PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 10: Voltando aos métodos de modelagem

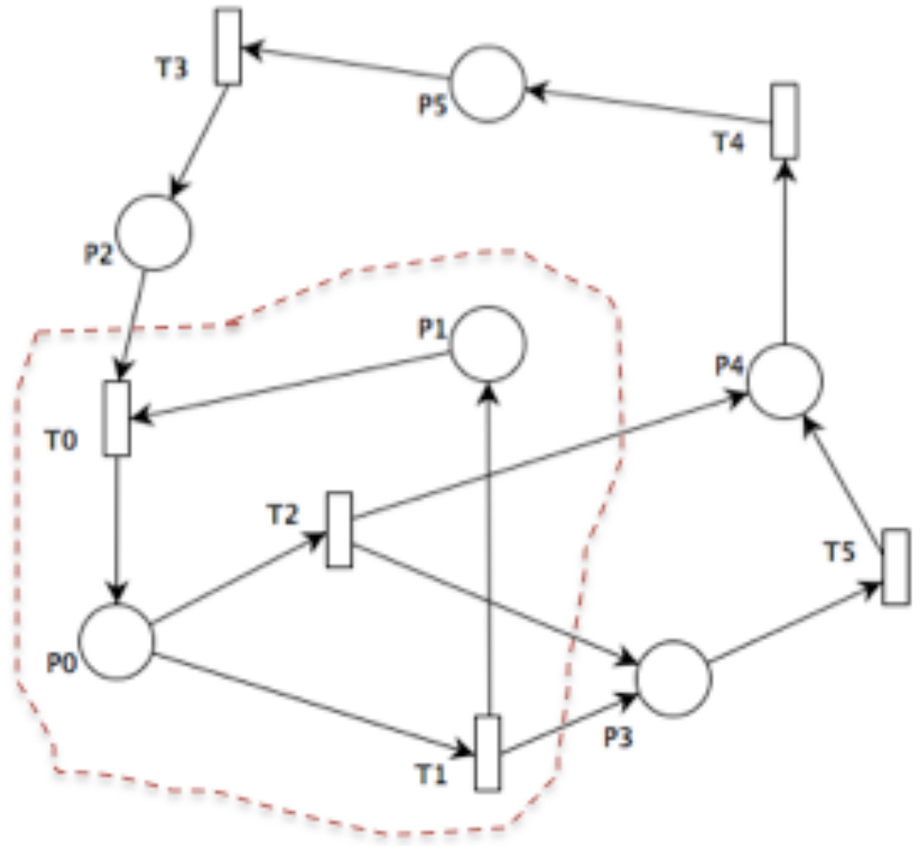
Prof. José Reinaldo Silva
reinaldo@poli.usp.br

Hierarchy

Hierarchy is not anything new and is actually connected with any kind of net, including the classical ones.

In design, hierarchy means to abstract the elements which properties are not relevant in an analysis phases.

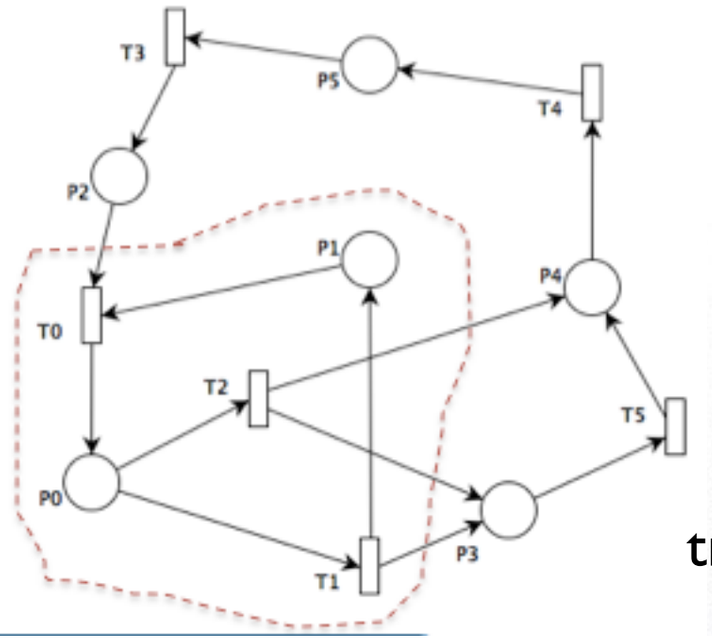
Hierarquia em redes clássicas

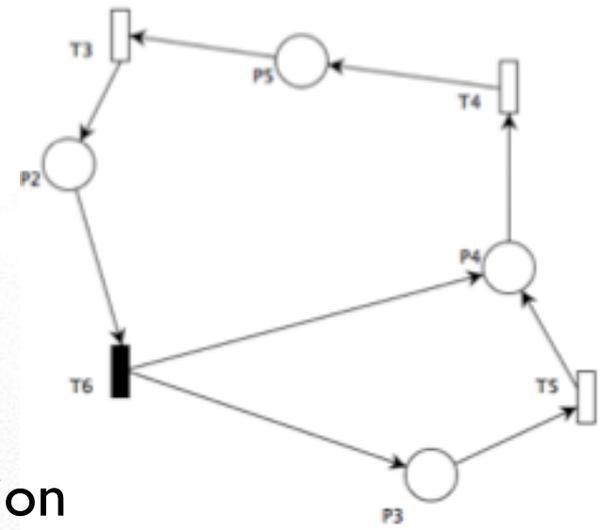


Definition 39

Seja uma estrutura de rede $N = (S, T; F)$. Seja $X = S \cup T$ e um sub-cojunto $Y \subseteq X$. Definimos uma borda de N como o conjunto $\partial(N) = \{y \in Y \,|\, \exists x \notin Y . x \in loc(y)\}$.

Substituição de uma sub-rede

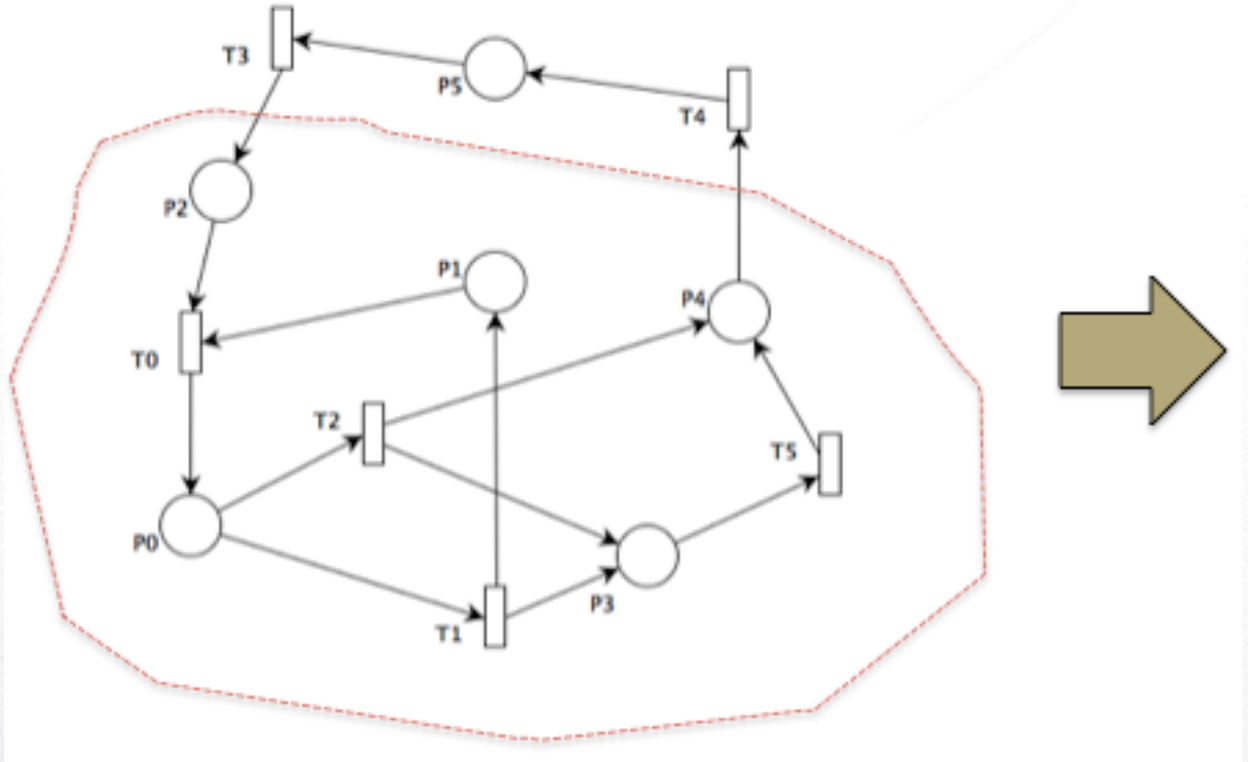


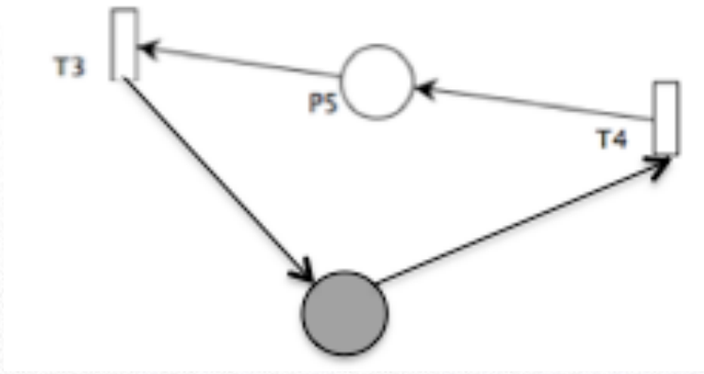


transition bounded substitution

Definition 40

Um sub-conjunto de elementos Y da rede $N = (S, T; F)$ é dito limitado por lugar (place bounded) ou aberto, se e somente se $\partial(Y) \subseteq S$. Similarmente, um sub-conjunto Y desta rede é dito limitado por transição (transition bounded), se e somente se $\partial(Y) \subseteq T$.





place bounded substitution

Se em uma rede com estrutura $N = (S, T; F)$ existe uma sub-rede Y limitada por transição, a substituição desta sub-rede Y gera uma rede $N' = (S', T'; F')$ onde:

(i) $S' = S \setminus Y$;

(ii) $T' = T \setminus Y \cup \{t_y\}$, onde t_y é o novo elemento que substitui Y;

(iii) $F' = F \setminus Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

.

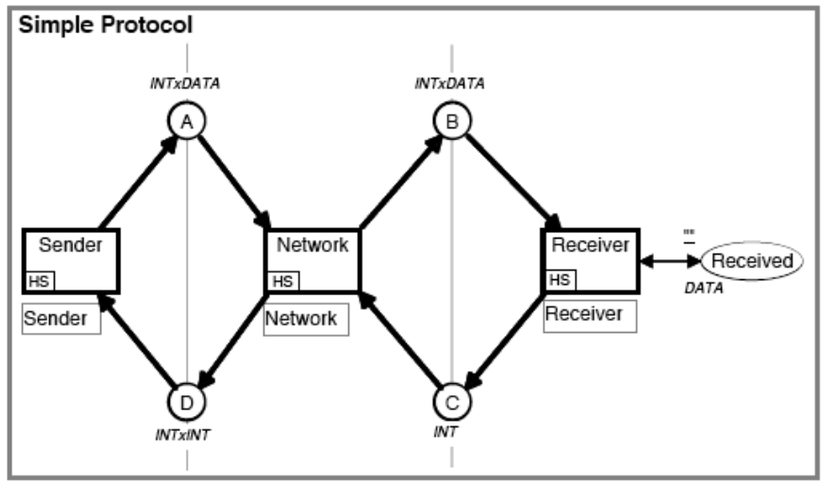
Similarmente, se a sub-rede Y é limitada por lugar,

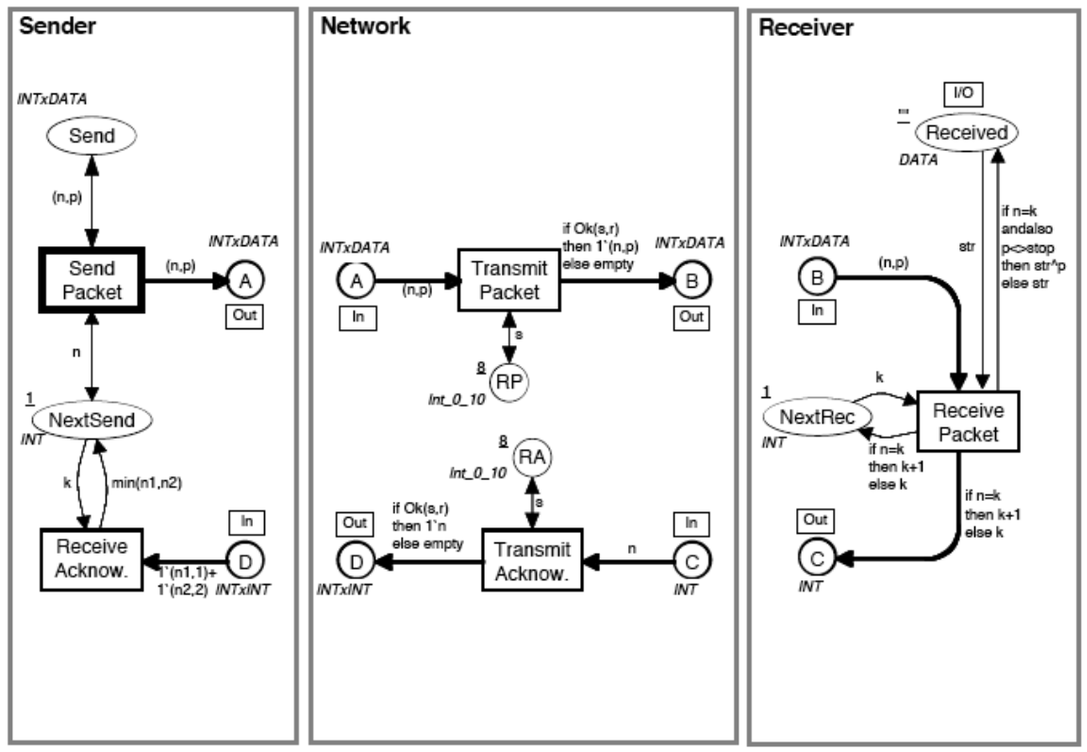
(i) $S' = S \setminus Y \cup \{s_y\}$, onde s_y é o novo elemento que substitui Y;

(ii) $T' = T \setminus Y$;

(iii) $F' = F \setminus Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

A hierarchical CP-net contains a number of *interrelated subnets*– called *pages*.





A page may contain one ore more *substitution transitions*.

- Each substitution transition is related to a *page*, i.e., a *subnet* providing a *more detailed description* than the transition itself.
- The page is a *subpage* of the substitution transition.

There is a *well-defined interface* between a substitution transition and its subpage:

- The places surrounding the substitution transition are *socket places*.
- The subpage contains a number of *port places*.
- Socket places are *related* to port places – in a similar way as actual parameters are related to formal parameters in a procedure call.
- A socket place has always the *same marking* as the related port place. The two places are just *different views* of the same place.

Substitution transitions work in a similar way as the refinement primitives found in many system description languages – e.g., SADT diagrams.

Fundamentos do método estruturado

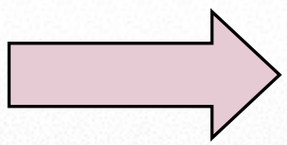
Um ***bloco*** é um conjunto genérico de instruções de programa, onde uma dada instrução é identificada como a entrada do bloco e outra (diferente da primeira) é identificada como a saída.

Se A e B são blocos de um mesmo programa, então A e B são ditos independentes se e somente se $A \cap B = \varnothing$.

Se A e B são tais que $A \cap B \neq \varnothing$ então $(A \subseteq B)$ OU $(A \subseteq B)$

Constituintes próprios e primos

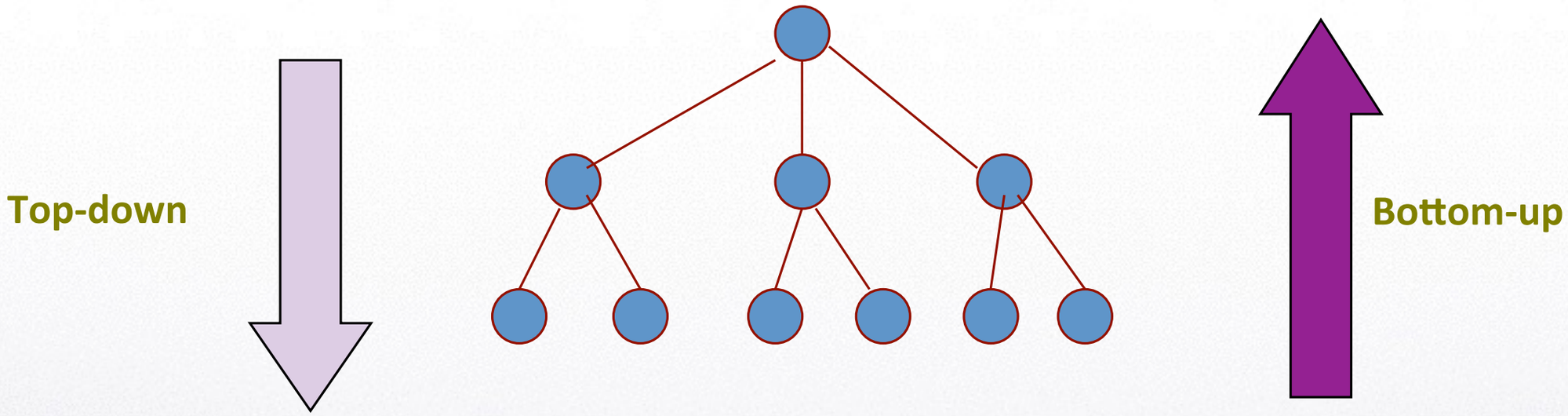
Elementos próprios



Base de elementos constituintes (componentes)

Elementos próprios indivisíveis são chamados primos. Um conjunto LI de elementos primos pode constituir uma base e portanto pode descrever qualquer programa.

Decomposição por refinamentos: o método estruturado



Overview

Apresentamos as redes de Petri como um esquema e uma representação formal para a modelagem e análise de sistemas e processos discretos, pertinentes a uma larga gama de domínios. Em particular, quase 70% dos sistemas automatizados acabam caindo nesta categoria, e o futuro nos reserva ainda possibilidade de ampliação deste escopo com a difusão dos "sistemas de serviço".

O importante é no entanto a introdução do formalismo de redes de Petri, que como vimos pode ser dividido em dois grandes grupos: o das redes ditas clássicas; o das redes de alto nível; e o das redes estendidas.

Redes de Petri

Redes Clássicas

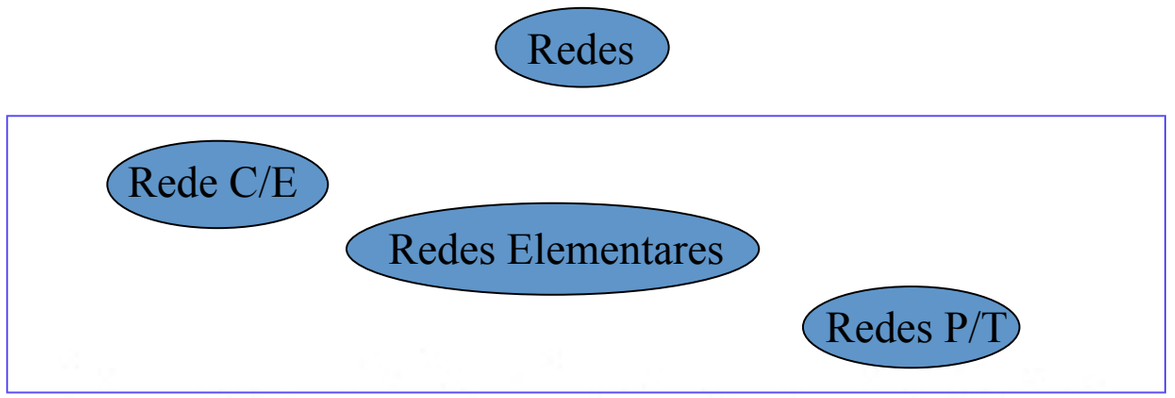
Redes P/T (seriam o padrão)

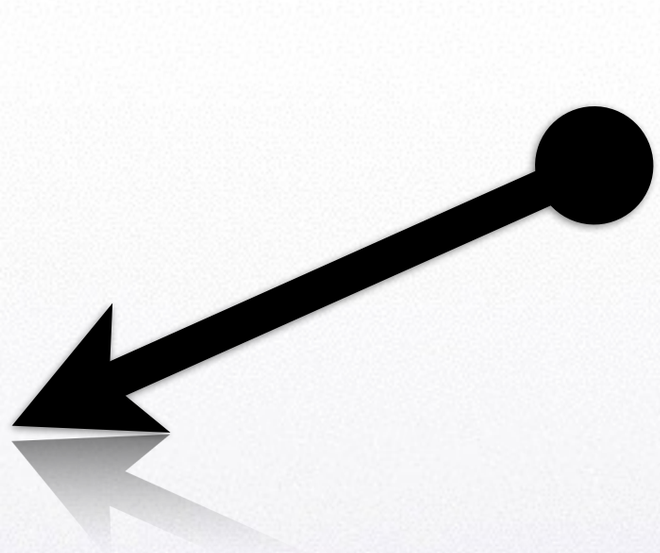
Redes de Alto Nível

(HLPNs, Redes Coloridas, etc)

Redes Estedidas

Redes com elementos estenditos (gates, pseudo-lugares, etc.)
Redes hierárquicas
Redes Orientadas a Objetos





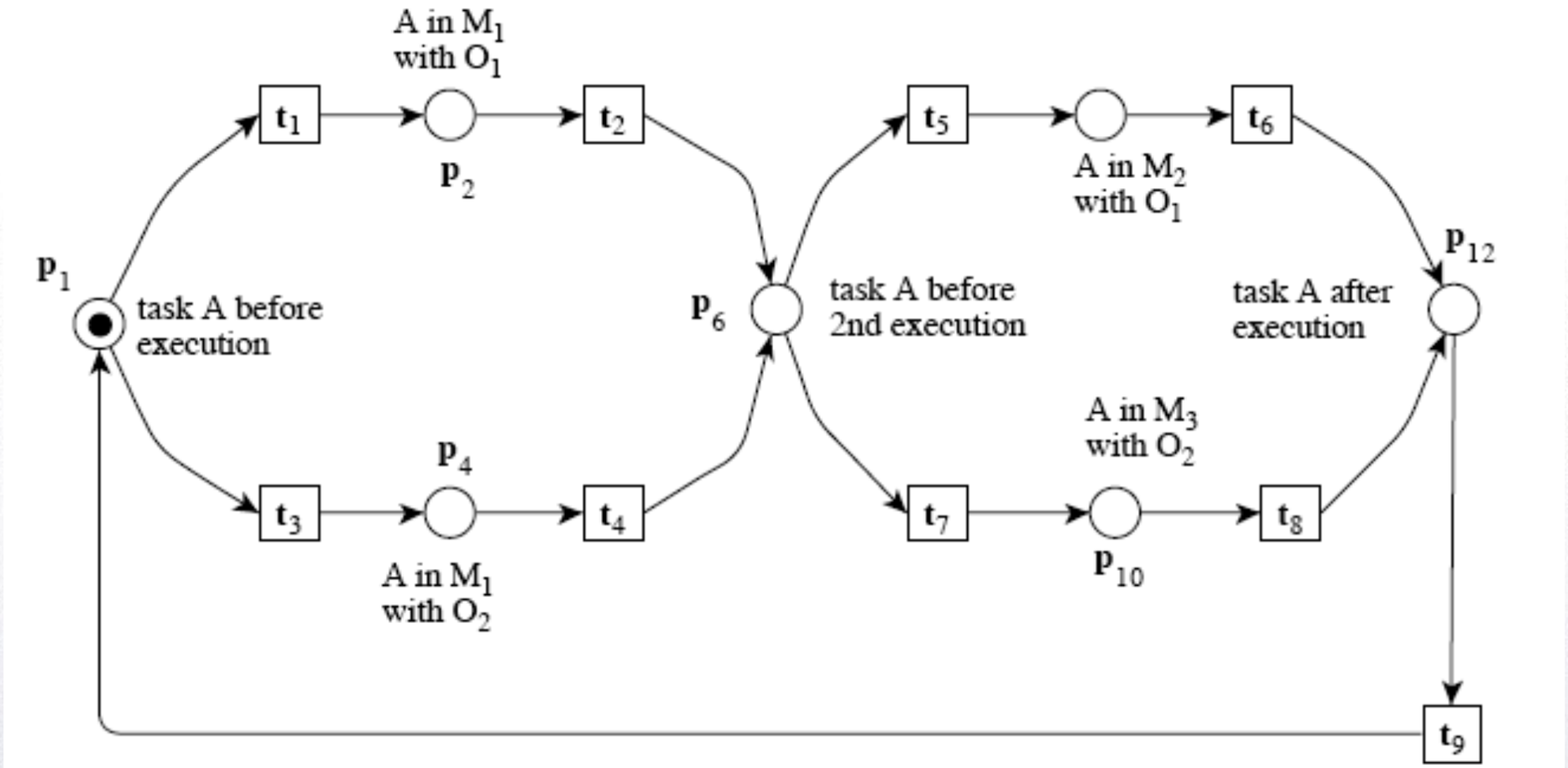
Modelagem de Sistemas Discretos

- Síntese da rede;
- Procedimentos de redução;
- Análise da rede (atingibilidade, deadlock, etc.);
- Simulação.

Procedimento de modelagem e análise

Requisitos do problema: *Seja um sistema de manufatura simples, composto de 3 máquinas DNC: M1, M2 e M3. Estas máquinas podem executar duas operações diferentes, O1 e O2, de modo que O1 pode ser executado nas máquinas M1 e M2 mas não simultaneamente. Similarmente, O2 pode ser executado em M1 e M3 mas não simultaneamente.*

Uma forma de tratar o problema é em primeiro lugar modelar a sequência de operações, sem levar em conta nenhuma restrição e nenhum recurso.



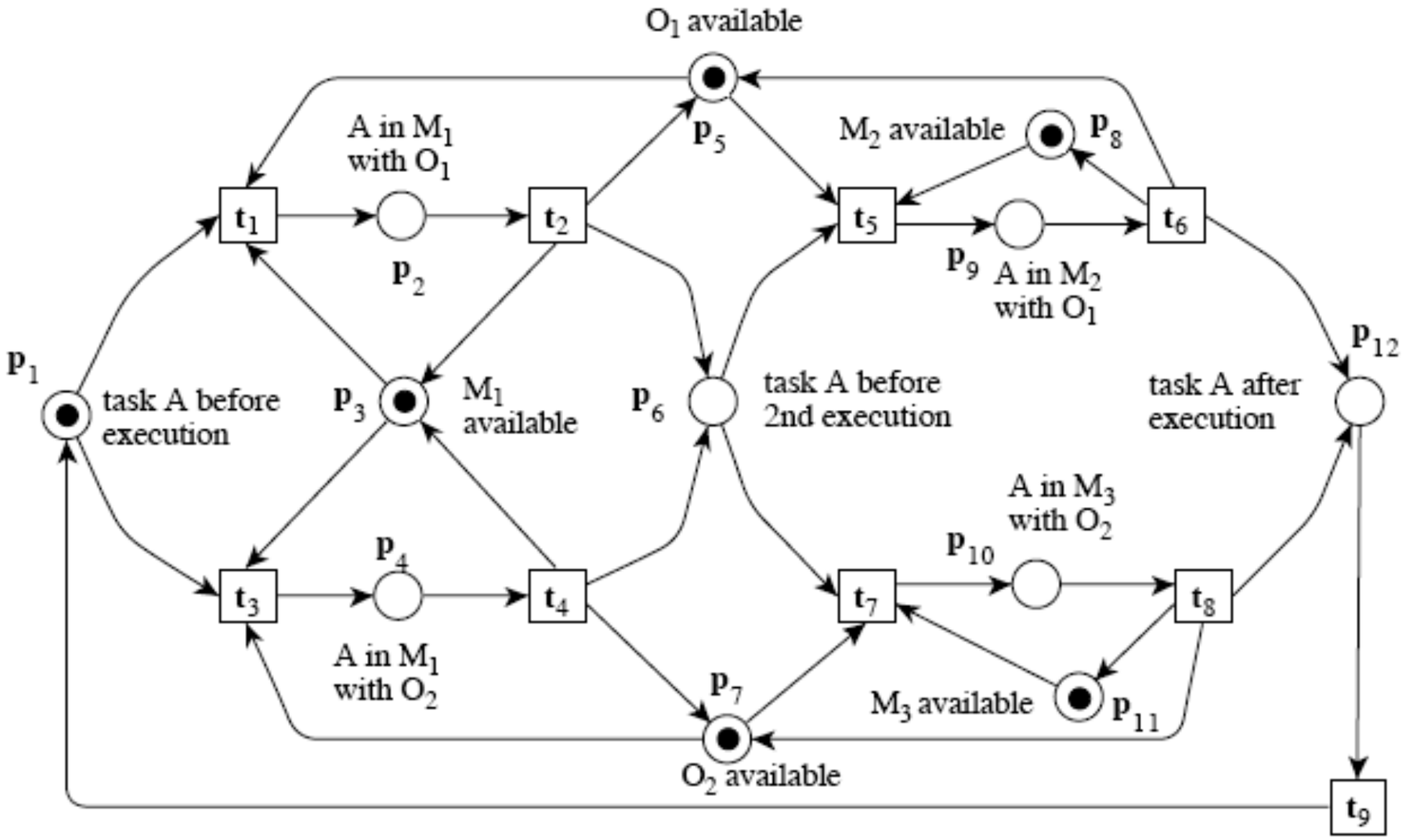
Girauld, C. and Valk, R.; Petri Nets for Systems Engineering, Springer-Verlag, 2003

Análise

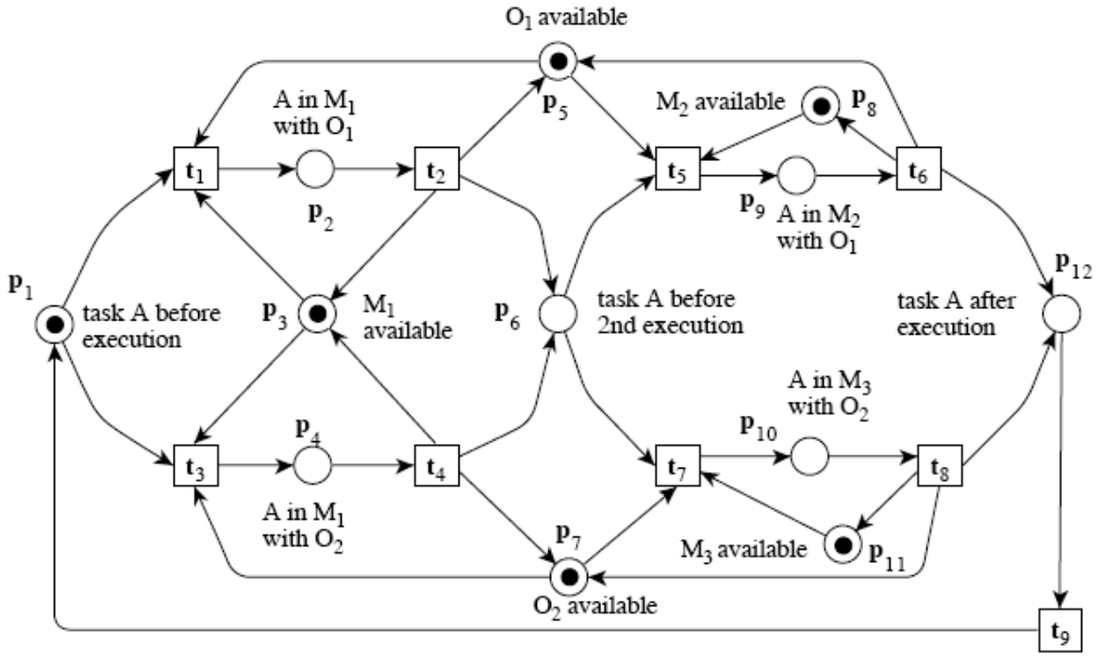
O sistema é cíclico, e permite a combinação das operações em qualquer ordem (e portanto o sistema estaria preparado para implementar qualquer receita de peça). O sistema é conservativo, de modo que cada peça seria representada por uma marca que deve estar em algum dos lugares já especificados.

Na especificação de requisitos, os recursos são representados pela disponibilidade das máquinas e pela sua capacidade de executar cada uma das operações. Uma vez feita a parte sequencial da rede devemos agora introduzir estas restrições, que devem alterar a rede ou a sucessão de estados desta.

Introduzindo os recursos, segundo a especificação de requisitos já apresentada, temos:



Análise de Invariantes



Análise de invariantes

Os invariantes podem ser analisados e servir como forma de verificação para o atendimento dos requisistos

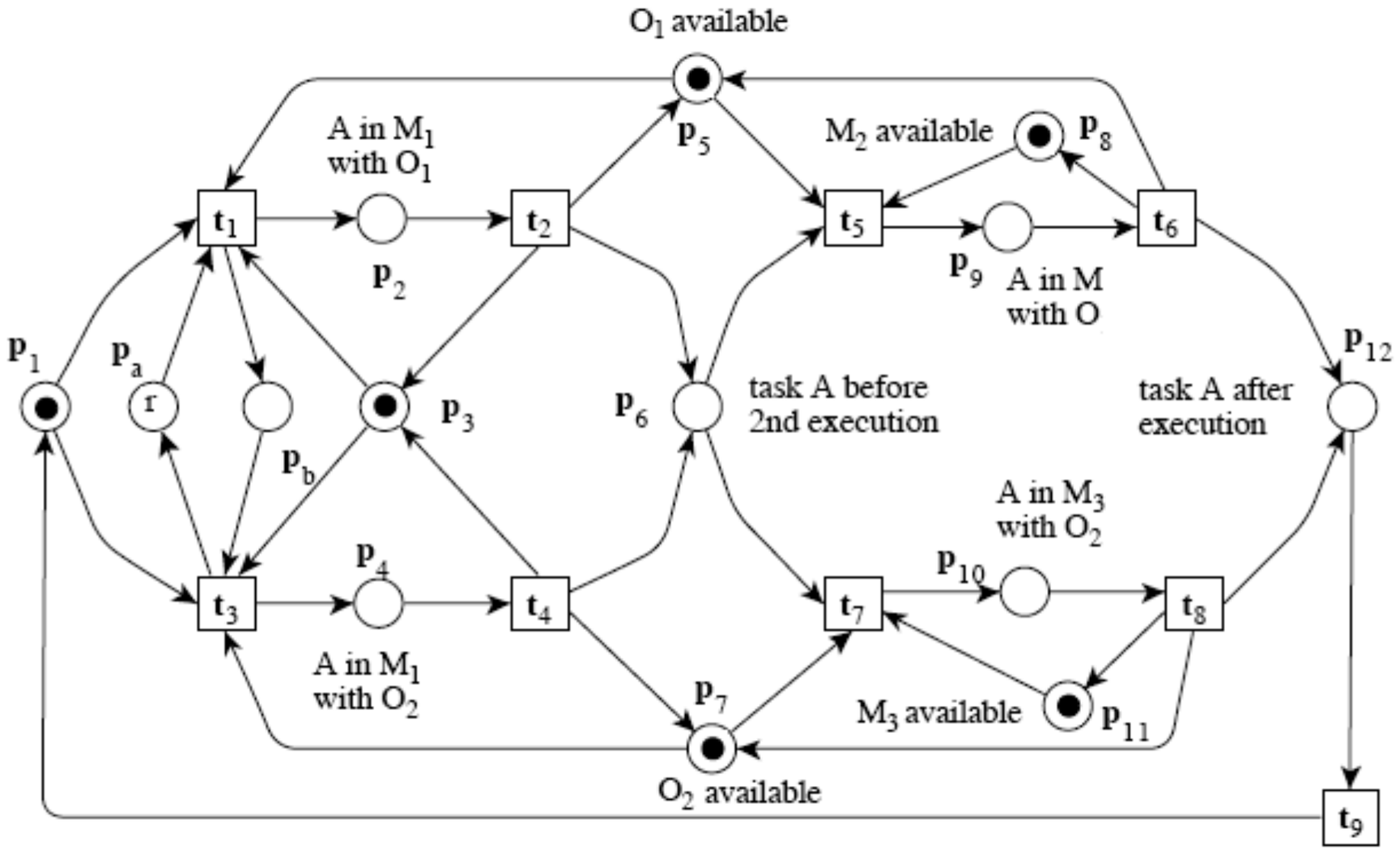
i) $m[p_2] + m[p_5] + m[p_9] = 1;$

ii) $m[p_4] + m[p_7] + m[p_{10}] = 1;$

iii) $m[p_2] + m[p_3] + m[p_4] = 1;$

iv) $m[p_1] + m[p_2] + m[p_4] + m[p_6] + m[p_9] + m[p_{10}] + m[p_{12}] = c$

Introduzindo Sincronização

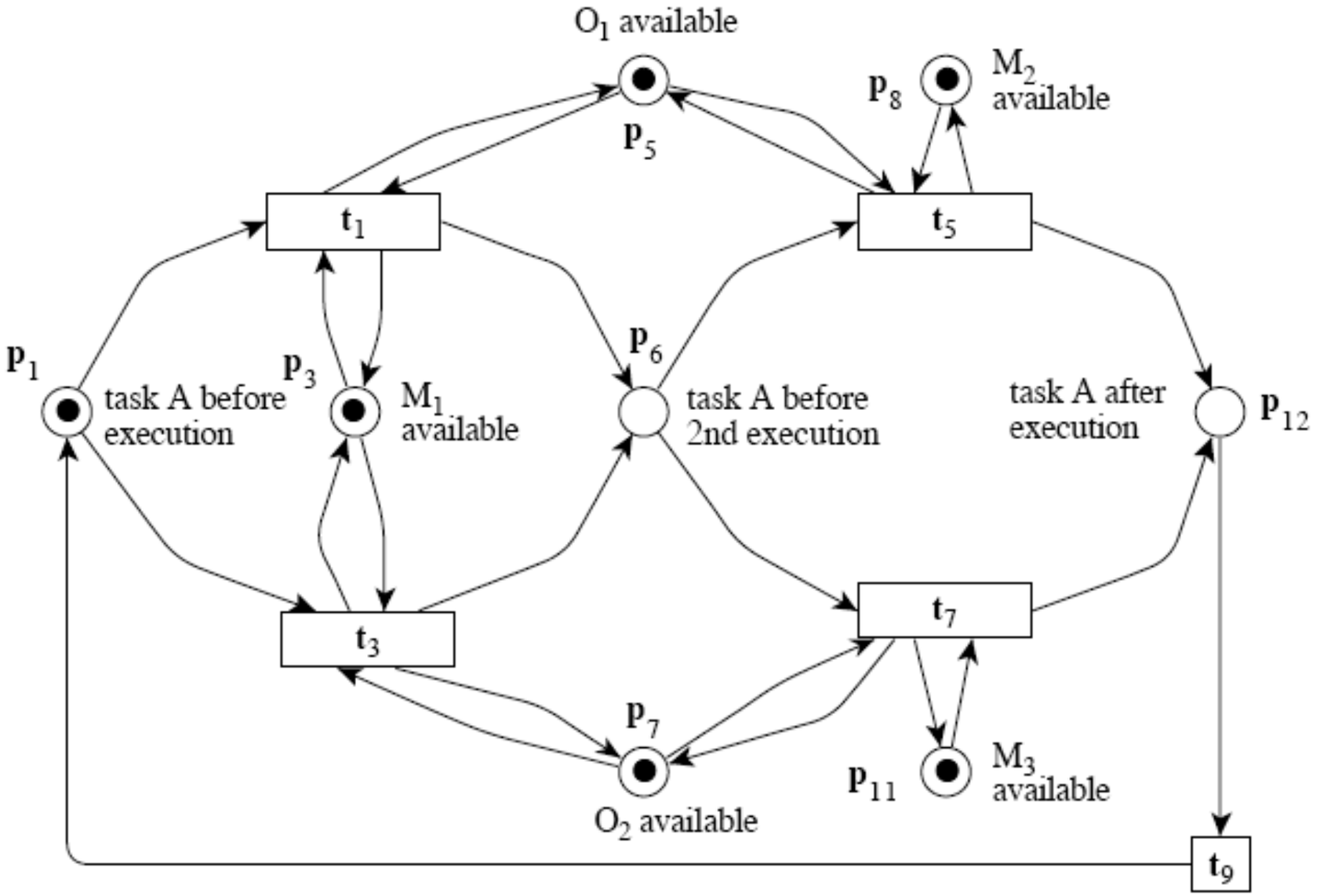


Distância Síncrona

Definition 45

Define-se como a distância síncrona entre duas transições t_1 e 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.

Reduções


O_1 available
p_5
p_8
M_2 available
t_1
t_5
p_1
task A before execution
p_3
M_1 available
p_6
task A before 2nd execution
task A after execution
p_{12}
t_3
t_7
p_7
O_2 available
p_{11}
M_3 available
t_9

Dobramentos (folding)

Um dobramento seria plenamente justificável se, para este exemplo, tivermos agora diferentes tipos de peças para fabricar onde cada tipo delas denota uma receita diferente, isto é, uma sucessão diferente de operações. Neste caso os conflitos da rede deveriam ser resolvidos com o tipo da peça.

Segundo a norma ISO/IEC 15.909-1

A **HLPN** is a structure $HLPN = (P, T, D; Type, Pre, Post, M_0)$ where

- P is a finite set of elements called Places.
- T is a finite set of elements called Transitions disjoint from P $(P \cap T = \emptyset)$.
- D is a non-empty finite set of non-empty domains where each element of D is called a type.
- $Type : P \cup T \longrightarrow D$ is a function used to assign types to places and to determine transition modes.
- $Pre, Post : TRANS \longrightarrow \mu PLACE$ are the pre and post mappings with

$$TRANS = \{(t, m) \mid t \in T, m \in Type(t)\}$$

$$PLACE = \{(p, g) \mid p \in P, g \in Type(p)\}$$

. . .

- $M_0 \in \mu PLACE$ is a multiset called the initial marking of the net.

NOTE: $\mu PLACE$ is the set of multisets over the set, $PLACE$ (see Annex A, section A.2).

Buscando um processo de projeto

Nos casos em que intuitivamente temos um sistema que é plenamente representado por uma rede clássica, e, mais do que isso, onde este modelo é facilmente e completamente interpretado modelo, é fácil de entender que somente uma demanda por múltiplos casos de simetria ou dobramento nos levaria a apelar para um sistema de alto nível.

No exercício que acabamos de ver poderíamos introduzir vários tipos de peça no processo de fabricação, cuja "receita" seria dada por diferentes combinações das operações utilizadas operando nas diferentes máquinas. Neste caso seria bastante atraente a distinção de marcas por tipos. Mas será esta a sequencia adequada em todos os casos?

Requisitos: o início de um grande problema

Certamente o início de todo projeto bem sucedido é a eliciação de um conjunto de requisitos que descreve com precisão as funcionalidades do sistema que deve ser modelado e implementado. Portanto para se chegar a um processo de projeto que termine na modelagem do sistema em Redes de Petri é preciso ter em conta de que este projeto deve começar com uma boa representação de requisitos. A representação mais usada e difundida para isso é com certeza a UML.

Análise de Requisitos, Síntese de redes, Building blocks

Uma hipótese bastante tentadoras seria ter um processo de projeto que pudesse ser reduzido a uma sequencia de transformações de transferência semântica entre linguagens, começando pela UML. Uma rede de Petri derivada de um ou mais diagramas UML poderia servir de base para um processo de **análise destes requisitos** e mais tarde com as devidas mudanças inseridas resultar no modelo do sistema.

Este processo poderia perfeitamente ser combinado com o método conhecido como ***building blocks*** onde várias partes da rede poderiam ser sintetizadas como descrito no parágrafo acima até se ter, por composição o sistema completo.

Partindo dos Casos de Uso

Casos de uso podem ser representados segundo uma forma diagramática como proposto em (Silva e Santos, 2004)

Symbol	Description
●	Start of a process (Use Case)
⊙	End of a process (Use Case)
→	Start of an event of basic flow of process
↳	Start of an event of alternative flow of process
◇	Start of a conditional event
$\rightsquigarrow^n$	Jump of current iteration to iteration ^{n}
||	Sequence of concurrent events

Silva, J.R. e Santos, E.A.; Applying Petri Nets to Requirements Validation, ABCM Symposium Series in Mechatronics, vol 1, pp. 508-517.

Um exemplo: o caixa eletrônico

Caso de uso textual

1. Initiate Withdraw – Customer inserts bank's card in the card reader on the ATM machine
2. Verify Bank Card – The ATM reads the account code from the magnetic strip on the bank card and checks if it is an acceptable bank card
3. Enter PIN – The ATM asks for the customer's PIN code (4 digits)
4. Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account. For this flow, the account is a valid account and the PIN is the correct PIN associated with this account
5. Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects "Cash Withdraw"
6. Enter Amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount ($10, $20, $50, or $100)
7. Authorization – The ATM initiates the verification process with the Banking System by sending the Card ID, PIN, Amount, and Account information as a transaction. For this flow, the Banking System is online and replies with the authorization to complete the cash withdrawal successfully and updates the account balance accordingly
8. Dispense – The Money is dispensed
9. Receipt – The receipt is printed and dispensed. The ATM also updates the internal log accordingly
10. Return Card – The Bank Card is returned

1 ● Initiate Withdraw – Customer inserts bank's card in the card reader on the ATM machine; (

1.1 ◇ Is the card valid? – Verify Bank Card – The ATM reads the account code from the magnetic strip on the bank's card and checks if it is an acceptable card;

1.1.1 ↳Send message of invalid card – If the card isn't an acceptable card, send an appropriate message. ⇝1.9

1.2 → Enter PIN – The ATM asks for the customer's PIN code (4 digits);
1.3 ◇ Is PIN Correct? – Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account;

1.3.1 ◇ Is the account valid? – Verify account code – The account code is verified;

1.3.1.1 ↳Send message of invalid account – The Banking system returns a code indicating the account could not be found or is not an account which allows withdrawals. ⇝1.9

1.3.2 ◇ Is the final try? – Checks the number of tries – The customer has three tries to enter the correct PIN;

1.3.2.1 ↳Send message of incorrect PIN – The ATM send an appropriate message. ⇝1.2

1.3.3 ↳Retain card – on the final try the card is retained, ATM returns to Ready State, and this use case terminates. ⇝2

1.4 ◇ Have money the ATM? – Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects "Cash Withdraw";

1.4.1 ↳Send message ATM out of Money – If the ATM is out of money, the "Cash Withdraw" option will not be available. ⇝1.4

1.5 ◇ Sufficient funds and does not exceed the daily limit? – Enter amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount ($10, $20, $50, or $100);

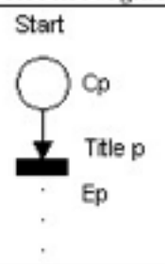
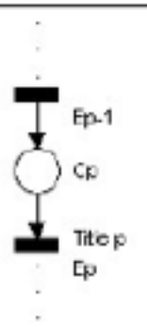
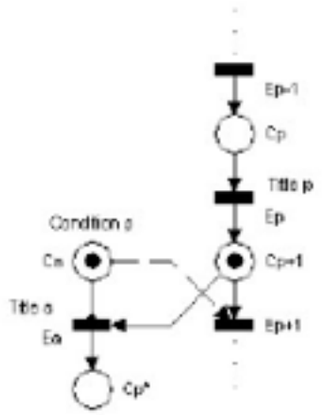
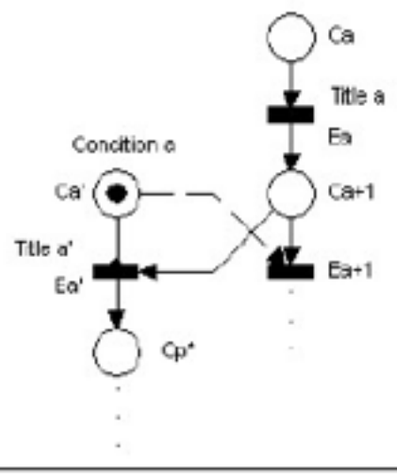
1.5.1 ◇ Sufficient funds? – Check sufficient funds – Check if the funds are sufficient;

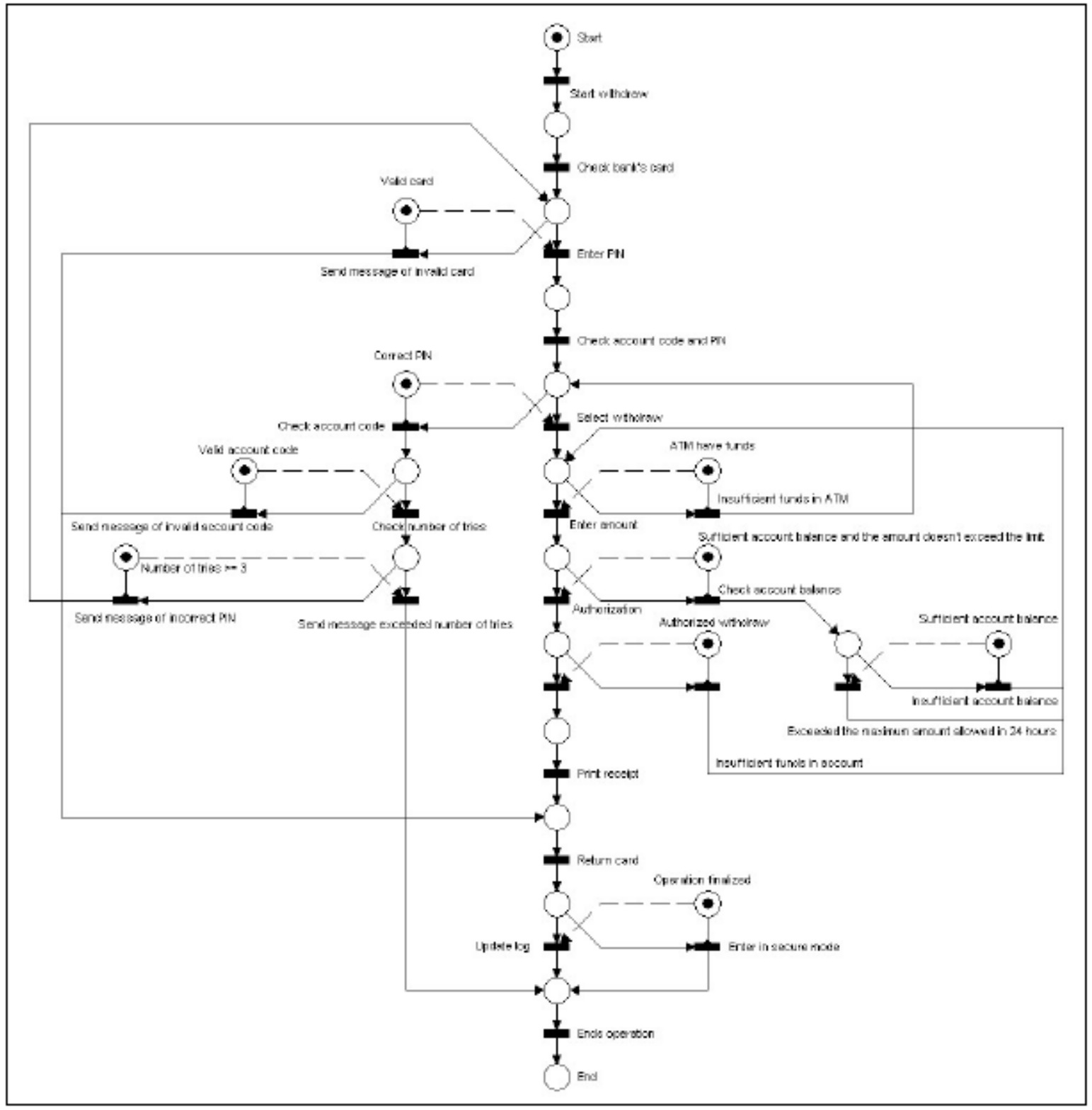
1.5.1.1 ↳Send message insufficient funds in ATM – The ATM contain insufficient funds to dispense the requested amount. ⇝1.5

```
<BF>          ::= <initial> { <event> } <final> { <AF> }
<initial>     ::= <label> "●" <title> "-" <description> ";" "("
<label>       ::= <number> { "." <number> }
<number>      ::= <sequential_number>
<title>       ::= <string>
<description> ::= <string>
<string>      ::= <any_caracter> { <any_caracter> }
<event>       ::= [ <eventB> | <eventA> ]
<eventB>      ::= <labelB> [ ( "→" <title> "-" <description> ";" ) |

                             ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                             ) ] { <eventB> } |

                  "||" "(" <eventB> { <eventB> } ")" "(" <eventB> { <eventB> } ")" "||" {
                  <eventB> }
<labelB>      ::= <label> "." <number>
<condition>   ::= <string>
<eventA>      ::= <labelA> [ ( "↳" <title> "-" <description> <branch> ) |

                             ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                             ) ] { <eventA> }

<labelA>      ::= <labelB> "." <number>
<branch>      ::= "~" [ <label> | <labelB> ]
<final>       ::= ")" <label> "◉" <title> "-" <description> ";"
<AF>          ::= <eventUA>
<eventUA>     ::= <label> "↳" <title> "-" <description> <branch> { <eventUA> }
```

Where:

- eventB → Event of Basic flow
- labelB → Event id of Basic flow
- eventA → Alternative Event of Basic flow
- labelA → Alternative Event id of Basic flow
- eventUA → Unconditional Event of Alternative Event
- labelUA → Unconditional Event id of Alternative Event

Event	BNF notation	Petri net segment
Initial	p ● Title - Description; (	
Basic	p → Title - Description;	
Basic conditional and alternative normal[1]	p◇Condition-Title-Descript.; a ↪ Title-Description ⇝p'	
Alternative conditional and normal	a◇Condition-Title-Descript.; a' ↪ Title-Description ⇝p'	


Start
Start withdraw
Check bank's card
Valid card
Send message of invalid card
Enter PIN
Check account code and PIN
Correct PIN
Check account code
Select withdraw
Valid account code
ATM have funds
Send message of invalid account code
Check number of tries
Insufficient funds in ATM
Enter amount
Number of tries == 3
Sufficient account balance and the amount doesn't exceed the limit
Check account balance
Authorization
Authorized withdraw
Sufficient account balance
Send message of incorrect PIN
Send message exceeded number of tries
Insufficient account balance
Exceeded the maximum amount allowed in 24 hours
Insufficient funds in account
Print receipt
Return card
Operation finalized
Update log
Enter in secure mode
Ends operation
End

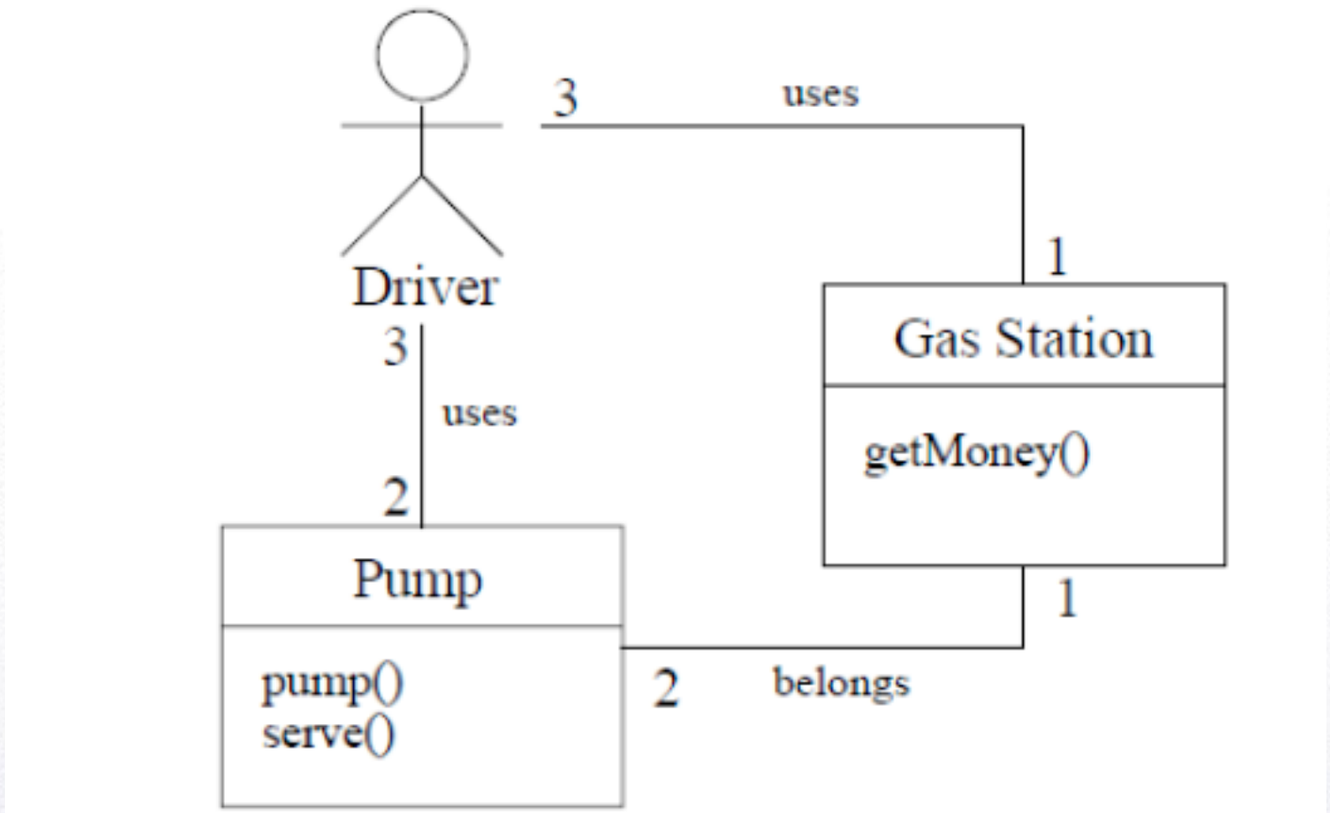
Um novo exemplo: atendimento em um posto de gasolina

Neste problema um certo número de clientes (motoristas) utilizam um mesmo posto de gasolina, que dispõe de duas bombas para atendê-los. O processo seguro demanda que os motoristas paguem antecipadamente pela quantidade especificada de combustível (especificado pelo custo ou pelo número de litros). O caixa programa uma das bombas para atender este cliente que deve usar a bomba que lhe foi indicada e servir-se.

Este problema pode também ser analisado por uma rede estendida temporizada.

Baresi, L. and Pezze, M., 2001. “Improving UML with petri nets”. In UNIGRA 2001, Uniform Approaches to Graphical Process Specification Techniques (a Satellite Event of ETAPS 2001). Elsevier, Vol. 44 of Electronic Notes in Theoretical Computer Science, pp. 107–119.

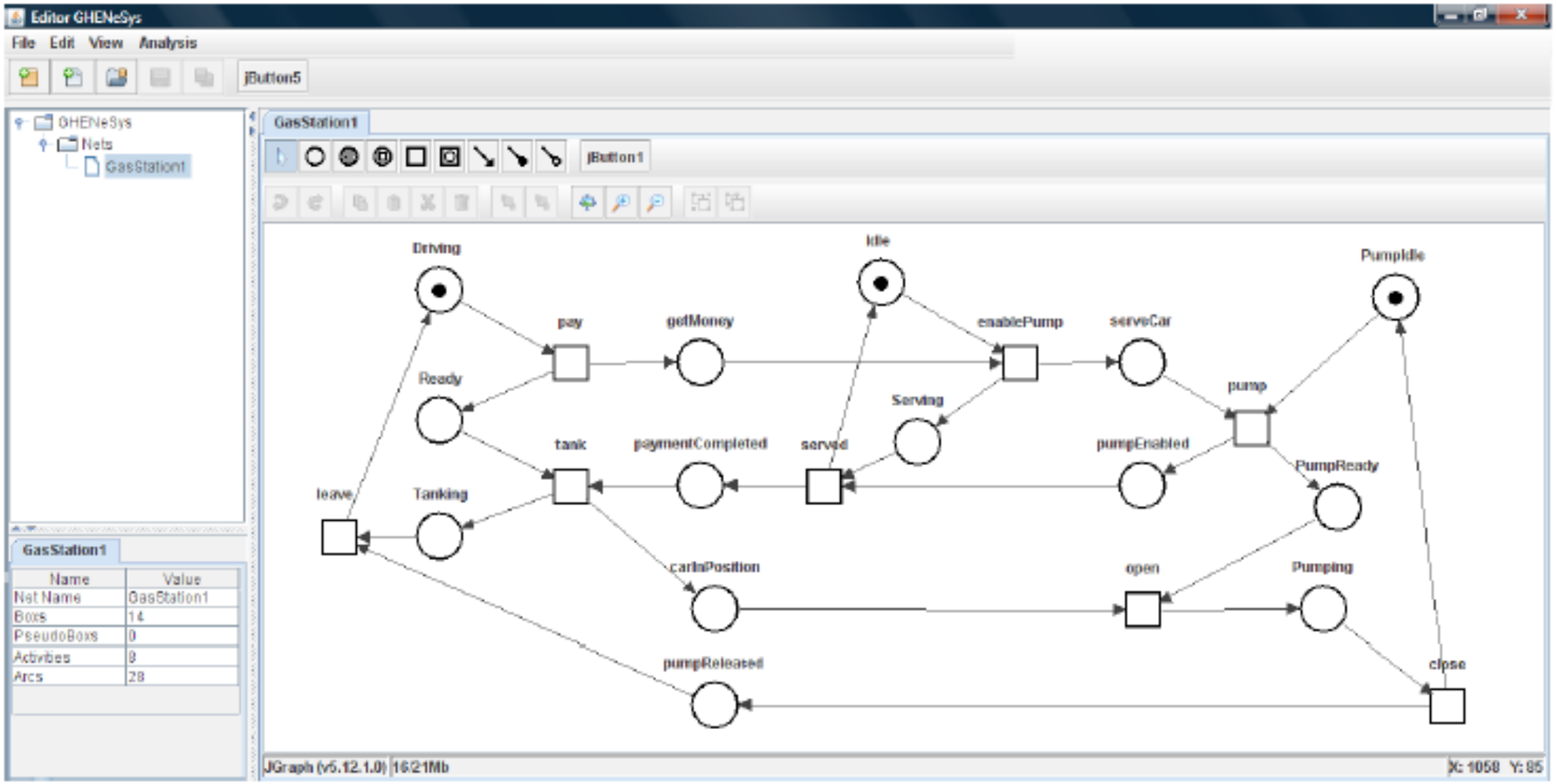
Diagrama de classe

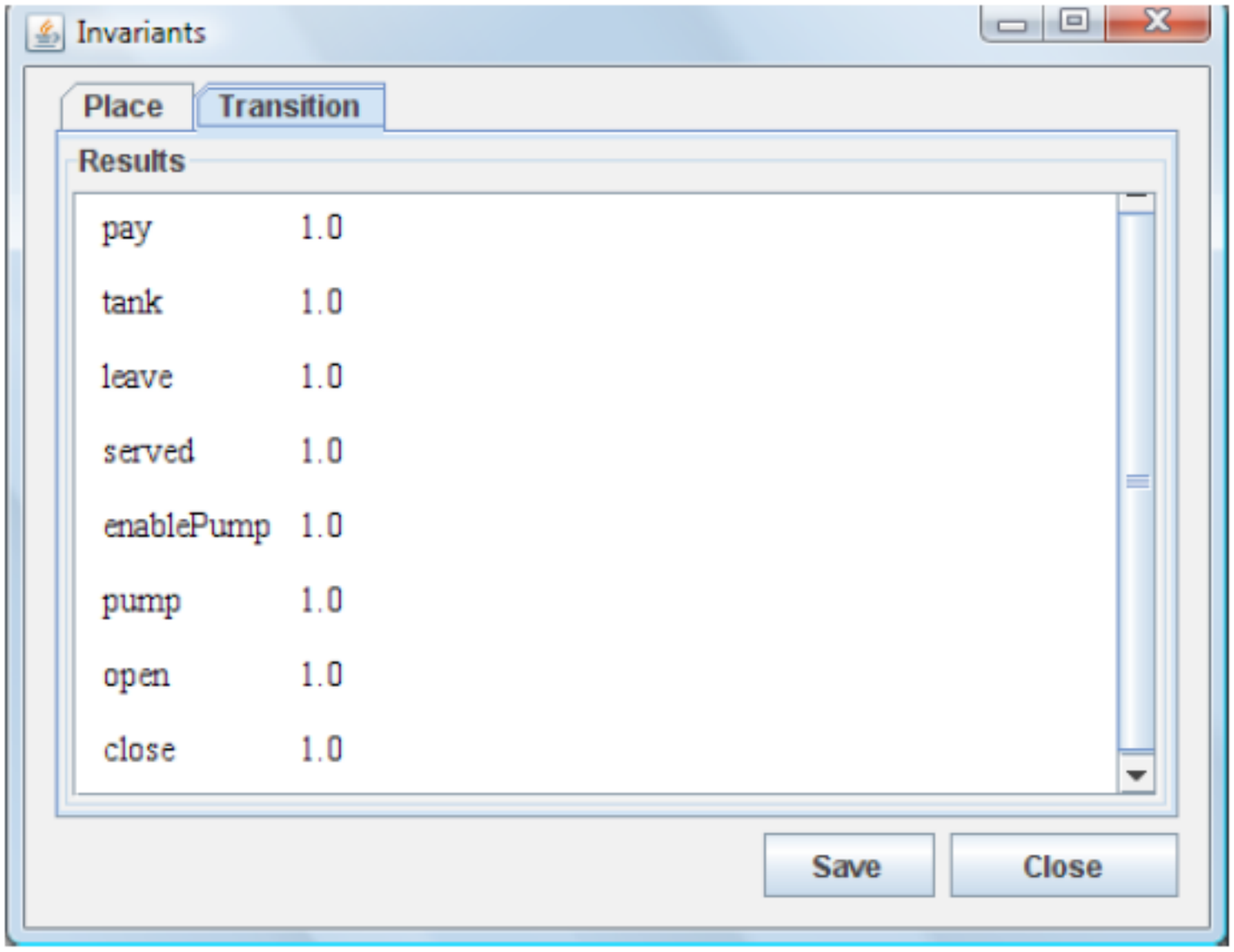


del Foyo, P.M.G., Olivera, A.Z.S., Silva, J.R.; Requirements Analysis of Automated Projects Using UML/Petri Nets, Proc. of the 21t. Int. Congress in Mechanical Engineering, Natal, Brazil, October, 2011.

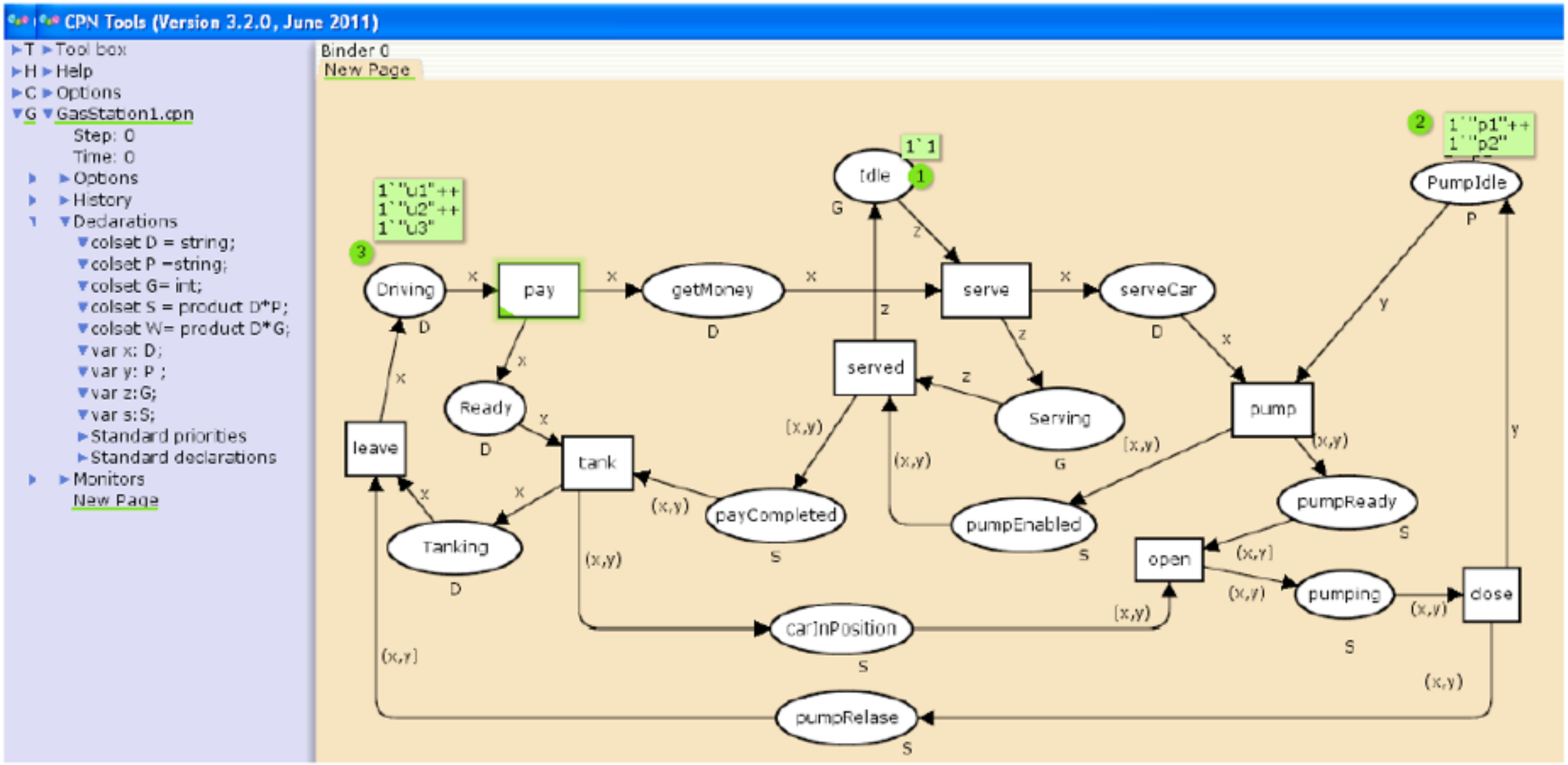
Redução do problema

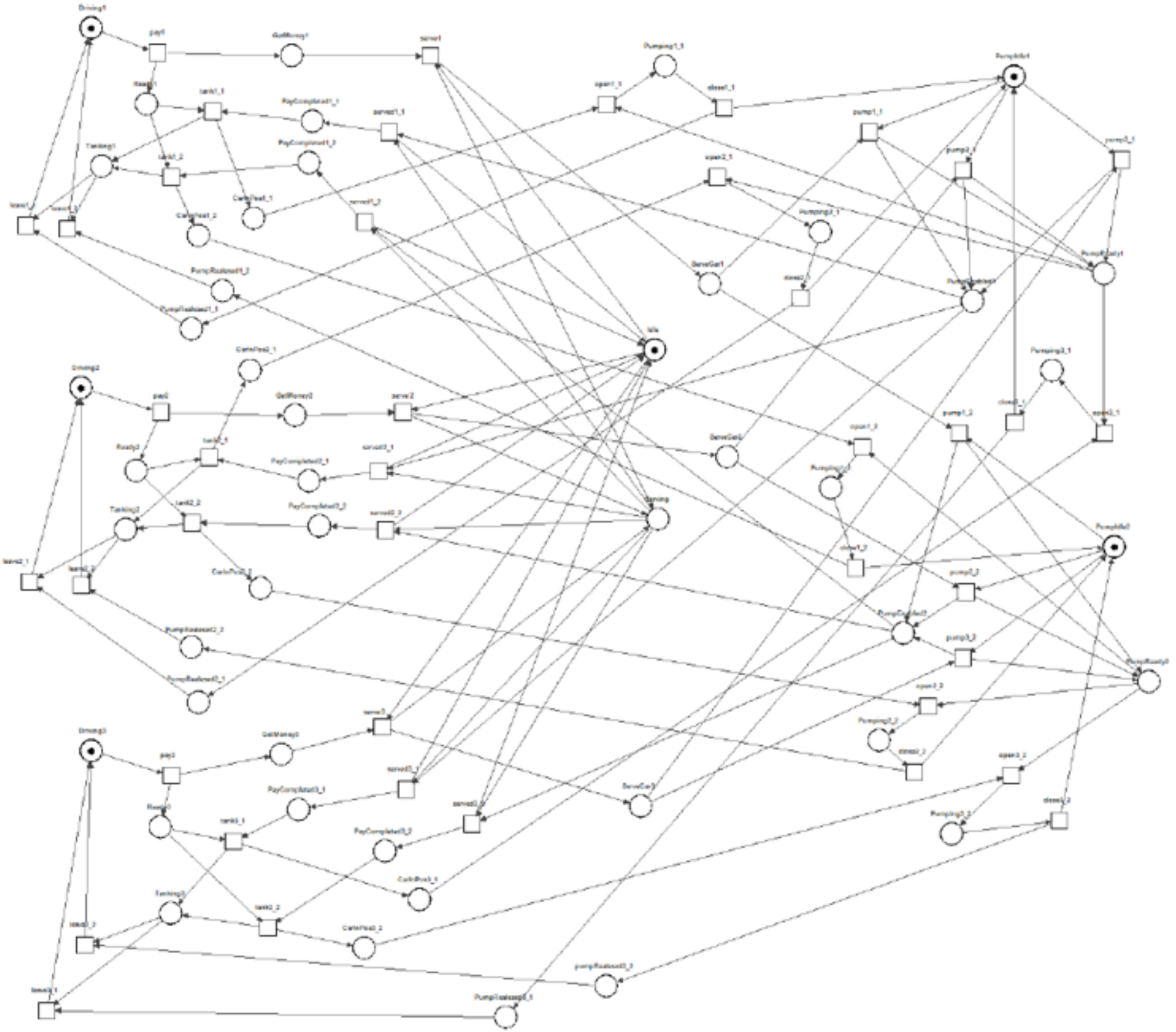
Um procedimento recomendável nestes casos seria analisar o processo de abastecimento de um motorista somente, utilizando uma das bombas.




Invariants
Place
Transition
Results
pay 1.0
tank 1.0
leave 1.0
served 1.0
enablePump 1.0
pump 1.0
open 1.0
close 1.0
Save
Close

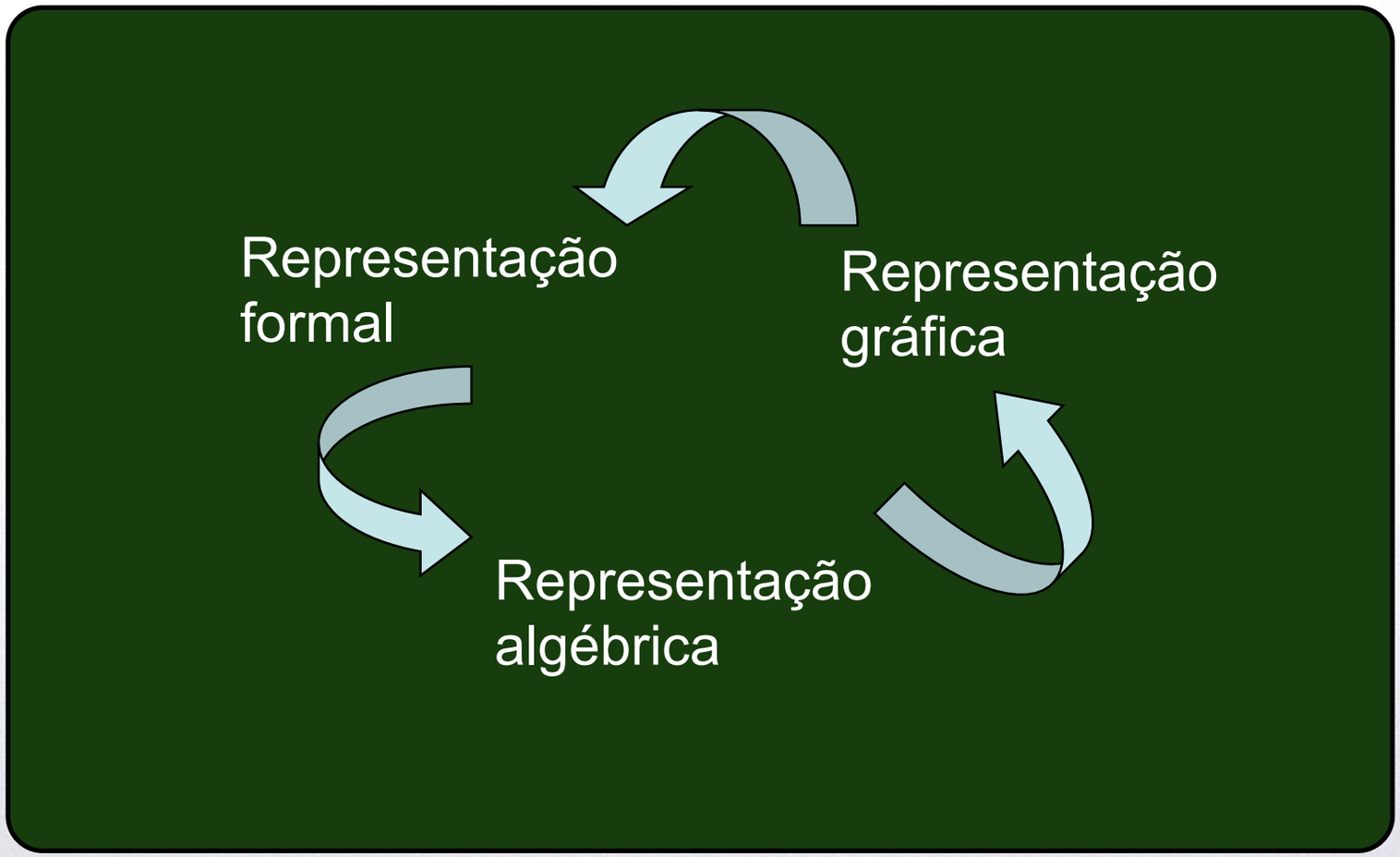
A multiplicidade dos motoristas nos leva novamente ao dobramento e às redes coloridas,

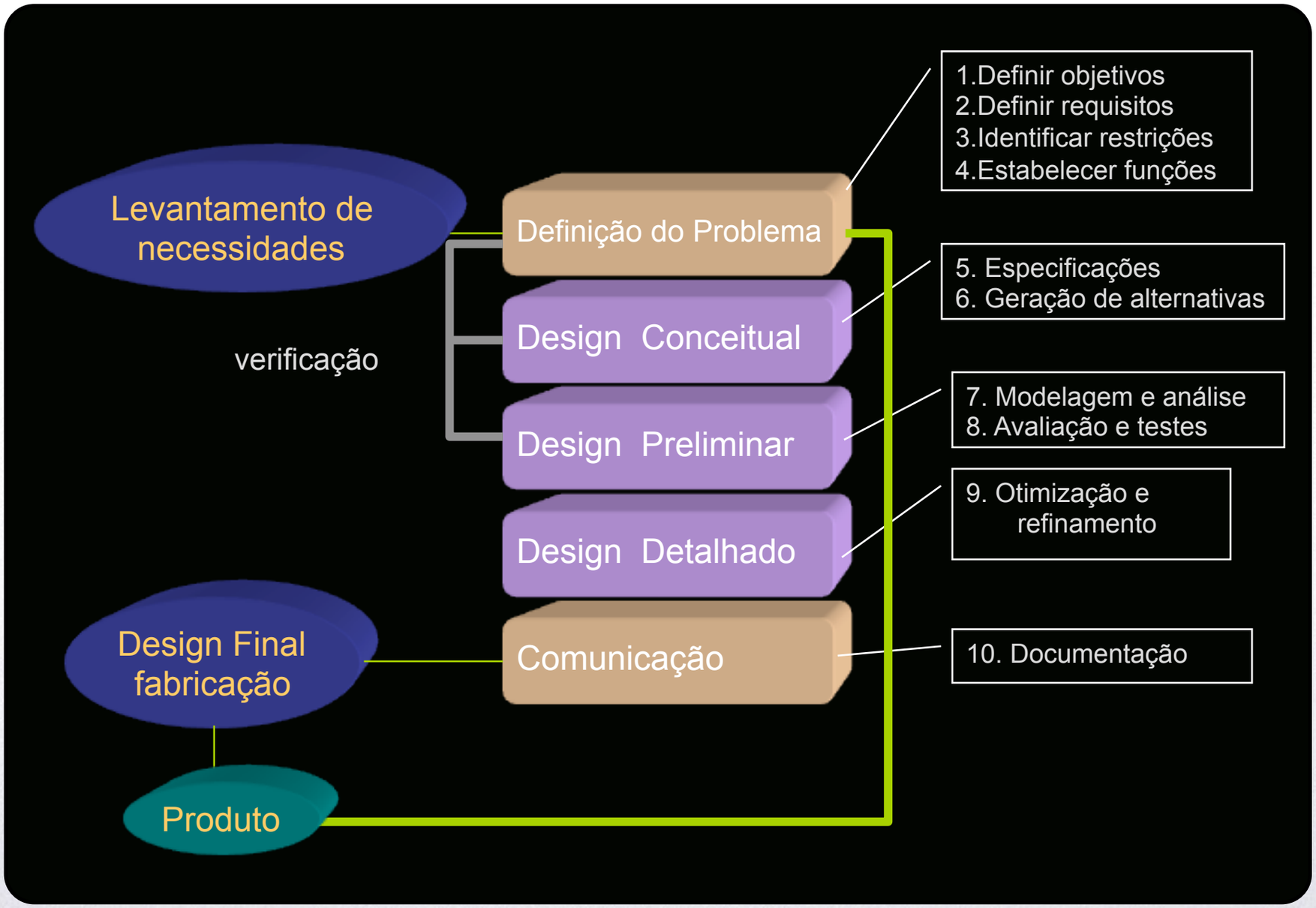




Seria possível sintetizar redes de alto nível diretamente dos diagramas UML?

O processo de modelagem e análise




Levantamento de necessidades
verificação
Definição do Problema
Design Conceitual
Design Preliminar
Design Detalhado
Comunicação
Design Final fabricação
Produto
1.Definir objetivos
2.Definir requisitos
3.Identificar restrições
4.Estabelecer funções
5. Especificações
6. Geração de alternativas
7. Modelagem e análise
8. Avaliação e testes
9. Otimização e refinamento
10. Documentação

State Space with Equivalence

O próximo nível – já além do escopo da nossa disciplina é a identificação de padrões no OS-graph que se caracterizem por sub-classes de equivalência. Assim, o comportamento destes blocos de rede se repete e podemos até abstrair o seu conteúdo.

A esta análise de chama SSE ou State Space with Equivalence.

Modelagem orientada a eventos (discretos)

Uma alternativa à modelagem orientada a estados (e ao espaço de estados) é a modelagem orientada a eventos (discretos), onde o centro do processo é a identificação dos eventos e qual a sequênica de eventos que denotam os procesos desejados (ou que levam aos estados desejados, em uma versão híbrida desta abordagem).

Um caso especial bem interessante é aquele onde existe um estado alvo único, identificado como um "planning problem".

The planning problem

Formally a planning algorithm has three inputs:

1. A description of the world in some formal language,
2. A description of the agent's goal in some formal language, and
3. A description of the possible actions that can be performed.

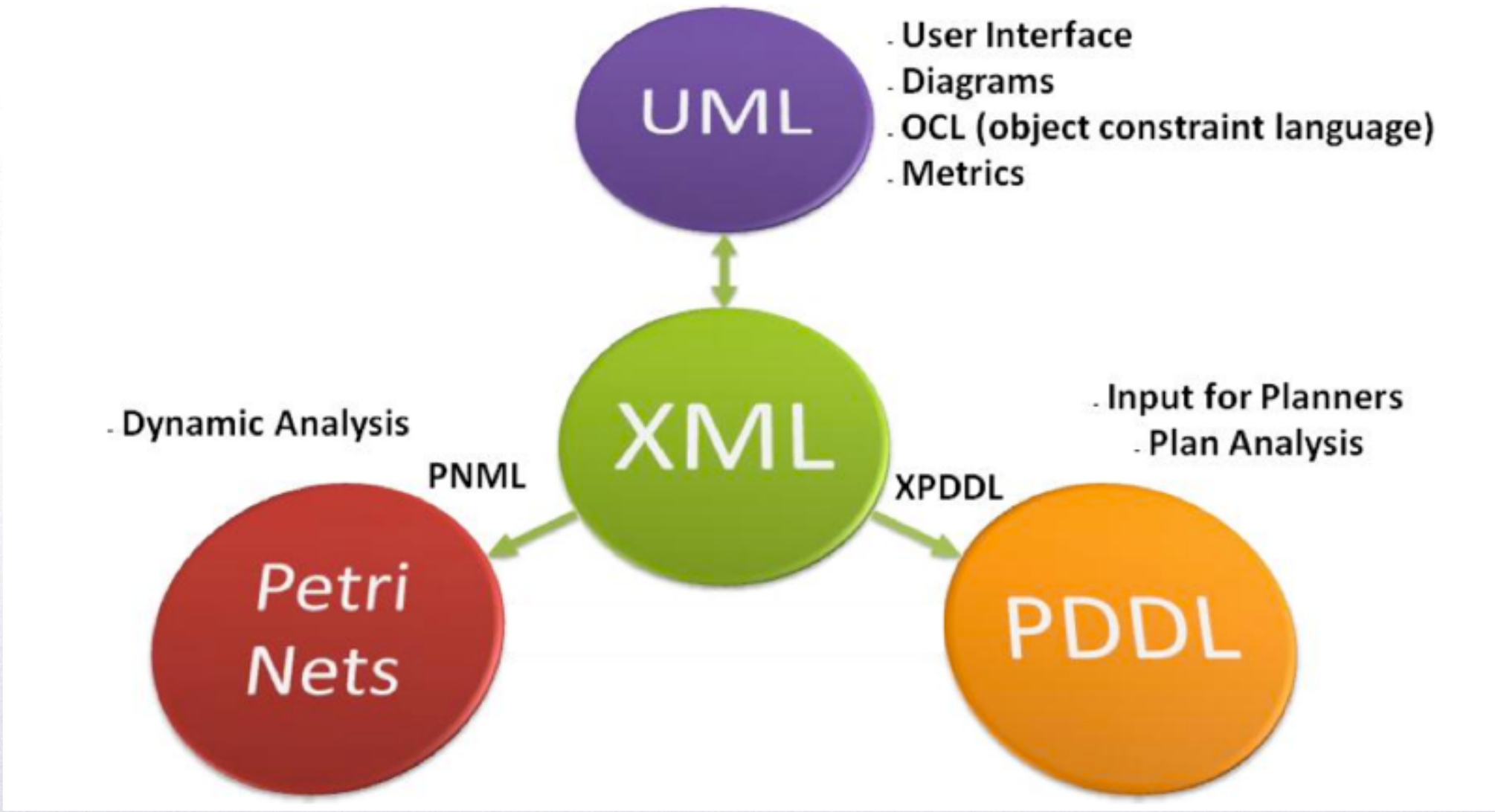
The planner's o/p is a sequence of actions which when executed in any world satisfying the initial state description will achieve the goal.

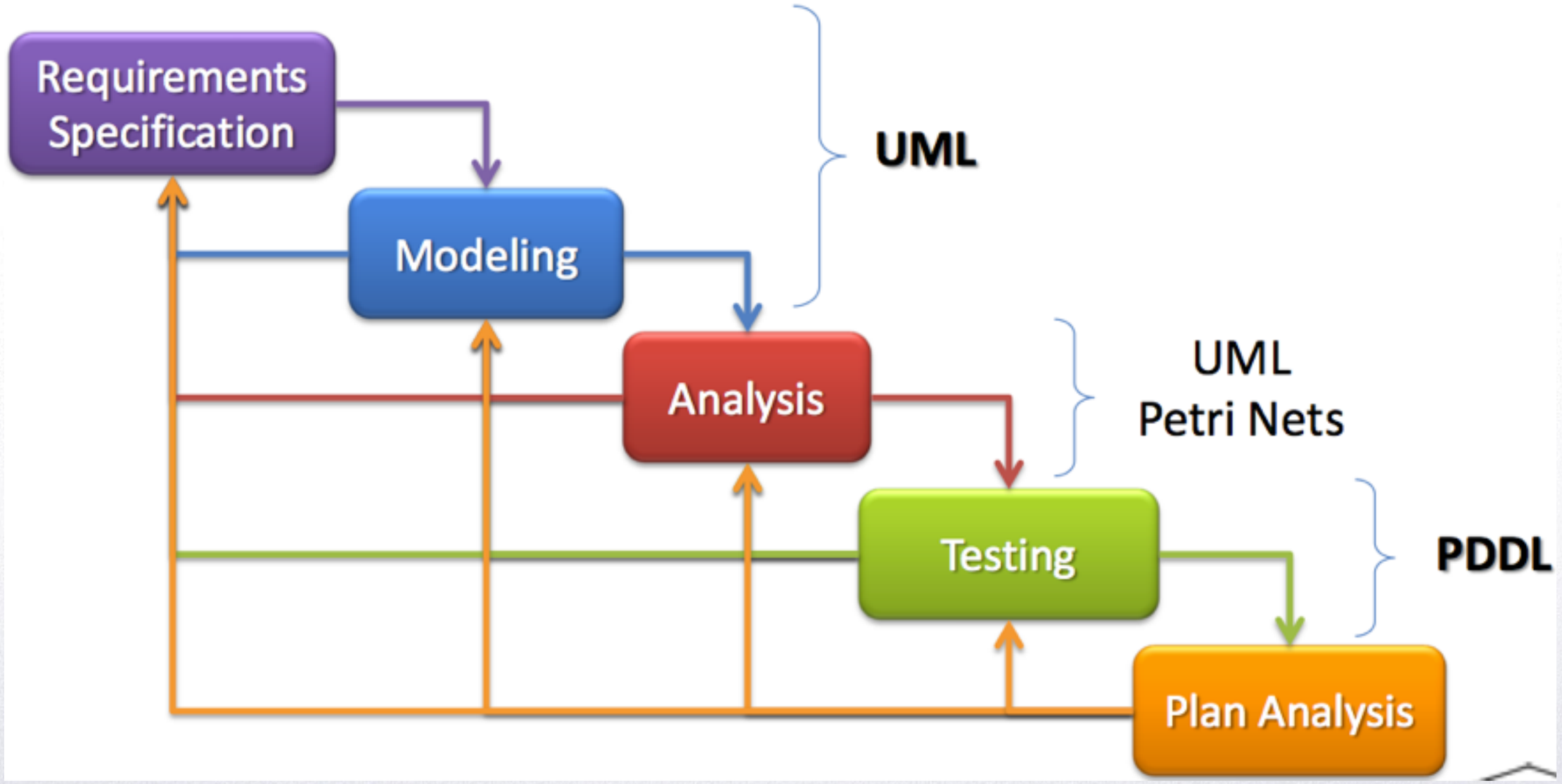
Fazendo uma analogia onde ações seriam equivalentes a eventos, poderemos redefinir um problema de planning como sendo o seguinte:

i) um problema de planning consiste em achar um passeio no grafo de estados que leve do estado inicial ao estado final especificados;

ii) um problema de planning consiste em achar uma sequencia de eventos (ações) que tenha como pre-condição genérica o estado inicial e como pós-condição genérica o estado final, e um conjunto de estados admissíveis.

itSIMPLE


UML
- User Interface
- Diagrams
- OCL (object constraint language)
- Metrics
XML
- Dynamic Analysis
PNML
Petri Nets
XPDDL
- Input for Planners
- Plan Analysis
PDDL


Requirements Specification
Modeling
Analysis
Testing
Plan Analysis
UML
UML
Petri Nets
PDDL

Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Are Automated Planners up to Solve Real Problems?

Fernando Moreira Sette*, Tiago Stegun Vaquero*, Song Won Park, Jose Reinaldo Silva***

**DesignLab, Mechatronic and Mechanical Systems Department*
Escola Politécnica – University of Sao Paulo, Brazil.
(e-mail: {fernando.sette, tiago.vaquero}@poli.usp.br, reinaldo@usp.br)
***Chemical Engineering Department. University of Sao Paulo. Brazil.*
(e-mail: sonwpark@usp.br)

Abstract: It is a well known fact that the AI planning community is very committed to apply the developments already achieved in this area to real complex applications. However realistic planning problems bring great challenges not only for the designers during design processes but also for the automated planners during the planning process itself. In addition, it is quite common to face issues about whether the available planners will be up to solve the problem being modeled during the initial design stages. In this paper we present the experience, results and issues that emerged from testing the performance of the recent planners when solving a real and complex problem such as the planning of daily

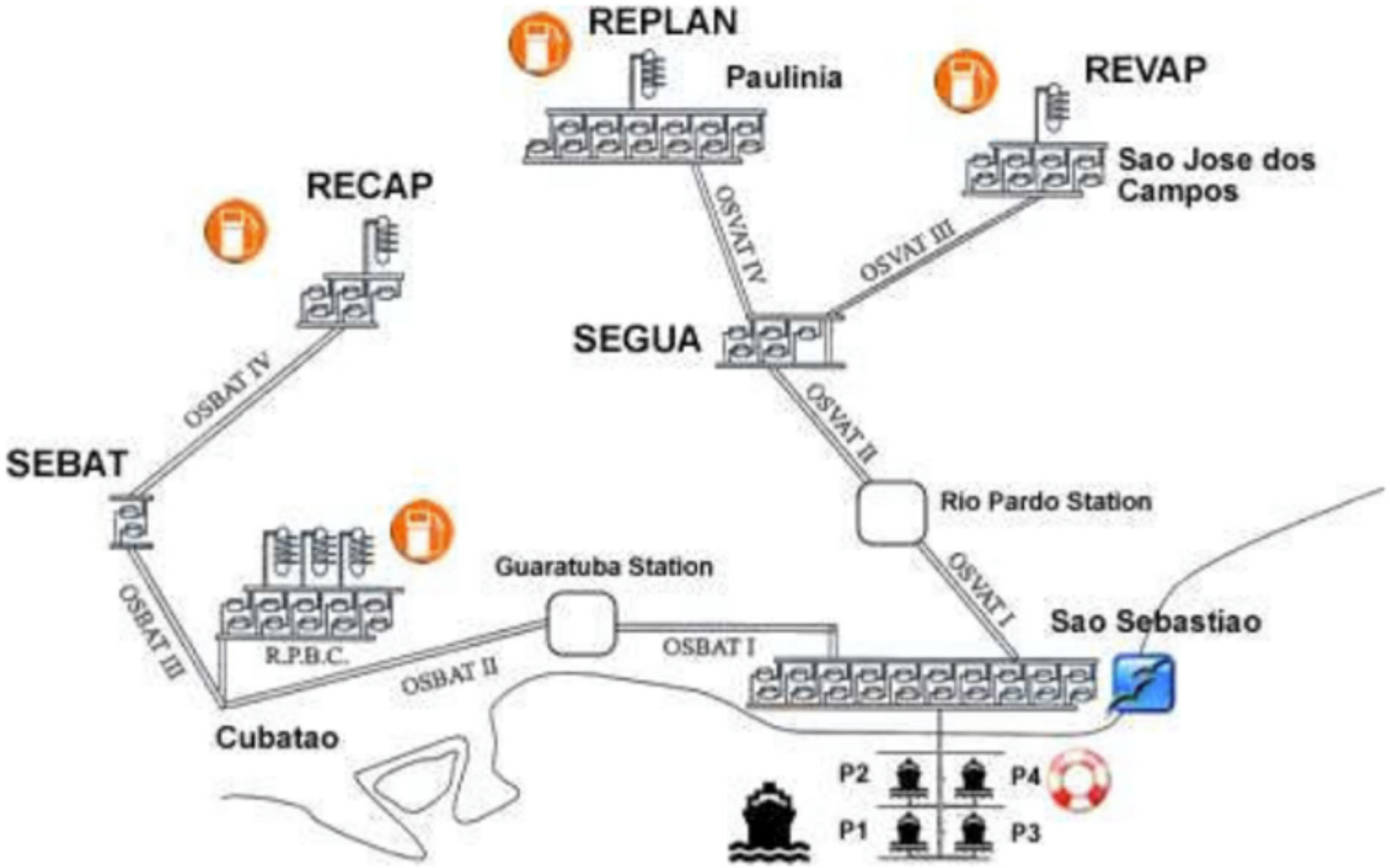


Fig. 1. Crude oil distribution infrastructure of Petrobras in the State of Sao Paulo, Brazil.

Table 1. Realistic problem without metric

# of Tanks	# of Tankers	Metric FF		SGPLAN		MIPS	
		Actions	Time	Actions	Time	Action	Time
18	1	7	0,46	7	0,46	13	1,14
18	2	17	1,82	17	0,97	25	49,84
18	3	30	3,24	31	4,66	TIME	TIME
18	4	X	X	62	54,18	TIME	TIME
18	5	FLUENTS	FLUENTS	75	120,48	TIME	TIME
18	6	FLUENTS	FLUENTS	78	42,74	TIME	TIME
18	7	FLUENTS	FLUENTS	TIME	TIME	TIME	TIME
18	8	FLUENTS	FLUENTS	97	74,43	TIME	TIME
18	9	FLUENTS	FLUENTS	TIME	TIME	TIME	TIME
18	10	FLUENTS	FLUENTS	TIME	TIME	TIME	TIME
18	11	FLUENTS	FLUENTS	115	303,78	TIME	TIME
18	12	FLUENTS	FLUENTS	126	386,72	TIME	TIME
18	13	FLUENTS	FLUENTS	132	478,73	TIME	TIME

Fim