Vérification des systèmes et logiciels répartis

Elie Najm

Elie.Najm@telecom-paristech.fr

46, Rue Barrault 75013 Paris

Quelques exemples d'échecs notoires

- système de routage de bagages de l'aéroport de Denver
  http://calleam.com/WTPF/?page_id=2086

- ATT (propagation de fautes => système down pendant 8 H)
  http://www.phworld.org/history/attcrash.htm

- Ariane V (Ariane IV - mauvaise réutilisation de code)
  http://www.astrosurf.com/luxorion/astronautique-accident-ariane-v501.htm

Quelle approche pour le développement du logiciel

- Adopter une méthode de développement
  - Qui s’appuie sur un cadre architectural
    - Et qui utilise la rigueur des mathématiques

Plan

- Rôle des méthodes de développement
- Rôle des Méthodes Formelles
- Rôle du cadre architectural
- Exemple de cadre architectural
- Un exercice de sémantique formelle : les machines à Etats de SDL
- Une introduction rapide aux Réseaux de Petri
- Une introduction rapide à la logique temporelle

Rôle des Méthodes de développement

Une méthode en 9 étapes pour le Développement du logiciel

1 - Commander les T-shirts pour l'équipe de développement

2 - Annoncer la disponibilité du produit

3 - Ecrire le code

4- Ecrire le manuel de l'utilisateur

5- Ecrire les spécifications (en conformité avec le code)

6 - Livrer le produit

7- Tester le produit (grâce aux utilisateurs)

8- En cas de bogues déclarer la logiciel corrigé comme une fonctionnalité

9 - Annoncer les nouvelles versions munies des "fonctionnalités"

Vérification versus Test

« Program testing can be used to show the presence of bugs, but never show their absence »

EdsgerW. Dijkstra

Vérifier grâce à la rigueur des mathématiques =>
Méthodes formelles

=> Spécifications non ambigües

=> Spécifications vérifiées

=> Raffinement vers l'implémentation : étapes vérifiées

Maitrise du développement du logiciel

le Capability Maturity Model (CMM) du CMU

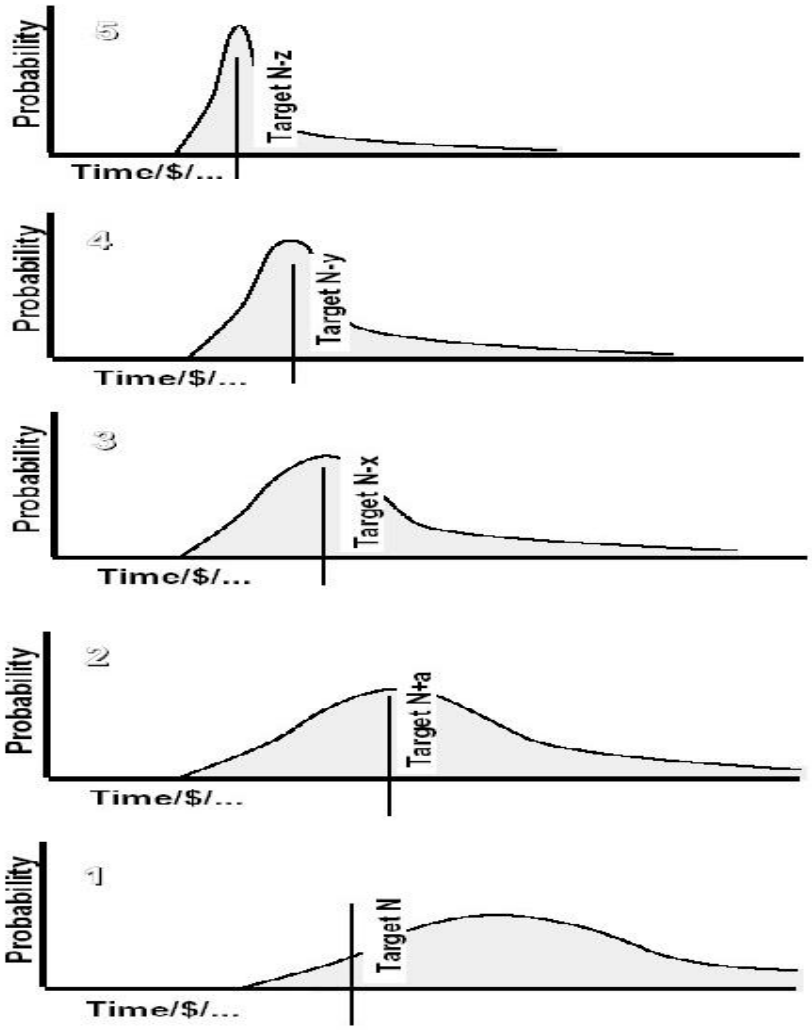
https://fr.wikipedia.org/wiki/Capability_maturity_model

5 niveaux de maturité :

niveau 1 - initial

niveau 5 - optimisé - utilisation de méthodes formelles

Niveaux CMM et predictabilité des projets



Performance continuously improves in Level 5 organizations

Based on quantitative understanding of process and product, performance continues to improve in Level 4 organizations

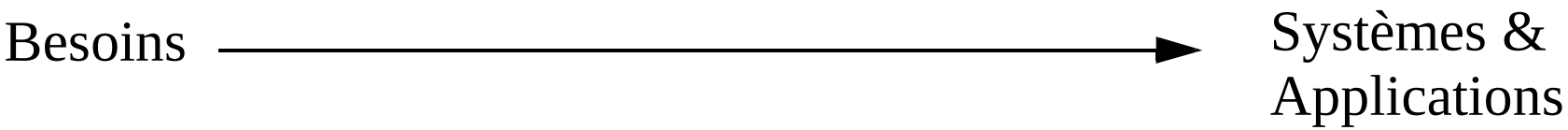
With well-defined processes, performance improves in Level 3 organizations

Plans based on past performance are more realistic in Level 2 organizations

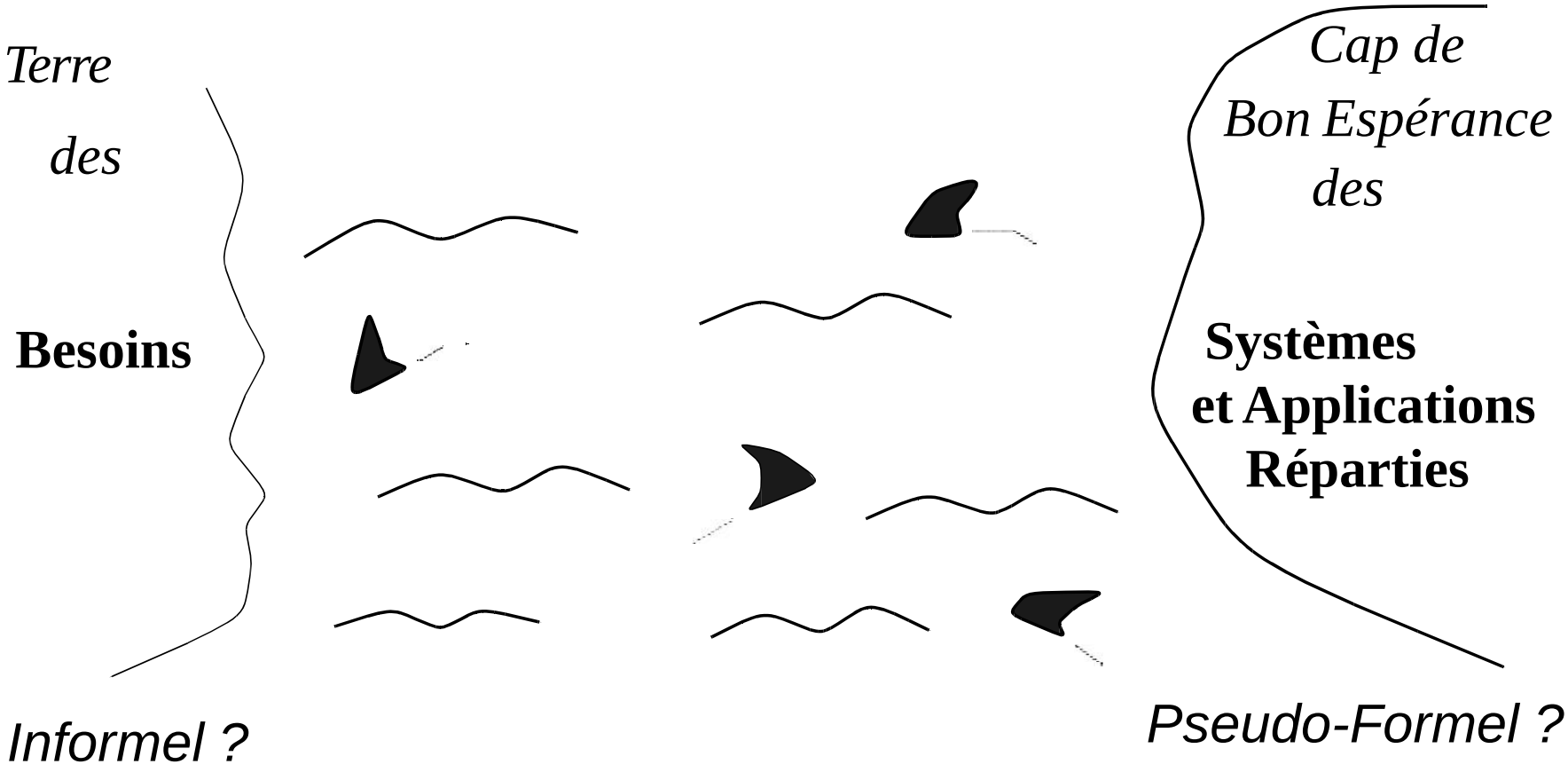
Schedule and cost targets are typically overrun by Level 1 organizations.

Rôle des Méthodes Formelles

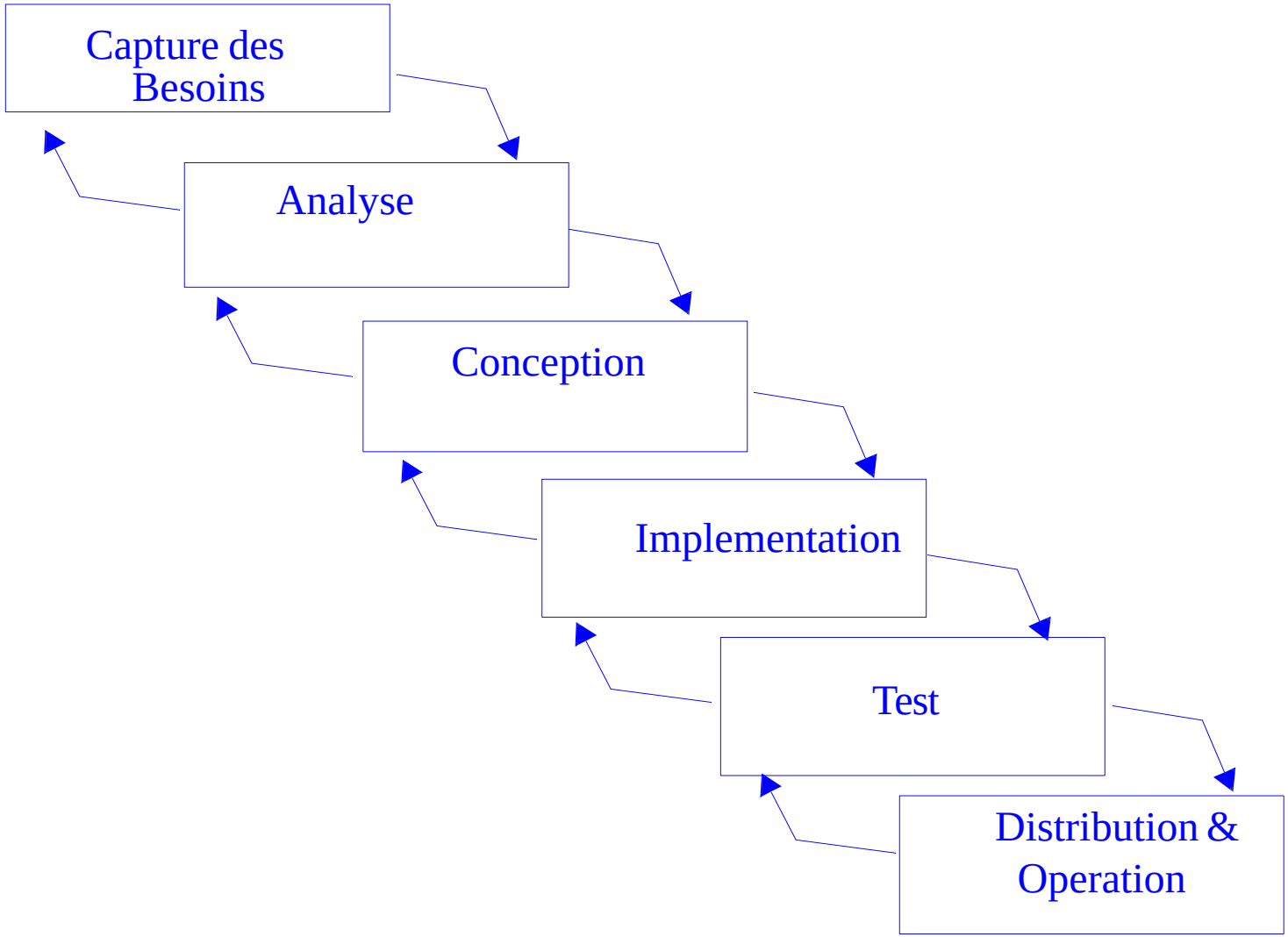
Ingénierie des Logiciels et Systèmes Répartis



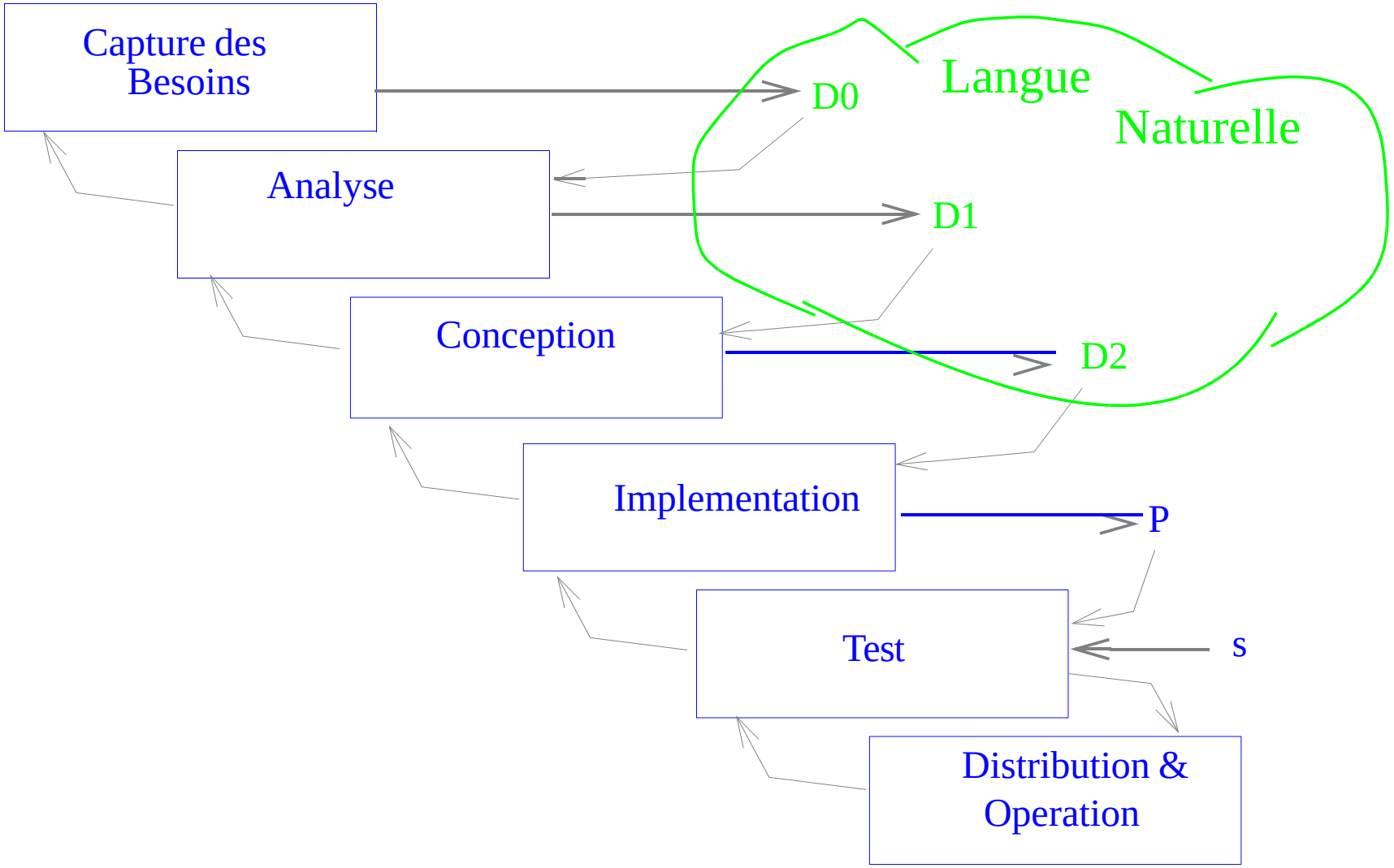
Ingénierie des Logiciels et Systèmes Répartis



Un Cycle de Développement du logiciel



Cycle de Développement du logiciel : approche non formelle



Langages de description de langage de programmation

Langages d'analyse et d'expression des Besoins

- Langues Naturelles
- Langues Naturelles Contôlées

- Langages Formels
  (semantique)

caracteristiques
- expression et verif de propriétés
- modélisent la réalité

Langages effectifs

- interprétables et exécutable par des :
  - ordinateurs
  - équipements de communication
  - autres équipenents
    (imprimantes, robots domotiques , ...)

caracteristiques :
agissent sur le monde réel

Apports de la formalisation

E_i E_j

E_j *suit* E_i

E_j *contredit* E_i

E_k *est sans blocage*

E_j *satisfait* E_i

E_j *raffine* E_i

E_j *est équivalent à* E_i

E_j *optimise* E_i

E_j *est un test pour* E_i

E_j *est une transformation correcte de* E_i

E_j *est une implémentation de* E_i

E_j *coopère avc* E_i

E_j *référence* E_i

E_j *utilise* E_i

Les approches de la vérification

- Model Checking : exploration de l'espace des états
  - Logique temporelle, Bisimulations et équivalences
  - Avantages : outils « push button », exhibe des contre-exemples,
  - Inconvénients : explosion combinatoire, systèmes finis

- Approche par preuve : automatisation du raisonnement
  - Démonstrateurs interactifs ou automatiques :
    - Coq (ex: CompCert), PVS, Isabelle
  - Preuves de raffinements sains :
    - B (Meteor – ligne 14 du métro), Z, OCL,
  - Avantages : peut s'appliquer sur des systèmes non finis
  - Inconvénient : courbe d'apprentissage, pas de contre-exemples

Propriétés vérifiables

- Absence de blocage total (deadlock)
- Absence de boucle bloquante (livelock)
- Absence de famine
- Equité
- Etats dangereux non atteignables (section critique)
- Equivalence entre comportements (raffinement correct)

Langages et formalismes formels

Expression du comportement

- Machines à états étendus : StateCharts, SDL, Promela, Réseaux de Petri
- Algèbres de processus : LOTOS, CCS, CSP, MuCrl
- Langages basés sur la théorie ensembliste : B, Z

Expression de propriétés

- Logiques du premier ordre, logiques temporelles

Formalismes et vérification

Formes de C sat P

C |= P spécification comportementale C est un modèle pour P
Exemple (Promela + logique temporelle)

C reussit T (T est une séquence de test dérivée de C)
Exemple (SDL + MSC)

B ~ B' Comportement B est *équivalent* à B'
Exemple (CCS + CCS)

I conf S Implémentation I est conforme à la spécification S

Rôle du cadre architectural

Vers un meilleur contrôle du Cycle de développement

1) Identifier les bons niveaux d'abstraction pour le cycle de développement

Exemple, les points de vues ODP https://en.wikipedia.org/wiki/RM-ODP

2) Doter chaque niveau d'abstraction d'un langage de description

3) Formaliser les niveaux d'abstraction =>

Langages ayant une **sémantique formelle**

=> Vérification de propriétés :

Le système (ou son abstraction) fait il ce qui est attendu

Le système (ou son abstraction) le fait il bien

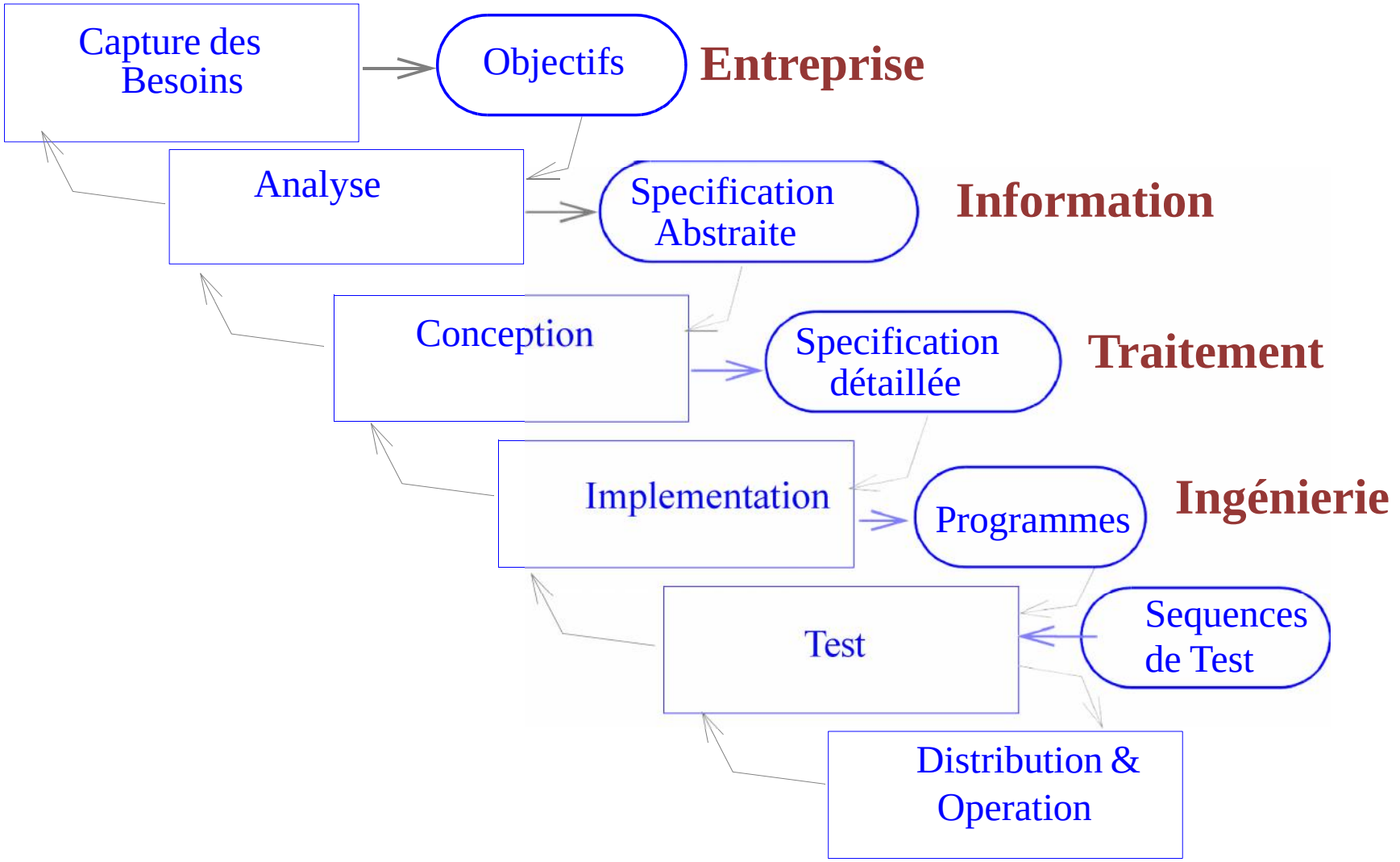
=> Préservation des propriétés :

La Transformation entre modèles est elle correcte

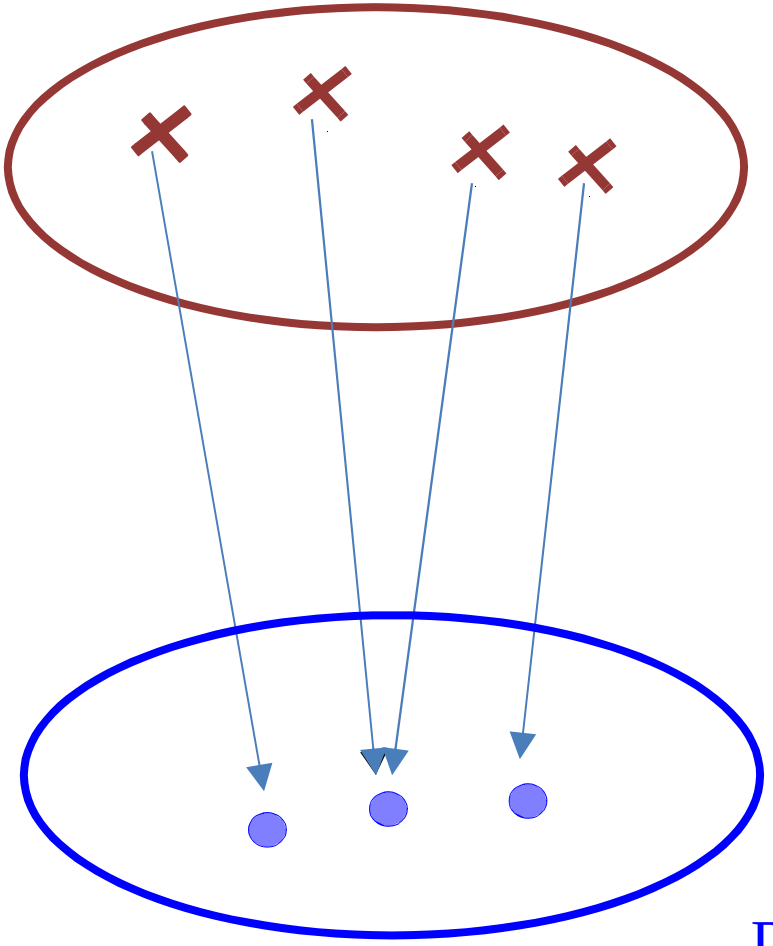
Les points de vue du Modèle de Référence ODP

- Entreprise
  - Objectif, besoins, stratégie, contexte réglementaire, partenariats
- Information
  - Structure de l'état du système et ses évolutions
- Traitement
  - Décomposition fonctionnelle du système en unités distribuables
- Ingénierie
  - Infrastructure support de la distribution
- Technologie
  - Choix de solutions technologiques pour l'implantation

Cycle de Développement du logiciel et **Points de Vue - *ODP***



Qu'est qu'un langage avec **Sémantique Formelle**



Un langage avec une syntaxe définie

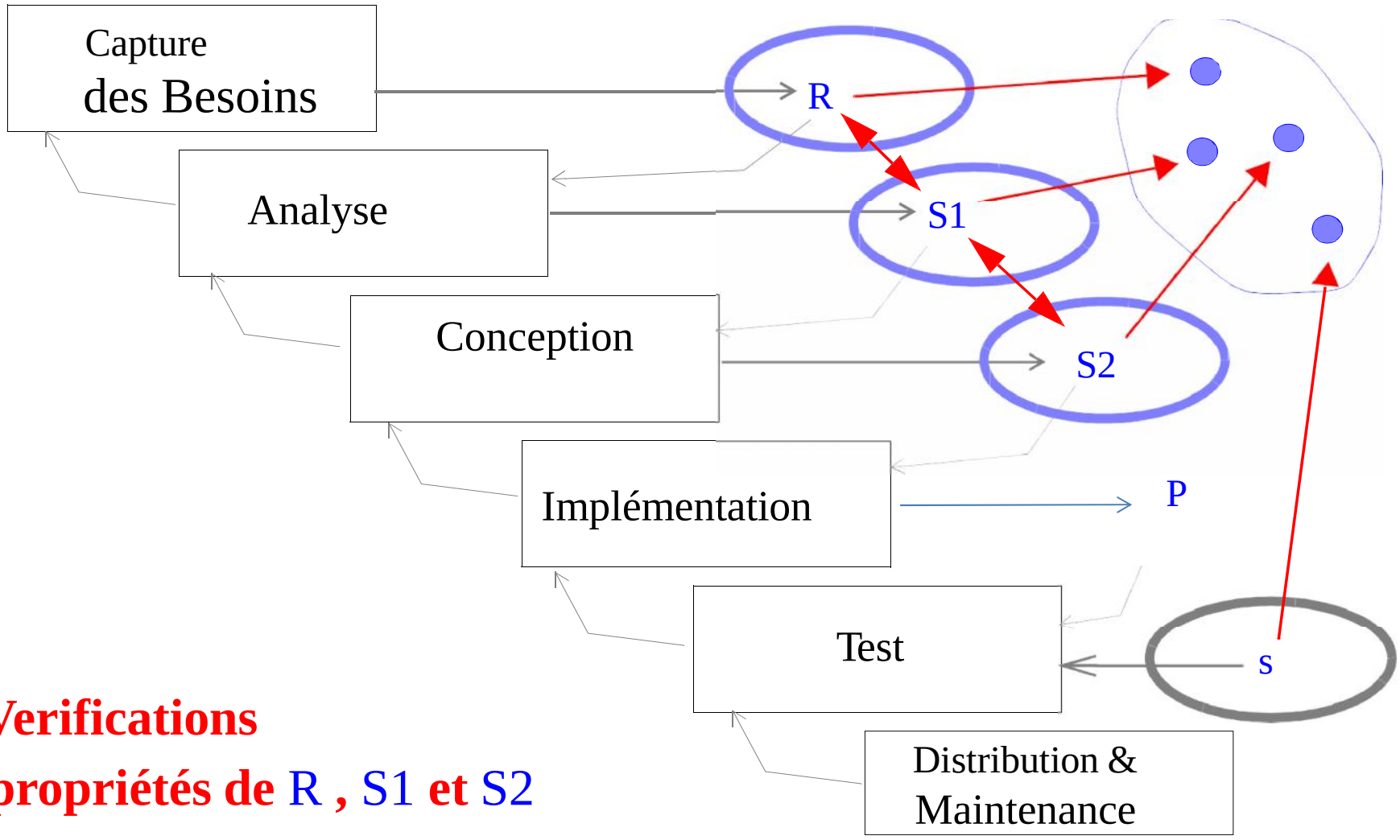
Une fonction d'interprétation

Un univers des interprétations

Domaine défini mathématiquement

exemple : les graphes étiquetés

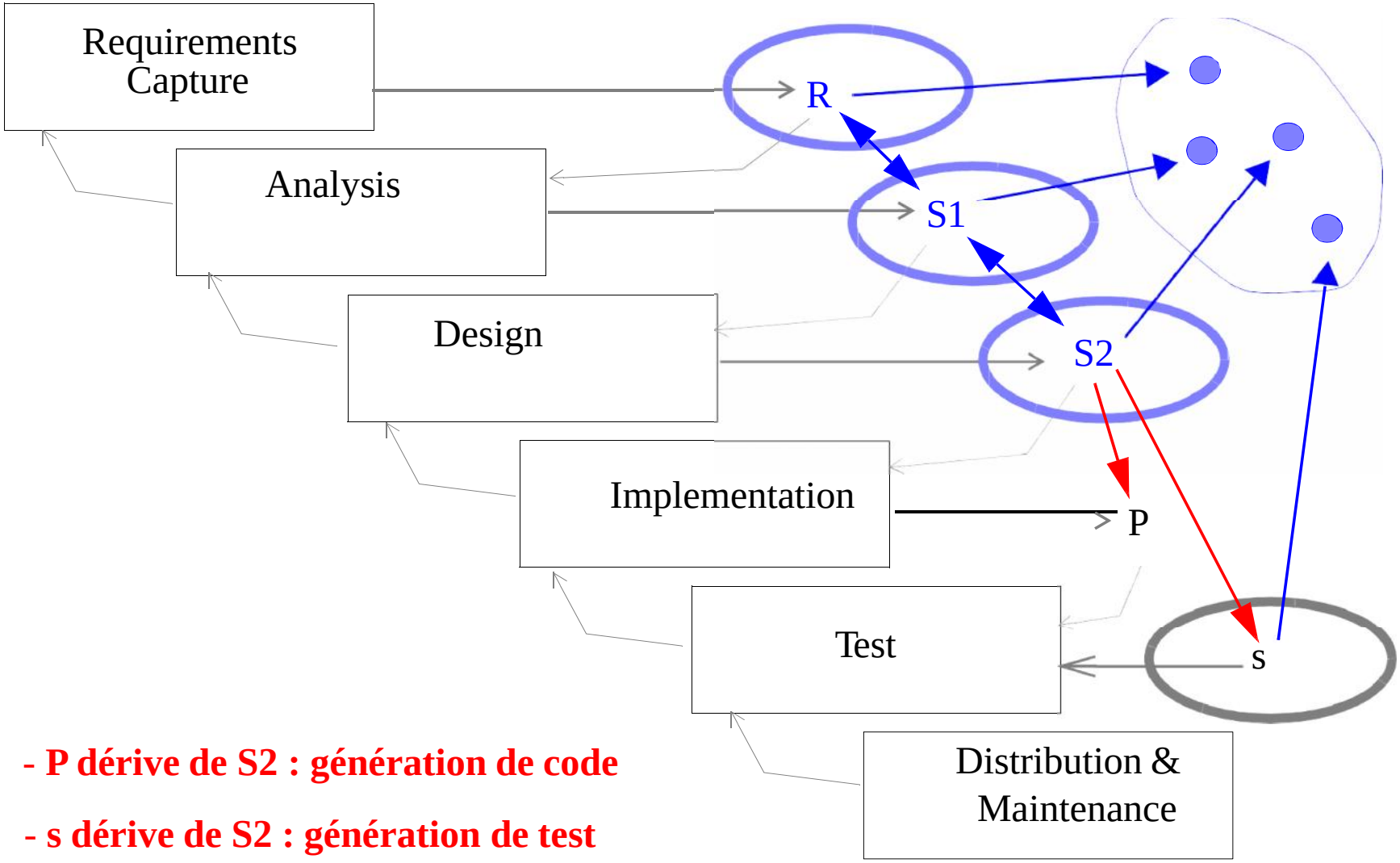
Cycle de Développement du logiciel : approche formelle



Verifications

- **propriétés de** R , S1 **et** S2
- **cohérences entre** R , S1 , S2 **et** s

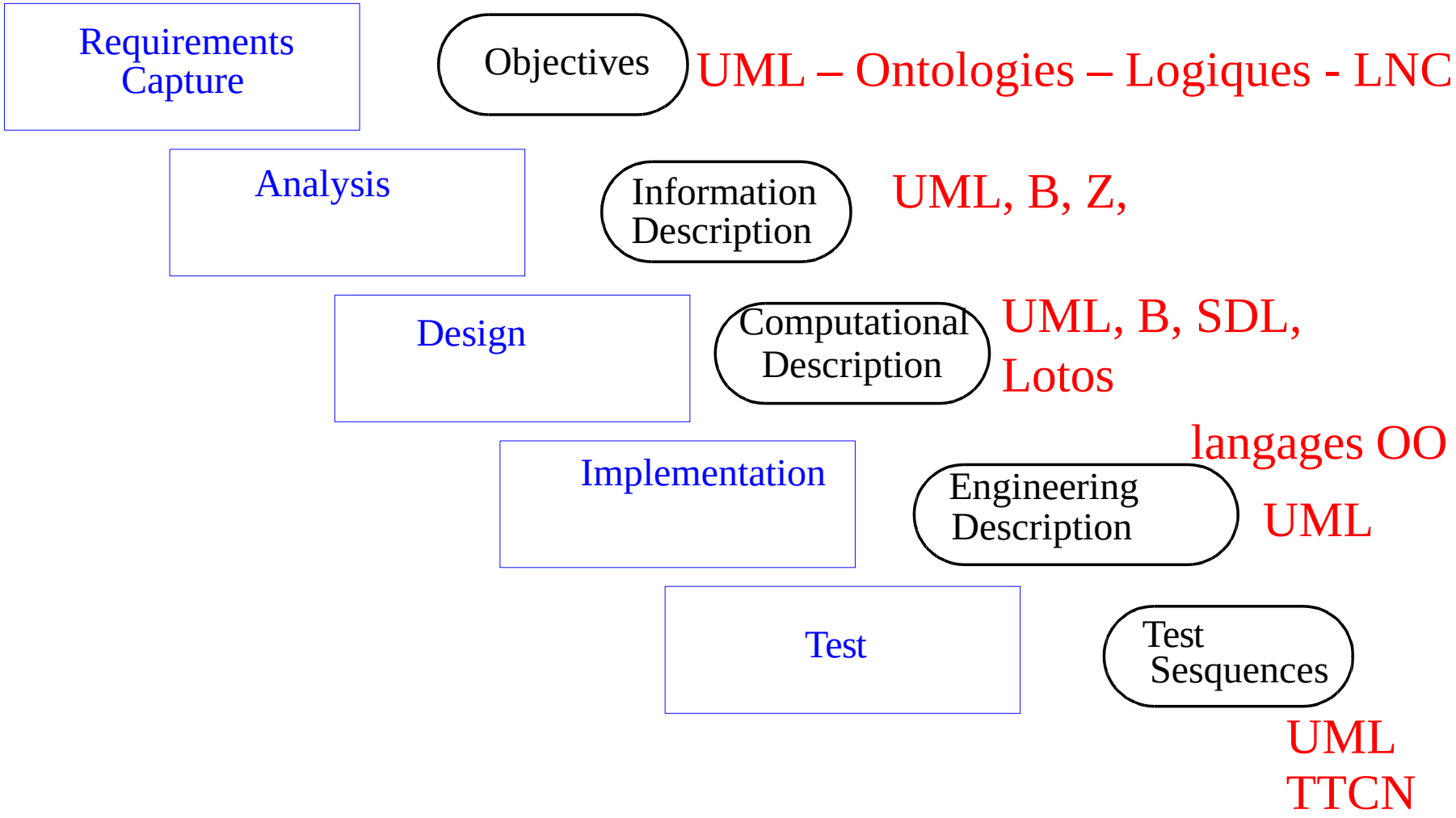
Cycle de Développement du logiciel : approche formelle



- P dérive de S2 : génération de code

- s dérive de S2 : génération de test

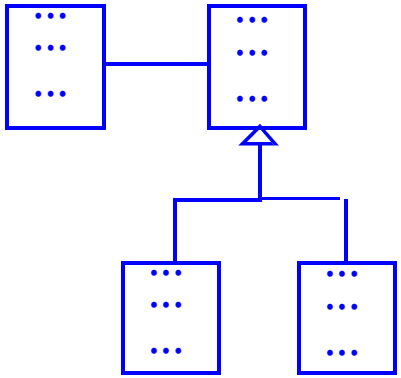
Cycle de Développement du logiciel et méthodes formelles



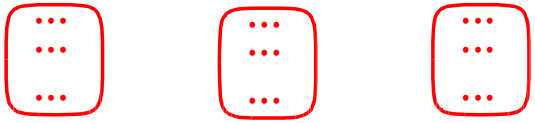
Exemple de cadre architectural : le service de téléphonie

Point de vue Information

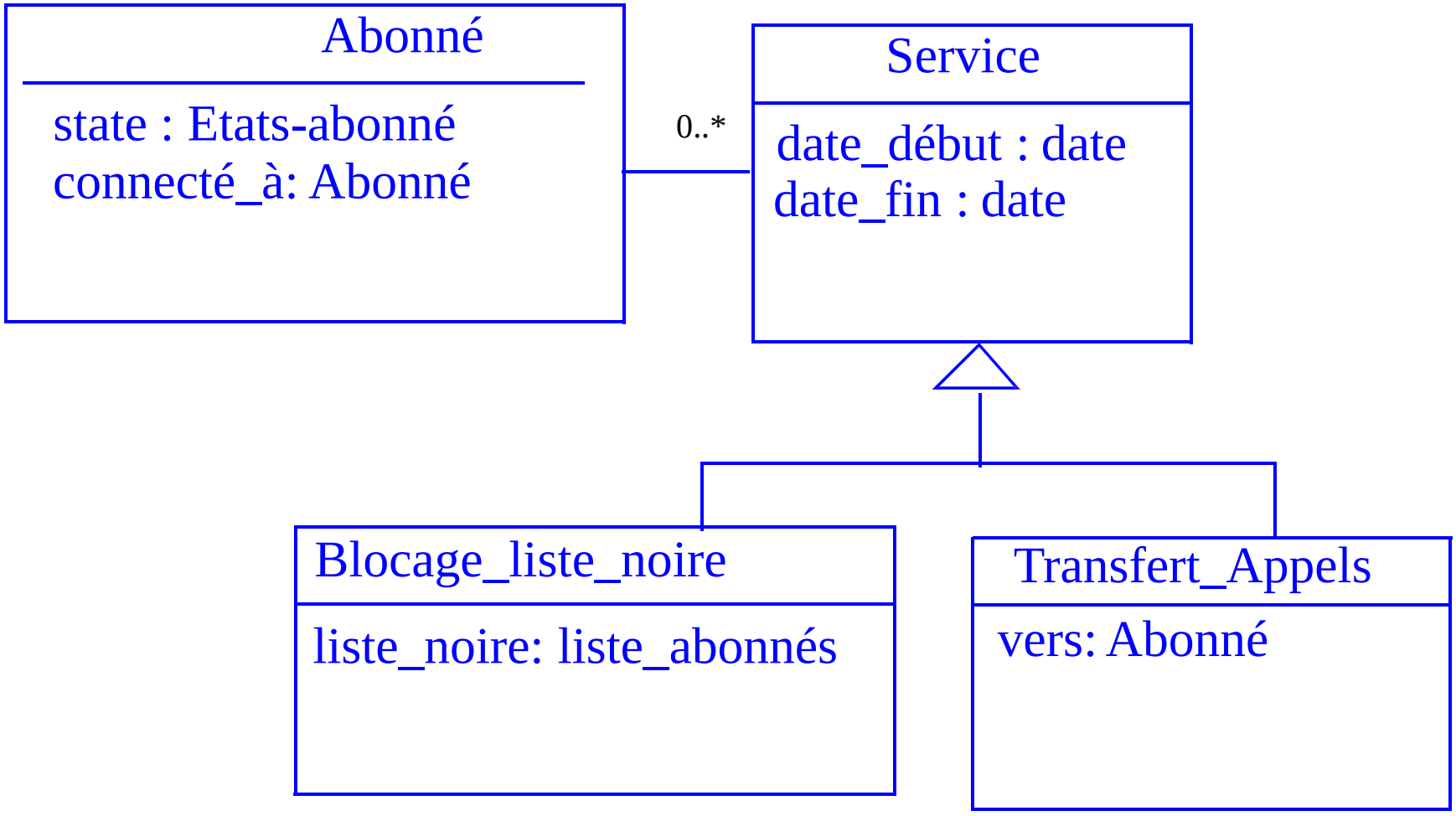
Structure d'état : classes & relations



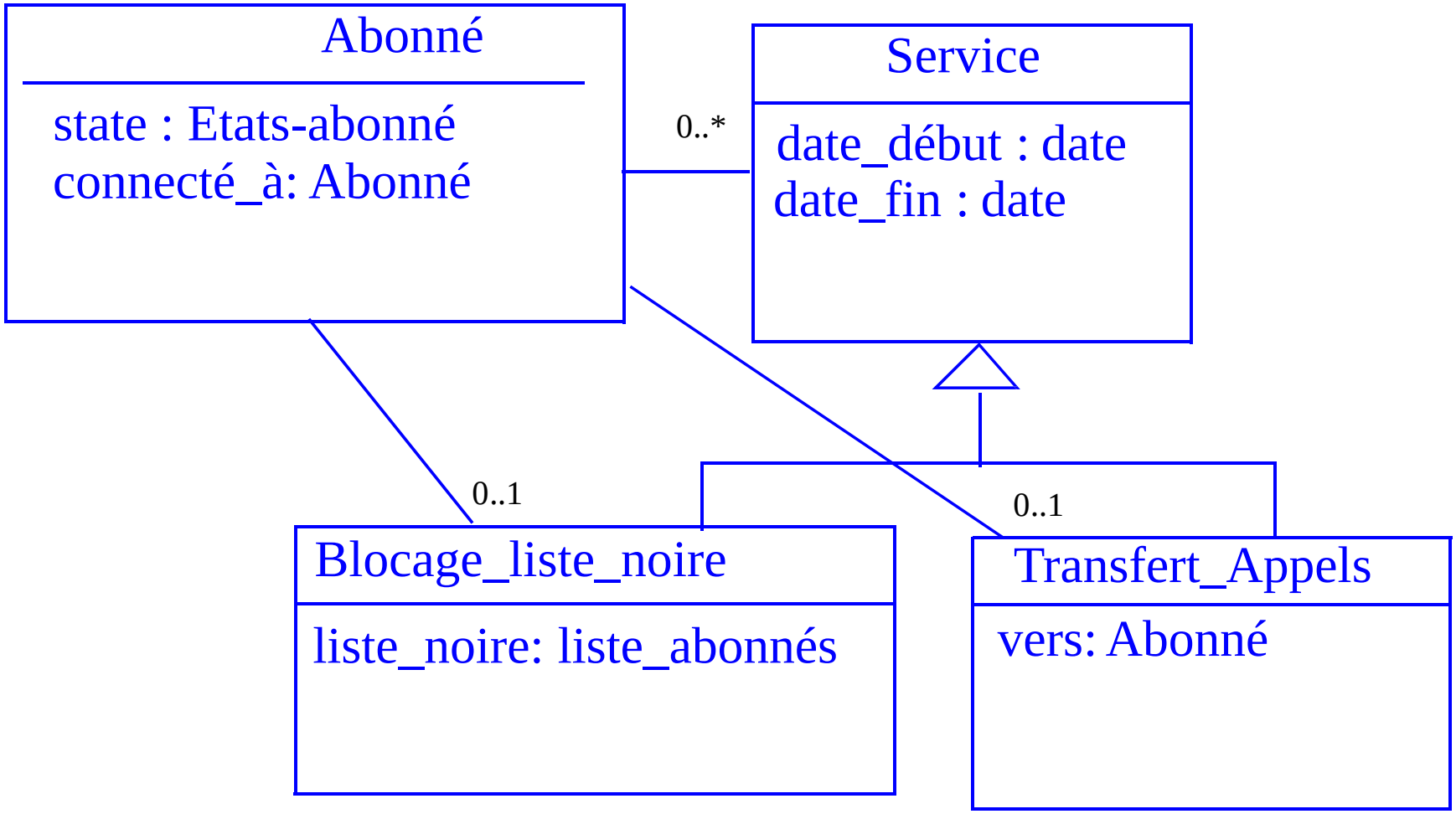
Evolutions de l'état : les actions



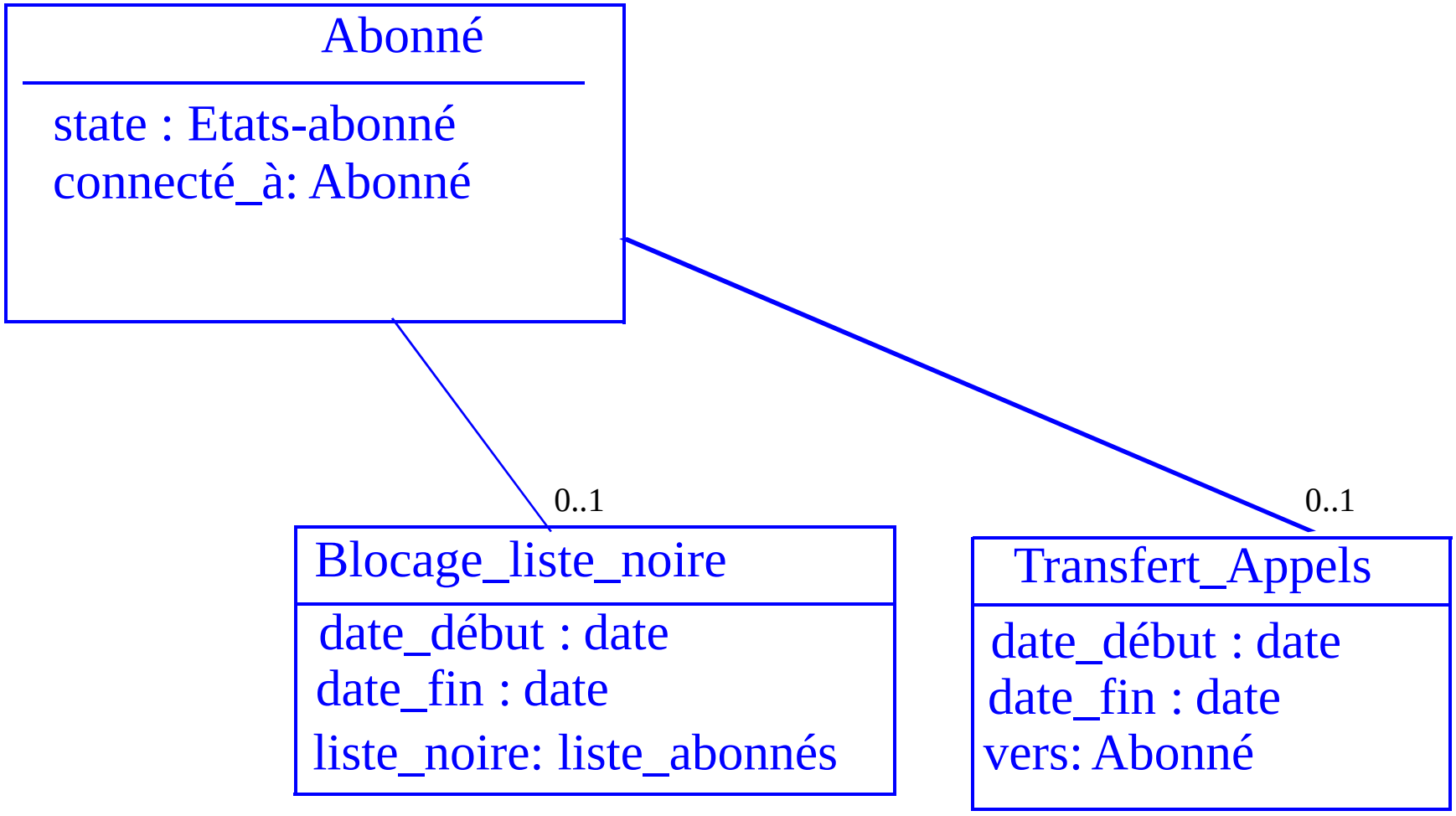
Point de Vue Information : structure d'état



Point de Vue Information : structure d'état



Point de Vue Information : structure d'état



Point de Vue Information : Actions

Action decrocher(a: Abonné)

Pre_Condition : a.state = raccroché

Post_Condition : a'.state = décroché

Action composer(a: abonné, b:abonné)

Pre_Condition : a.state = décroché

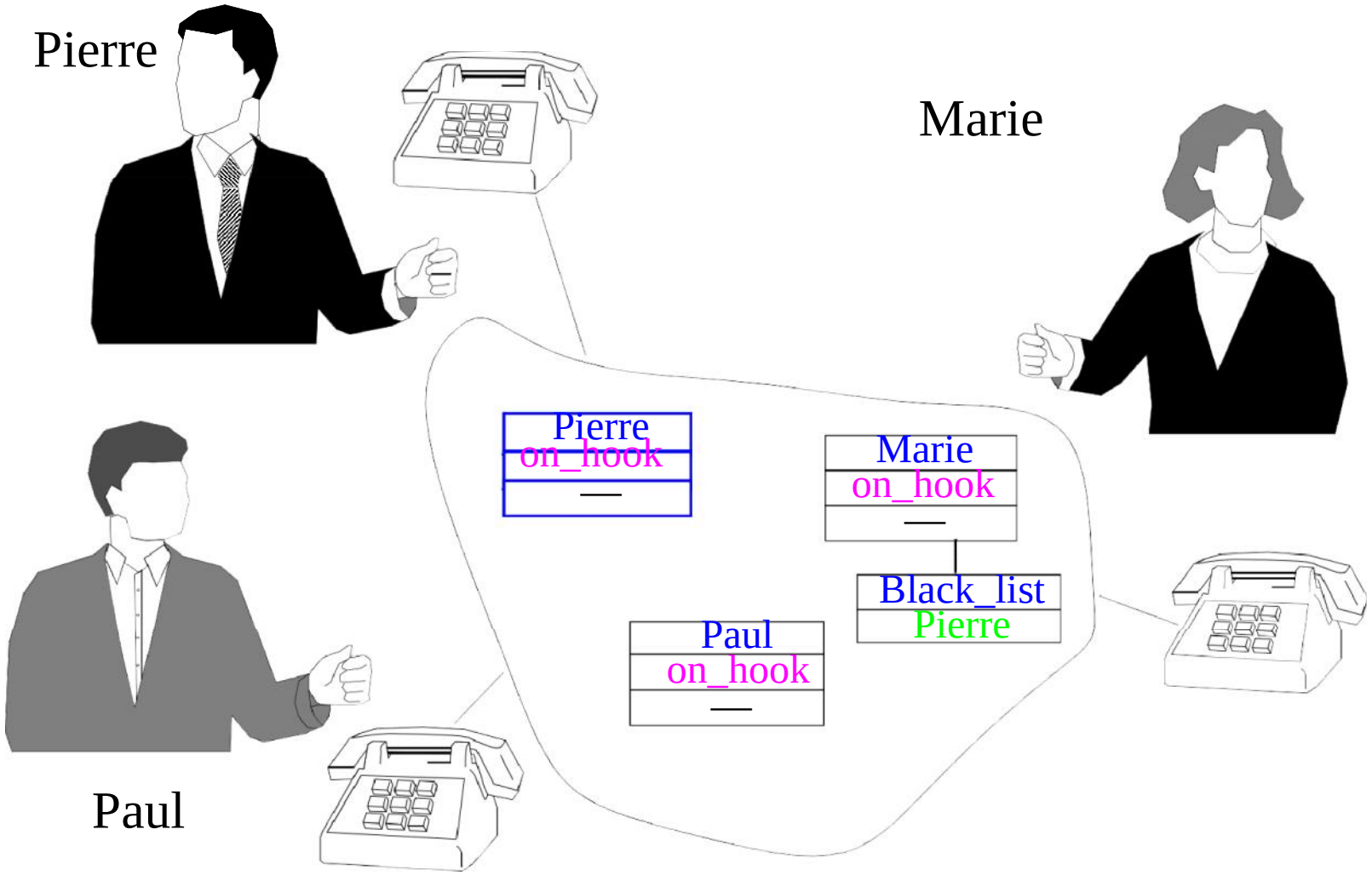
Post_Condition :

if b.state = raccroché & a $\notin$ b.liste_noire

then b'.state = sonnerie & b'.connecté_à = a &
a'.state = indication & a'.connecté_à = b

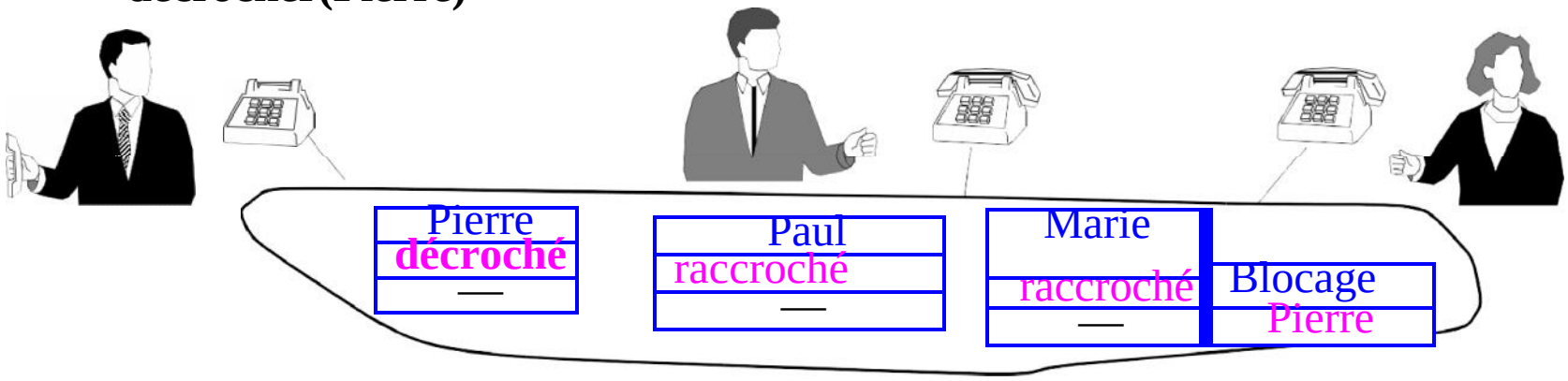
else b'.state = b.state & a'.state = occupé

Point de Vue Information : scénario

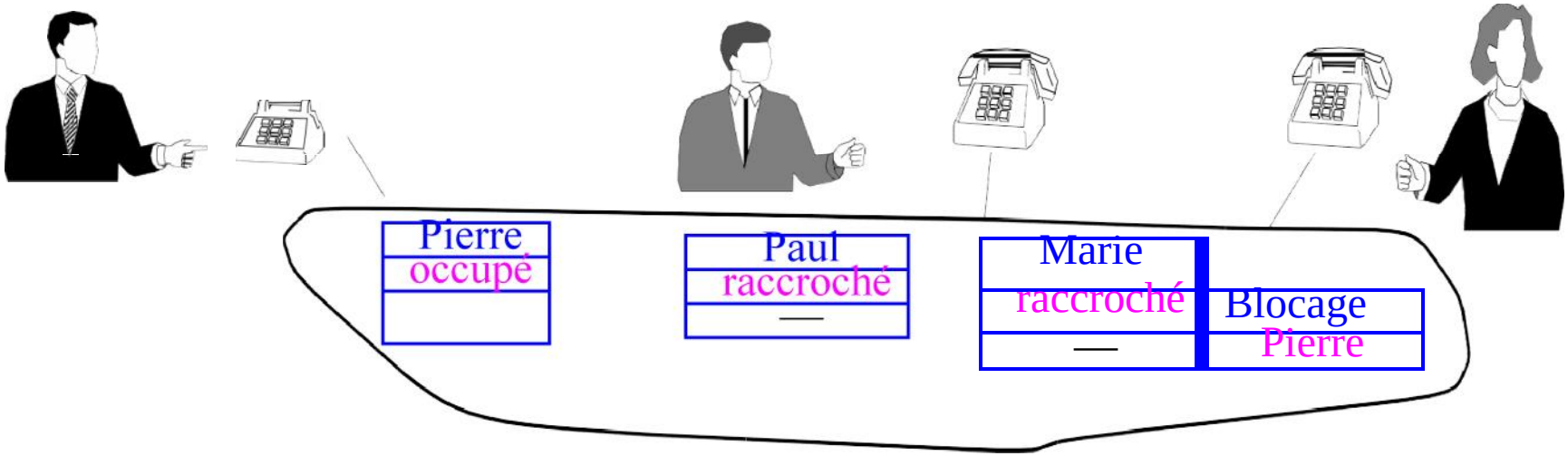


Point de Vue Information : scénario

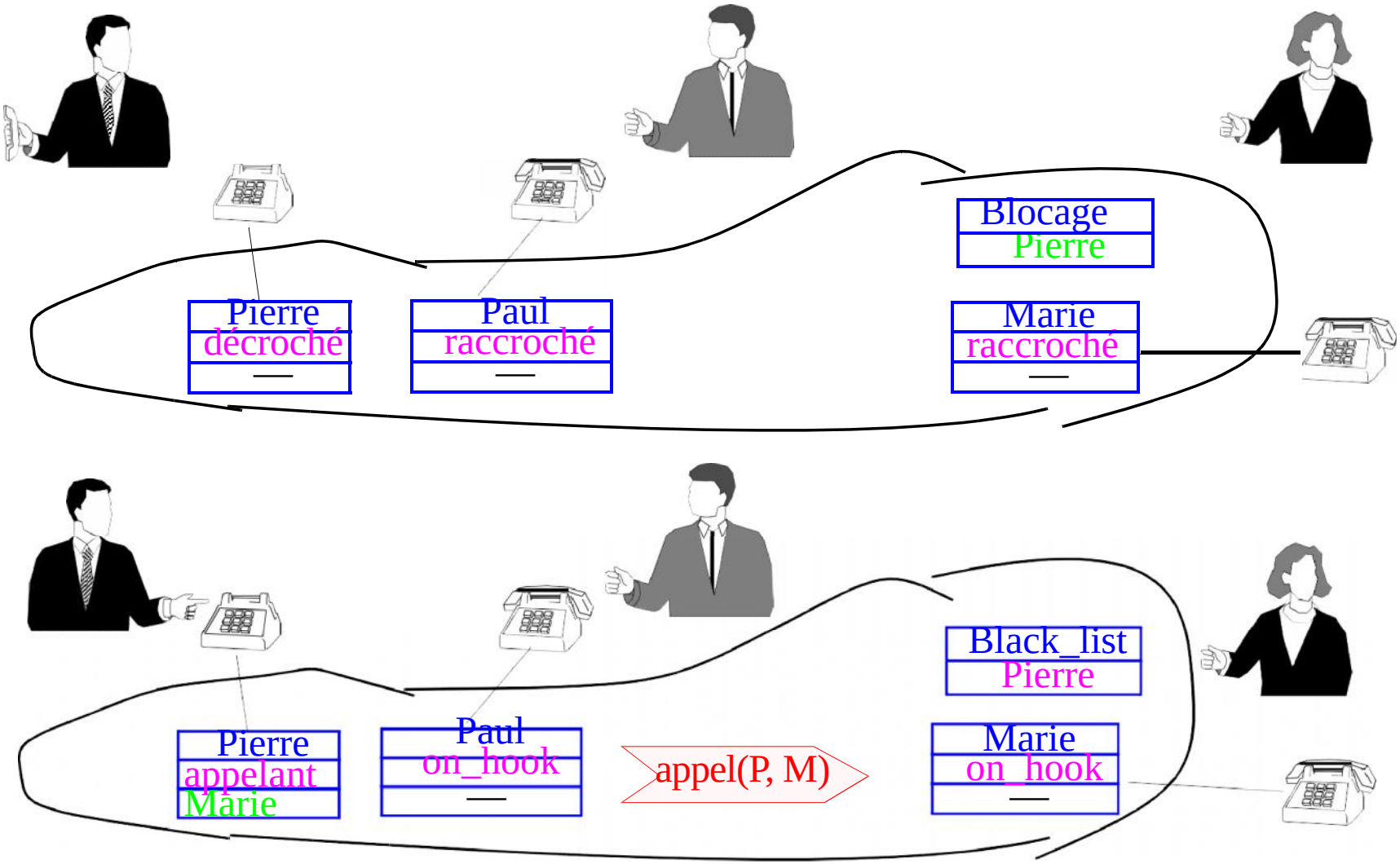
decrocher(Pierre)



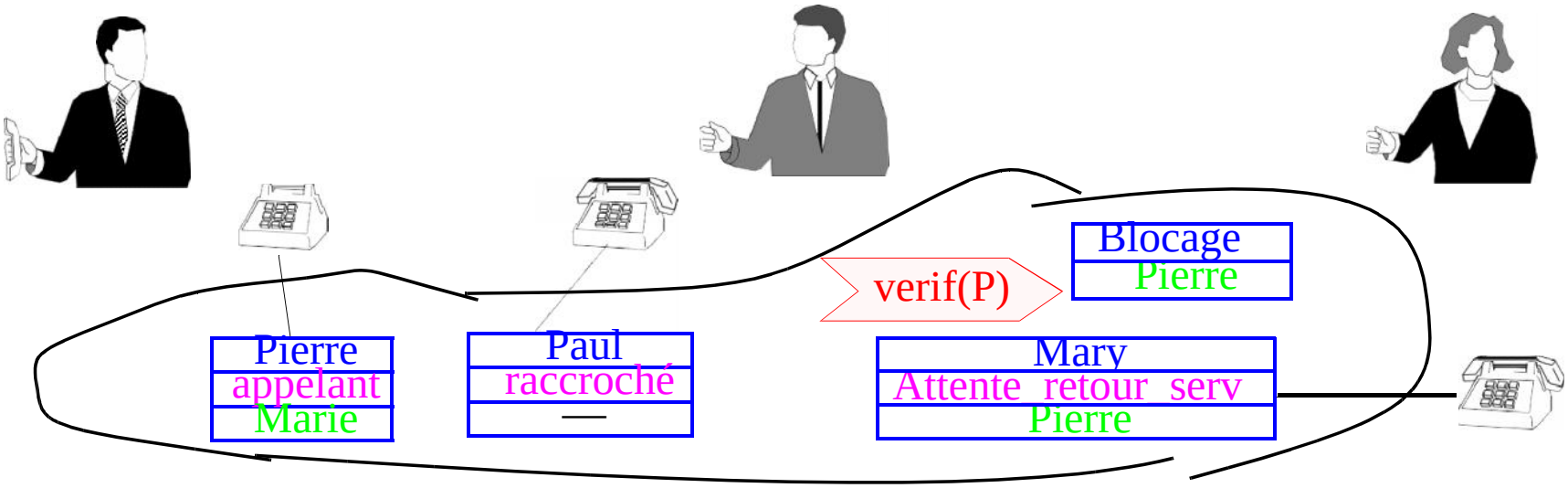
composer(Pierre, Marie)

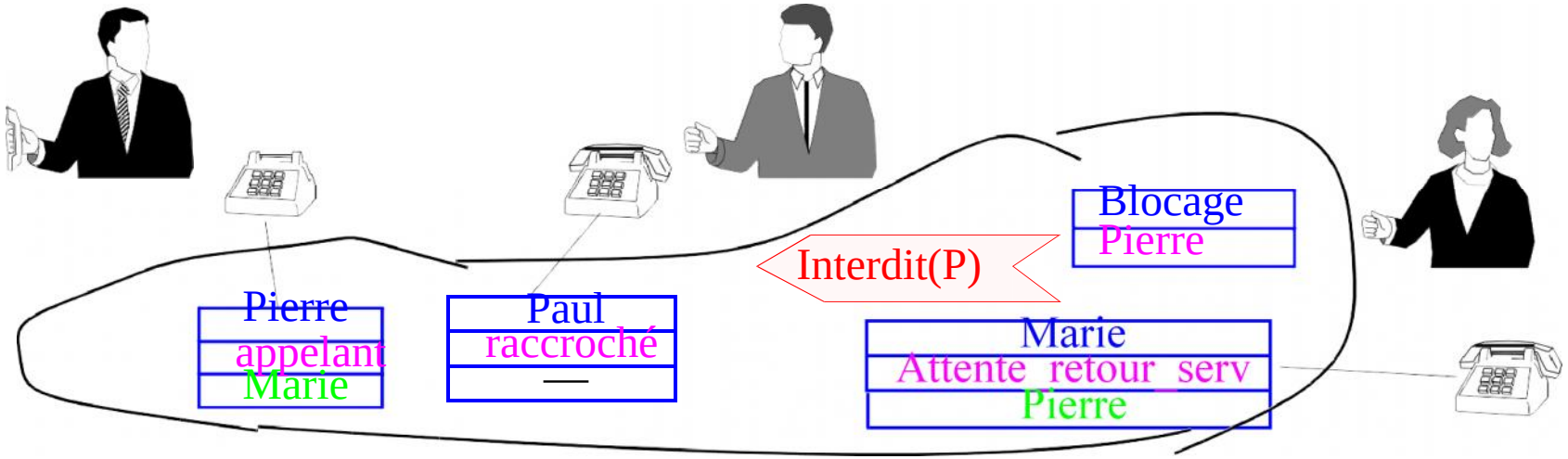


Point de Vue Traitement: scénario

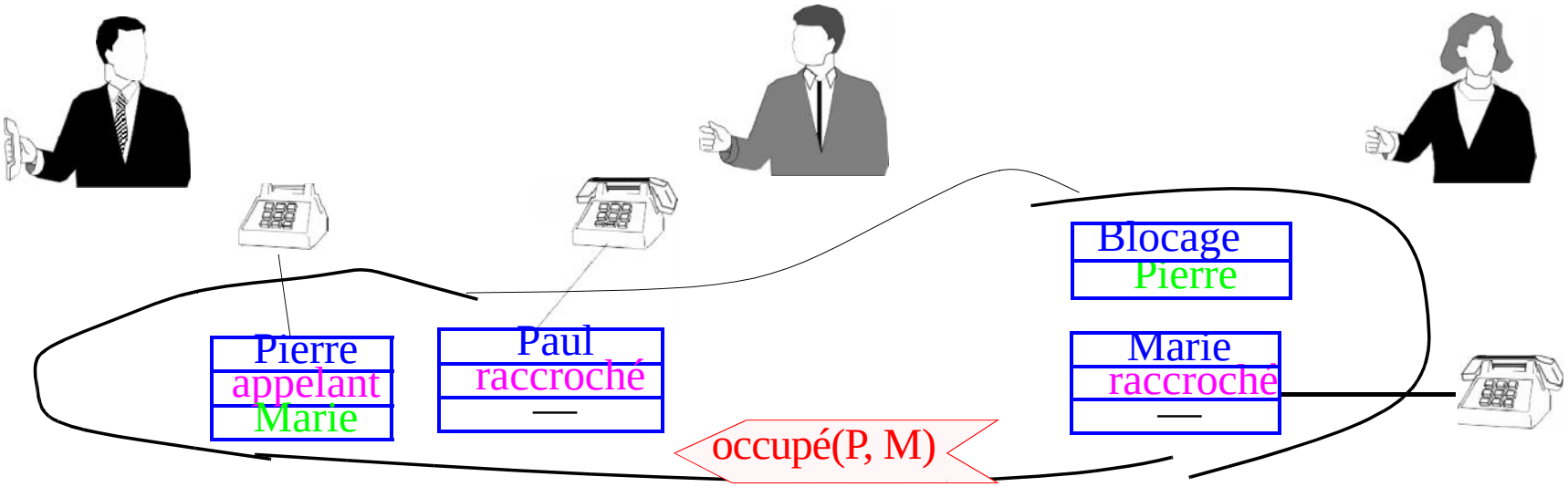


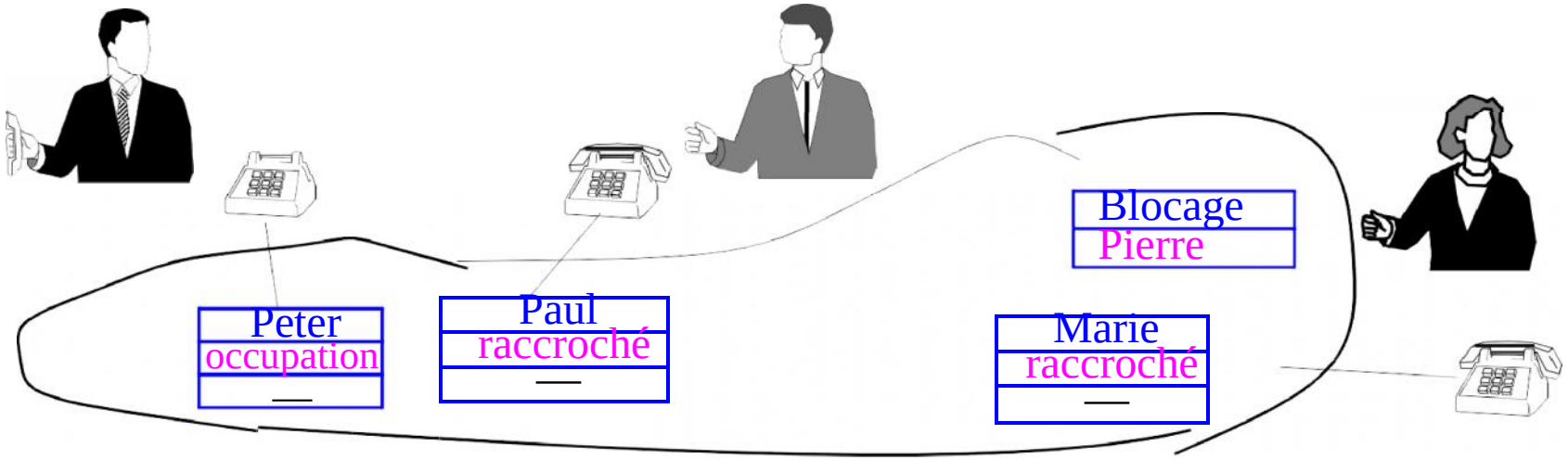
Point de Vue Traitement: scénario



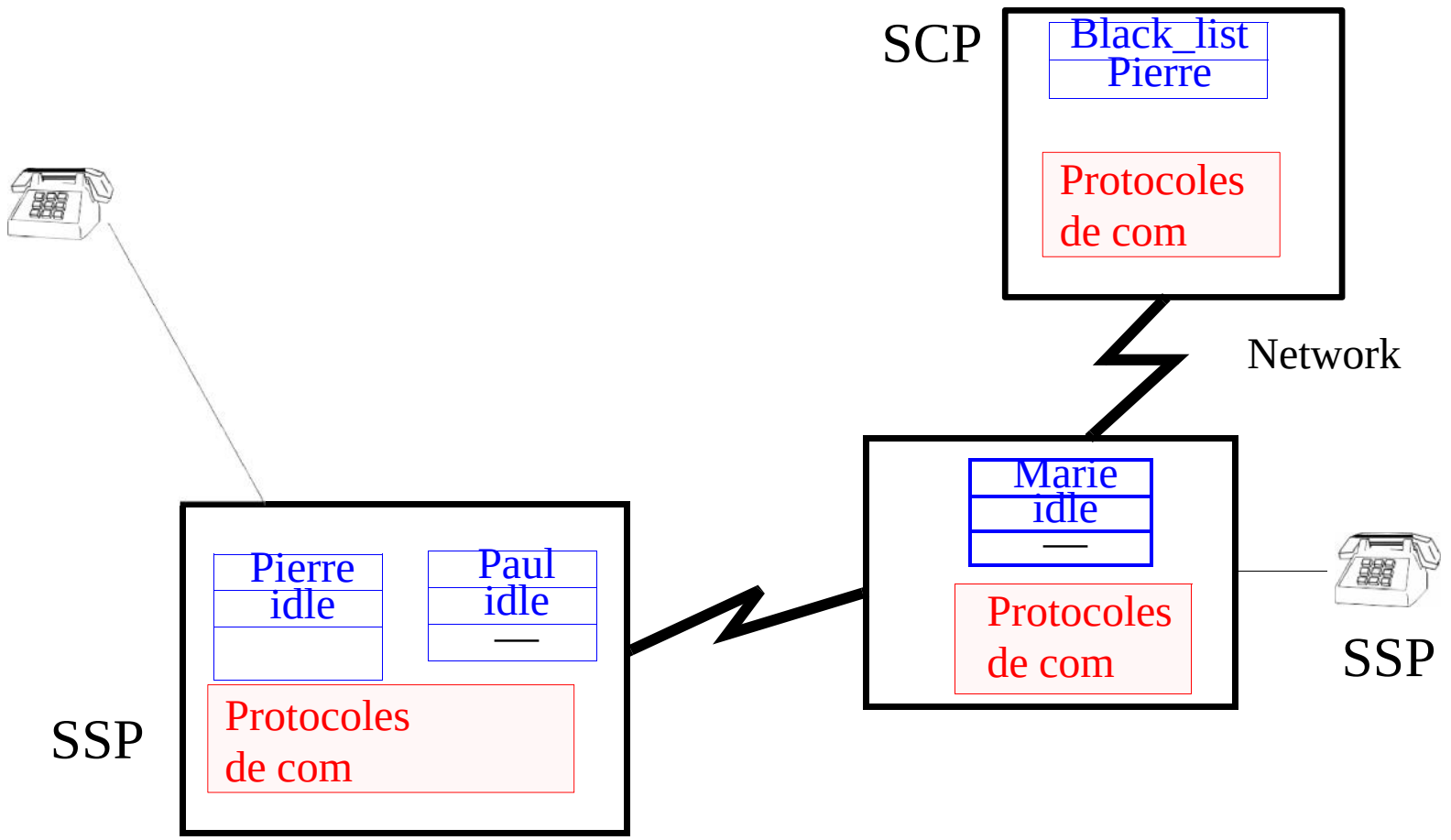


Point de Vue Traitement : scénario





Point de Vue Ingénierie



Exemple de sémantique formelle

SDL - un langage de l'ITU

SDL-76
SDL/GR notation Graphique
Concep ts Telecom

SDL- 80
SDL/PR notation textuelle
1 page semantique

SDL- 84
semantique pour données et structure

Liaison with ISO

SDL-88
semantique Mathematiquel
nouveaux concepts (Service)

SDL- 92
orientation Objet
non-determinisme
test, librairies

SDL-96
Améliorations minimes

SDL-2000
State Charts et UML

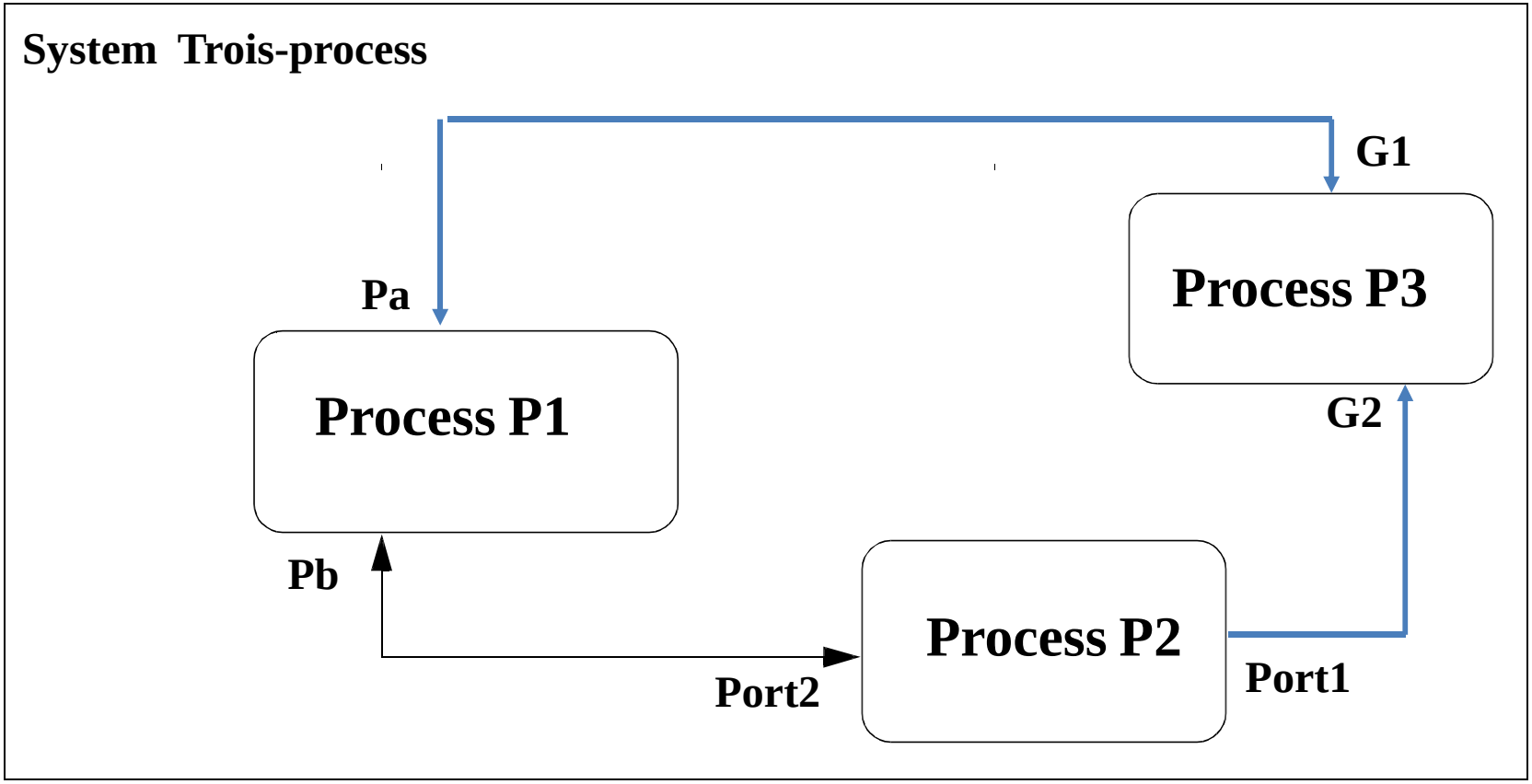
SDL - un langage de l'ITU

- **Syntaxe**: Textuelle et Graphique
- **Comportement** :

  Machines à Etats Finis Etendues Communicantes
  Variables locales à chaque machine

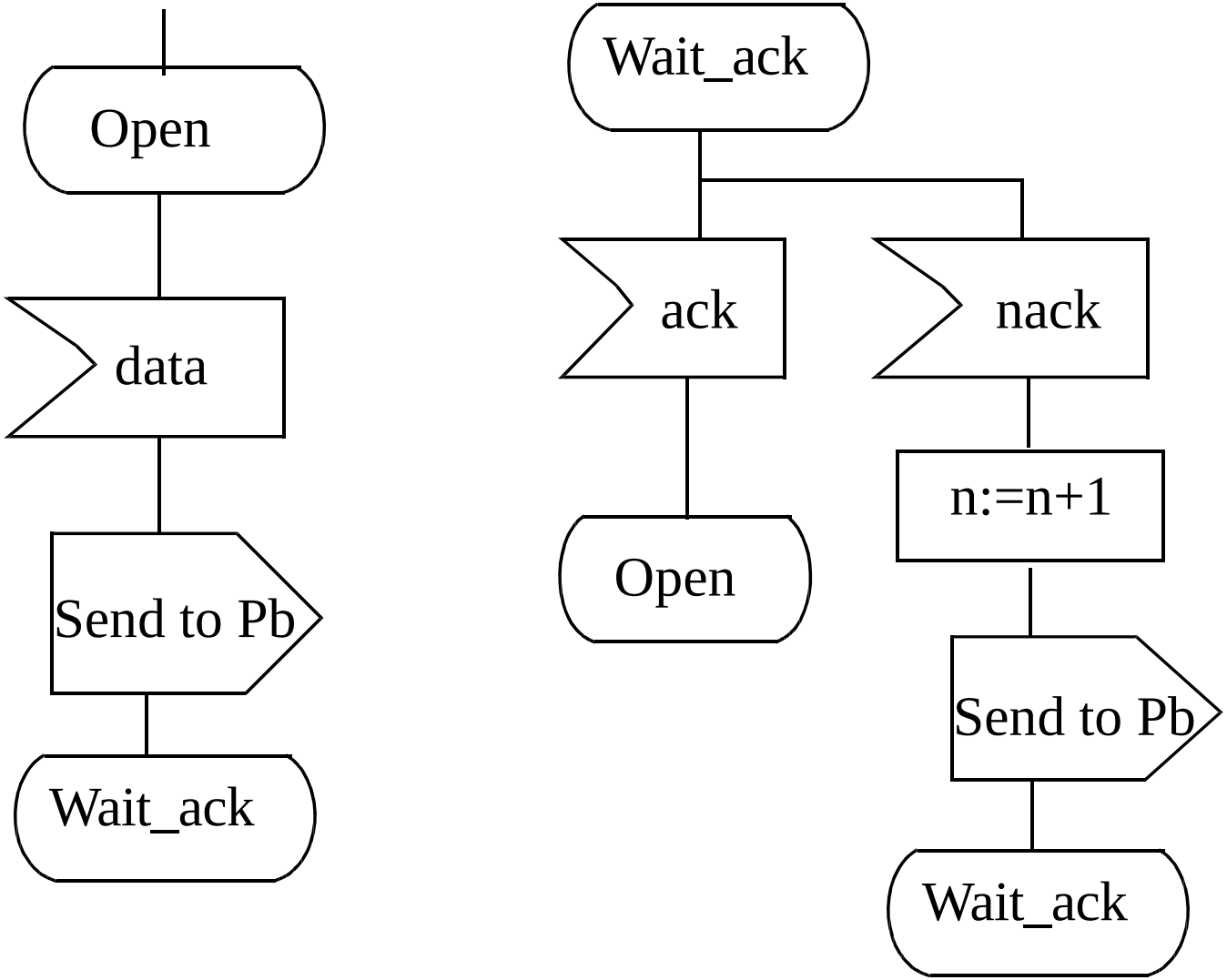
- **Communication** : par messages typés et à travers des cannaux FIFO
- **Concurrence** : Les Machines s'executent en parallèle asynchrone

Un système SDL est un ensemble de process interconnecté

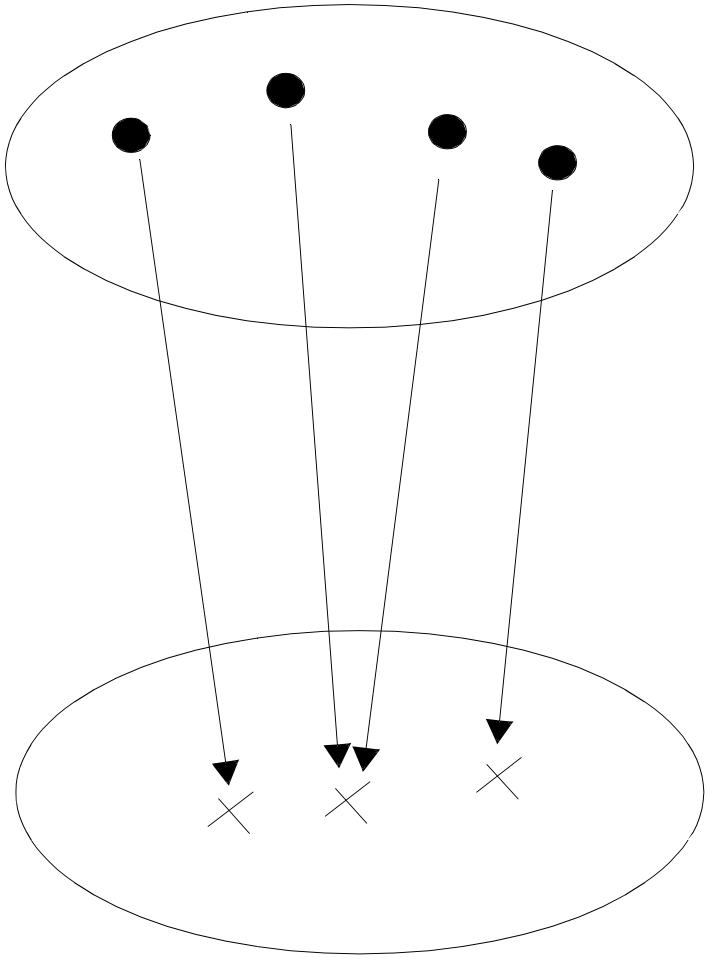


Comportement des process donné par un diagramme d'Etat

PROCESS P1



Modèle sémantique de SDL



Syntaxe : SDL graphique
ou SDL textuel

Domaine sémantique :

Machines à Etats finis étendues

Sémantique de SDL : les automates communicants

Automate communicant $\mathcal{A}$ = <q, Q, M, T, P> avec :

- Q un ensemble d'états $Q = Q_A \cup Q_T$
(une union disjointe d'états d'attente Q_A, et d'états transiants Q_T)

- q état initial

- T un ensemble de transitions $T = T_S \cup T_R \cup T_I$
(une union disjointe de transitions de réception T_R, d'émission T_S, et de transitions internes T_I)

- $T_S \subseteq Q_T \times M \times P \times Q$ $\quad T_R \subseteq Q_A \times M \times Q$ $\quad T_I \subseteq Q_T \times \{\tau\} \times Q$
(τ représente l'action interne)

- P est un ensemble fini de ports

Exemple : diagramme d'Etat du Process P1

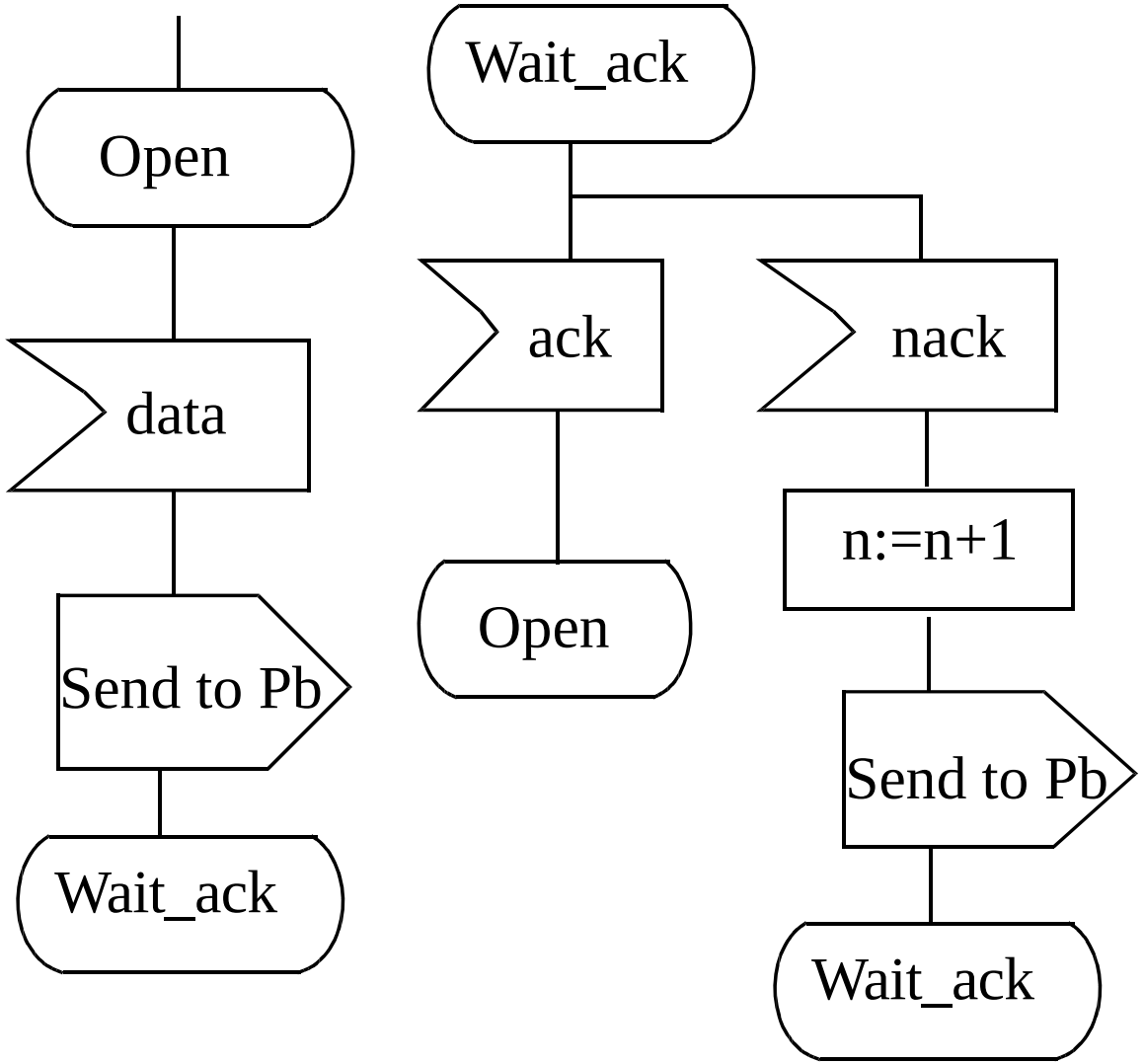
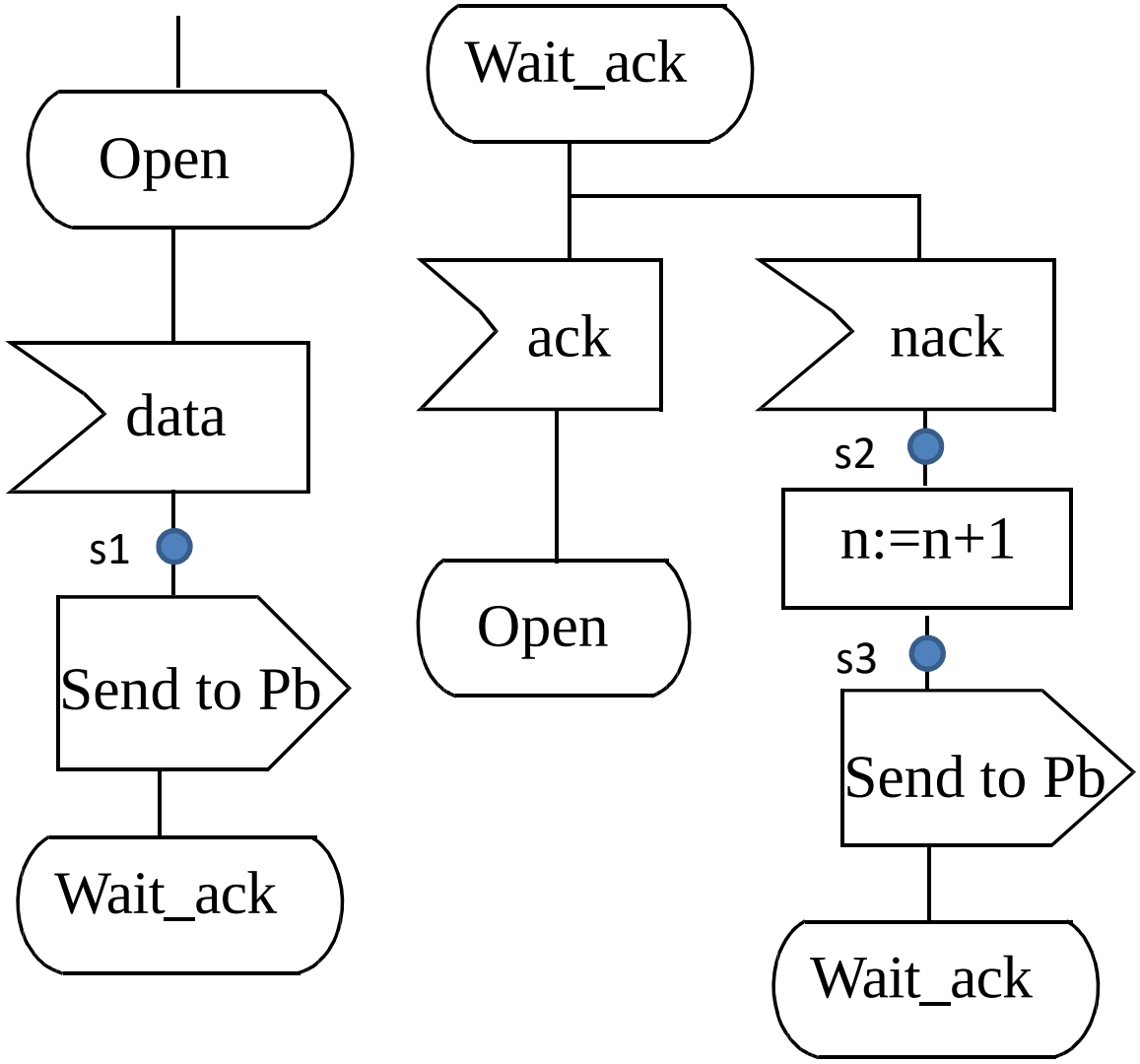


Diagramme d'Etat du Process P1 – nommage des états transiants



Exemple : Automate du Process P1 $\mathcal{A}_{P1} = \langle q, Q, M, T, P \rangle$

- Q ensemble d'états $Q = Q_A \cup Q_T$

avec $Q_A = \{ \text{Open, Wait\_Ack} \}$
et $Q_T = \{ s1, s2, s3 \}$

- état initial : $q =$ Open

- T un ensemble de transitions $T = T_S \cup T_R \cup T_I$

Réceptions $T_R = \{$ (Open, data, s1), (Wait_ack, ack, open), (Wait_ack, nack, s2)$\}$

Emissions $T_S = \{$ (s1, send, Pb, Wait_ack), (s3, send, Pb, Wait_ack) $\}$

Internes $T_I = \{$ (s2, τ, s3) $\}$

- P est un ensemble fini de ports = { Pa, Pb }

Un système d'automates communicants

Soit $\mathcal{A}_1$ ... $\mathcal{A}_n$ n automates communiquants et soit $\mathcal{P}$ l'union des ports P_i et soit $\mathcal{L}$ un ensemble de liens entre les ports, çàd : $\mathcal{L} \subseteq \mathcal{P} \times \mathcal{P}$
avec $\mathcal{L}$ vérifiant : $\forall$ p, p_1, p_2 si $(p,p_1) \in \mathcal{L}$ et $(p, p_2) \in \mathcal{L}$ alors $p_1=p_2$

* un système d'automates communicant est un ensemble d'automates munis de liens : $\mathcal{S}$ = ($\mathcal{A}_1$ ... $\mathcal{A}_n$, $\mathcal{L}$)

* un système est fermé lorsque tous les ports de ses process sont liés:
  $\forall$ p $\in \mathcal{P}$ $\exists$ p' $\in \mathcal{P}$ avec (p,p') $\in \mathcal{L}$ ou (p',p) $\in \mathcal{L}$

* l'état global, s, d'un système est le produit cartésien des états des états des automates et des états des files d'attentes :

$$s \in ((Q_1 \times \mathcal{M}^*) \times ... \times (Q_n \times \mathcal{M}^*))$$

où $\mathcal{M}=M_1 \cup ... \cup M_n$ et $\mathcal{M}^*$ est l'ensemble de séquences d'éléments de $\mathcal{M}$

s est donc de la forme $(q_1, fifo_1)$, ... , $(q_n, fifo_n)$

Un système d'automates communicants : Trois-process

$\mathcal{S}_{\text{Trois-process}} = (\mathcal{A}_{P1}, \mathcal{A}_{P2}, \mathcal{A}_{P3}, \mathcal{L})$

Avec $\mathcal{L} = \{ (G1, Pa), (Pa, G1), (Port2, Pb), (Pb, Port2), (Port1, G2) \}$

Transitions globales d'un système d'automates communicants

Emission : si $q_i \in Q_{iT}$ et $q_i \xrightarrow{m,p} q'$ et $(p,p') \in \ell$ et $p' \in P_j$

alors : $(q_1, fifo_1) \ldots (q_i, fifo_i) \ldots\ldots (q_j, fifo_j) \ldots\ldots (q_n, fifo_n)$

$\longrightarrow$ $(q_1, fifo_1) \ldots (q', fifo_i) \ldots (q_j, enfile(m,fifo_j)) \ldots (q_n, fifo_n)$

Réception: $q_i \in Q_{iA}$ et $q_i \xrightarrow{m} q'$ et $m = tete(fifo_i)$ et $\forall\, j \neq i : q_j \in Q_{jA}$

alors : $(q_1, fifo_1) \ldots\ldots (q_i, fifo_i) \ldots\ldots (q_j, fifo_j) \ldots (q_n, fifo_n)$

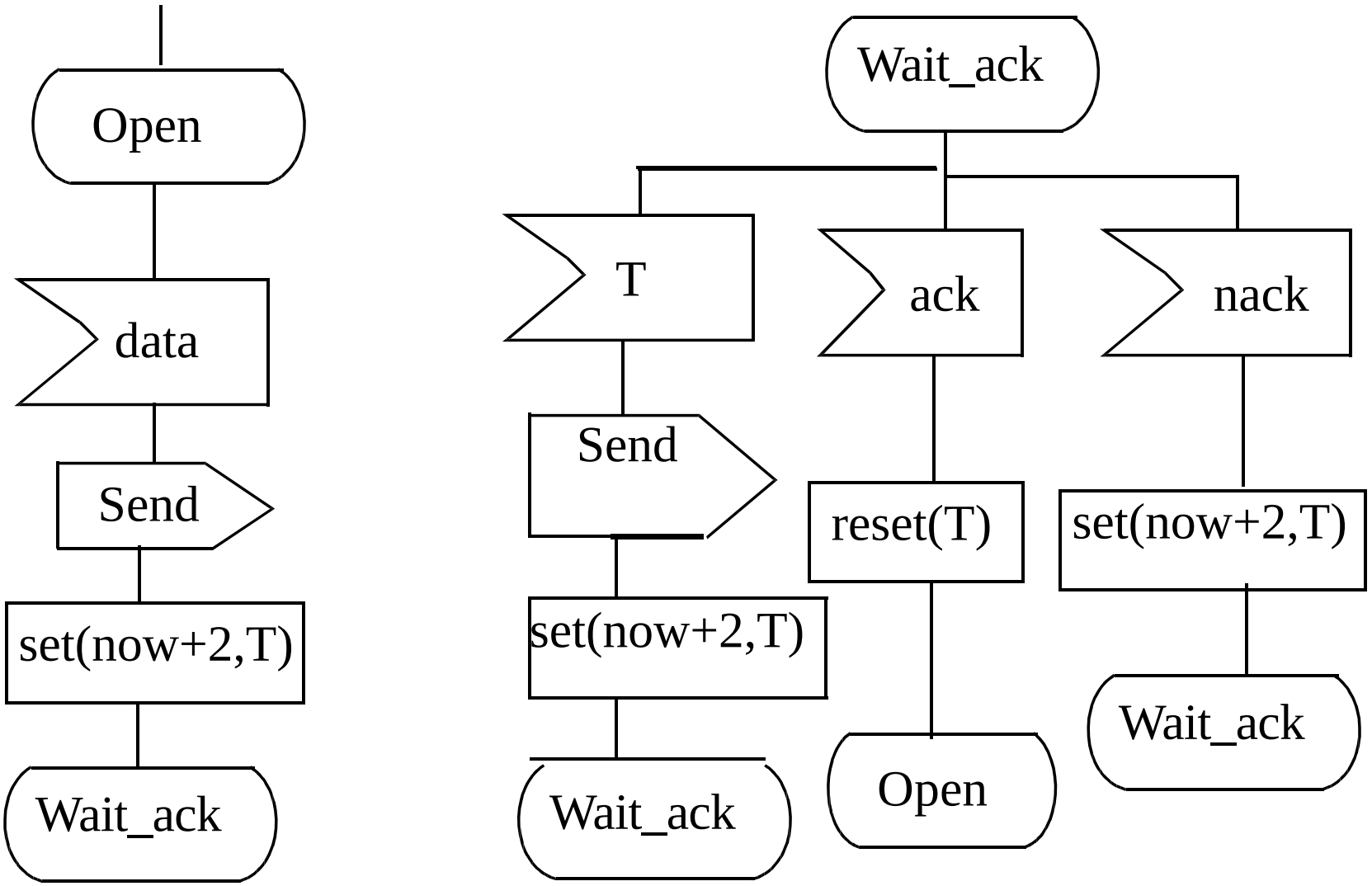
$\longrightarrow$ $(q_1, fifo_1) \ldots (q', defile(fifo_i)) \ldots (q_j, (fifo_j) \ldots (q_n, fifo_n)$

Transition interne : si $q_i \in Q_{iT}$ et $q_i \xrightarrow{\tau} q'$

alors : $(q_1, fifo_1) \ldots (q_i, fifo_i) \ldots (q_j, fifo_j) \ldots (q_n, fifo_n)$

$\longrightarrow$ $(q_1, fifo_1) \ldots (q', fifo_i) \ldots (q_j, (fifo_j) \ldots (q_n, fifo_n)$

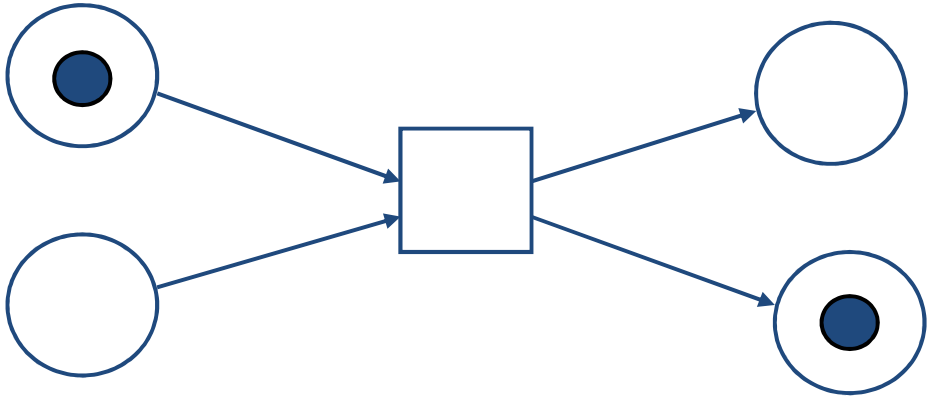
Un exemple de diagramme d'Etat (Process P1 avec timers)



Introduction rapide aux réseaux de Petri

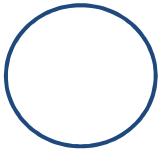
Elie Najm

Réseaux de Petri



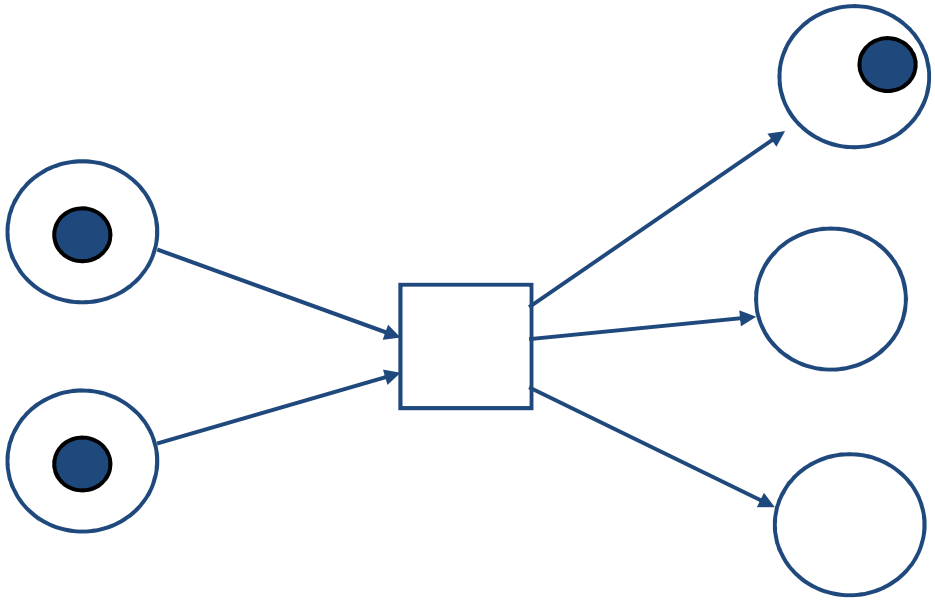
pre-conditions

post-conditions



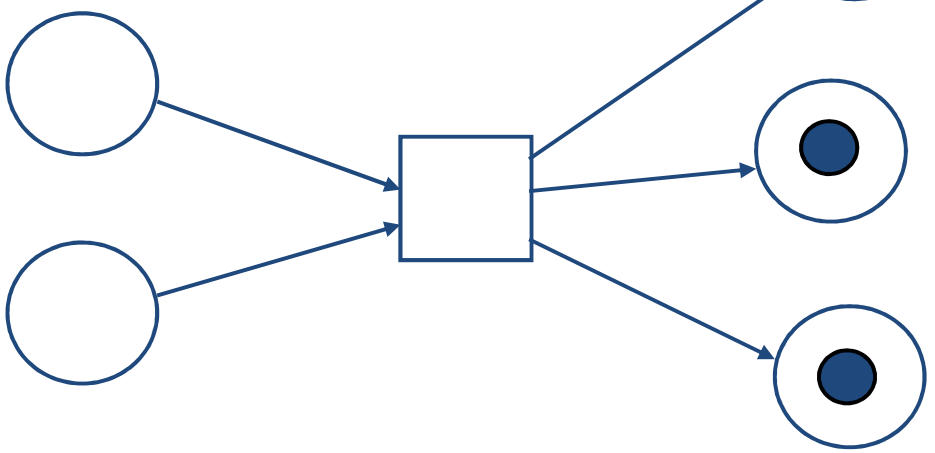
Place

Transition



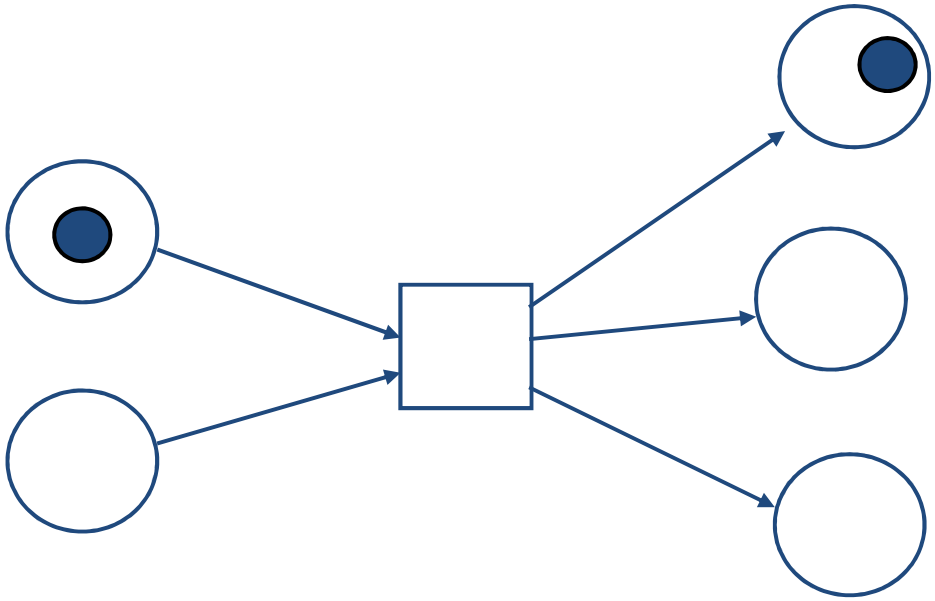
pre-conditions

post-conditions

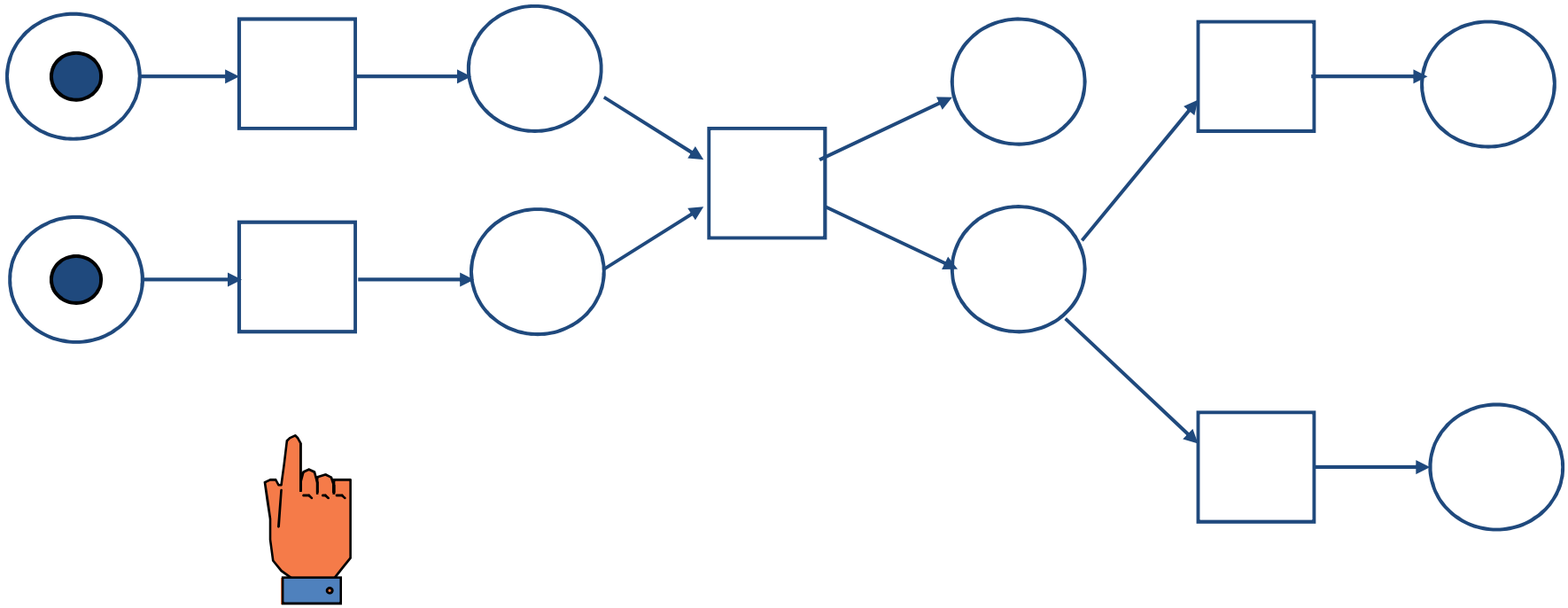


pre-conditions

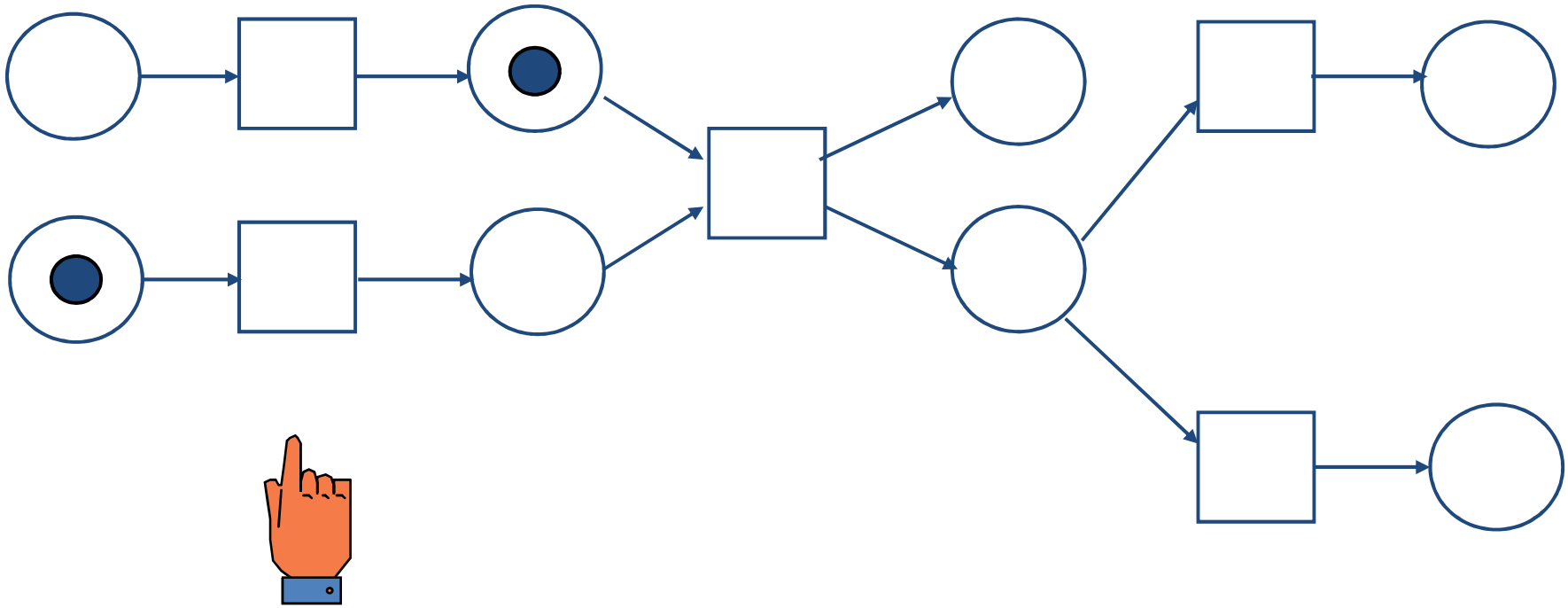
post-conditions



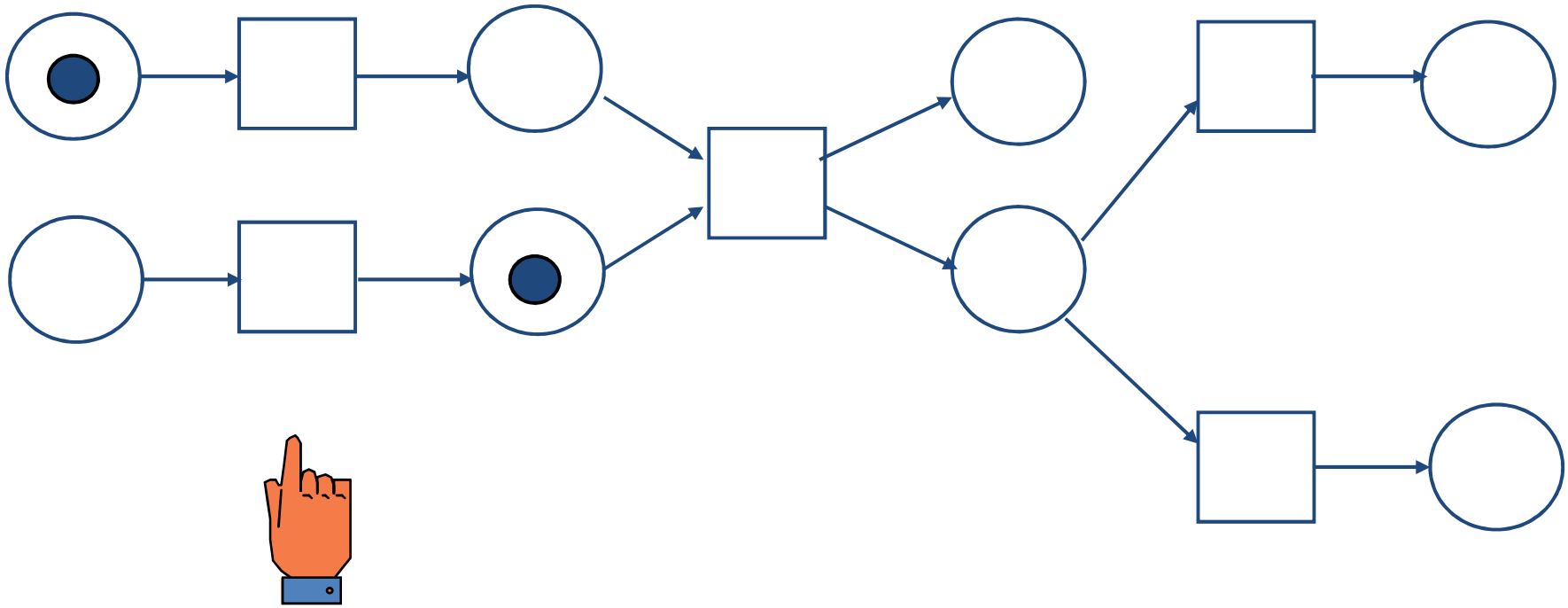
Transition non tirable



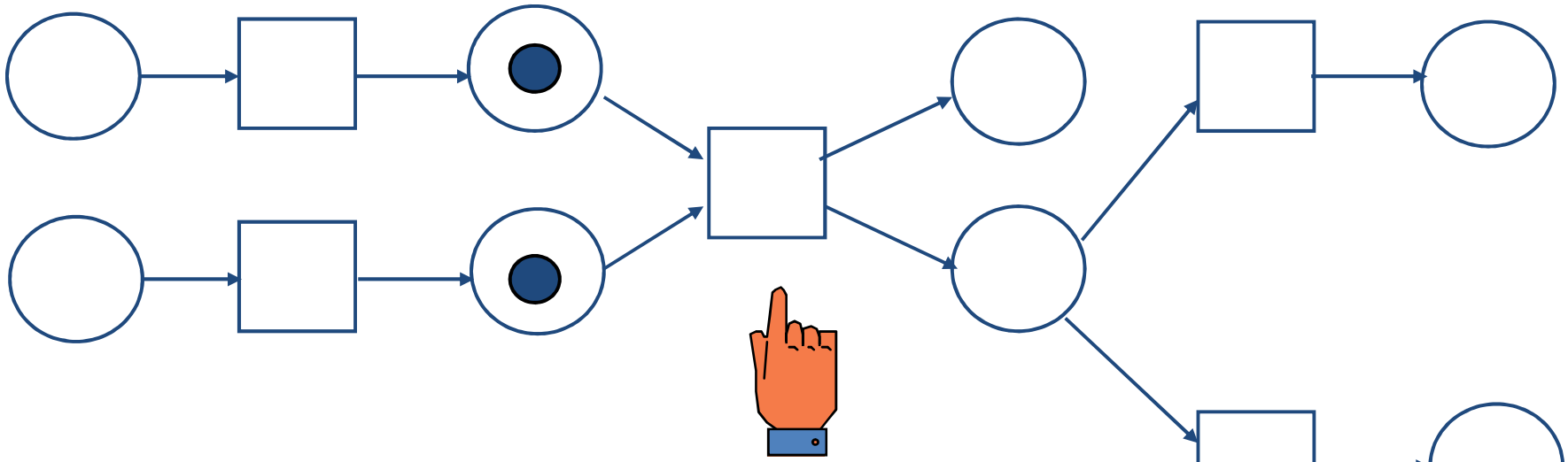
CONCURRENCE



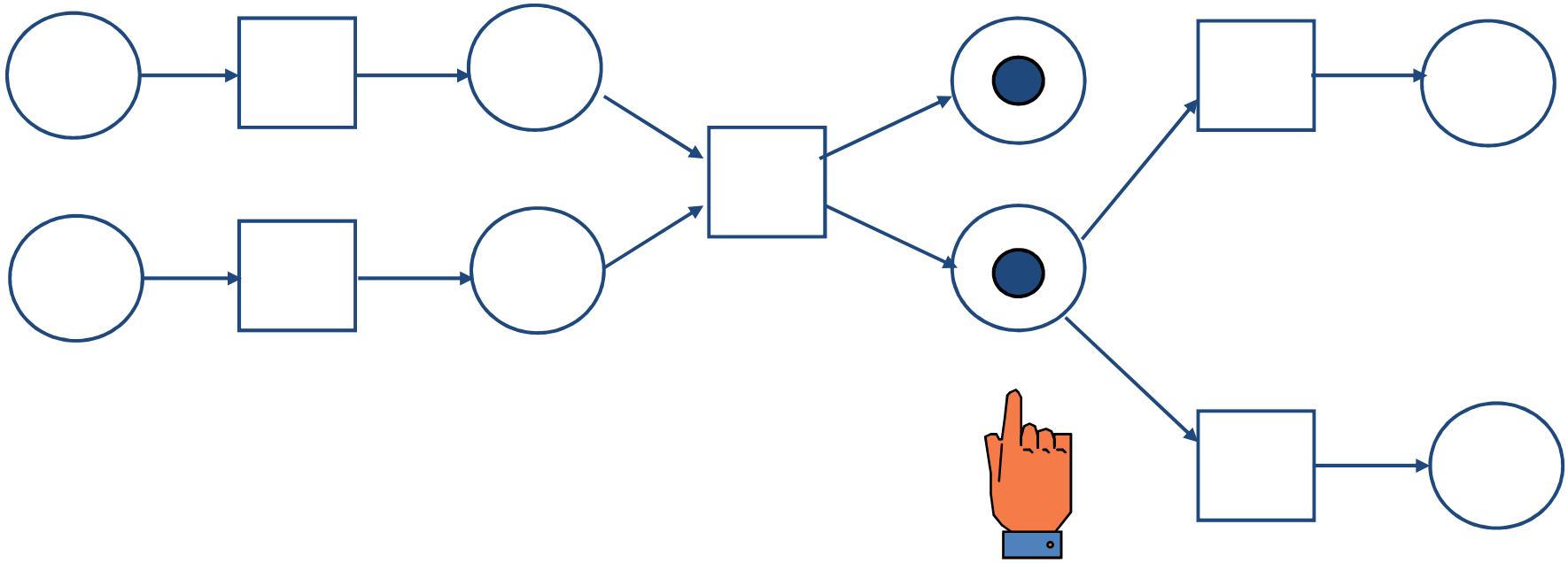
CONCURRENCE



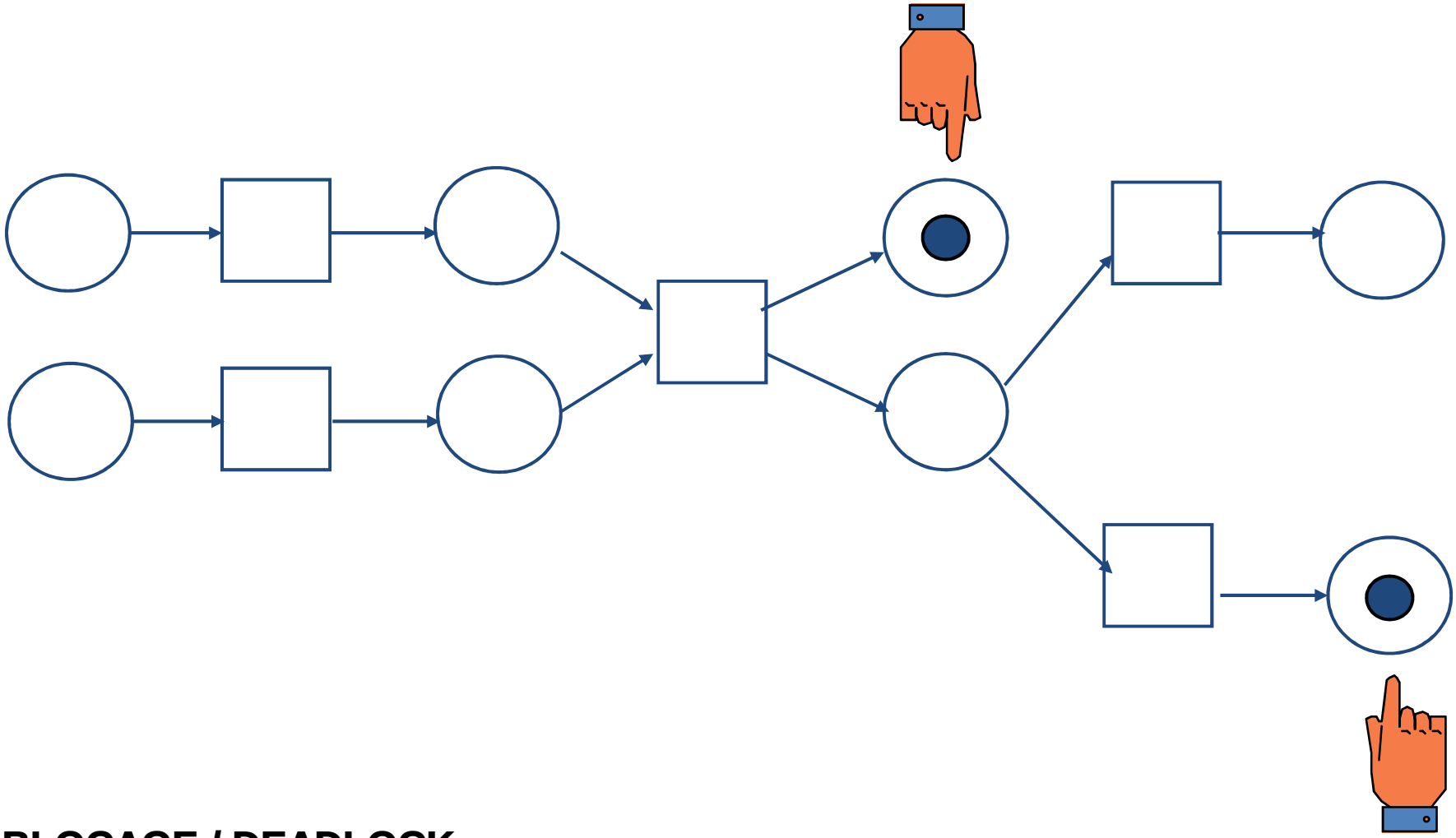
CONCURRENCE



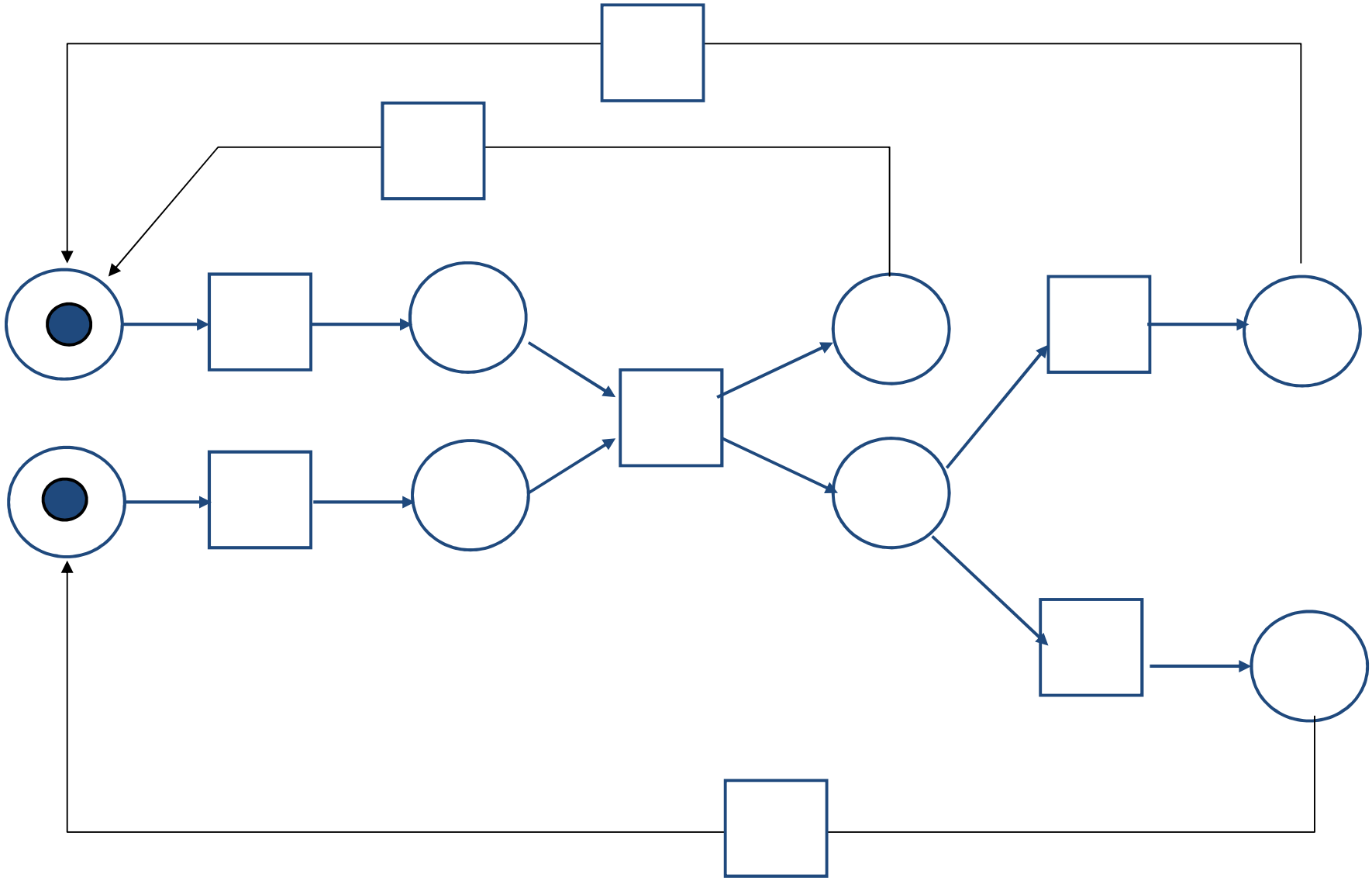
SYNCHRONISATION



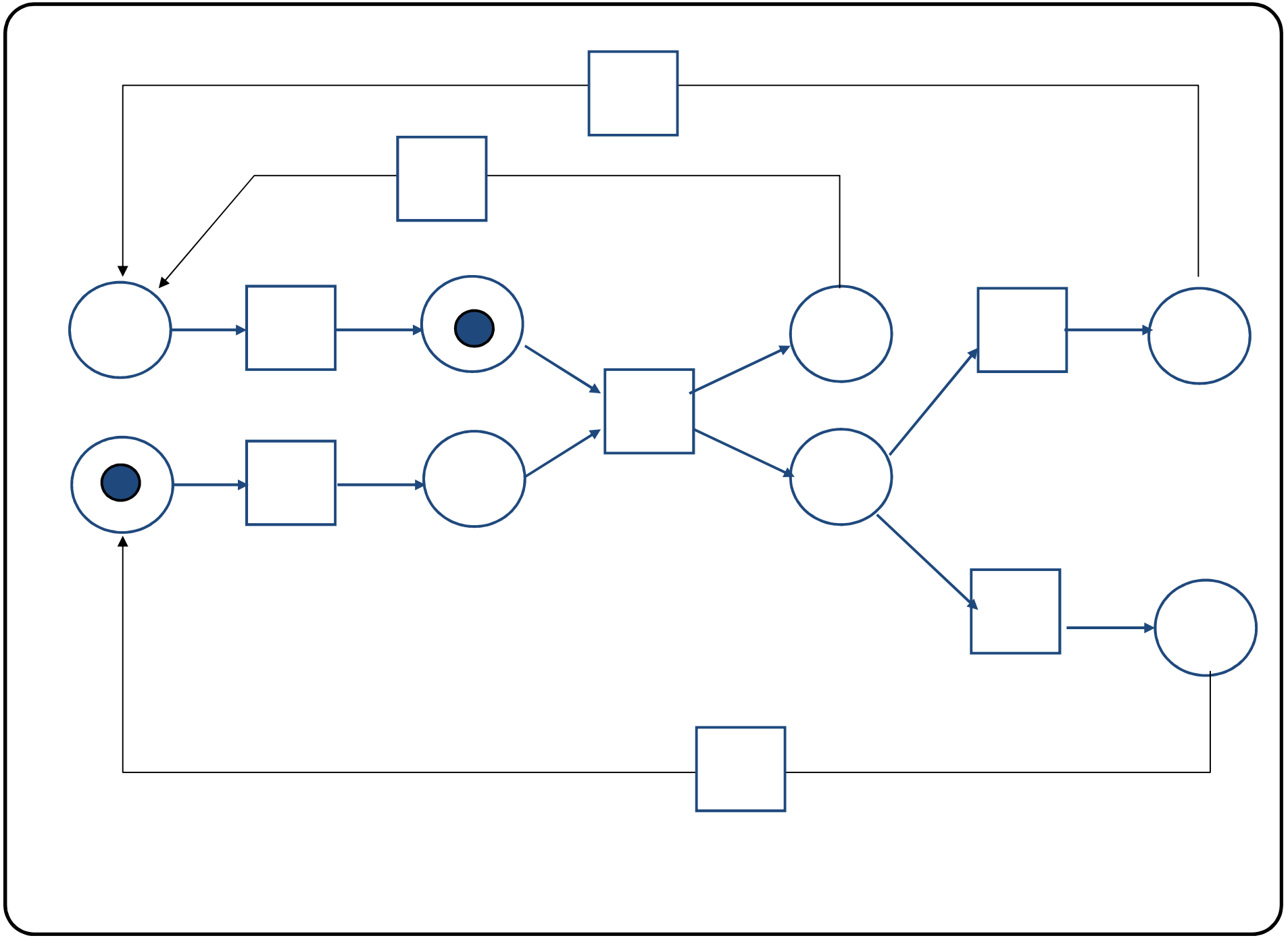
CONFLIT

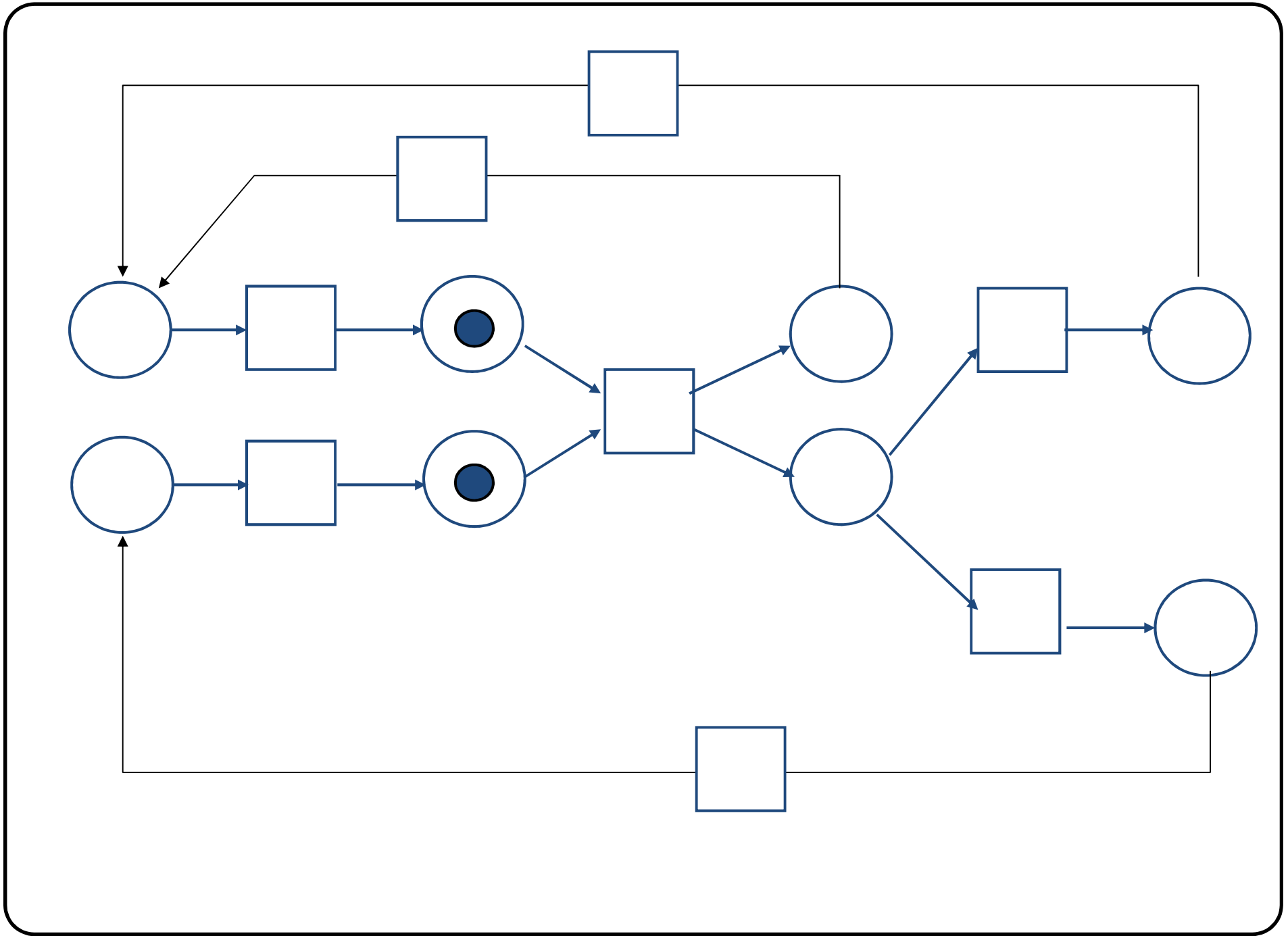


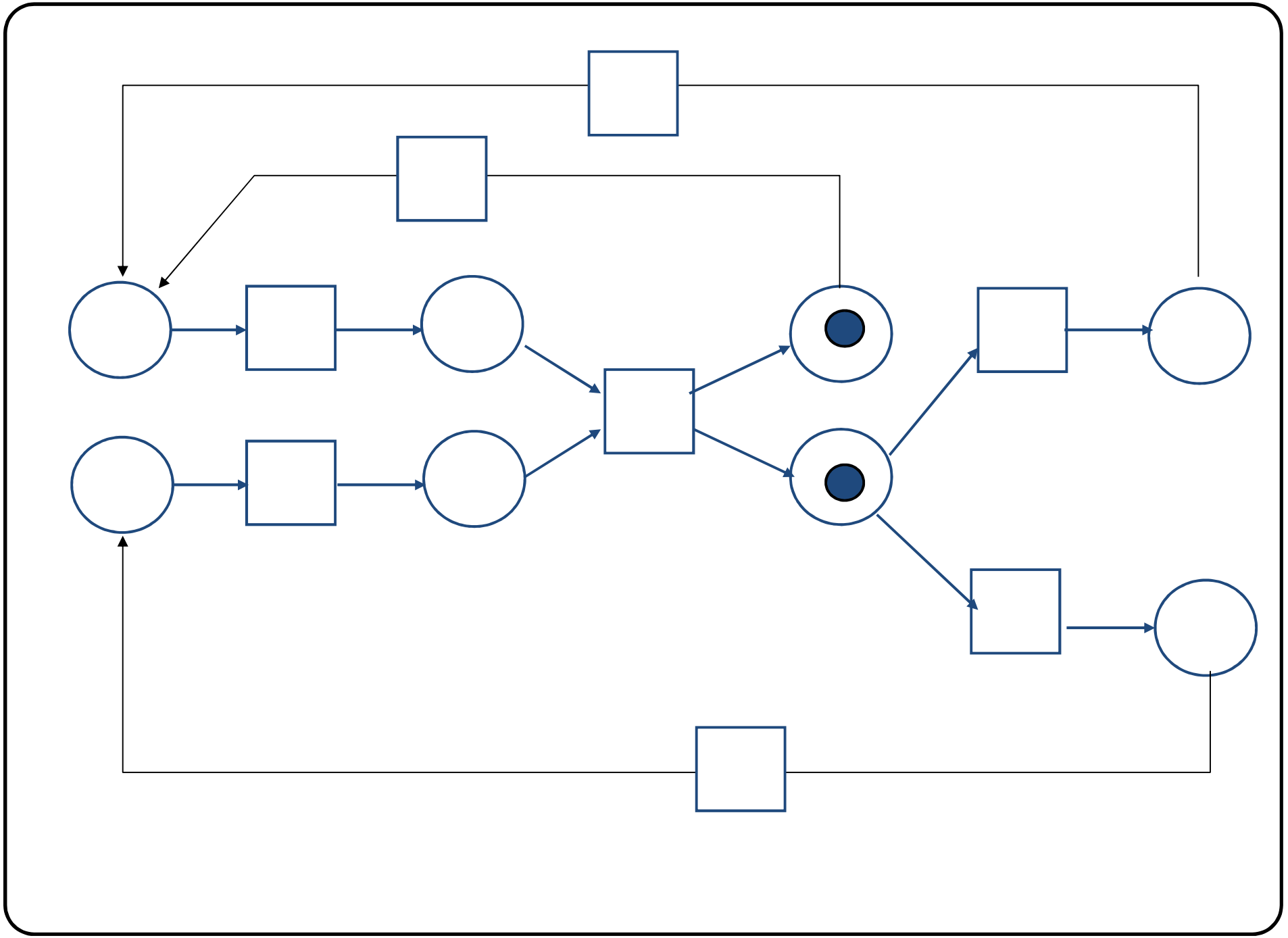
BLOCAGE / DEADLOCK

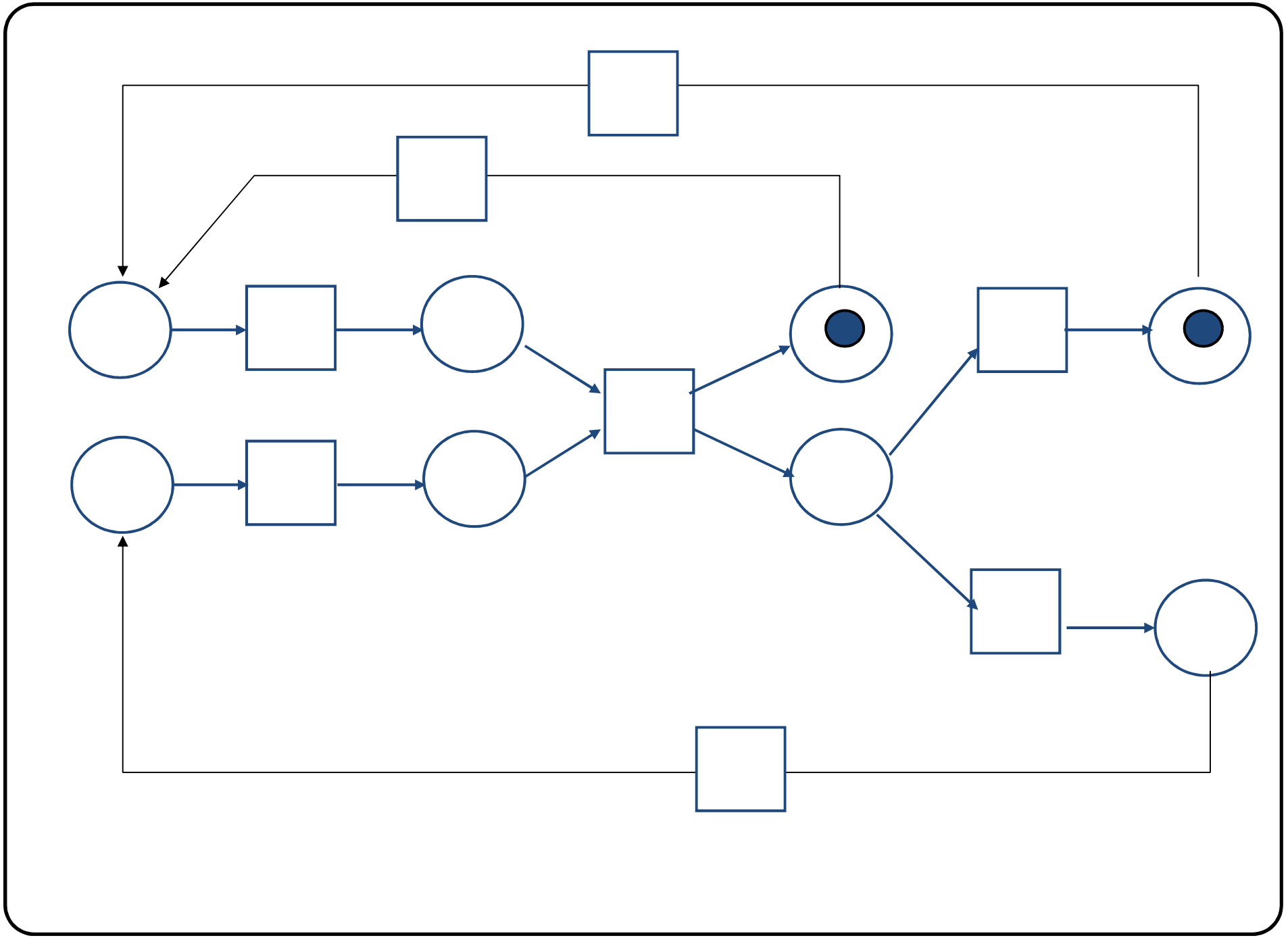


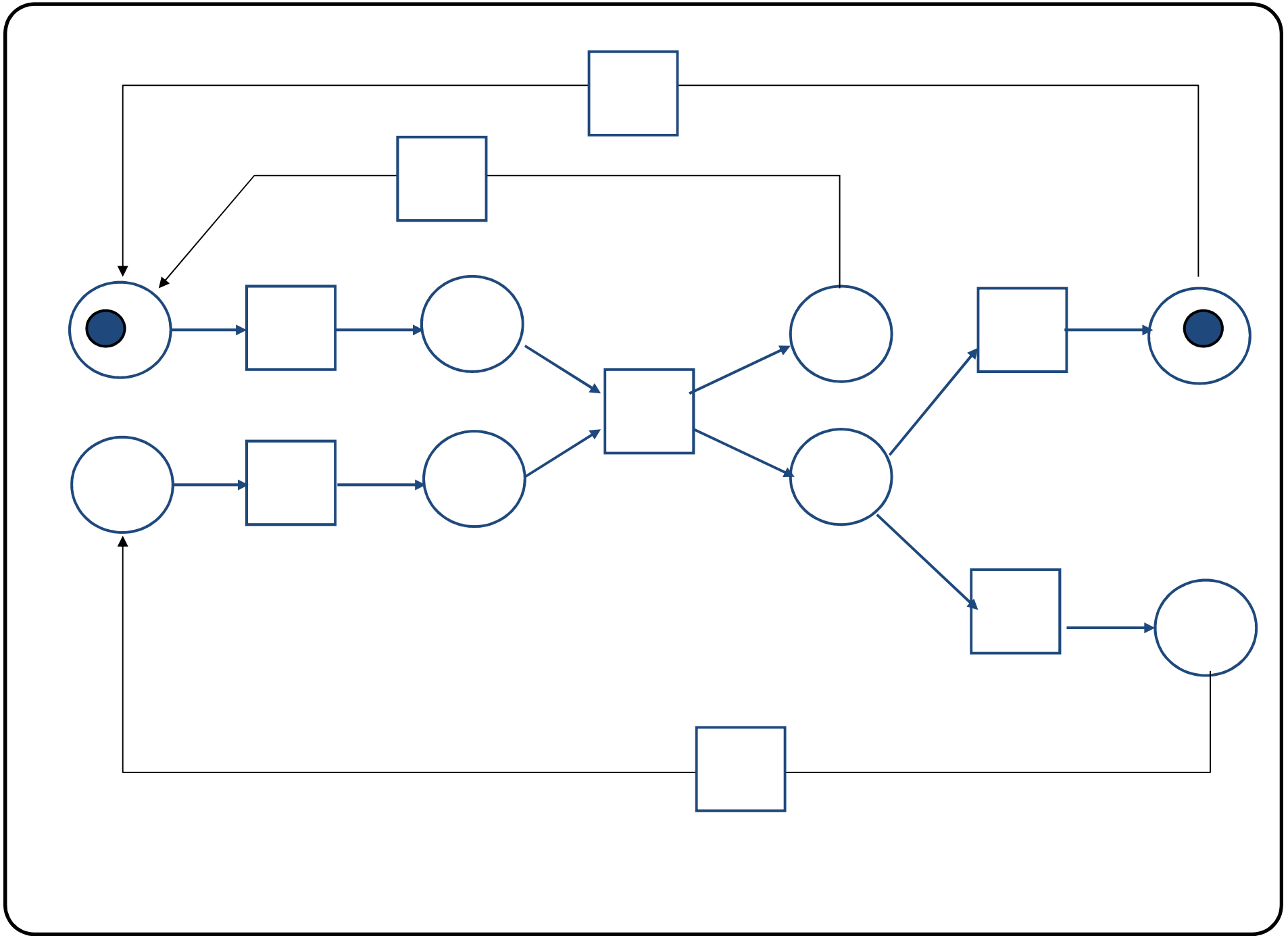
Ce réseau de Petri possède t il un état de blocage ?

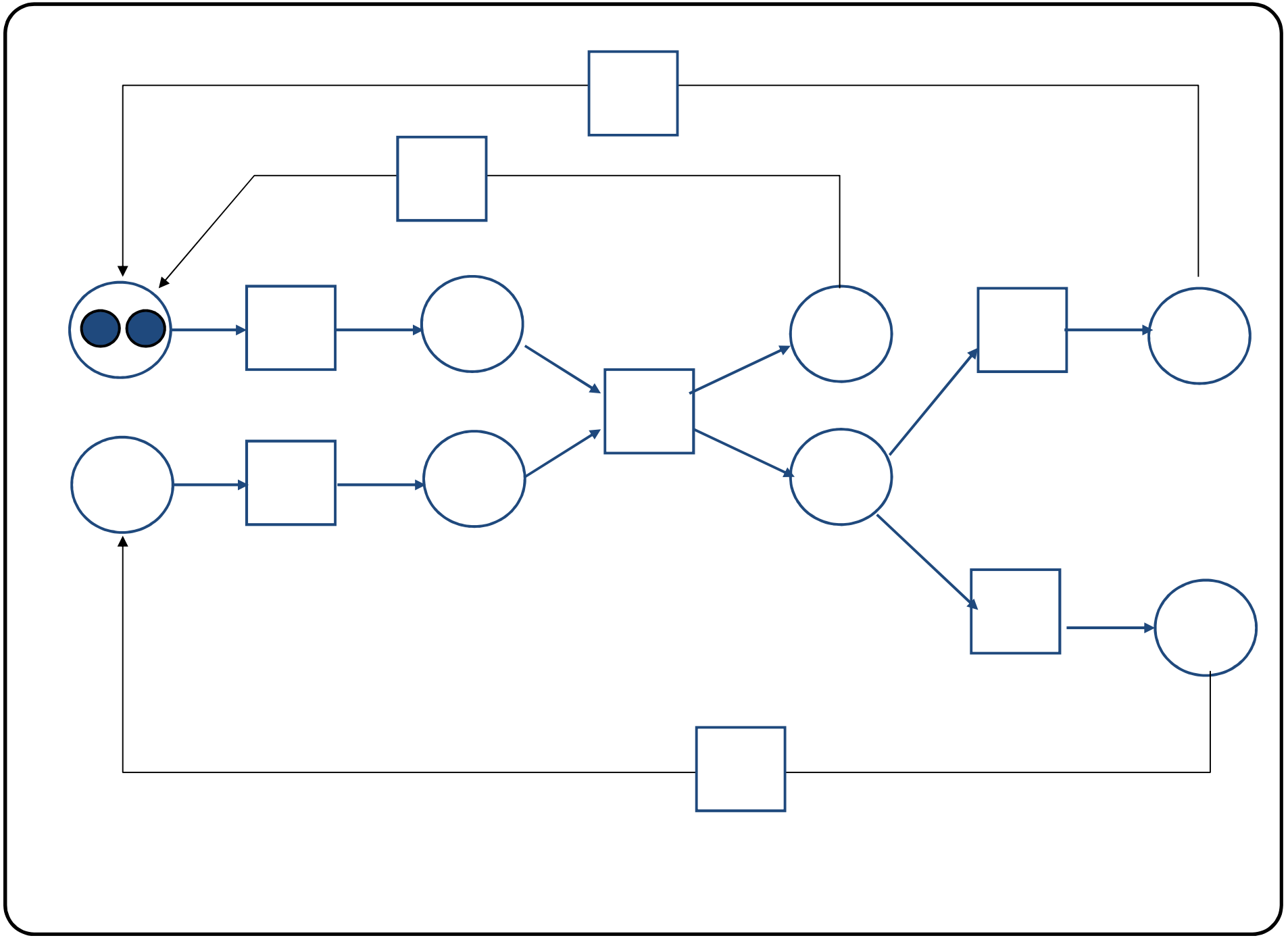


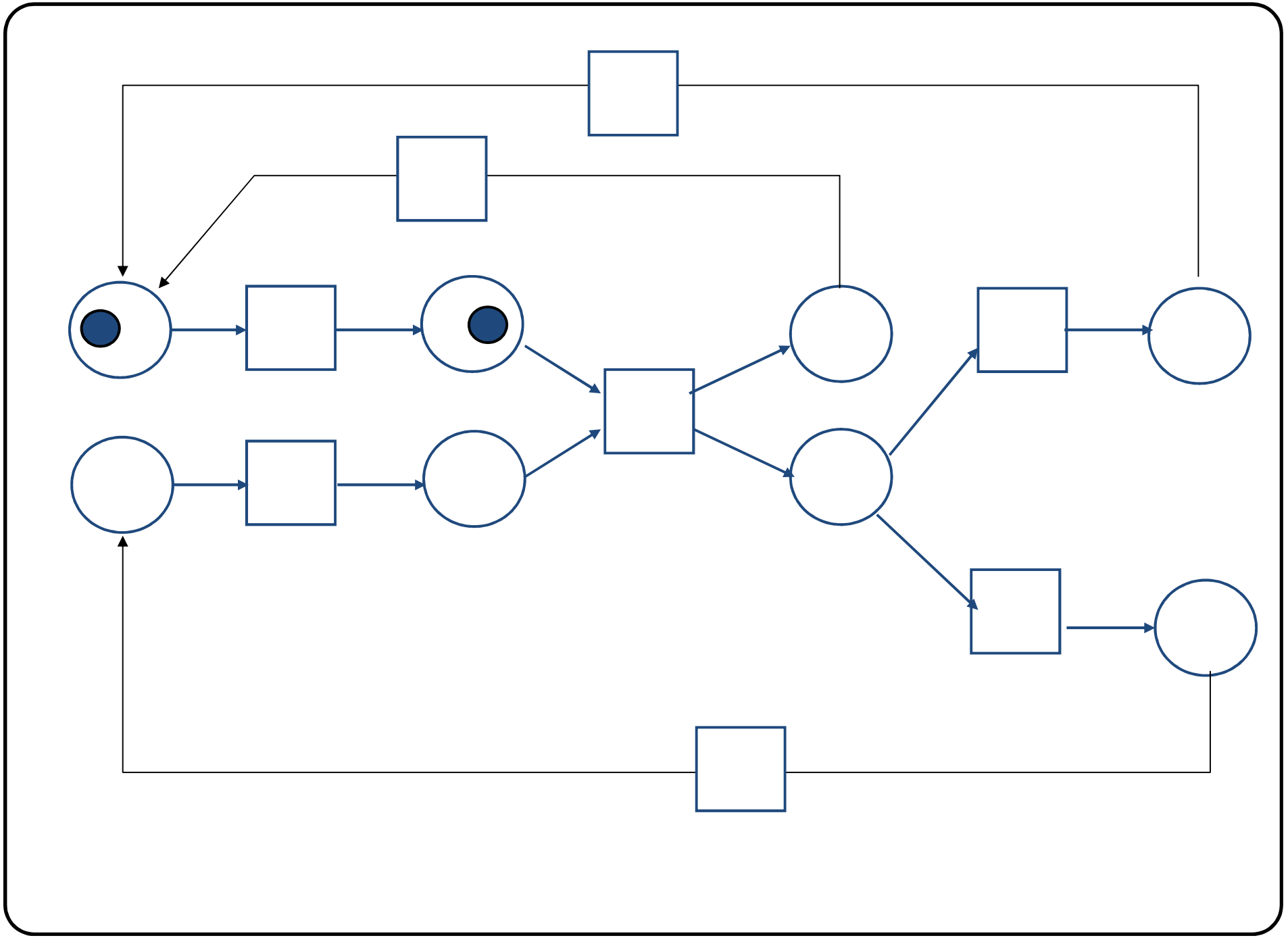


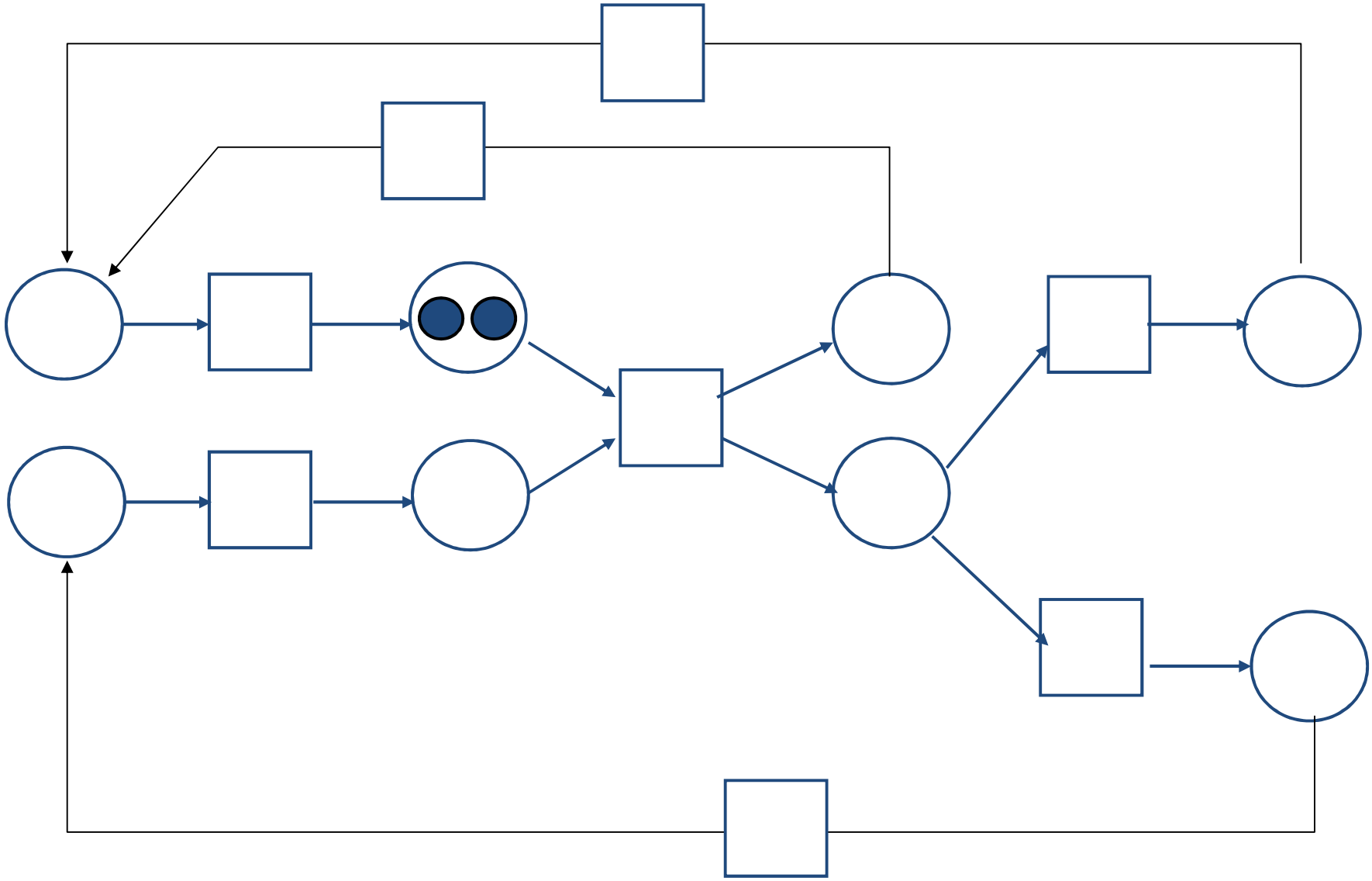












Ce réseau de Petri possède t il un état de blocage ? Oui

Introduction rapide à la logique temporelle

Logique Temporelle

φ , $/\backslash$, $\backslash/$, vrai, faux, =>

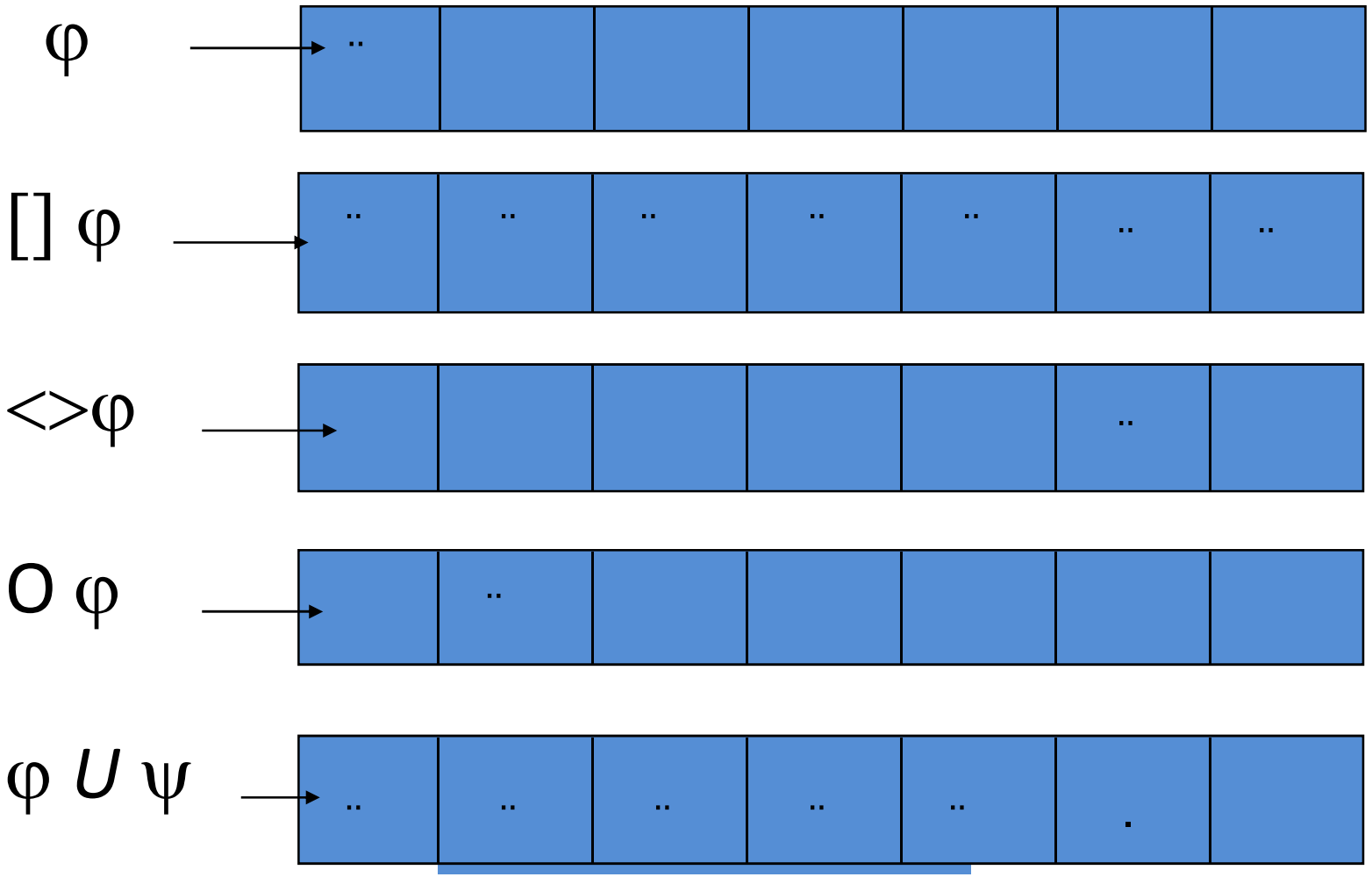
$[]\,\varphi$

$<>\varphi$

$O\,\varphi$

$\varphi\ U\ \psi$

Sémantique : les traces



Combinaisons :

[]<>p
“p aura lieu infiniment souvent”

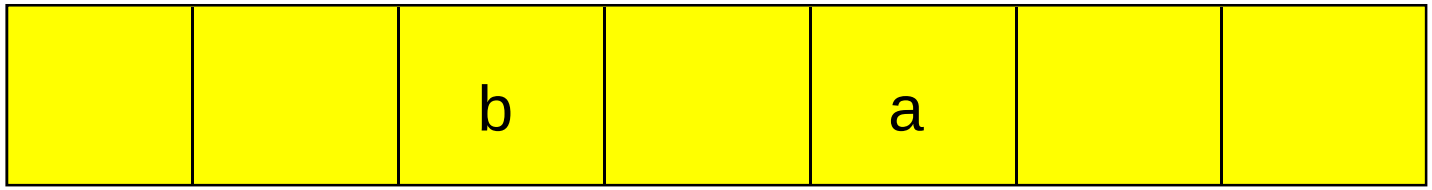
<>[]p
“à partir d’un certains point, p aura lieu tout le temps”.

([]<>p) => ([]<>q)
“si p a lieu infiniment souvant, alors q aura lieu infiniment souvent”.

Quelques autres combinaisons :

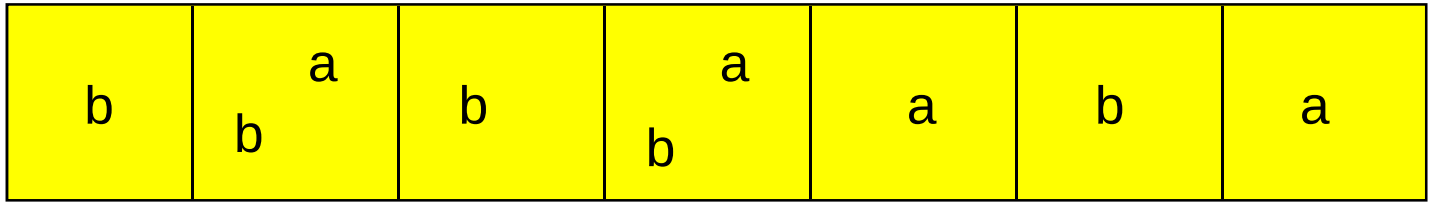
$[](a/\backslash b)=([]a)/\backslash([]b)$

Mais $<>(a/\backslash b)\neq(<>a)/\backslash(<>b)$



$<>(a\backslash/b)=(<>a)\backslash/(<>b)$

Mais $[](a\backslash/b)\neq([]a)\backslash/([]b)$



Exercices

([]<>A)/\([]<>B)	?	[]<>(A/\B)?	<=
([]<>A)\/([]<>B)	?	[]<>(A\/B)?	=
(<>[]A)/\(<>[]B)	?	<>[](A/\B)?	=
(<>[]A)\/(<>[]B)	?	<>[](A\/B)?	=>

Formal methods wiki page:

https://en.wikipedia.org/wiki/Formal_methods

Controlled Natural Language :

https://en.wikipedia.org/wiki/Controlled_natural_language

Réseaux de Petri :

https://en.wikipedia.org/wiki/Petri_net

SDL (Specification and Description Language)

http://www.sdl-forum.org/

SPIN and PROMELA

http://netlib.bell-labs.com/netlib/spin/whatispin.html

The official website of ISO/IEC JTC1/SC7 for standards in the area of Software Engineering and Open Distributed Processing (ODP):

http://saturne.info.uqam.ca/Labo_Recherche/Lrgl/sc7/

Object Management Group and UML

http://www.omg.org/uml/

The CMM documents

http://www.sei.cmu.edu/cmm/papers/cmm.pdf

http://www.sei.cmu.edu/sema/pdf/SW-CMM/2003sepSwCMM.pdf

Major failures

http://www.siam.org/siamnews/general/ariane.htm

http://www.around.com/ariane.html

http://www.cs.berkeley.edu/~nikitab/courses/cs294-8/hw1.html