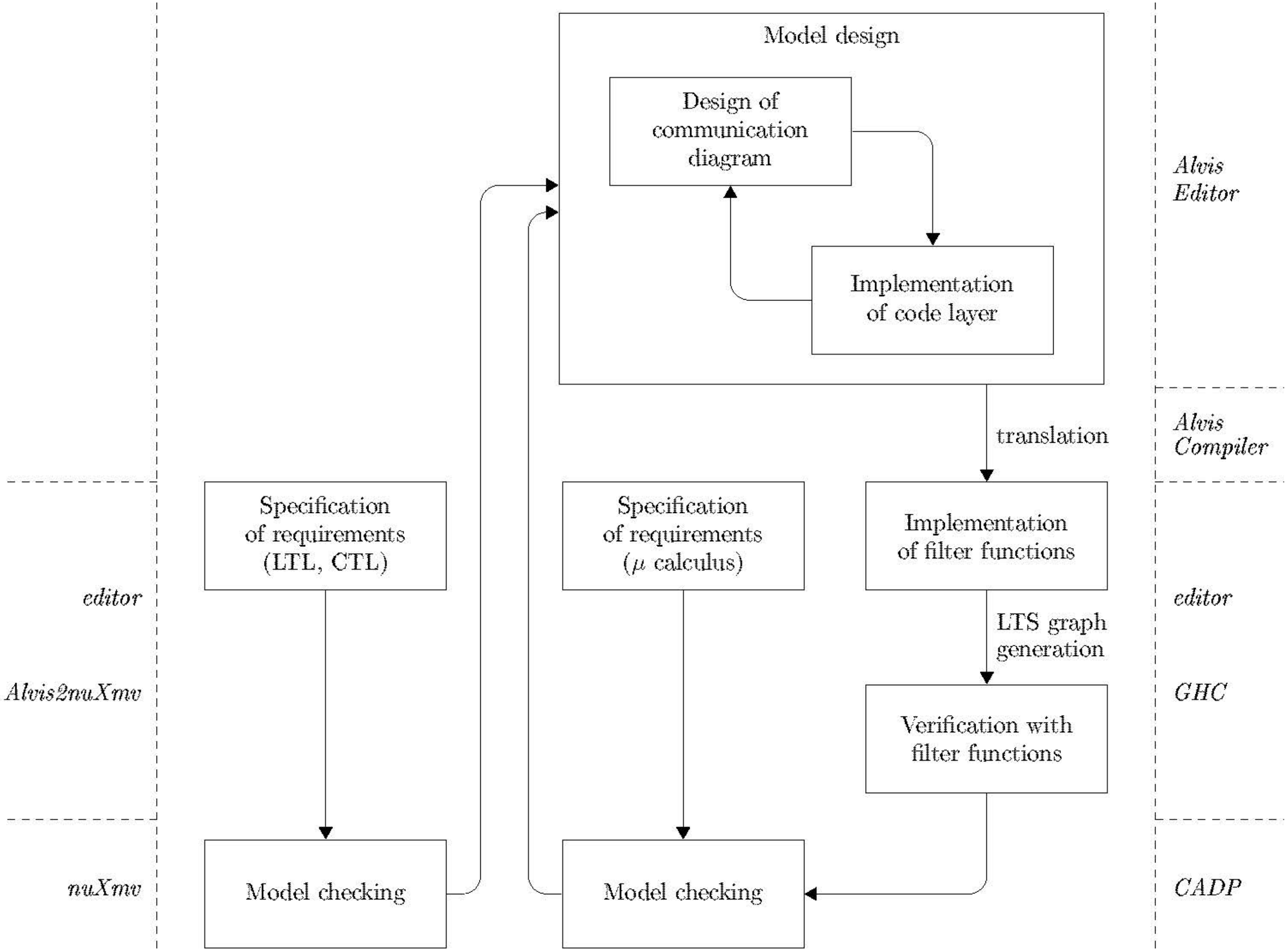


ALVIS – editing layers

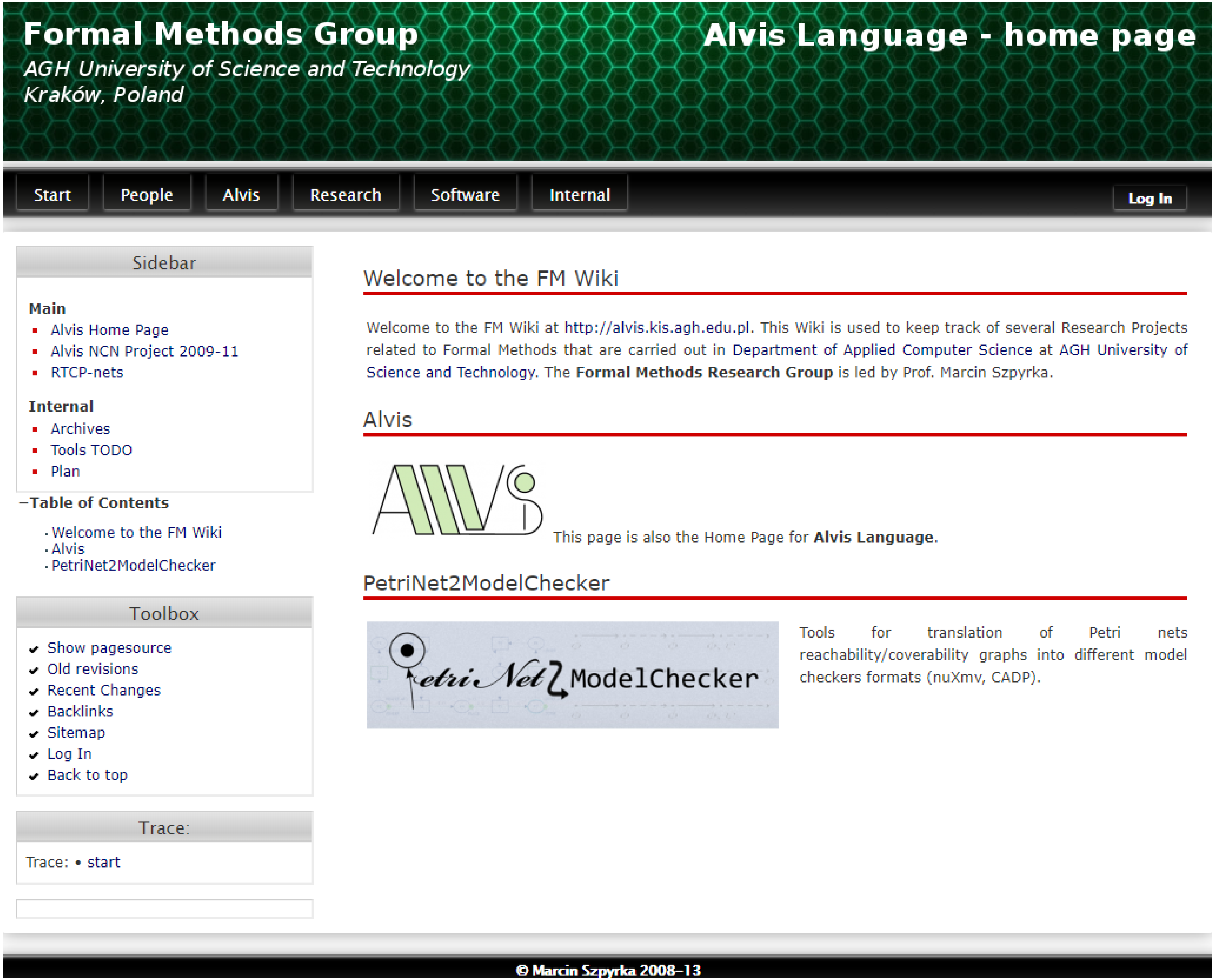


M. Szpyrka, P. Matyasik, M.Wypych, J. Biernacki, and Ł. Podolski: *Alvis modeling language. Manual*, 2017, http://alvis.kis.agh.edu.pl/wiki/start

Alvis process



M. Szpyrka, P. Matyasik, M.Wypych, J. Biernacki, and Ł. Podolski: *Alvis modeling language. Manual*, 2017, http://alvis.kis.agh.edu.pl/wiki/start



http://alvis.kis.agh.edu.pl/wiki/start

Conclusions

1. Formal methods may improve software development esp. from integration & consistency point of view
2. Advanced methods of state space reductions extend applicability to industrial systems
3. Automation of the translations and integrated development systems may encourage developers for using formal methods
4. Rigorous use of formal methods may reduce testing costs

Thank you for your attantion!

Tomasz Szmuc
AGH University of Science and Technology
Department of Applied Computer Science
tsz@agh.edu.pl