PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 9: Técnicas de modelagem de sistemas com RdP

Prof. José Reinaldo Silva
reinaldo@usp.br

Plano de Aulas

Aula	Tema	Data
Aula7	Propriedades das redes P/T e Coloridas	9/11
Aula8	Análise de Invariantes	16/11
Aula9	Técnicas de modelagem	23/11
Aula10	Métodos de Design orientados a estados	30/11
Aula11	Métodos de Design orientados a eventos	07/12
Aula12	Perspectivas de pesquisa em modelagem de sistemas discretos com RdP	14/12

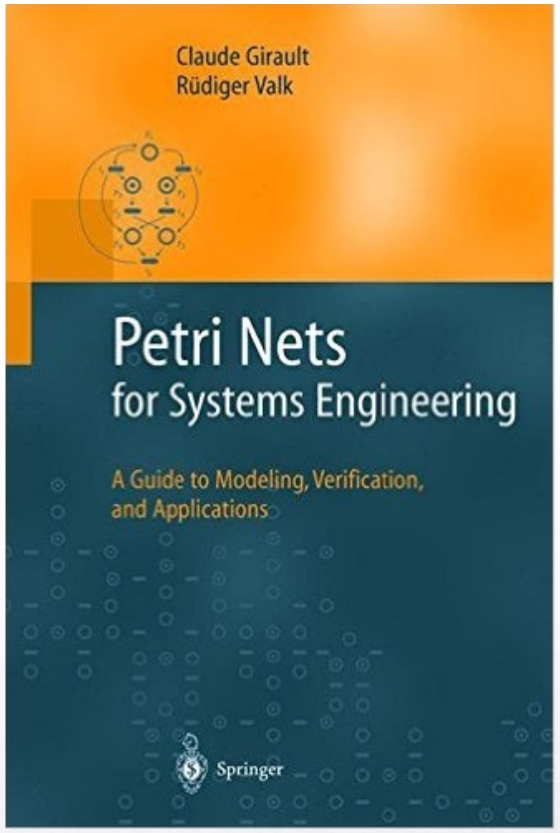
Part II. Modelling

7. Introduction 81

8. Modelling and Analysis Techniques by Example 85
- 8.1 Nets, Refinement, and Abstraction 85
- 8.2 Place/Transition Nets and Resource Management 92
- 8.3 Coloured Nets, Abstraction, and Unfolding 97

9. Techniques 105
- 9.1 Building Blocks 105
- 9.2 Combining Nets 108
  - 9.2.1 Place Fusion 108
  - 9.2.2 Arc Addition 110
  - 9.2.3 Transition Fusion 110
- 9.3 High-Level Nets 112
  - 9.3.1 Coloured Nets 112
  - 9.3.2 Fairness, Priority, and Time 115
- 9.4 Decomposing Nets 116
- 9.5 Conclusion 116

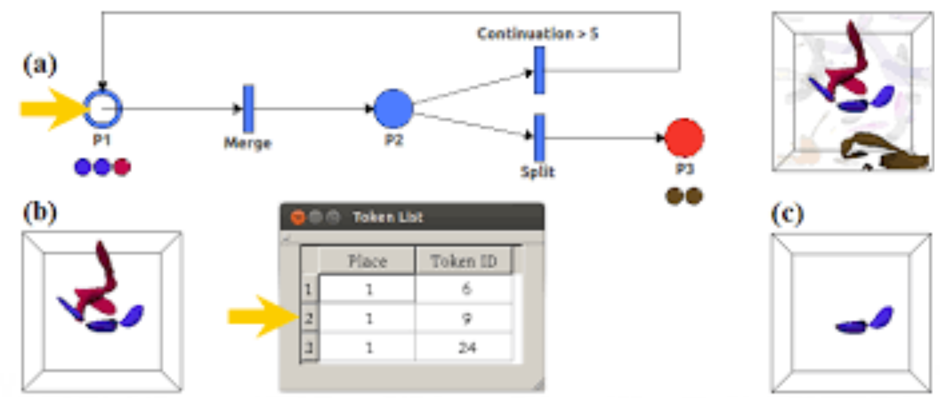
10. Methods 119
- 10.1 State-Oriented Modelling 120
  - 10.1.1 Specification 123
  - 10.1.2 Design 124
  - 10.1.3 Implementation 133
  - 10.1.4 Conclusion 134
- 10.2 Event-Oriented Modelling 135
  - 10.2.1 High-Level Modelling 135
  - 10.2.2 Protocol Modelling 137
  - 10.2.3 Verification 142
  - 10.2.4 Conclusion 145
- 10.3 Object-Oriented Modelling 146
  - 10.3.1 Objects vs Petri Nets 146
  - 10.3.2 Integration Approaches 148
  - 10.3.3 A Multi-Formalism Approach Including Nets 152
  - 10.3.4 Conclusion 157



Prometemos tratar nesta aula do problema da análise de invariantes nas redes coloridas e voltaremos à discussão sobre o uso da análise de propriedades na modelagem e análise de sistemas automatizados.

PN Basic Properties

1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not work at all.
3) *reversibility*, characterising recoverability of the initial marking from any reachable marking.
4) *mutual exclusion*, dealing with the impossibility of simultaneous *submarkings* (p-mutex) or *firing concurrency* (t-mutex).



Seja qual for a opção para análise de invariantes (em redes clássicas ou coloridas) o nosso objetivo é nos certificar de que o ambiente de modelagem segue o formalismo, e, se for o caso usá-lo para determinar os invariantes.

Proposition 9.8. *Let $SS = (N_{SS}, A_{SS})$ be the finite state space of a Coloured Petri Net, and let $SG = (N_{SG}, A_{SG})$ be the SCC graph. Then the following holds:*

1. *A marking M' is reachable from a marking $M \in \mathscr{R}(M_0)$ if and only if*

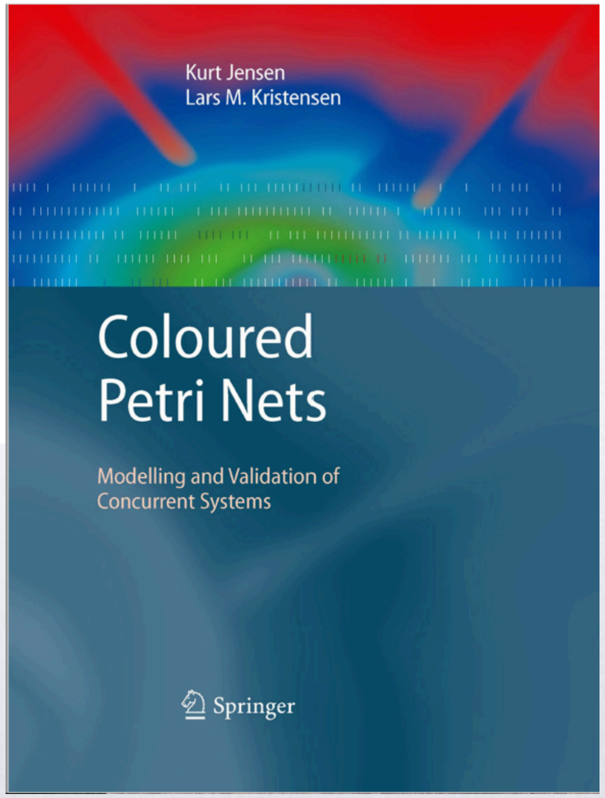
$$M' \in \mathscr{R}_{SS}(M)$$

2. *A marking M' is reachable from a marking $M \in \mathscr{R}(M_0)$ if and only if*

$$SCC(M') \in \mathscr{R}_{SG}(SCC(M))$$

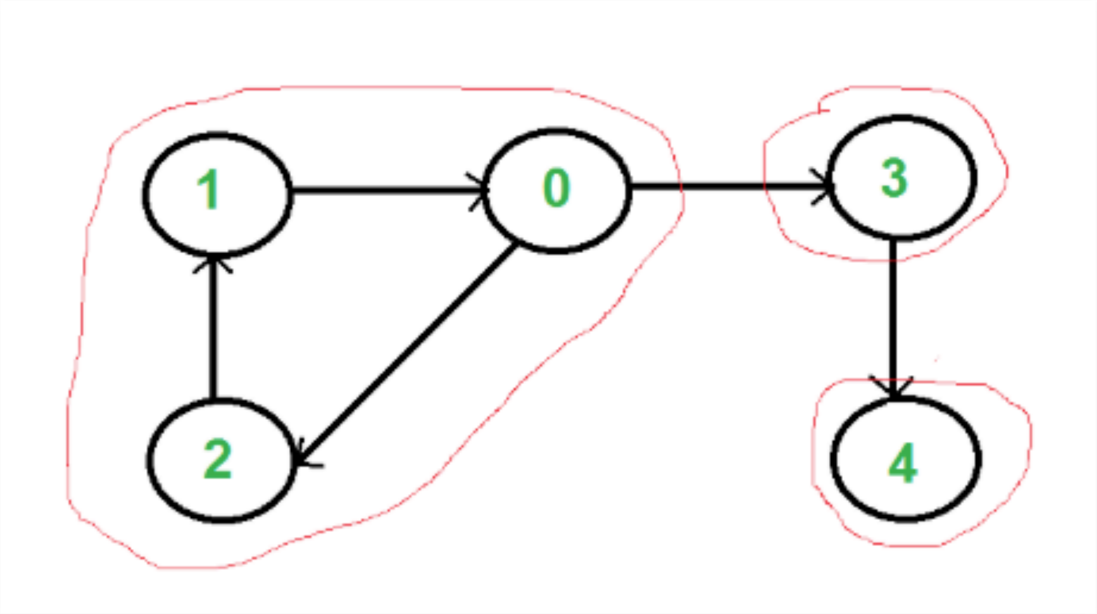
3. *A predicate ϕ on markings is reachable if and only if*

$$\exists M \in N_{SS} : \phi(M)$$



Strongly Connected Components

A directed graph is strongly connected if there is a path between all pairs of vertices. A strongly connected component (**SCC**) of a directed graph is a maximal strongly connected subgraph. For example, there are 3 SCCs in the following graph.



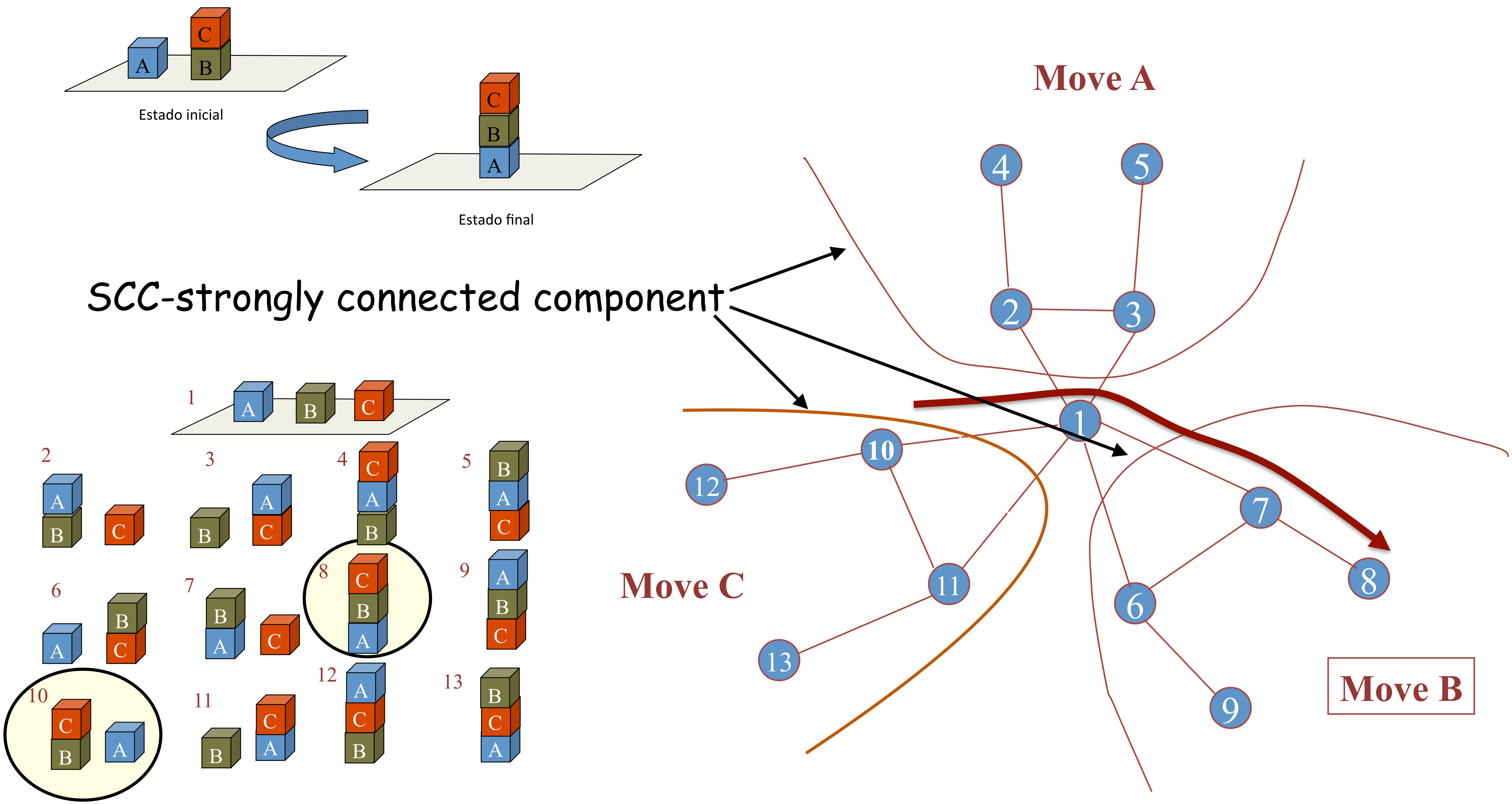
We can find all strongly connected components in O(V+E) time using Kosaraju's algorithm. Following is detailed Kosaraju's algorithm.

1) Create an empty stack 'S' and do DFS traversal of a graph. In DFS traversal, after calling recursive DFS for adjacent vertices of a vertex, push the vertex to stack. In the above graph, if we start DFS from vertex 0, we get vertices in stack as 1, 2, 4, 3, 0.

2) Reverse directions of all arcs to obtain the transpose graph.

3) One by one pop a vertex from S while S is not empty. Let the popped vertex be 'v'. Take v as source and do DFS (call DFSUtil(v)). The DFS starting from v prints strongly connected component of v. In the above example, we process vertices in order 0, 3, 4, 2, 1 (One by one popped from stack).

Lembrando a Aula2…



Fairness

Many different notions of fairness have been proposed in the literature on Petri nets. We present here two basic fairness concepts: bounded-fairness (B-fairness) and unconditional (global) fairness (U-fairness).

B-fairness

Two transitions t1 and t2 are said to be in a bounded-fair (or B-fair) relation if the maximum number of times that either one can fire while the other is not firing is bounded. A Petri net (N, M0) is said to be a B-fair net if every pair of transitions in the net are in a B-fair relation.

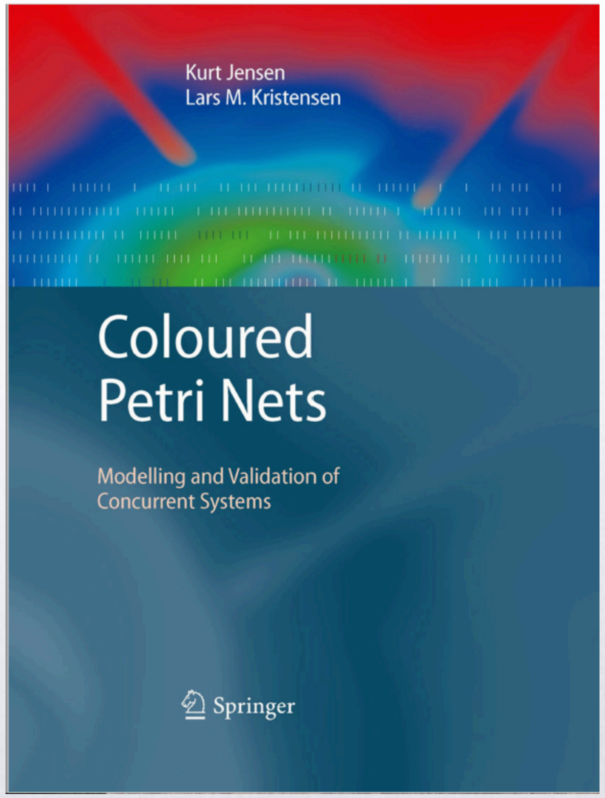
Fairness

Many different notions of fairness have been proposed in the literature on Petri nets. We present here two basic fairness concepts: bounded-fairness (B-fairness) and unconditional (global) fairness (U-fairness).

U-fairness

A firing sequence σ is said to be unconditionally (globally) fair if it is finite or every transition in the net appears infinitely often in a. A Petri net (N, M0) is said to be an unconditionally fair net if every firing sequence σ from M in R(M0) is unconditionally fair

The fairness properties give information about how often transitions occur in infinite occurrence sequences. We denote by OS_∞ the set of infinite occurrence sequences starting in the initial marking. For a transition $t \in T$ and an infinite occurrence sequence $\sigma \in OS_\infty$ we use $OC_t(\sigma)$ to denote the number of steps in which t occurs.



Fairness properties

- Are only relevant if there are Infinite Firing Sequences (IFS), otherwise CPN Tools reports: "no infinite occurrence sequences".
- Given a transition t it is often desirable that t appears infinitely often in an IFS.
- Properties reported by CPN Tools
  - t is **impartial**: t occurs infinitely often in every IFS.
  - t is **fair**: t occurs infinitely often in every IFS where t is enabled infinitely often.
  - t is **just**: t occurs infinitely often in every IFS where t is continuously enabled from some point onward
  - **No fairness**: not just, i.e., there is an IFS where t is continuously enabled from some point onward and does not fire anymore

Distância Síncrona

Definition 45

Define-se como a distância síncrona entre duas transições t_1e t_2 de uma rede P/T (N, M_0), ao número inteiro,
$d_{1,2} = max\,|\overline{\sigma}(t_1) - \overline{\sigma}(t_2)|$,
onde $\overline{\sigma}(t_i)$ é a variância no número de disparos de t_i.

HOW TO FIND INVARIANTS FOR COLOURED PETRI NETS

Kurt Jensen
Computer Science Department
Aarhus University, Ny Munkegade
DK - 8000 Aarhus C, Denmark

Abstract: This paper shows how invariants can be found for coloured Petri Nets. We define a set of transformation rules, which can be used to transform the incidence matrix, without changing the set of invariants.

1. INTRODUCTION

In [2] coloured Petri Nets are defined as a generalization of place/transition-nets, and it is shown how to generalize the invariant-concept, [4], to coloured Petri nets. The elements in the involved matrices are no longer integers but functions, and matrix multiplication is generalized to involve composition/application of these functions. In [2] it is shown how to use invariants when proving various properties for the considered systems. In the present paper it will be shown how to find invariants by a sequence of transformations mapping the incidence matrix into gradually simpler matrices with the same set of invariants. The present paper is a continuation of [2], and it will use the definitions and notations from [2] without further explanation.

In section 2 we define four transformation rules, which can be used to transform the incidence matrix of a coloured Petri net. The four transformation rules are inspired by the method af Gauss-elimination, which is used for matrices, where all elements belong to a field. We prove that the transformation rules are sound, i.e. they do not change the set of invariants. The matrix elements for coloured Petri nets are not contained in a field, but only in a non-commutative ring, and thus division of two elements may be impossible. For this situation no general algorithm is known to solve homogeneous matrix equations. Thus we cannot expect our set of transformation rules to be complete, i.e. it is in general not possible to find all invariants only by means of the rules.

An Introduction to the Theoretical Aspects of Coloured Petri Nets

Kurt Jensen
Computer Science Department, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark

Phone: +45 89 42 32 34
Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk

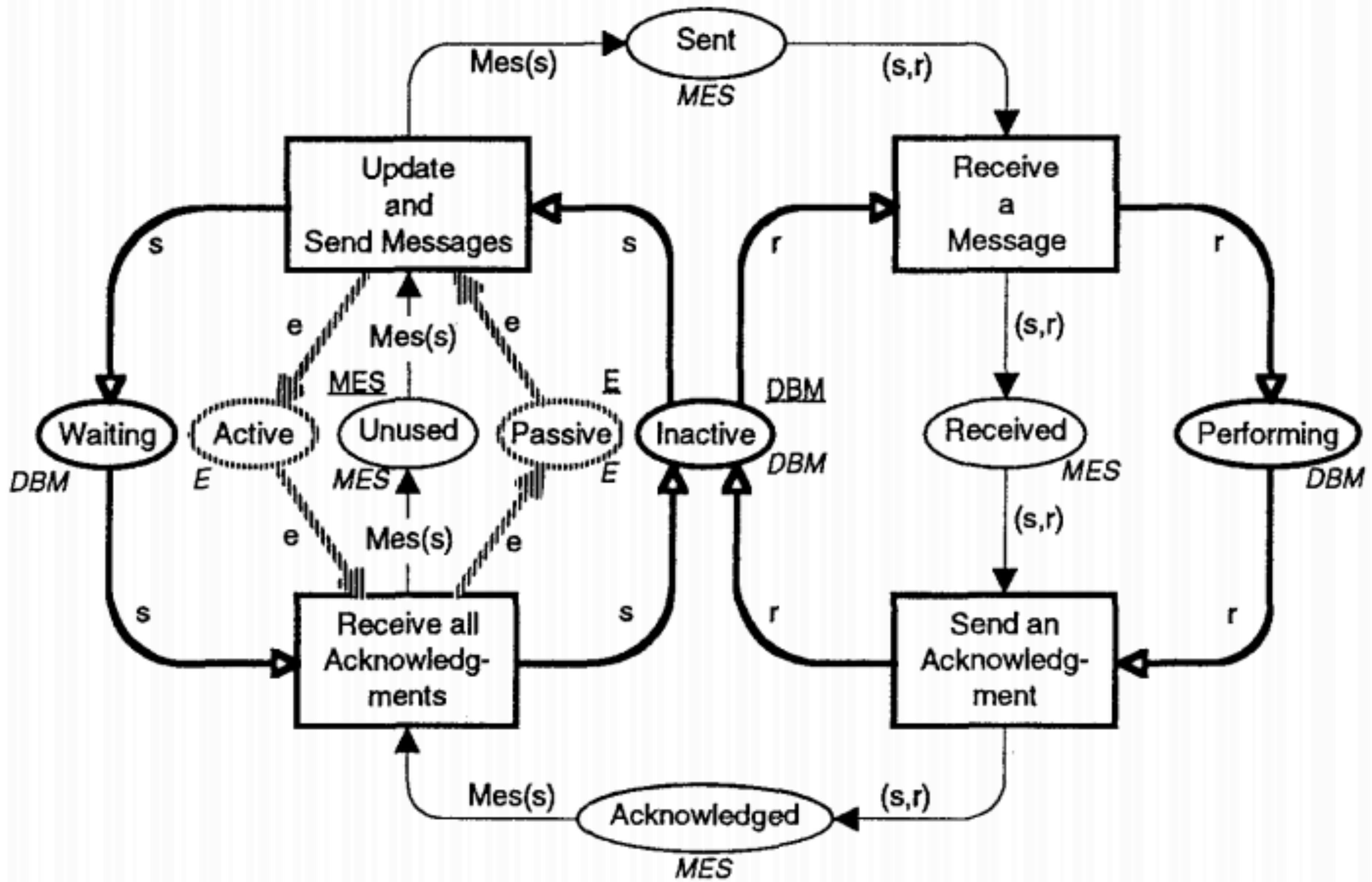
Abstract: This paper presents the basic theoretical aspects of Coloured Petri Nets (CP-nets or CPN). CP-nets have been developed, from being a promising theoretical model, to being a full-fledged language for the design, specification, simulation, validation and implementation of large software systems (and other systems in which human beings and/or computers communicate by means of some more or less formal rules). The paper contains the formal definition of CP-nets and their basic concepts (e.g., the different dynamic properties such as liveness and fairness). The paper also contains a short introduction to the analysis methods, in particular occurrence graphs and place invariants.

The development of CP-nets has been driven by the desire to develop a modelling language – at the same time theoretically well-founded and versatile enough to be used in practice for systems of the size and complexity that we find in typical industrial projects. To achieve this, we have combined the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for the description of the synchronisation of concurrent processes, while programming languages provide the primitives for the definition of data types and the manipulation of their data values.

The paper does not assume that the reader has any prior knowledge of Petri nets – although such knowledge will, of course, be a help.

Keywords: Petri Nets, High-level Petri Nets, Coloured Petri Nets.

```
val n = 5;
color DBM = index d with 1..n  declare ms;
color PR = product DBM * DBM  declare mult;
fun diff(x,y) = (x<>y);
color MES = subset PR by diff  declare ms;
color E = with e;
fun Mes(s) = mult'PR(1`s,DBM-1`s);
var s, r : DBM;
```



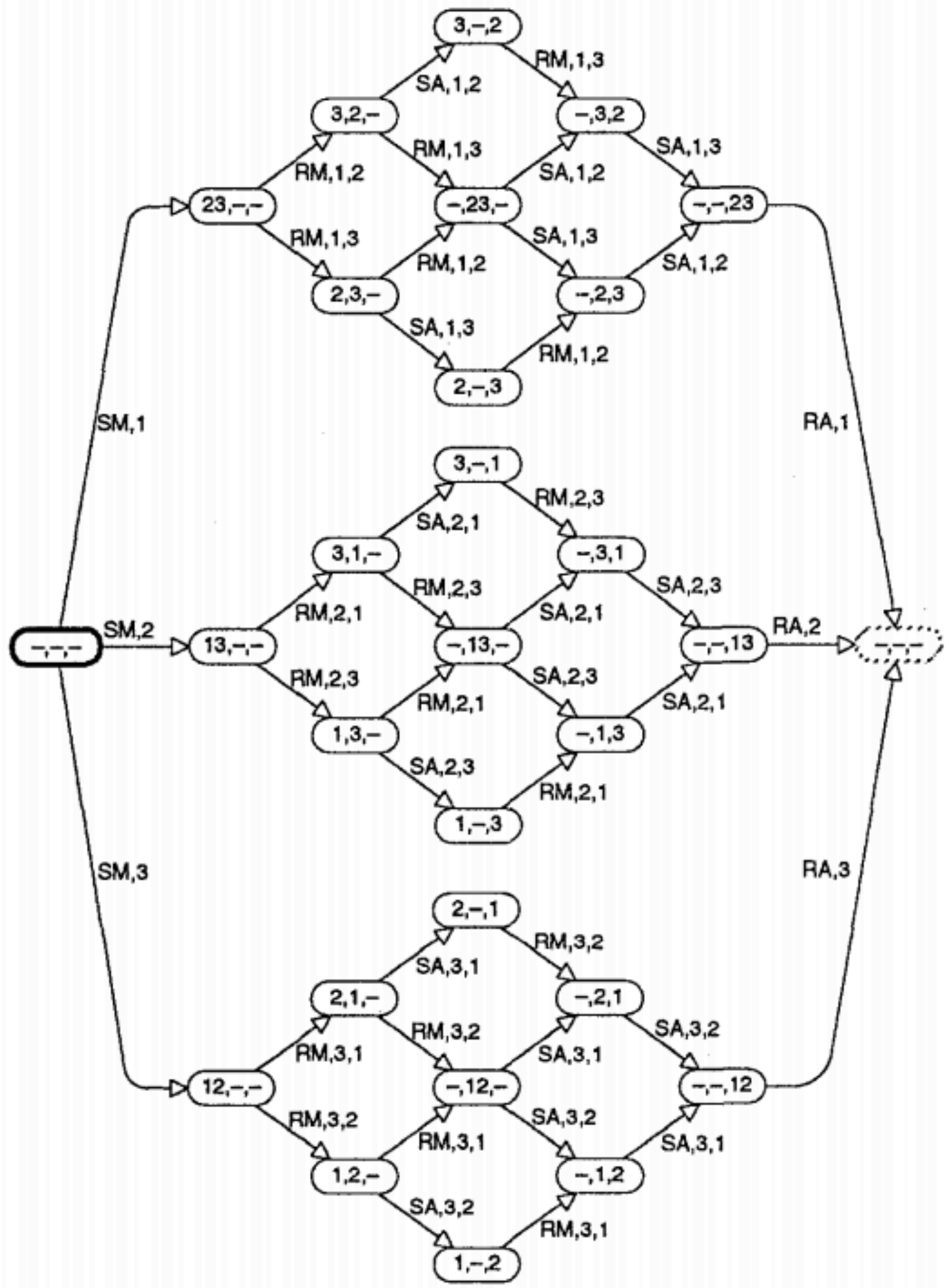
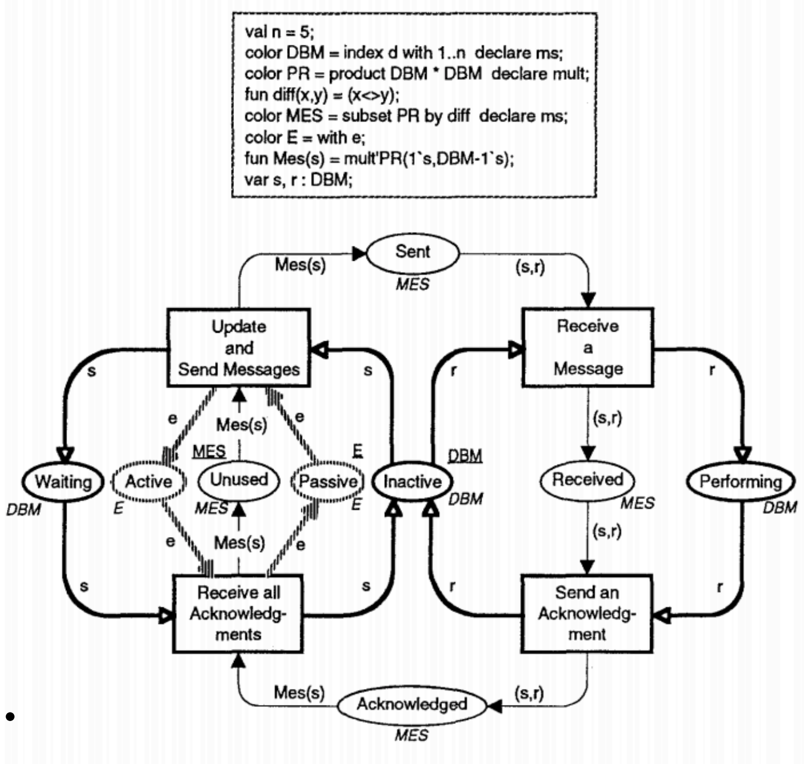


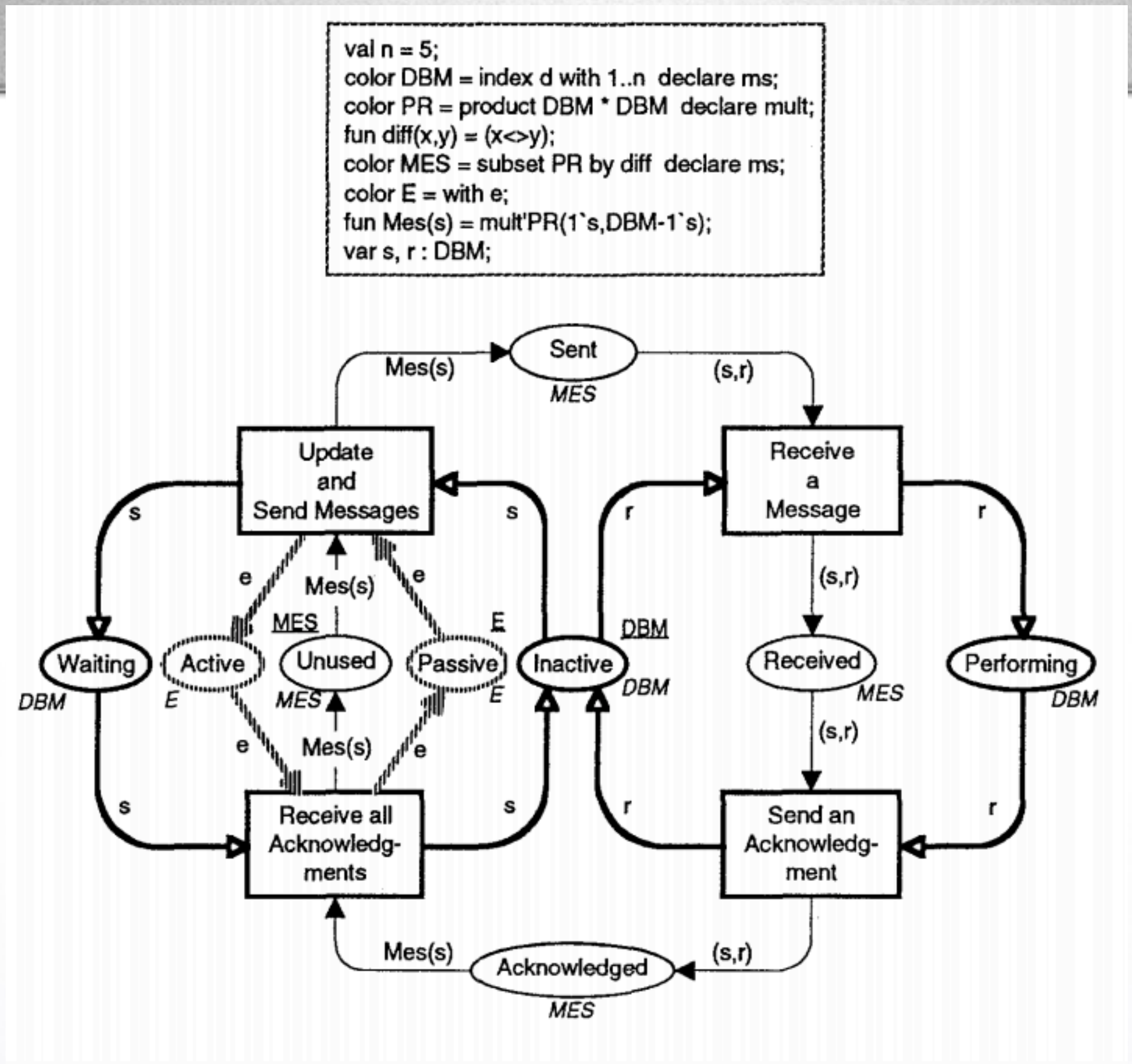
Fig. 2. Occurrence graph for data base system with 3 managers



The basic idea behind place invariants is to construct equations which are satisfied for all reachable markings. In the data base system we expect each manager to be either Inactive, Waiting or Performing.

This is expressed by the following equation satisfied for all reachable markings

M: M(Inactive) + M(Waiting) + M(Performing) = DBM(x).



If we now look at the messages properties...

M(Unused) + M(Sent) + M(Received) + M(Acknowledged) = MES(s)

M(Active) + M(Passive) = 1'e.

An Introduction to the Theoretical Aspects of Coloured Petri Nets

Kurt Jensen
Computer Science Department, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark

Phone: +45 89 42 32 34
Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk

Abstract: This paper presents the basic theoretical aspects of Coloured Petri Nets (CP-nets or CPN). CP-nets have been developed, from being a promising theoretical model, to being a full-fledged language for the design, specification, simulation, validation and implementation of large software systems (and other systems in which human beings and/or computers communicate by means of some more or less formal rules). The paper contains the formal definition of CP-nets and their basic concepts (e.g., the different dynamic properties such as liveness and fairness). The paper also contains a short introduction to the analysis methods, in particular occurrence graphs and place invariants.

The development of CP-nets has been driven by the desire to develop a

Each of the above equations can be written on the form:

$$W_{p_1}(M(p_1)) + W_{p_2}(M(p_2)) + \ldots + W_{p_n}(M(p_n)) = m_{inv}$$

where $\{p_1, p_2, \ldots, p_n\} \subseteq P$. Each **weight** W_p is a function mapping from the type of p into some common type $A \in \Sigma$ shared by all weights. Finally m_{inv} is a multi-set. It can be determined by evaluating the left-hand side of the equation in the initial marking (or in any other reachable marking).

An Introduction to the Theoretical Aspects of Coloured Petri Nets

Kurt Jensen
Computer Science Department, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark

Phone: +45 89 42 32 34
Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk

Abstract: This paper presents the basic theoretical aspects of Coloured Petri

Definition 7.1: Let $A \in \Sigma$ be a type and let $W = \{W_p\}_{p \in P}$ be a set of linear functions such that $W_p \in [C(p)_{WS} \rightarrow A_{WS}]$ for all $p \in P$.

(i) W is a **place flow** iff:

$$\forall (t,b) \in BE: \sum_{p \in P} W_p(E(p,t)<b>) = \sum_{p \in P} W_p(E(t,p)<b>).$$

(ii) W determines a **place invariant** iff:

$$\forall M \in [M_0\rangle: \sum_{p \in P} W_p(M(p)) = \sum_{p \in P} W_p(M_0(p)).$$

An Introduction to the Theoretical Aspects of Coloured Petri Nets

Kurt Jensen
Computer Science Department, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark

Phone: +45 89 42 32 34
Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk

Abstract: This paper presents the basic theoretical aspects of Coloured Petri Nets (CP-nets or CPN). CP-nets have been developed, from being a promising theoretical model, to being a full-fledged language for the design, specification, simulation, validation and implementation of large software systems (and other systems in which human beings and/or computers communicate by means of some more or less formal rules). The paper contains the formal definition of CP-nets and their basic concepts (e.g., the different dynamic properties such as liveness and fairness). The paper also contains a short introduction to the analysis methods, in particular occurrence graphs and place invariants.

The development of CP-nets has been driven by the desire to develop a modelling language – at the same time theoretically well-founded and versatile enough to be used in practice for systems of the size and complexity that we find

Theorem 7.2: W is a place flow ⇔ W determines a place invariant.

⇒ is satisfied for all CP-nets.

⇐ is only satisfied when the CP-net does not have dead binding elements.

Invariants now mens to preserve the type of marks, or to establish a common type to be preserved in different places.

Still, there is no well-accepted “algorithm” to perform invariant calculus in Colored Petri Nets.

Part II. Modelling

7. Introduction

8. Modelling and Analysis Techniques by Example
8.1 Nets, Refinement, and Abstraction
8.2 Place/Transition Nets and Resource Management
8.3 Coloured Nets, Abstraction, and Unfolding

9. Techniques
9.1 Building Blocks
9.2 Combining Nets
9.2.1 Place Fusion
9.2.2 Arc Addition
9.2.3 Transition Fusion
9.3 High-Level Nets
9.3.1 Coloured Nets
9.3.2 Fairness, Priority, and Time
9.4 Decomposing Nets
9.5 Conclusion

10. Methods
10.1 State-Oriented Modelling
10.1.1 Specification
10.1.2 Design
10.1.3 Implementation
10.1.4 Conclusion
10.2 Event-Oriented Modelling
10.2.1 High-Level Modelling
10.2.2 Protocol Modelling
10.2.3 Verification
10.2.4 Conclusion
10.3 Object-Oriented Modelling
10.3.1 Objects vs Petri Nets
10.3.2 Integration Approaches ... 148
10.3.3 A Multi-Formalism Approach Including Nets ... 152
10.3.4 Conclusion ... 157

9. Techniques*

In this chapter we give general principles of modelling with Petri nets. We will concentrate on the aspects that are specific to Petri nets. We shall discuss how the specific building blocks of Petri nets (places, transitions, arcs, and tokens) are used to model components and aspects of the problem.

A large part of this chapter is devoted to composition and decomposition of net models. A bottom-up modelling strategy starts by building models for simple subsystems and combining them into more complex ones until the desired model is obtained. The top-down approach decomposes the system to be modelled into subsystems, and decomposes these subsystems into smaller subsystems to the point where subsystems can simply be modelled as nets. Often the two approaches are combined. The gap between the system to be modelled and the building blocks of the modelling paradigm is narrowed by both decomposing the system and constructing some higher-level building blocks.

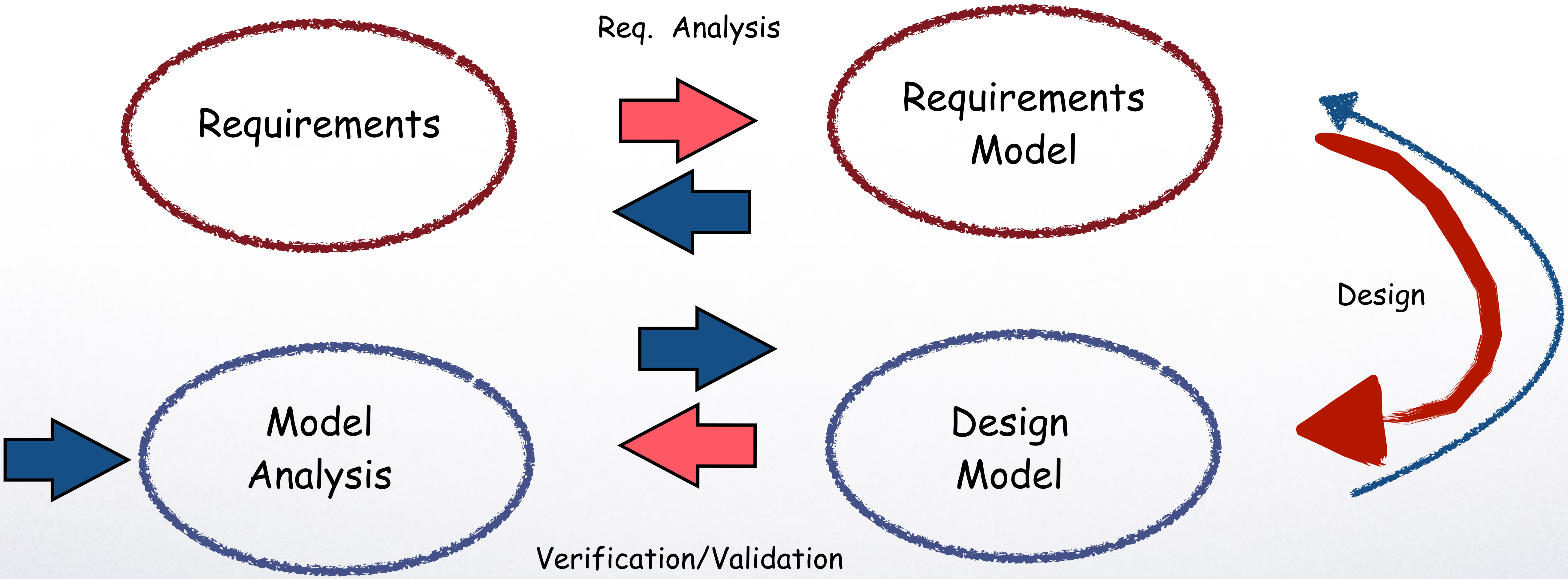
In the sections to come, we will discuss the use of Petri net building blocks for modelling. Then, we will consider the synthesis and decomposition of nets. We start with simple (place/transition) nets and then move on to extensions including colour, priority and time.

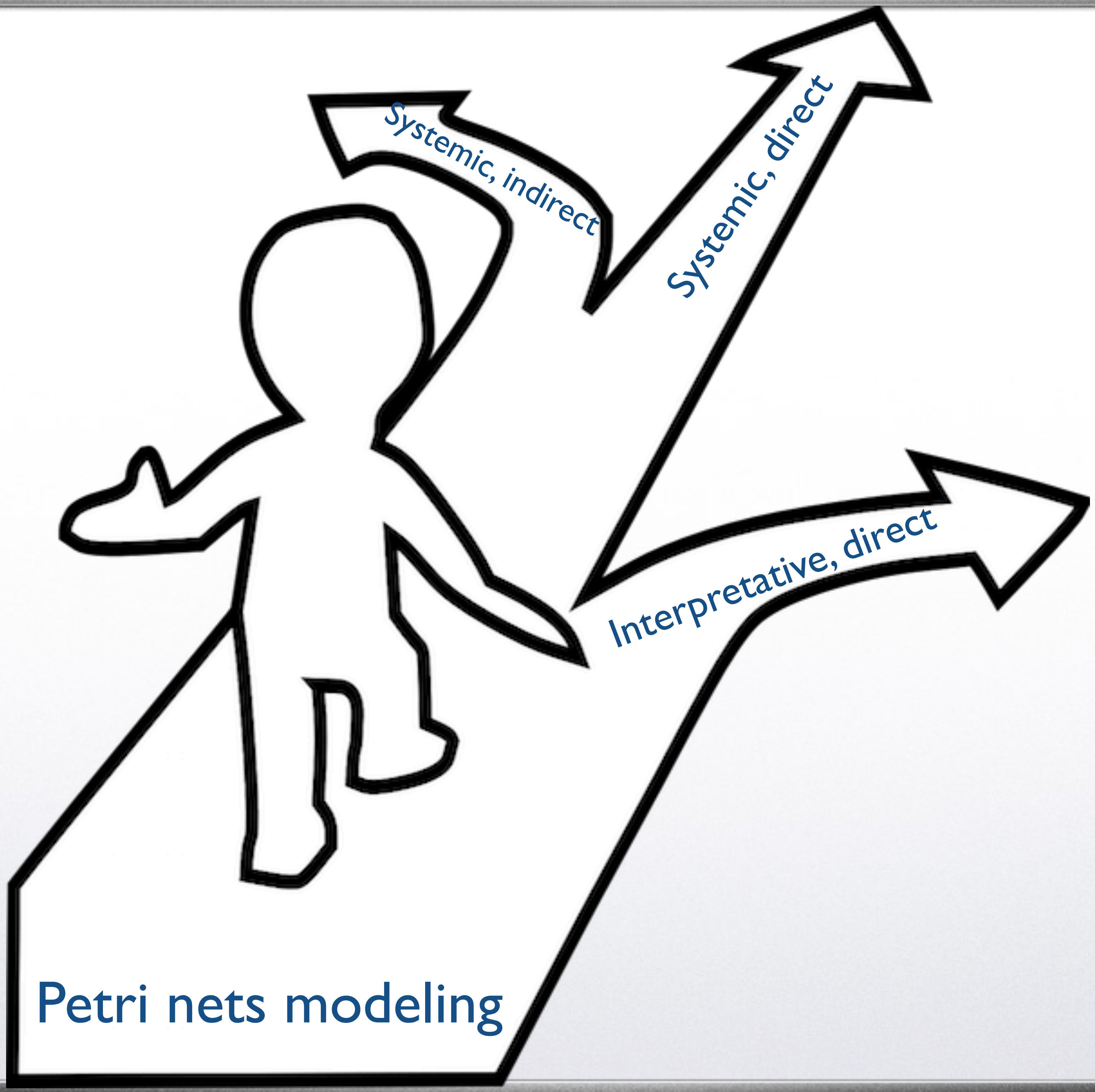
9.1 Building Blocks

Petri nets consist of places (circles), transitions (squares or rectangles), di-

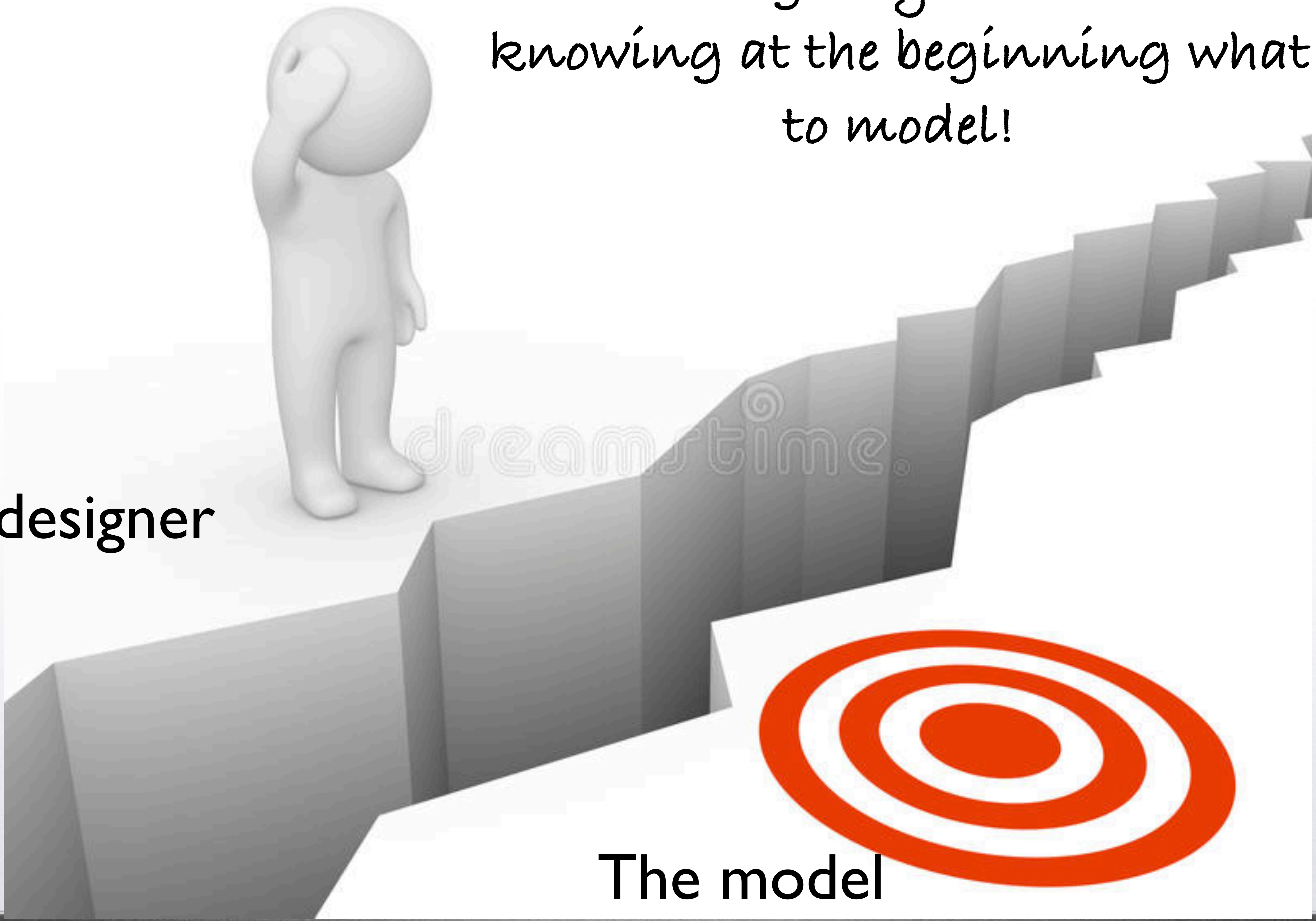
Petri nets modeling: where to begin?

Use of Petri Nets in Design




Systemic, indirect
Systemic, direct
Interpretative, direct
Petri nets modeling

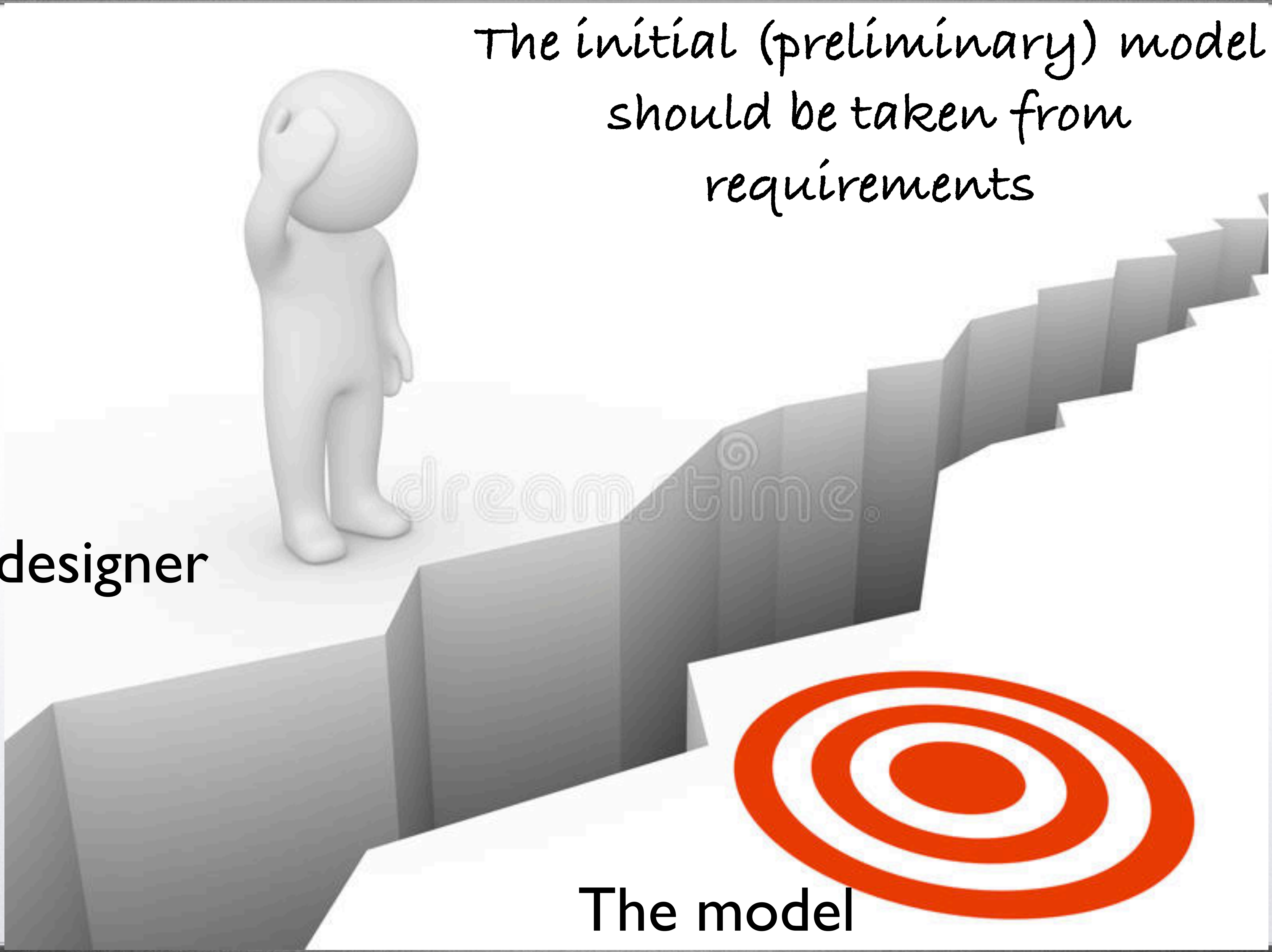
Modeling a system means knowing at the beginning what to model!



The designer

The model

The initial (preliminary) model should be taken from requirements



d-l@b

MECATRÔNICA

2009 ISECS International Colloquium on Computing, Communication, Control, and Management

Transformation of Usecase and Sequence Diagrams to Petri Nets

Sima Emadi
Engineering Department,
Meybod Branch, Islamic Azad University,
Yazd, Iran
emadi@srbiau.ac.ir

Fer
Computer I
Shahid
f_sl

Abstract—**With the growing use of UML diagrams for software design description and the importance of nonfunctional requirements evaluation at software development process, transforming these diagrams to executable models is considered to be significant. Nonfunctional requirements can not be evaluated directly by UML diagrams. Software designers are not usually familiar with non-functional requirement analysis and are not able to analyze such requirements easily. Therefore the designer should produce an executable model from software design description to be ready for analysis. usecase and sequence diagrams are the most important UML diagrams for software design description. In this paper, we propose new algorithms that enable a designer to transform usecase and sequence diagrams into executable models based on Petri nets and then we show how to use this Petri net models for simulation. Finally, to represent the usage of our proposed algorithms, we consider a case study as an example.**

Keywords-usecase diagram, sequence diagram, executable model, petri net, software design, nonfunctional requirement evaluation

I. INTRODUCTION

Nowadays, one of the most noticeable tasks of a designer

specific nonfunctional
When we apply ther
reliability, we should
one. Elkoutbi et al.
structure to color Pe
transformed usecase t
(OSAN) [6]. However
transformation of use
approaches have be
sequence diagram to
Bernardi et al. all s
transformed to Gen
Ourdani et al. have t
sequence diagram to
between two transfo
approach the transfo
messages as well as
approach, the transfor
receiver component.
utilized all structures
transformation. On th
based on Petri nets, v
measured, we just atta
tokens and adopt ex
attribute value from th

Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets

Christoph Eichner, Hans Fleischhack, Roland Meyer,
Ulrik Schrimpf, and Christian Stehno

Parallel Systems Group,
Department for Computing Science,
Carl von Ossietzky Universität,
D-26111 Oldenburg, Germany
forename.surname@informatik.uni-oldenburg.de

Abstract. With the introduction of UML 2.0, many improvements to diagrams have been incorporated into the language. Some of the major changes were applied to sequence diagrams, which were enhanced with most of the concepts from ITU-T's Message Sequence Charts, and more. In this paper, we introduce a formal semantics for most concepts of sequence diagrams by means of Petri nets as a formal model. Thus, we are able to express the partially ordered and concurrent behaviour of the diagrams natively within the model. Moreover, the use of coloured high-level Petri nets allows a comprehensive and efficient structure for data types and control elements. The proposed semantics is defined compositionally, based on basic Petri net composition operations.

1 Introduction

The long-standing and successfully applied modelling technique of Message Sequence Charts (MSC) [11] of ITU-T has finally found its way to the most widely applied software modelling framework, the Unified Modelling Language (UML) [18]. In its recent 2.0 version, sequence diagrams (SD, interaction diagram) were enhanced by important control flow features. This change is one of

Requirement Analysis Method for Real World Systems in Automated Planning

Rosimarci Tonaco Basbaum[1] and Tiago Stegun Vaquero[2] and José Reinaldo Silva[1]
[1]Department of Mechatronic Engineering, University of São Paulo, Brazil
[2]Department of Mechanical & Industrial Engineering, University of Toronto, Canada
rosimarci@usp.br, tiago.vaquero@mie.utoronto.ca, reinaldo@usp.br

On the intelligent design field the requirement analysis phase has a fundamental role in automated planning- especially for "real life" systems - because it has the ability to identify or redesign variables which can potentially increase the model accuracy generated by the automated planner. A great effort has been made today in the area of Artificial Intelligence for defining reliable automated planning systems that can be applied for real life applications. That leads to the need for systematic design process, in which the initial phases are not neglected and where Knowledge and Requirement Engineering tools have a fundamental role for supporting designers. This paper intent to investigate design methods as well as perform a more detailed study on the adoption of UML and Petri Nets in the requirement analysis phase using the itSIMPLE framework as a KE tool.

Introduction

Planning characterizes a specific kind of design problem where the purpose is to find a set of admissible actions to

eling language
the UML diag
analysis will b
such as invaria
1989b). The fir
tween the requ
diagrams. The
that are hidder
well as additio
interpreted for
constraints, in
istics that coul
The tool we
(Vaquero et al.
to performe the
ing the devopm
In the secti
UML in auton
tion of Petri Ne
the results and

SBAI 2017
XIII Simpósio Brasileiro de Automação Inteligente
Porto Alegre – RS, 1º – 4 de Outubro de 2017

GORE METHODS TO MODEL REAL WORLD PROBLEM DOMAINS IN AUTOMATED PLANNING

Javier Martínez Silva*, José Reinaldo Silva*
*Department of Mechatronics Engineering, University of São Paulo, São Paulo, Brazil
Emails: javsilva@usp.br, reinaldo@usp.br

Abstract— In the field of intelligent design, the early phase, dedicated to requirements modeling and analysis, plays a fundamental role, especially when analytic formal solutions are not suitable. Automated planning falls in that category - particularly when the target are "real world" systems. In requirement analysis Knowledge Engineering is explored to provide clues that can facilitate a convergence for a good planning solution. Therefore, a great effort has been made today in the area of Artificial Intelligence to define a reliable design process for automated planning that includes Knowledge Engineering treatment in the early phase, coupled to requirements modeling and analysis. This paper presents an integrated approach to requirements analysis based on GORE (Goal Oriented Requirements Engineering) that starts by modeling a knowledge architecture based on a domain and planning requirements represented in KAOS and converted in Petri Nets (PN) to analysis. A software tool called rekPlan is proposed to generate the PN graph. The analysis is made in another software tool proposed in our Lab that, GHENESys (General Hierarchical Enhanced Net System), that support unified Petri Nets following ISO/IEC 15.909. A real case study is presented, based on classic problems of pre-salt petroleum industries.

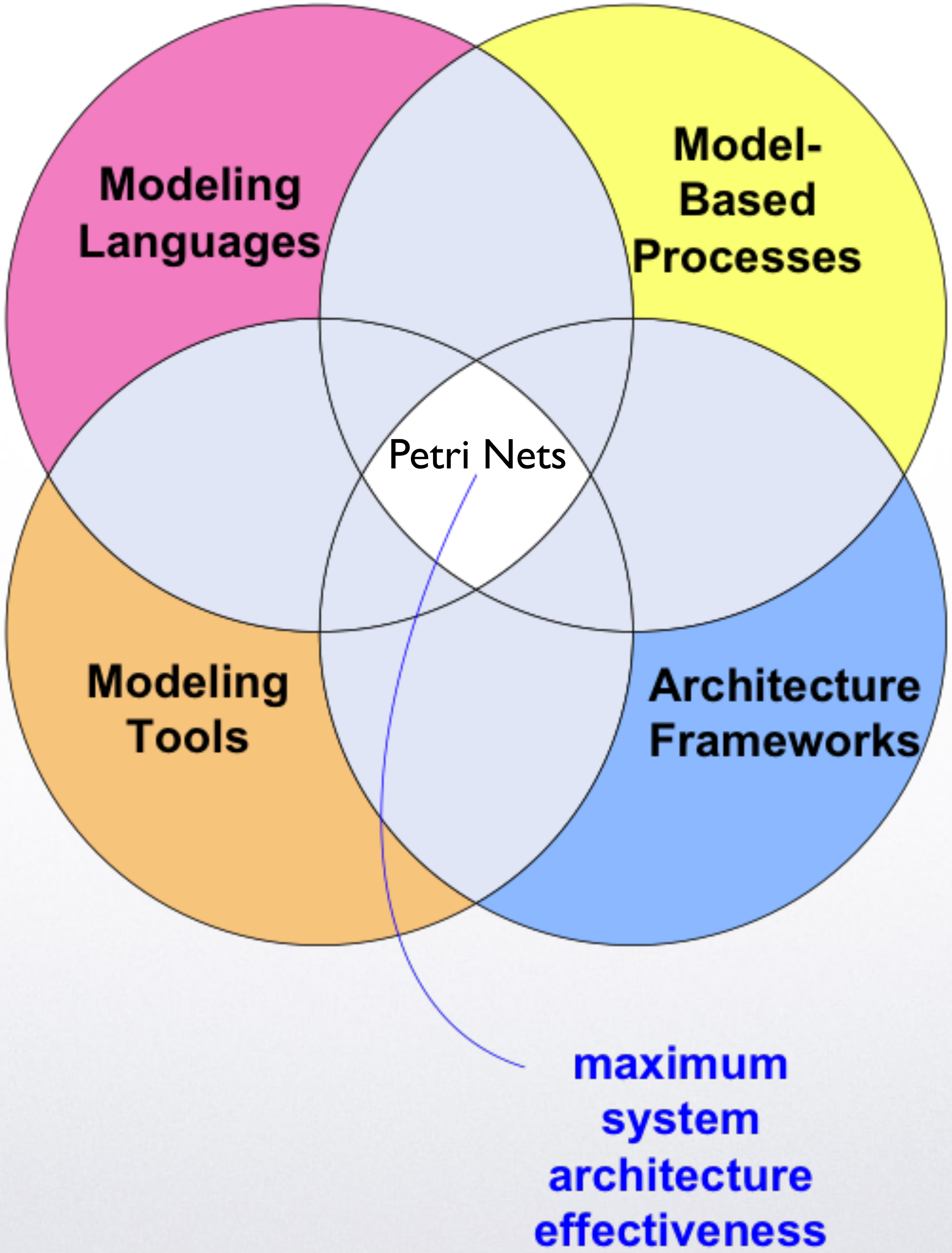
Keywords— Requirement Engineering, GORE, Petri Net, Automated Planning

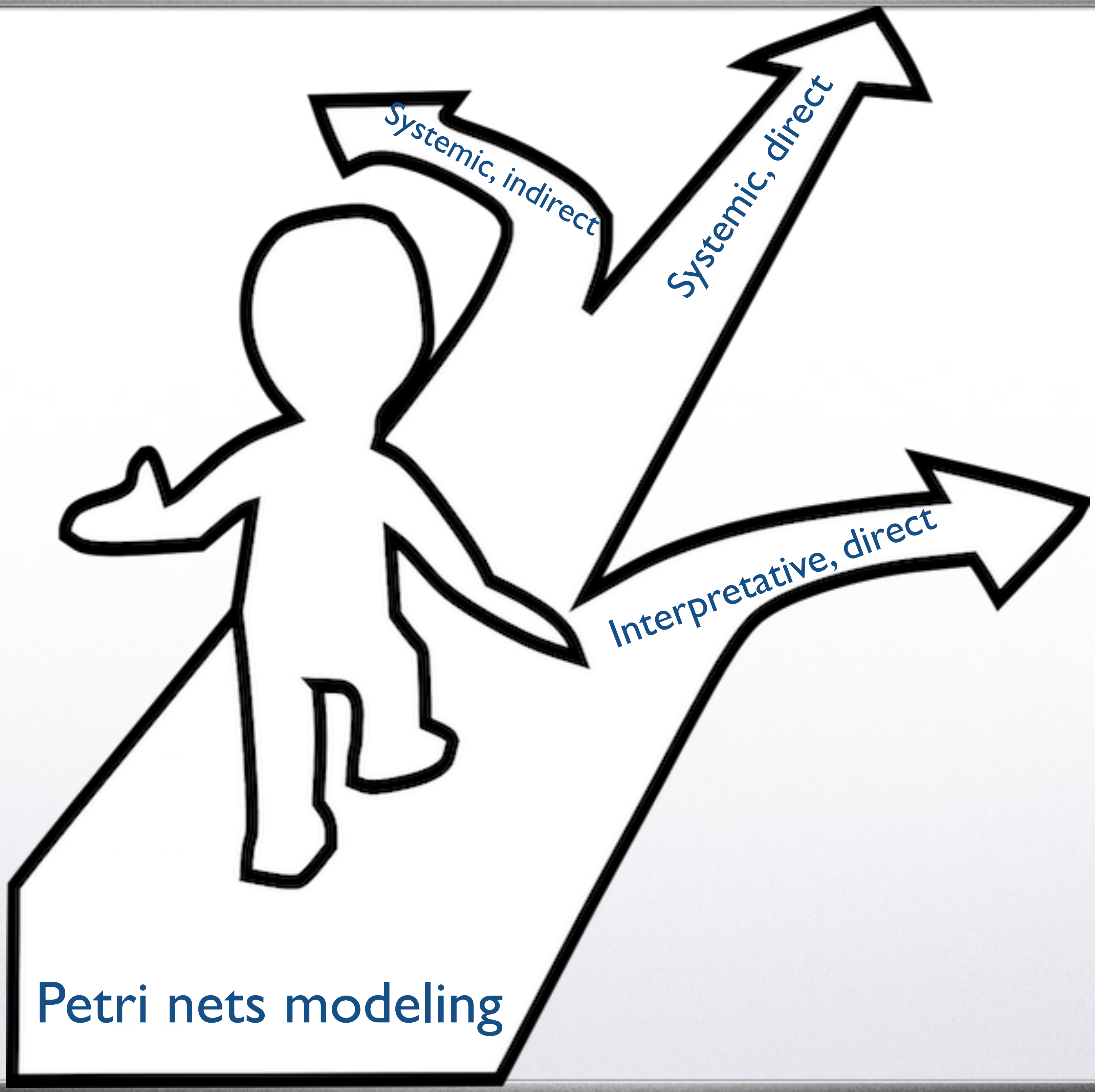
1 Introduction

Planning characterizes a specific type of design problem where the purpose is to find an admissible sequence of actions to bring the system from a given initial state to a target final state. Current approaches in the literature aim to improve the performance of automated planners by trying to optimize the search algorithms and the general solution (Lipovetzky and Geffner, 2017). In addition, most existing work on this direction is conceptually tested in synthesized artificial problems (closed problems that have limited set of actions) as. On the other hand, due to the extensive development in this area, some authors started to apply planning techniques to real world problems as well - like logistic problems - with a considerable higher number of variables, where the classic domain independent approaches are computationally prohibitive (Vaquero et al., 2012). Such alternative approach could bring some light and/or good results to challenge problems and could also gave some feedback to solve a fully automated, domain-independent problem.

tics problem proposed in ROADEF has a map of cities connected by airline routes. Transportation inside cities uses a truck (there is one truck in each city). Cities are abstracted, being treated as destination points. Inside a city, a truck can go from any point to any destination at no cost (Lin et al., 2016). However, in the real world, transportation within a city is a sub-problem that can involve considerable costs.

This paper intents to propose a requirement analysis formal procedure, based on hierarchical models, that starts by taking requirements of planning problems represented in KAOS (Keep All Object Satisfied) and proceed to analysis based on classic Petri Nets. This approach were inserted in a knowledge based tool called reKPlan (Requirement Engineering based on KAOS for Planning Problems) framework, that performs the KAOS/GHENeSys net translation (and eventually translates from GHENeSys to Linear Temporal Logic, LTL). Thus, the KAOS diagram is translated to a Petri net through a transference algorithm, proposed by (MARTINEZ SILVA, 2016)


Modeling Languages
Model-Based Processes
Petri Nets
Modeling Tools
Architecture Frameworks
maximum system architecture effectiveness


Systemic, indirect
Systemic, direct
Interpretative, direct
Petri nets modeling

There are invariants that could be inferred at the very beginning, it does not matter is we are using P/T or CPN. We will call it "**design invariants**".

Other invariants can be only calculated (specially in large projects) and then there is the problem of synthesising a P/T net - or unfolding the net - before doing it. We will call that **operational invariants**.

Tradeoff

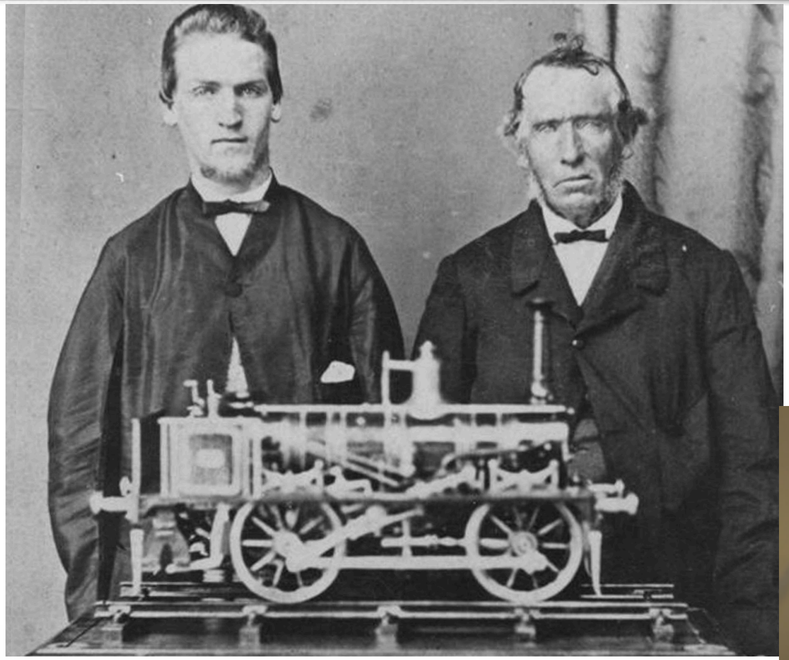
- More information in tokens
  - color sets, functions, etc.
  - behavior may be hidden in "code"
  - extreme case: all behavior folded into one place and one transition
- More information in network
  - possibly spaghetti networks to encode simple things
  - behavior may be incomprehensible
  - cannot be parameterized
  - extreme case: (infinite) classical Petri net

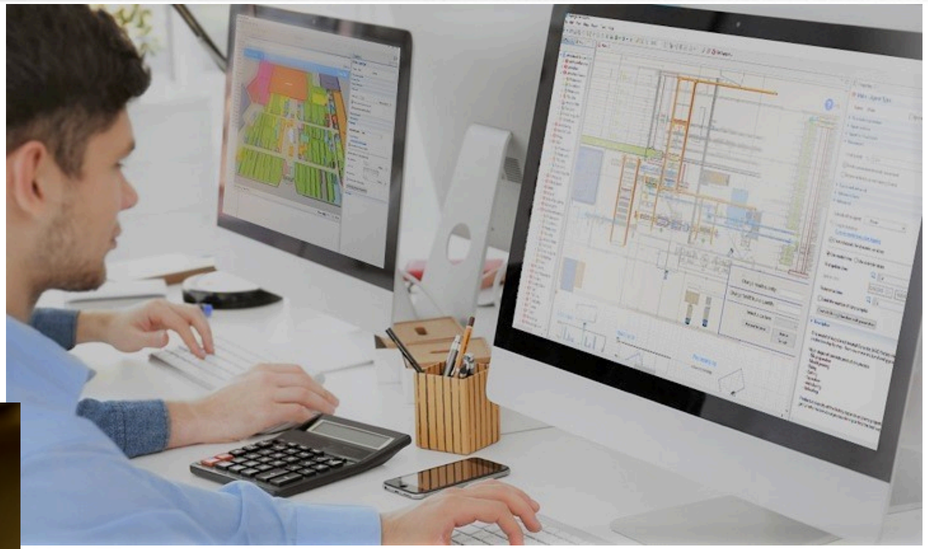
What is a model?

model and modelling
in painting, the *use of light and shade to simulate volume in the representation of solids*. In sculpture the terms denote a technique involving the use of a pliable material such as clay or wax. As opposed to carving, modelling permits addition as well as subtraction of material and lends itself to freer handling and change of intention. The technique is exemplified also by those works in cast metal and plaster that are made from the mold of a clay original. The mold is made by the process of cire perdue. The noun model is used to describe such an original and also *any three-dimensional scale model for a larger or more elaborate project in architecture, landscaping, or industry*. It also denotes a person or object used as an aid to representation in painting.

The Columbia Encyclopaedia, Sixth Edition. 2001.

Abstract representation, scale model of future design

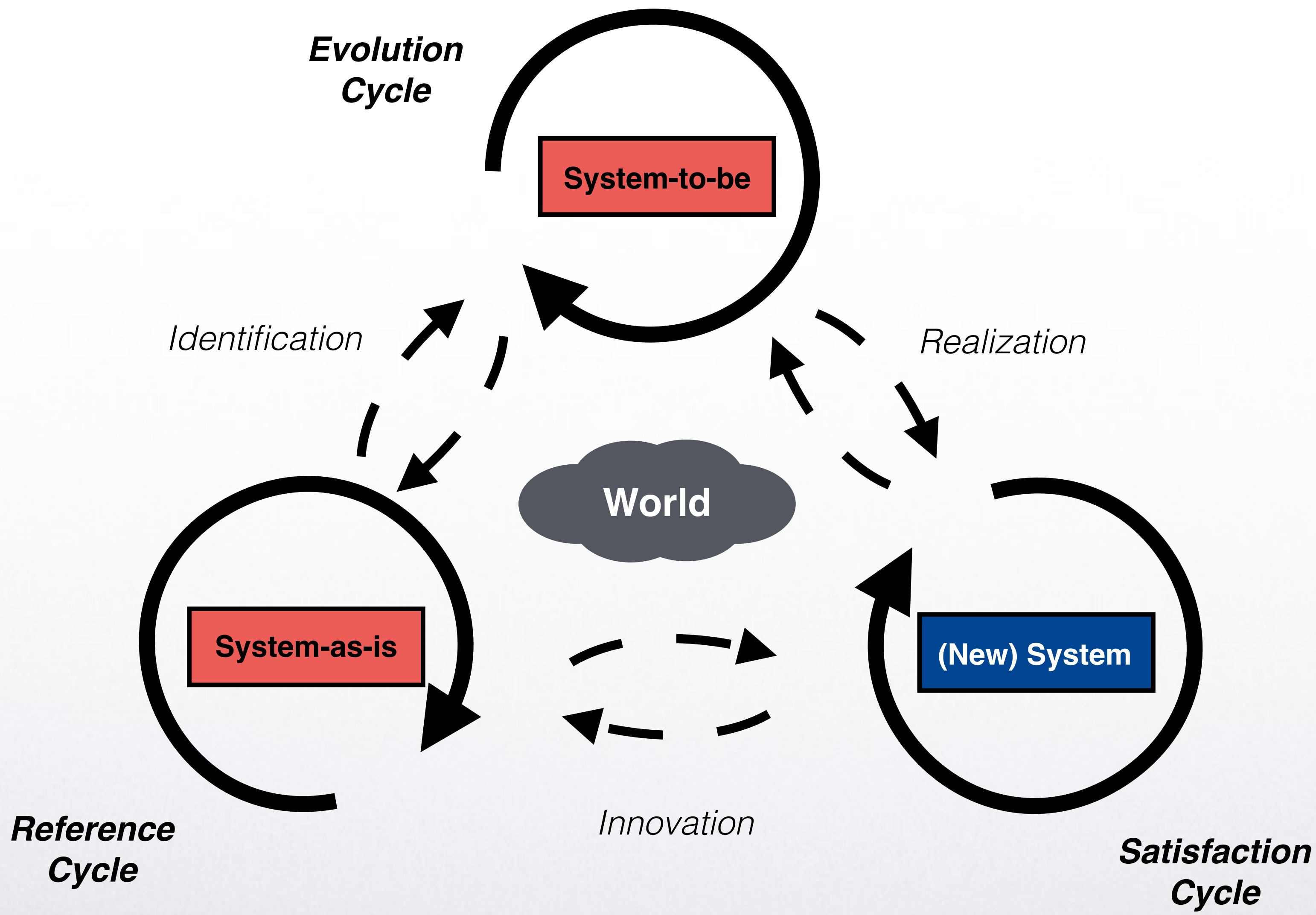


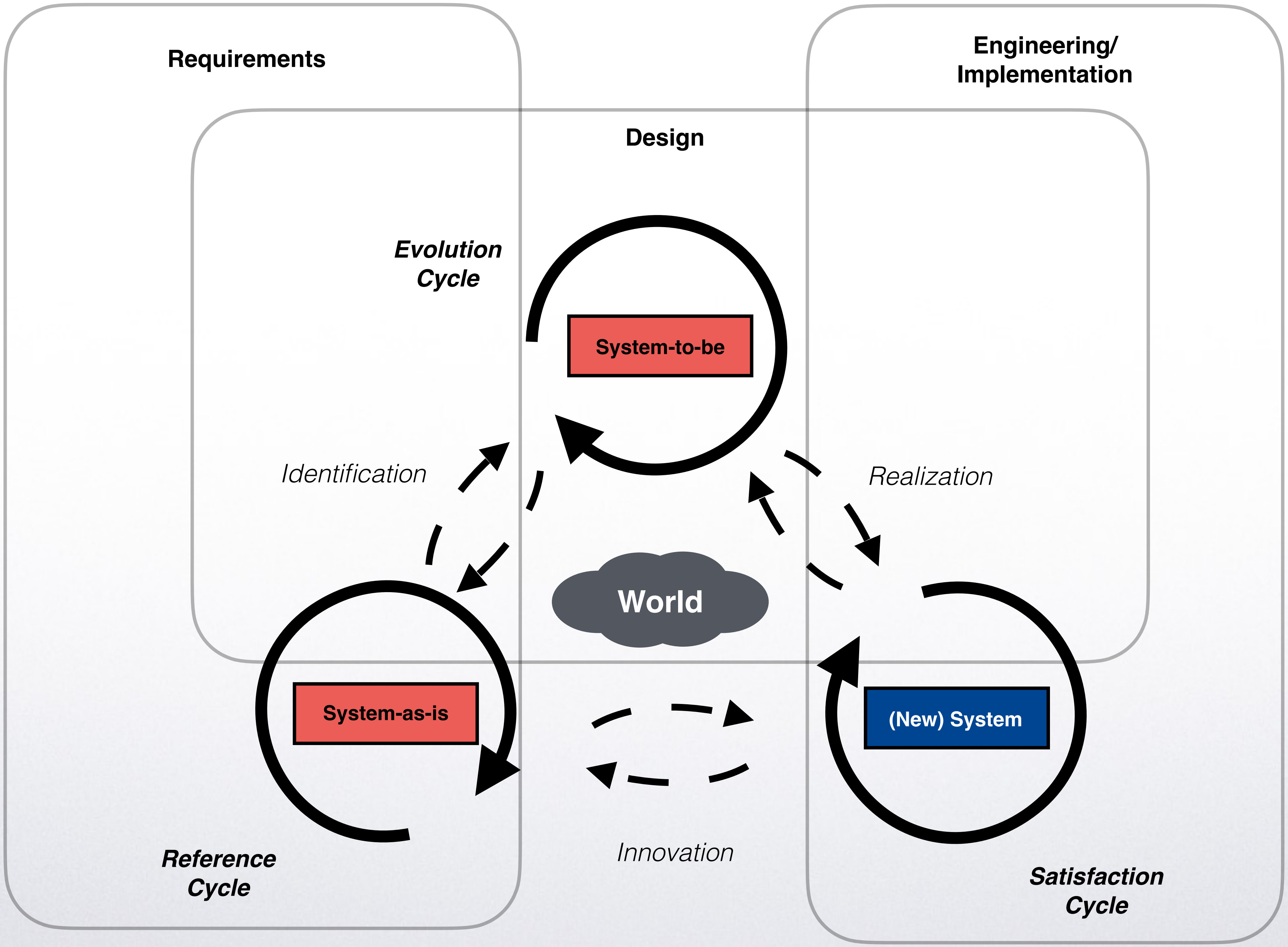


Formal Modelling

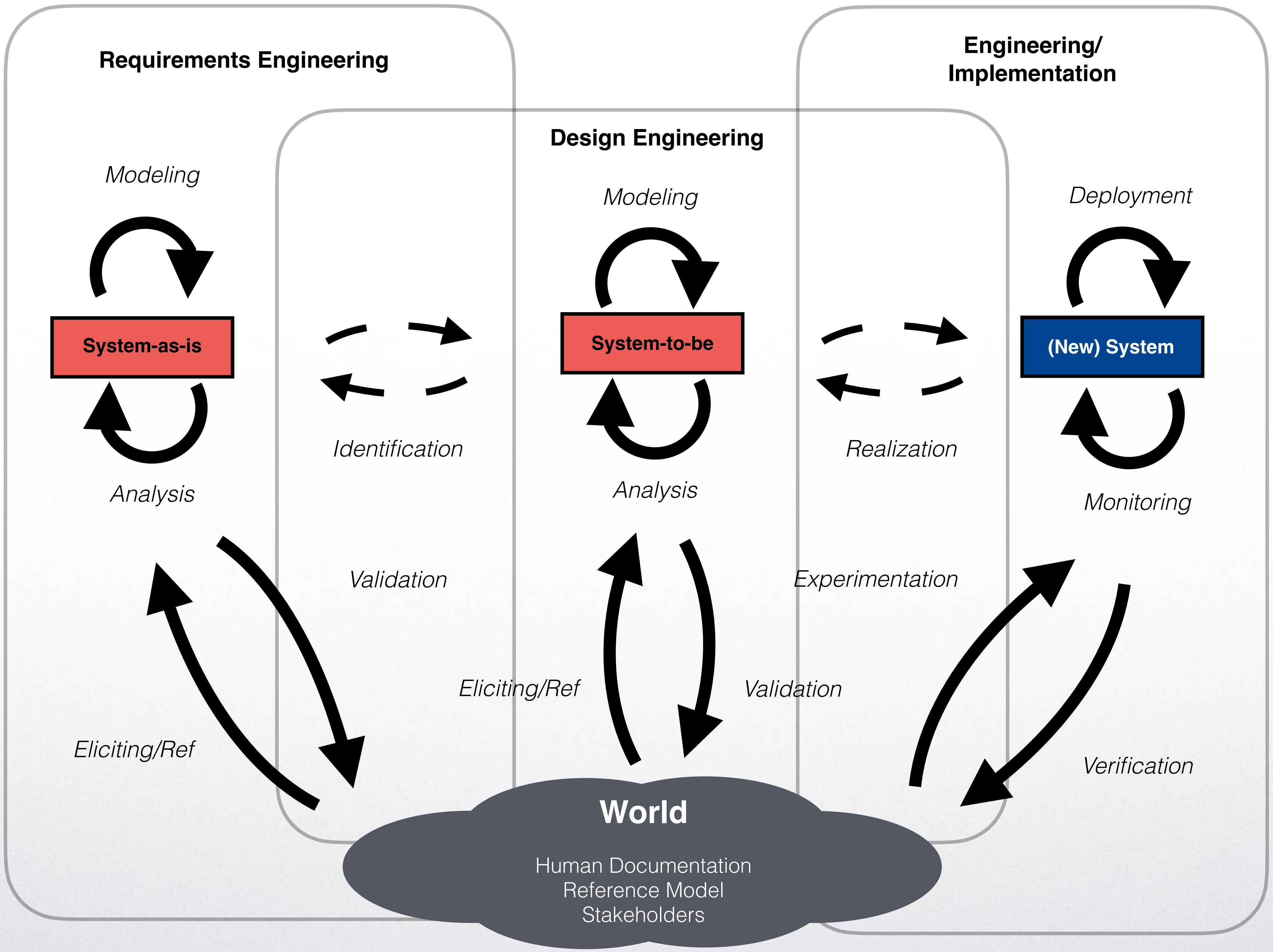
Formal modelling means representing a system by a formal model. Formal modelling is well defined and described as the application of a fairly broad variety of theoretical computer science fundamentals, in particular logic calculi, formal languages, automata theory, and program semantics, but also type systems and algebraic data types to problems in software and hardware specification and verification (Monin, 2003).

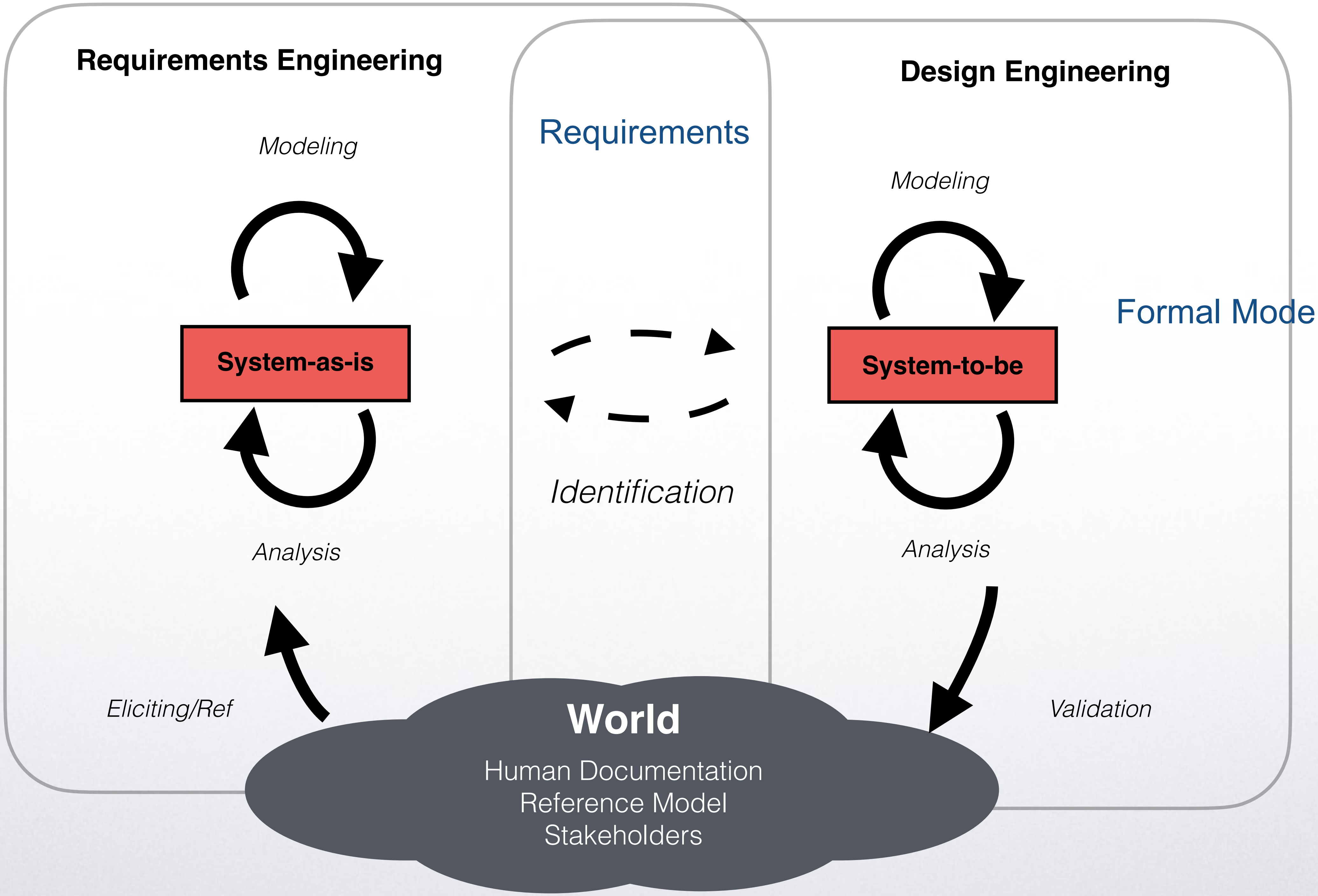
Systems Modeling





Requirements
Engineering/
Implementation
Design
Evolution
Cycle
System-to-be
Identification
Realization
World
System-as-is
(New) System
Innovation
Reference
Cycle
Satisfaction
Cycle


Requirements Engineering
Design Engineering
Engineering/
Implementation
Modeling
System-as-is
Analysis
Identification
Validation
Eliciting/Ref
Modeling
System-to-be
Analysis
Eliciting/Ref
Validation
Realization
Experimentation
Deployment
(New) System
Monitoring
Verification
World
Human Documentation
Reference Model
Stakeholders



Requirements Engineering
Modeling
System-as-is
Analysis
Eliciting/Ref
Requirements
Identification
Design Engineering
Modeling
System-to-be
Formal Model
Analysis
Validation
World
Human Documentation
Reference Model
Stakeholders

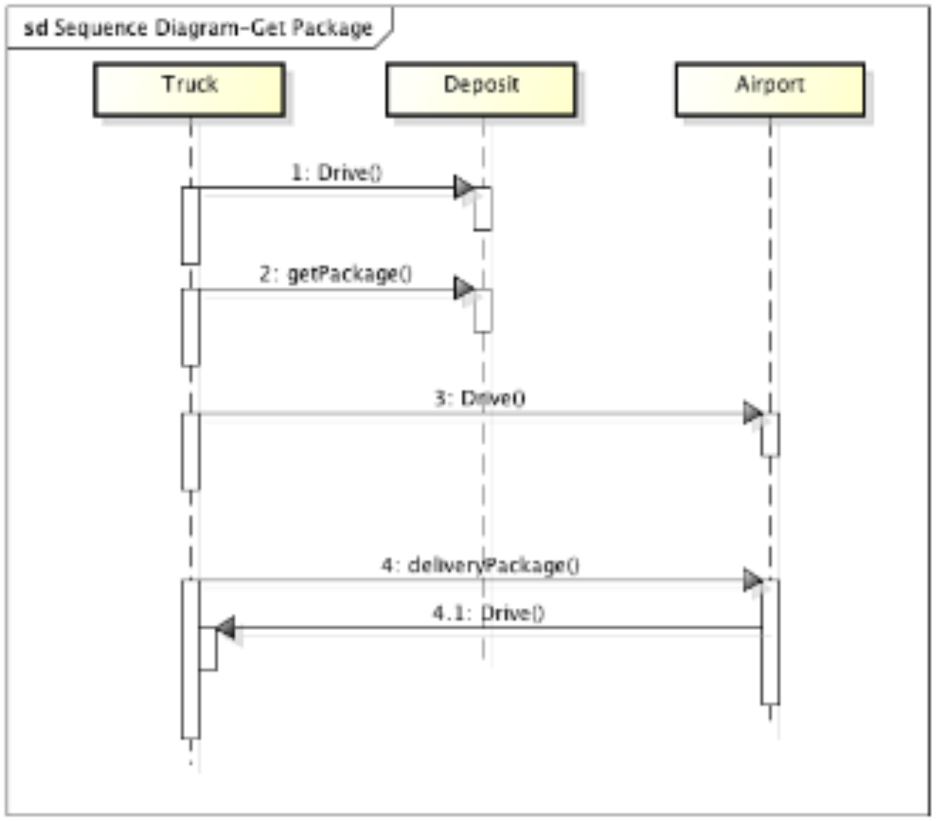
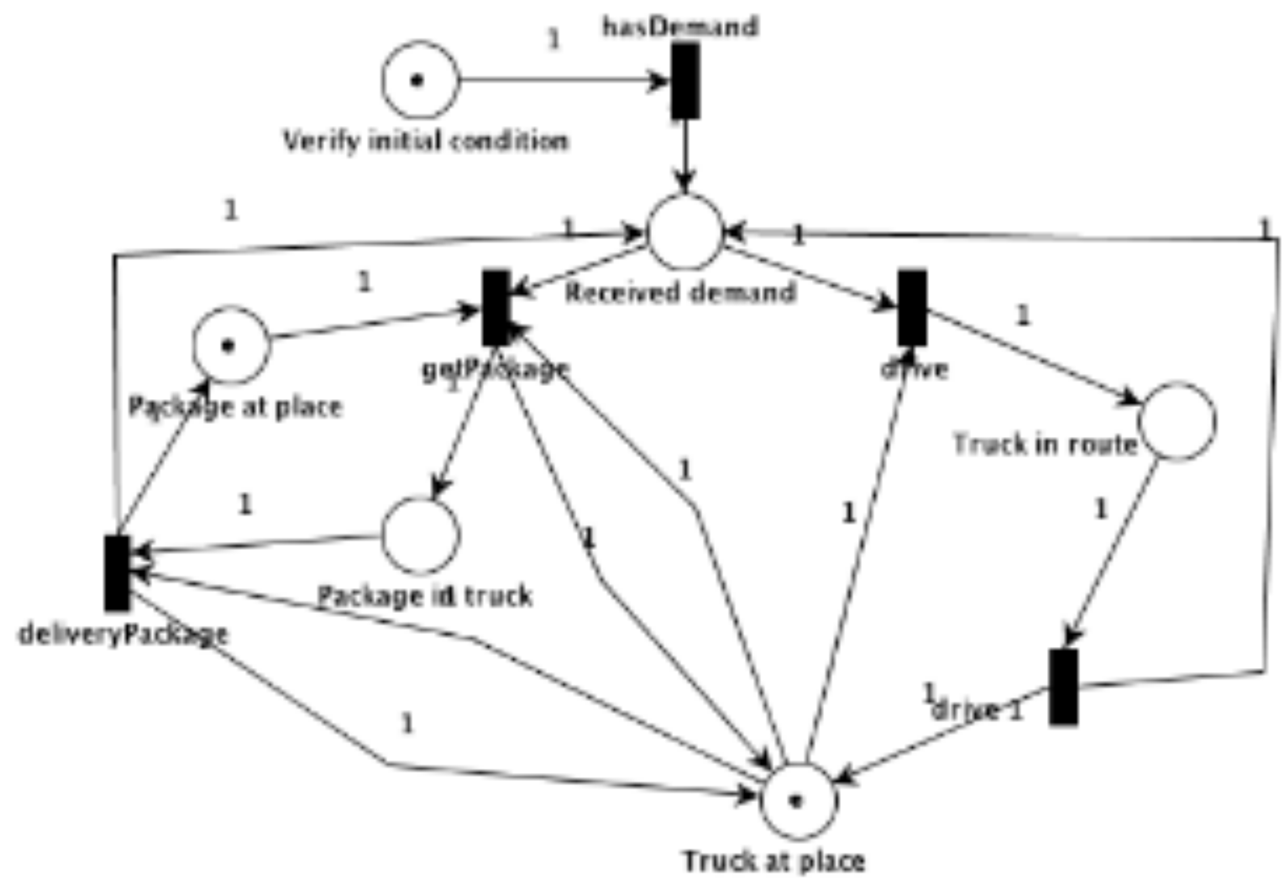
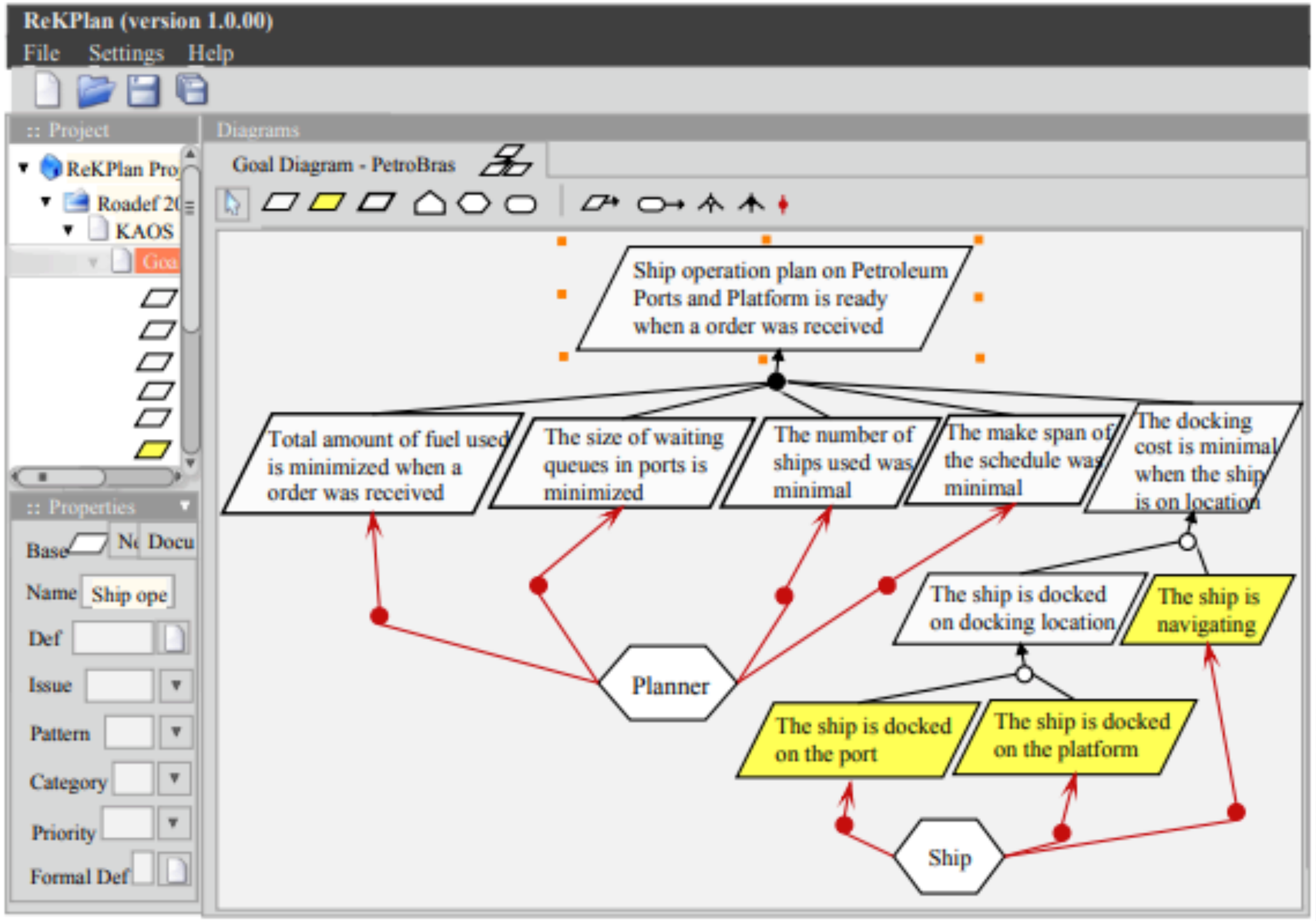
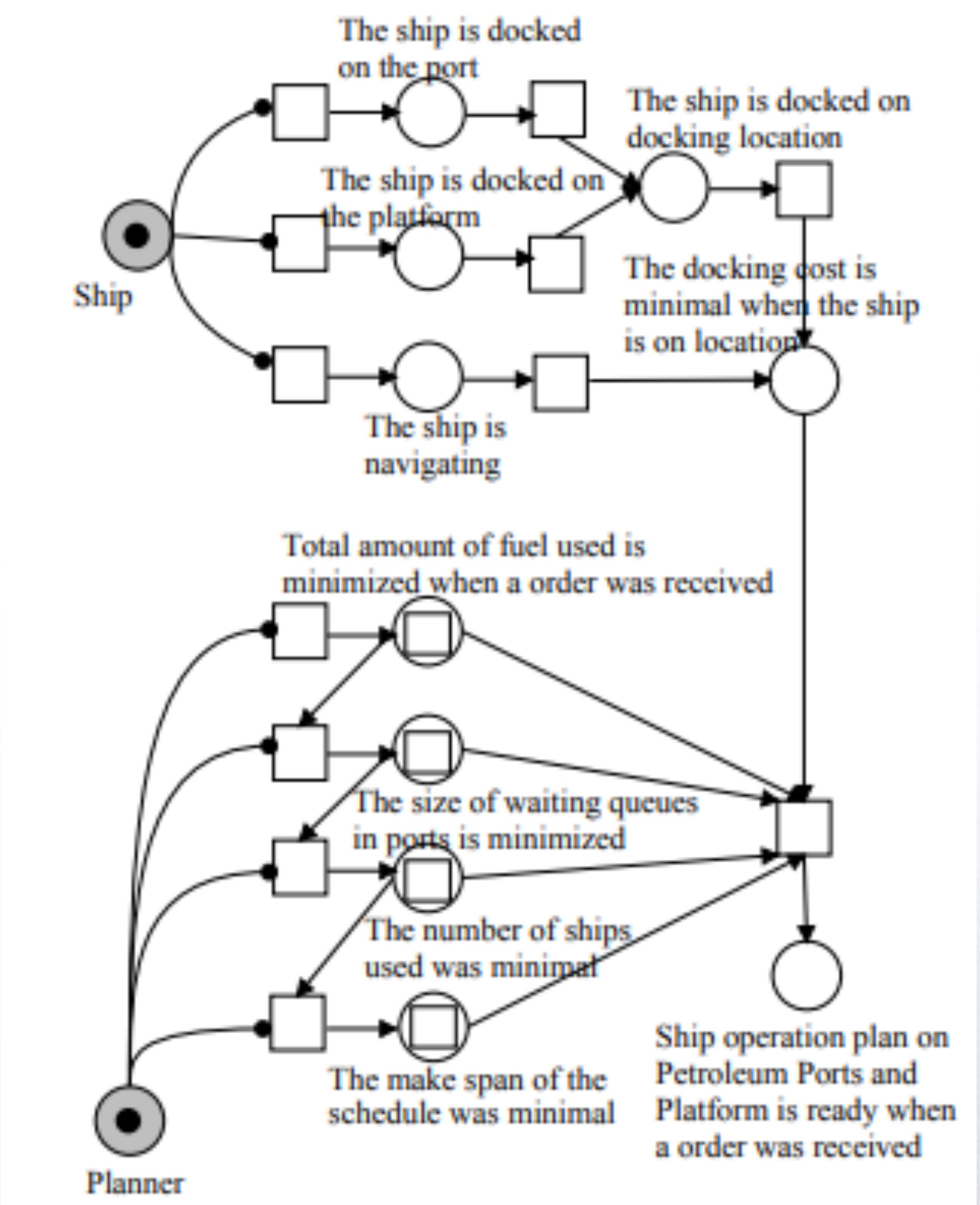


Figure 6: Sequence Diagram designed to represent the planning problem.

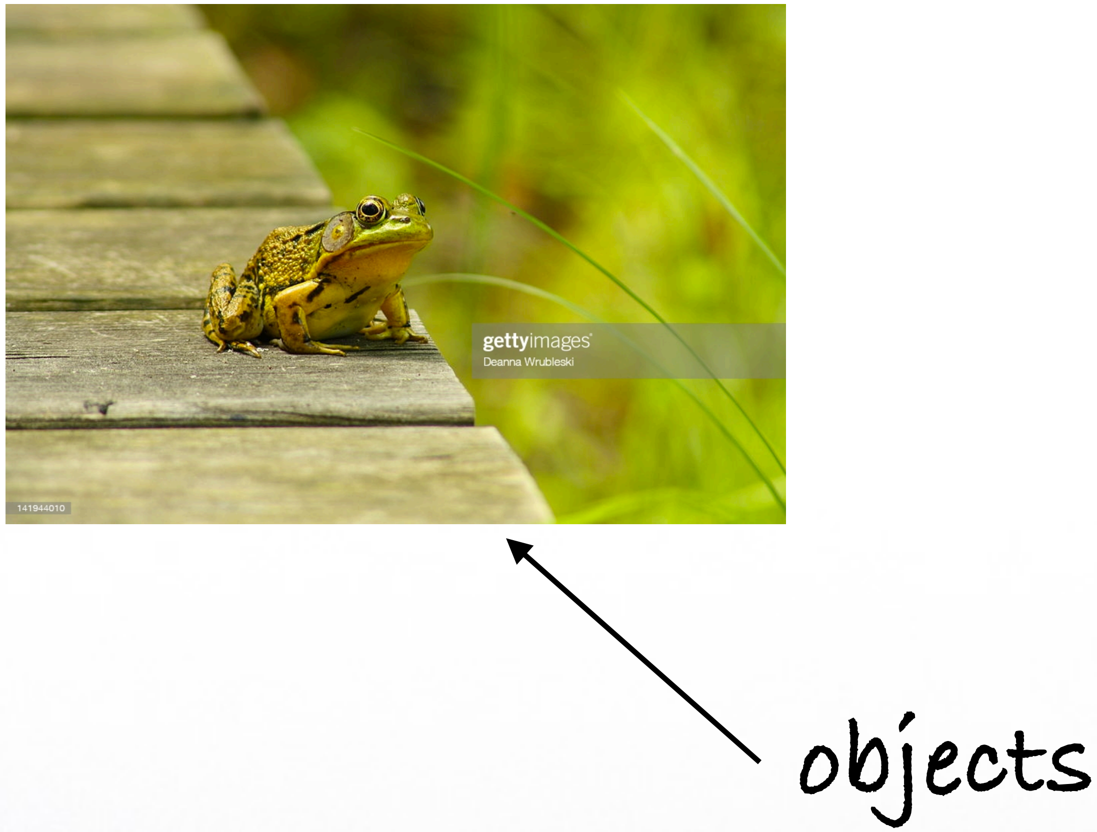


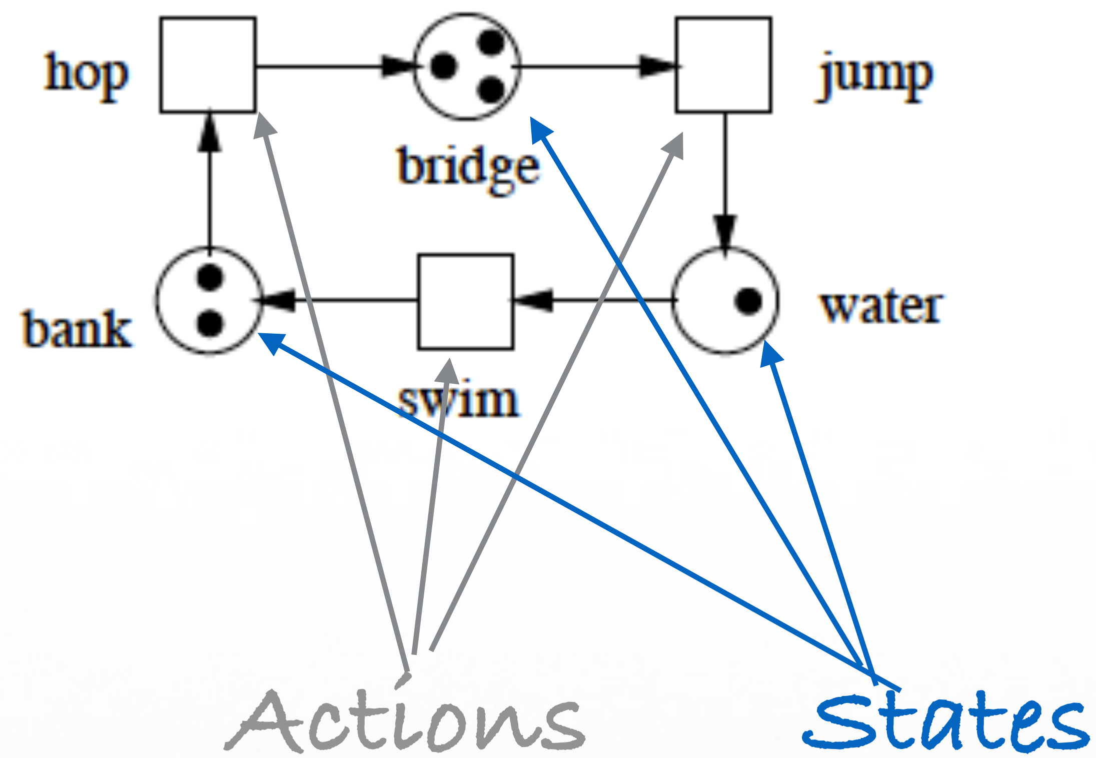


ReKPlan (version 1.0.00)
File Settings Help
:: Project
Diagrams
Goal Diagram - PetroBras
KAOS
:: Properties
Base
Name Ship ope
Def
Issue
Pattern
Category
Priority
Formal Def
Ship operation plan on Petroleum Ports and Platform is ready when a order was received
Total amount of fuel used is minimized when a order was received
The size of waiting queues in ports is minimized
The number of ships used was minimal
The make span of the schedule was minimal
The docking cost is minimal when the ship is on location
The ship is docked on docking location
The ship is navigating
Planner
The ship is docked on the port
The ship is docked on the platform
Ship



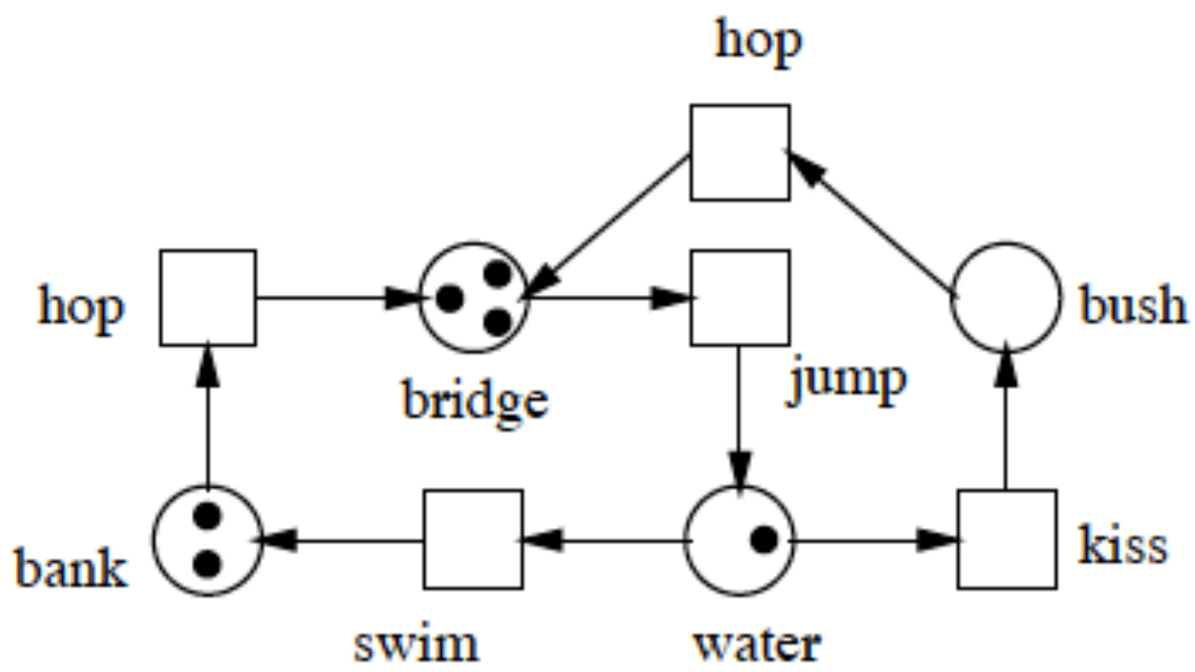
The ship is docked on the port
The ship is docked on docking location
The ship is docked on the platform
Ship
The docking cost is minimal when the ship is on location
The ship is navigating
Total amount of fuel used is minimized when a order was received
The size of waiting queues in ports is minimized
The number of ships used was minimal
The make span of the schedule was minimal
Ship operation plan on Petroleum Ports and Platform is ready when a order was received
Planner





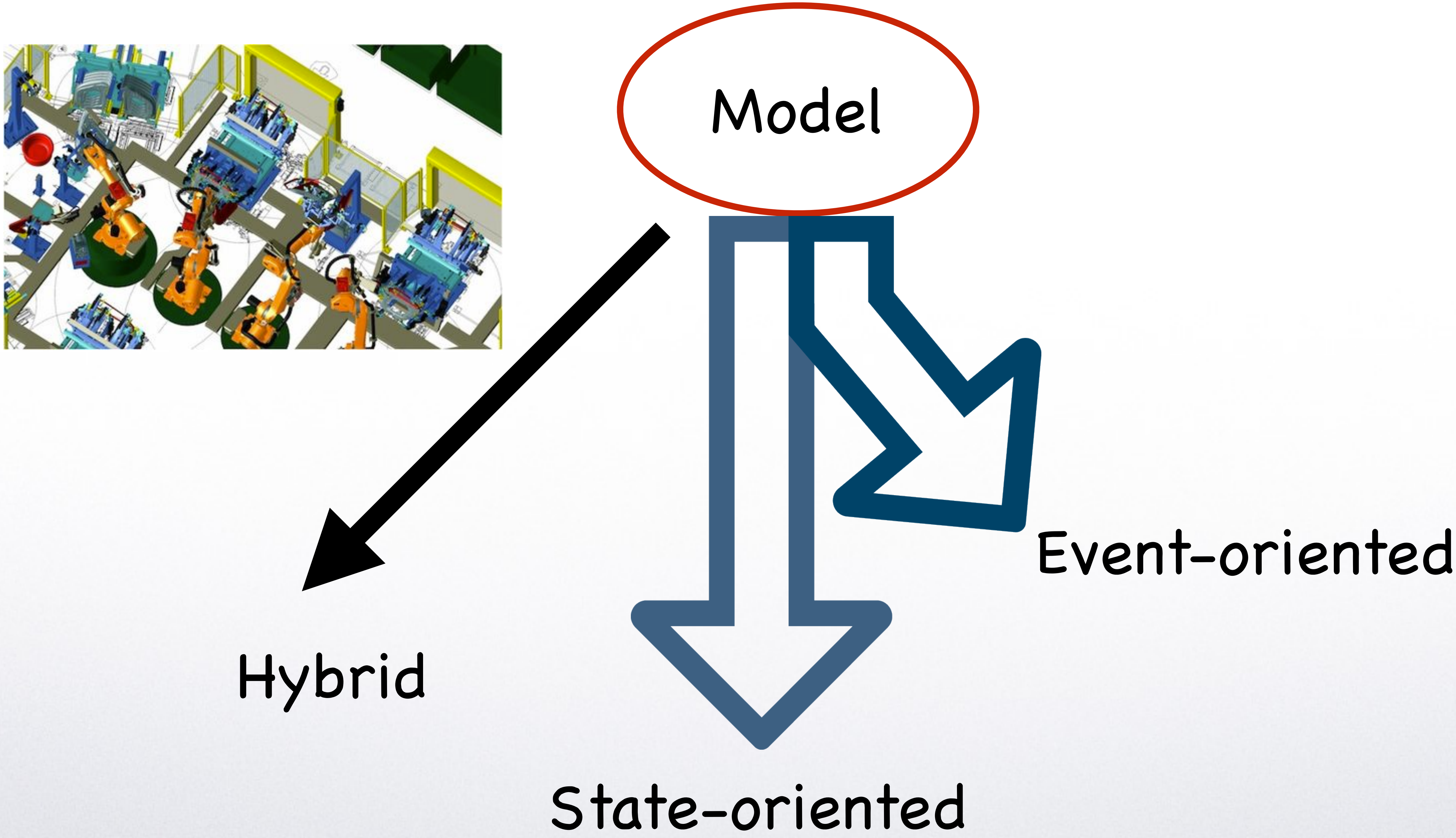
Here the tokens represent frogs that amuse themselves with jumping into a stream from a bridge, swimming to the bank and then hopping back to the bridge and starting all over again. Clearly, the frogs are the objects with three possible states and three actions that alter the state in a fixed order.

Incremental development



We can complicate the frog model by adding a beautiful girl who sometimes catches a frog that jumped from the bridge and kisses it. When the frog fails to become a prince, she disappointedly throws it into a nearby bush. The frog then hops back to the bridge to resume its play.

In this simple and intuitive system, objects (frogs and the princess) are not "created" dynamically or destroyed, they are both system resources. Therefore, there is a p-flow associated with that. Any available resource agent belongs either to the class "frog" or to the class "princess".


Model
Event-oriented
Hybrid
State-oriented

Structured approach

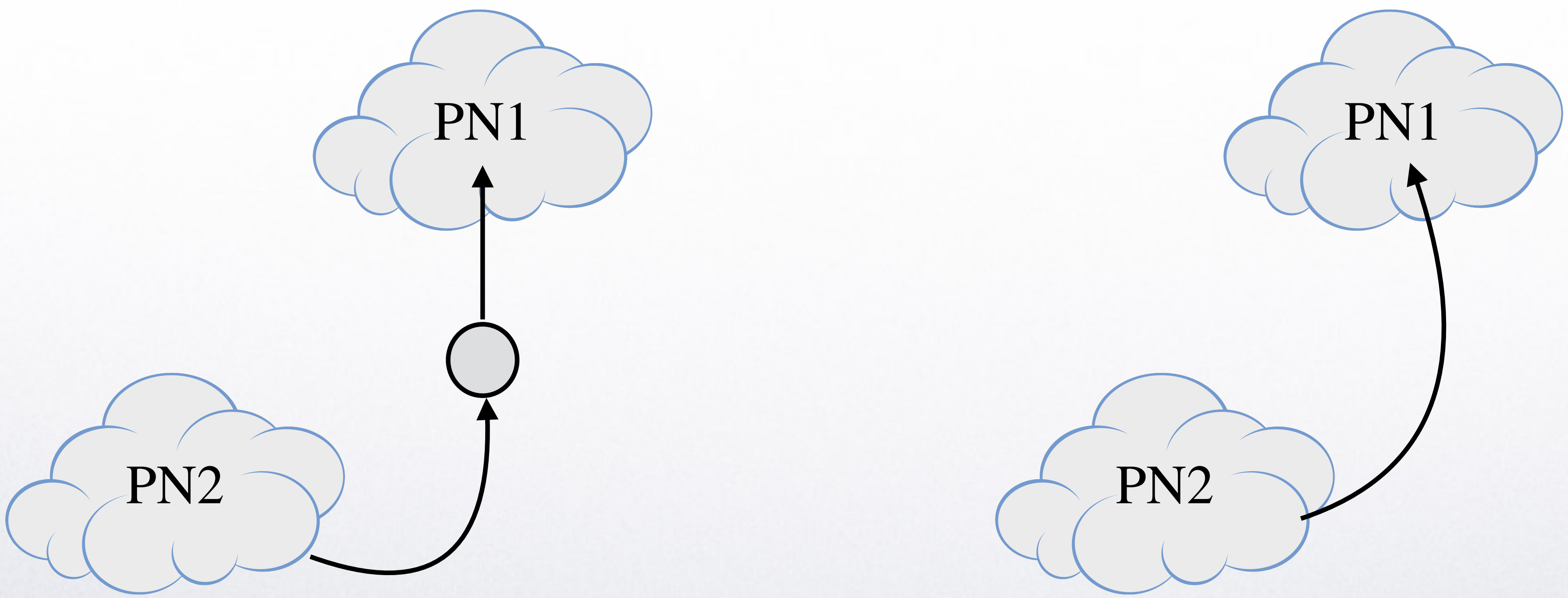
Reusability,
Based on compositionality

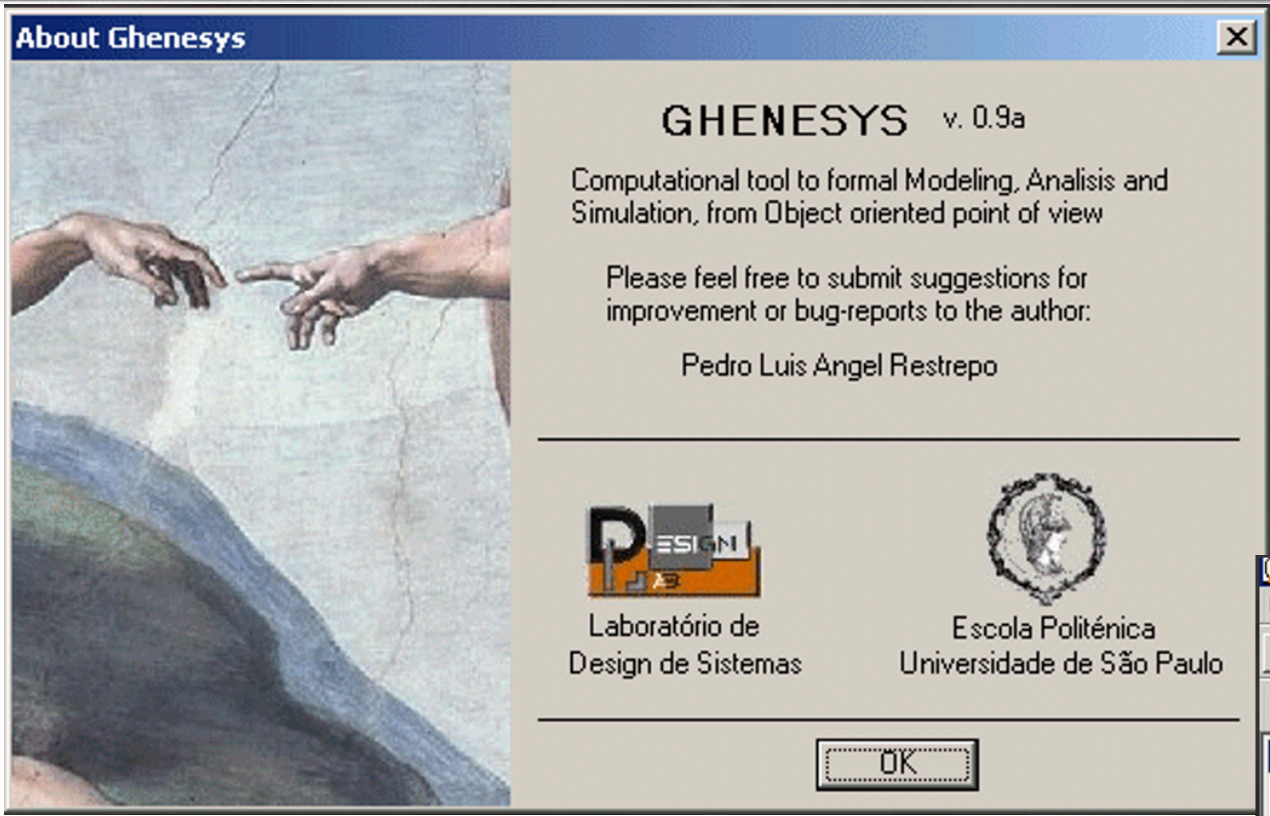
Systemic,
Based on refinemets

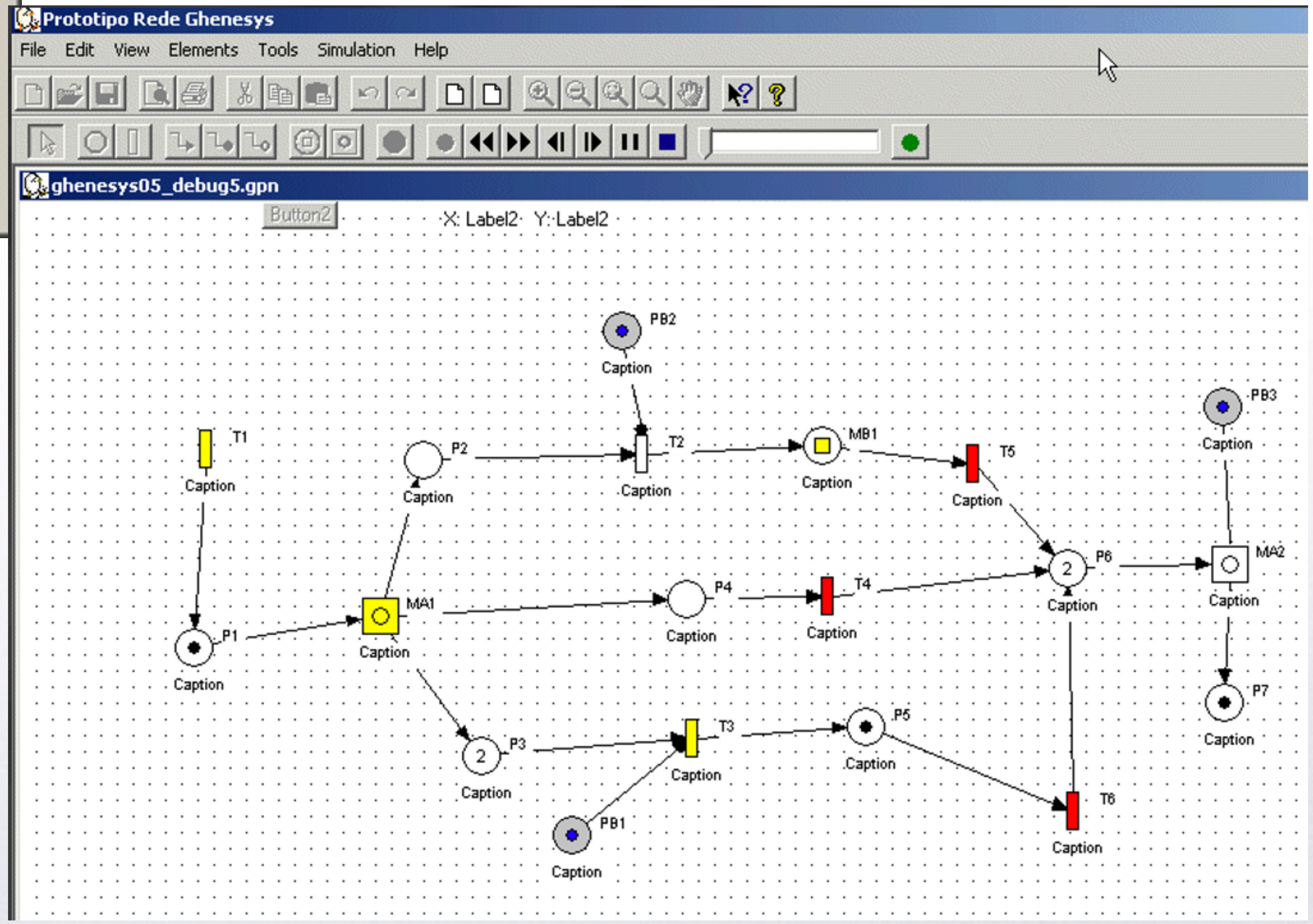
Components - preferably SCC's - can be developed altogether with the system, by refinements, or can be developed separately and joined to the general model. In such a case, we have systems and sub-systems that communicate with each other.

Communication should be synchronous or asynchronous...

... asynchronous communication must be represented by place fusion or arc addition.




About Ghenesys
GHENESYS v. 0.9a
Computational tool to formal Modeling, Analisis and Simulation, from Object oriented point of view
Please feel free to submit suggestions for improvement or bug-reports to the author:
Pedro Luis Angel Restrepo
Laboratório de Design de Sistemas
Escola Politécnica Universidade de São Paulo
OK


Prototipo Rede Ghenesys
File Edit View Elements Tools Simulation Help
ghenesys05_debug5.gpn
Button2
X: Label2 Y: Label2
PB2
PB3
T1
P2
T2
MB1
T5
MA2
P6
P4
T4
MA1
P1
P7
P3
T3
P5
T6
PB1
Caption

We define a context matrix D as a diagonal matrix which brace is given by a vector such as,

$$[d]_{1j} = \begin{cases} 1, \text{ if j is a box} \\ 0 \text{ . if j is a pseudobox} \end{cases}$$

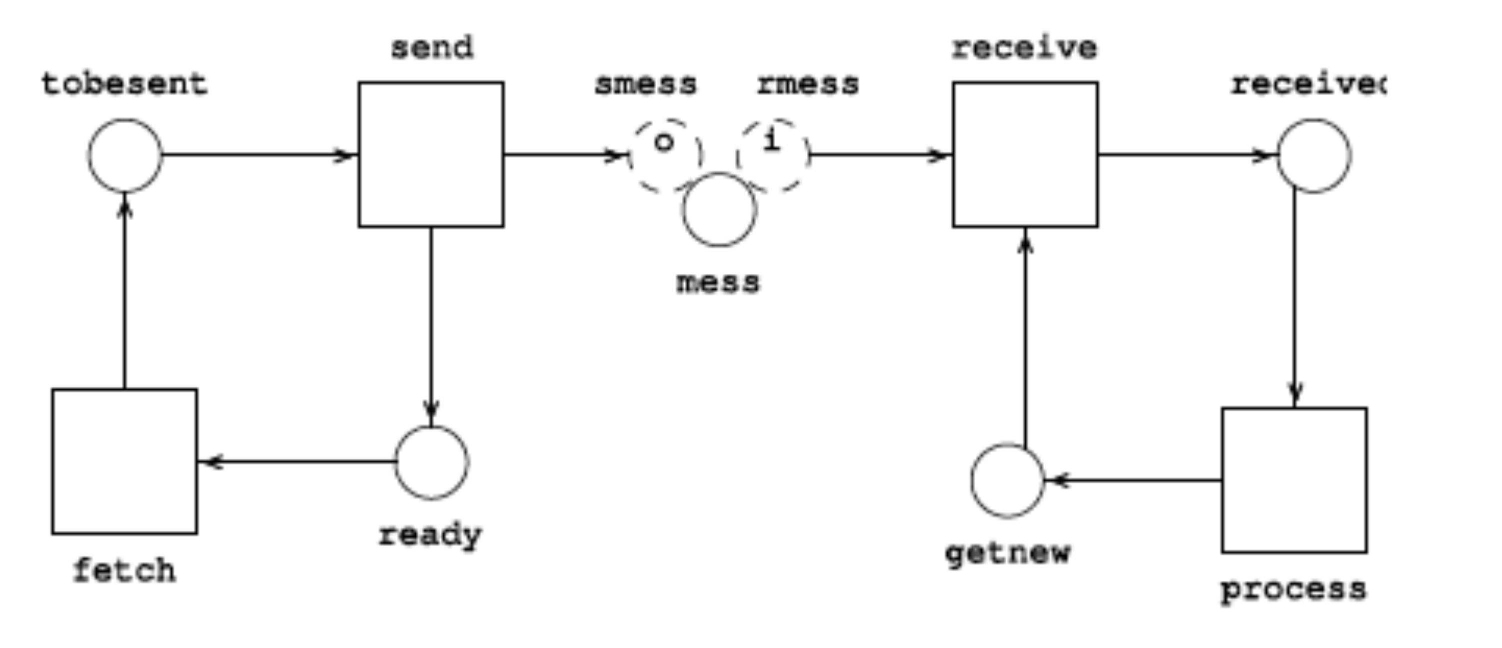
d = [1 1 1 1 1 ...0 0]

the new state equation for the net is given by

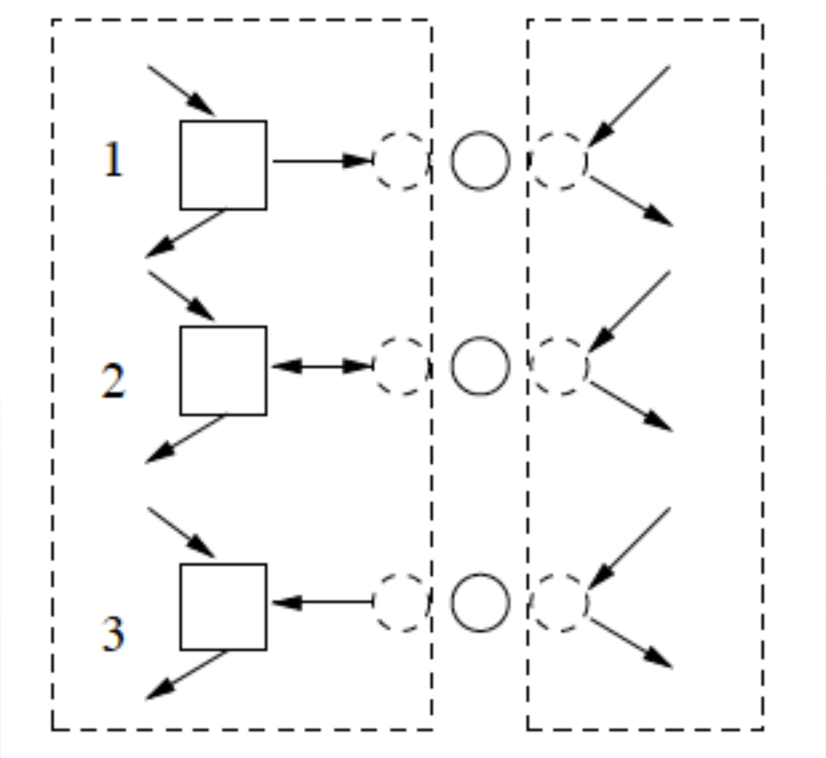
$$\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{DC}v_k$$

which is quite similar to the classical state equation

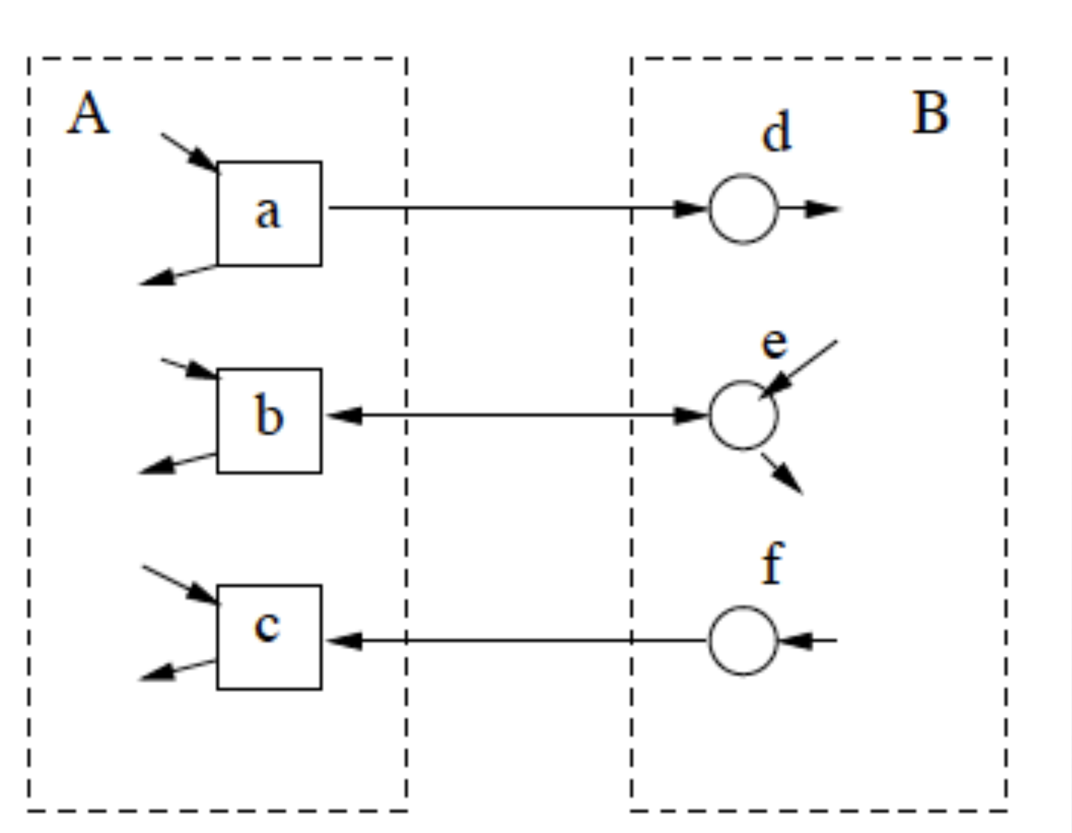
In the canonical definition fusion places (smess, rmess) belong to different components. When the components are matched the corresponding fusion places are collapsed (fusion).



Os importantes são: i) quando uma rede coloca marcas em um lugar que é retirado pela outra (channel, caso 1), ii) onde o fusion place (pin) pode desabilitar uma transição na outra rede (2); iii) quando ambas as as redes podem usar o token do fusion place (3) causando possível mutex.



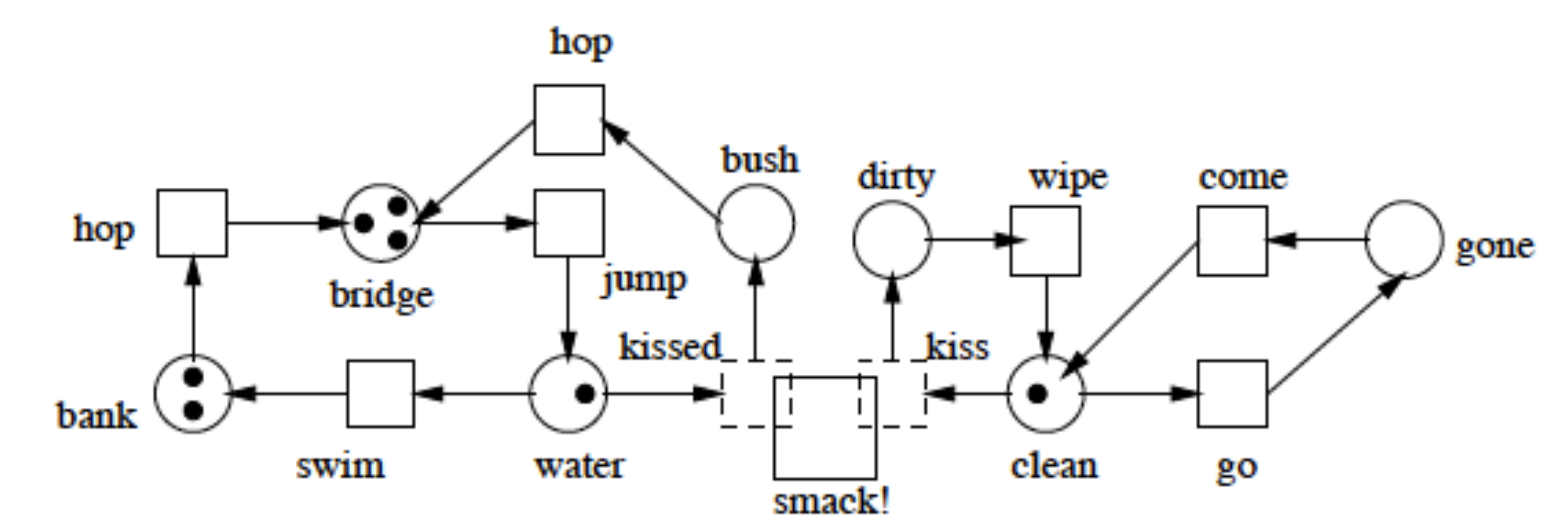
Outra forma de comunicação assíncrona é o arc addition uma transição em uma rede é ligada a um lugar da outra (sem interrupção de fluxo) - input addition - ou ao contrário - output addition. Uma terceira possibilidade é com combinar input e output in a I/O addition.



Comunicação síncrona: transition fusion

No caso anterior a ativação ocorre em duas fases: uma das redes produz um token em um fusion place e a outra pode retirar depois. Na comunicação síncrona fundiremos transições, assim, se uma fusion transition está habilitada a ocorrência promove fluxo em ambas as redes causando o sincronismo.

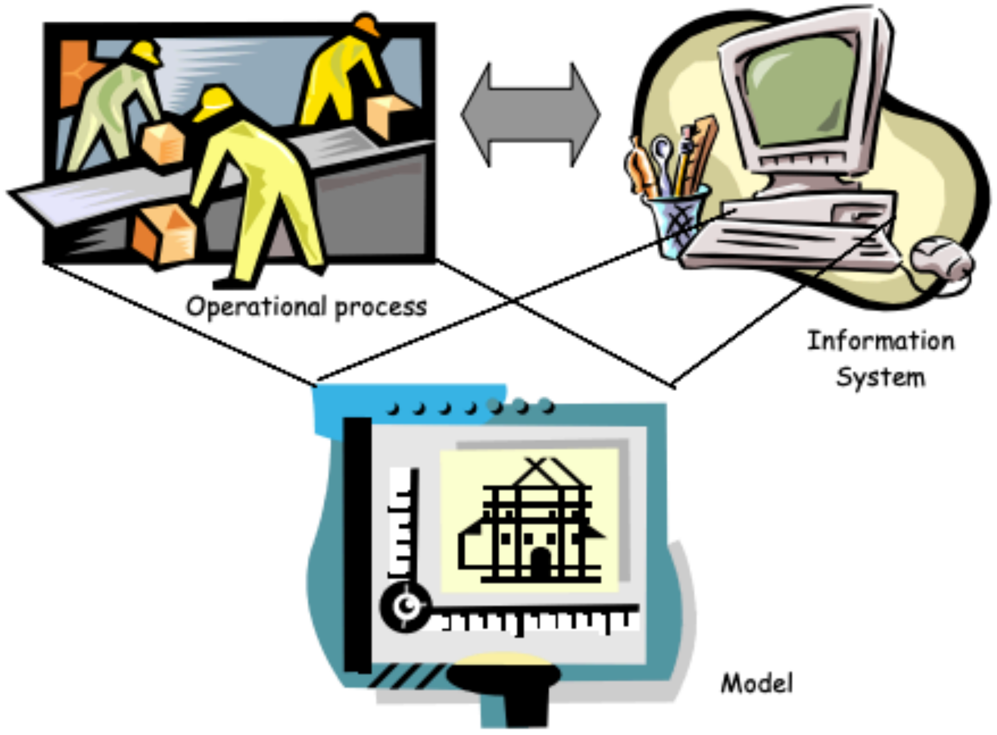
Em uma versão revisada podemos modelar o ciclo dos sapos e da princesa como componentes distintas, sincronizadas na ação de "beijar o sapo" para eventualmente se transformar em príncipe. A princesa pode ainda se retirar do lago, deixando os sapos brincando em paz!



O objetivo do processo de síntese é produzir um modelo que será analisado por simulação e/ou por análise de propriedades (usando o modelo P/T ou de redes de alto nível ou coloridas).

Wil van der Aalst

Conclusion analysis



- Analysis is typically model-driven to allow e.g. what-if questions.
- Models of both operational processes and/or the information systems can be analyzed.
- Types of analysis:
  - *validation (interactive simulation/gaming)*
  - *verification (state-space analysis, place and transition invariants, siphons, traps, etc.)*
  - *performance analysis (simulation)*

Na próxima aula vamos ver como aplicar a técnica de building blocks nas redes de alto nível e seguiremos na discussão dos métodos de modelagem de sistemas usando Redes de Petri.