9ième Atelier en Evaluation de Performances
Aussois 1-4 juin 2008

Modélisation et évaluation de performance des systèmes basés composants

N.Salmi, P.Moreaux, M.Ioualalen

LISTIC, Polytech'Savoie

Annecy, France

LSI, USTHB

Alger, Algérie

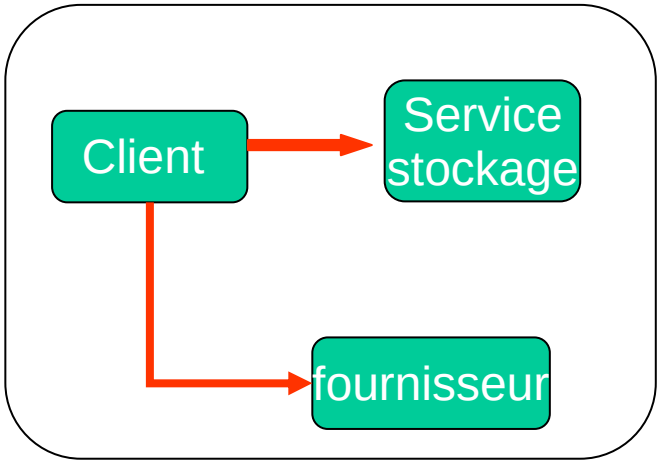
Plan

- Motivations : Systèmes basés composants (CBS) et analyse de performance
- Etat de l'art
- Méthodologie d'analyse des performances des CBS
  - Modèle Formel : Réseau de Petri Stochastique bien formé (Stochastic Well-Formed Net, SWN)
  - Etapes de la méthode
- Traduction d'un CBS en modèle SWN
- Analyse de performance du CBS
- Application aux CBS Fractal
- Conclusion & travaux futurs

Motivations

- **Après l'approche orientée objet ??**
- **Objectifs :** - Produire des applications de qualité supérieure, rapidement,
  - Coût diminué du développement, réutilisation de code
  - Plus de modularité, flexibilité et maintenabilité

Tout nouveau développement doit repartir à zéro ?



Nouvelle application

Conception de systèmes basés composant

Systèmes basés composants
CBS

CBS =assemblage de composants

Boîte noire munie de :

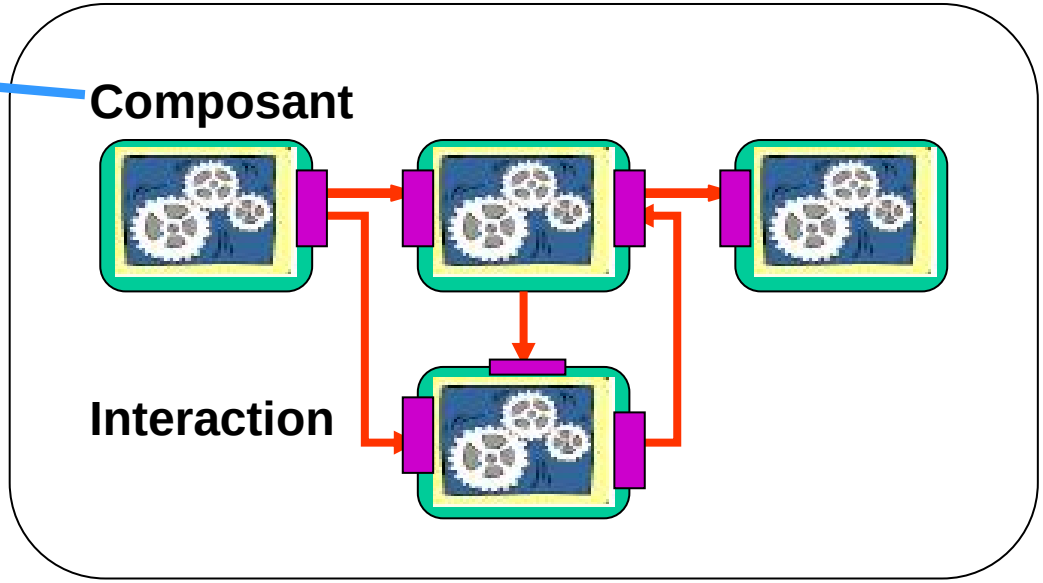
Interfaces

offrant/requérant

des services

comportement

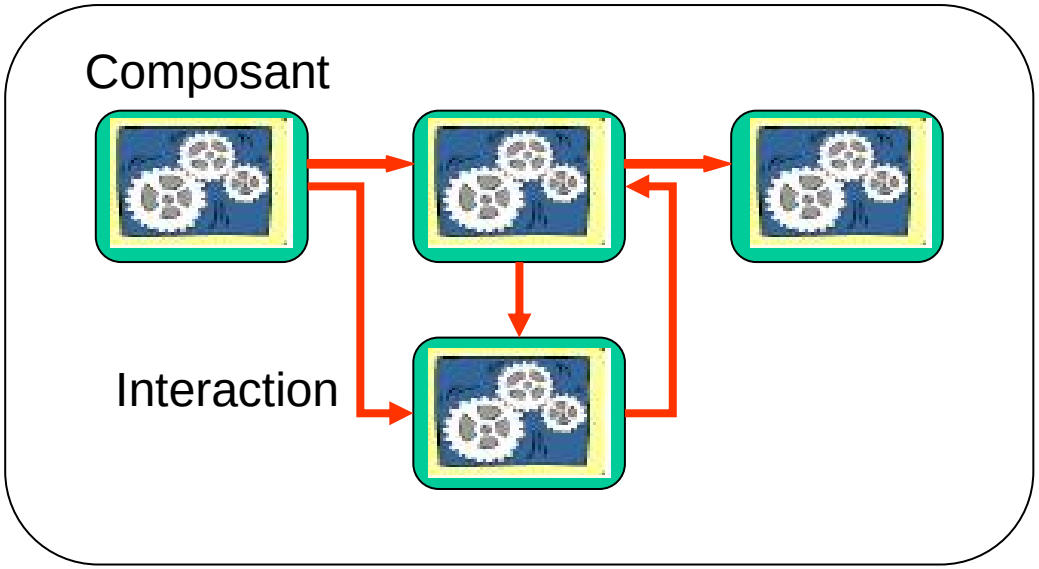
algorithmes/fonctions lisant/écrivant les données sur les interfaces



- **Multitude de modèles de composant : CCM, .Net, Fractal, EJB, AADL, ...**
- **Outils : Architecture Description Language (ADL)**
- **Plusieurs domaines d'applications : systèmes embarqués, e-business, …**

Motivations

Complexité des CBS → L'assemblage est-il correct ?



Qualitative

Blocage ?

Un état A accessible ?

Quantitative

Temps de réponse à une requête de service ?

Throughput (débit) ?

Utilisation moyenne d'une ressource ?

Etat de l'art

Analyse Qualitative de CBS

- Fusion de plusieurs modèles de composants dans un seul modèle, puis **Model checking** sur le modèle global (Kupferman, Verdi 1998).

- Vérification de **propriétés comportementales** basée sur les systèmes de transitions étiquetées (**LTS**) modélisant les composants et leur composition (DaSilva et al. 2005).

- Spécification et vérification des systèmes Fractal, basées sue les **LTS**. Utilisation de la **logique temporelle** (Barros et al. 2006).

Analyse Quantitative de CBS

- Généralement faite par des **mesures** sur les systèmes existants.

- Traduction de la description d'un système décrit dans le langage AADL en un Réseau de Petri Stochastique Généralisé (**GSPN**) pour des mesures de **dépendabilité** (Rugina et al.2006).

- Traduction de la conception d'un système (décrit en UML) vers des modèles de performance tels que les files d'attente à couches, **LQN**, puis analyse du modèle global (Wu et Woodside 2004, Grassi et al.2007).

Objectif

- **Analyse de Performance** plutôt que analyse qualitative.

- Proposer une approche structurée pour l'analyse de performance des CBS :

  - exploite la propriété de compositionalité pour réduire la complexité d'analyse (temps de calcul et mémoire) et éviter le problème d'explosion de l'espace d'états.

  - se base sur l'analyse des modèles de composants plutôt que du modèle global.

- Instanciation de la méthode à des Systèmes Basés Composants Fractal.

Notre méthodologie

- **Deux étapes principales :**
  - A partir de la description ADL d'un CBS, génération d'un modèle global du CBS, composition des modèles de composants et des interactions.

  - Analyse de Performance de l'assemblage des modèles de composant.

- **Modèle formel : Réseau bien formé ou Well-formed Net (WN)**

WN = Réseau de Petri coloré structuré, SWN = version stochastique d'un WN

- Types structurés des places et transitions,
- Fonctions d'arcs et gardes des transitions structurées.
- Domaine de couleur (noeud)= Produit cartésien de classes de couleur

Dépliage

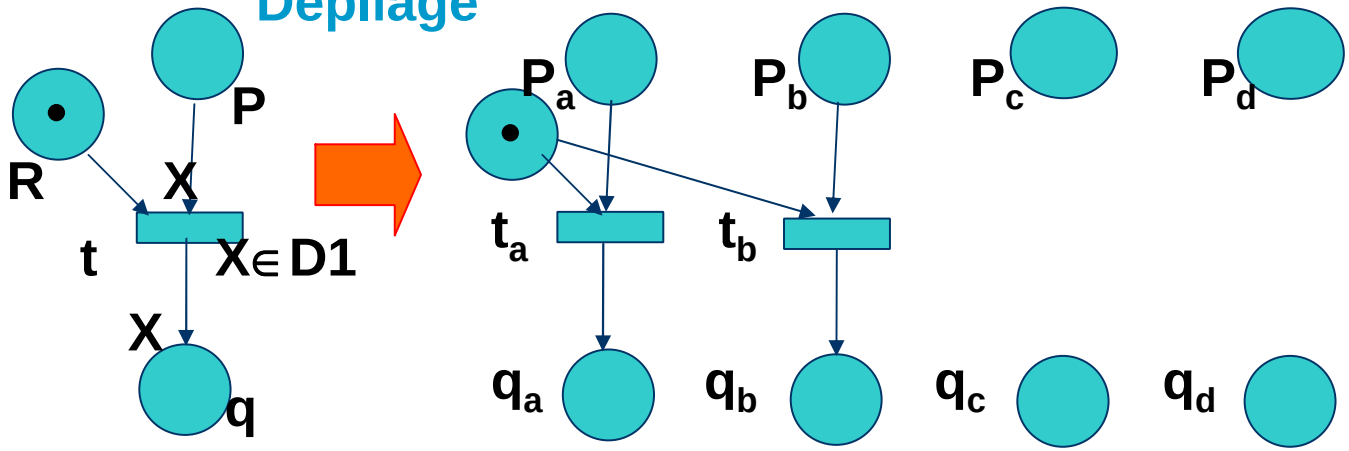
$C=D_1 \cup D_2$

$D_1=\{a,b\}$

$D_2=\{c,d\}$

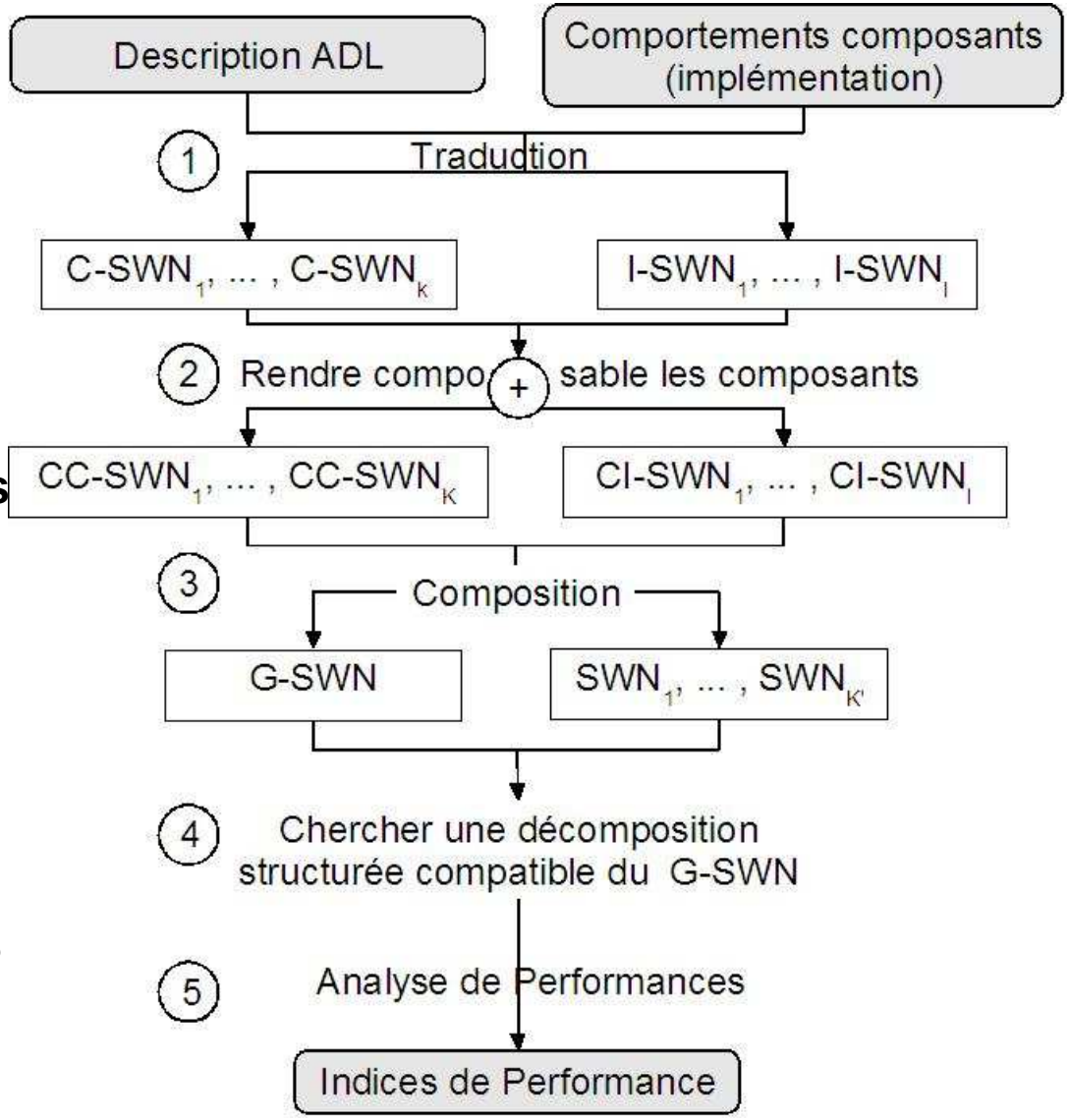
$C(p)=C(q)=C$

$X(c)=c$



Etapes de notre méthode

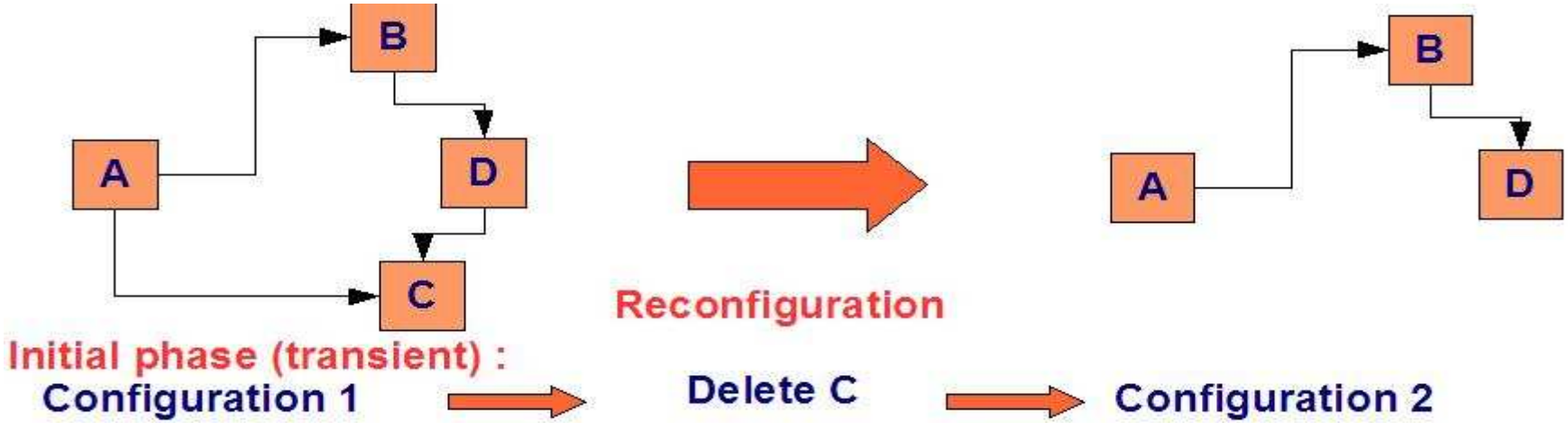
- **Component SWNs (C-SWN$_k$) :**
  Modèles des composants primitifs
- **Interaction SWNs (I-SWN$_j$) :**
  Modèles des interactions entre C-SWNs
- **Composable C-SWNs (CC-SWN$_k$).**
- **Composable I-SWNs (CI-SWNk).**
- **Global SWN G-SWN :**
  Composition des CC-SWNs et CI-SWNs par fusion de places/transitions.



Etape 1 : Modélisation

Considérations générales

1. Configurations stables considérées, pas de phase de reconfiguration (transitoire).



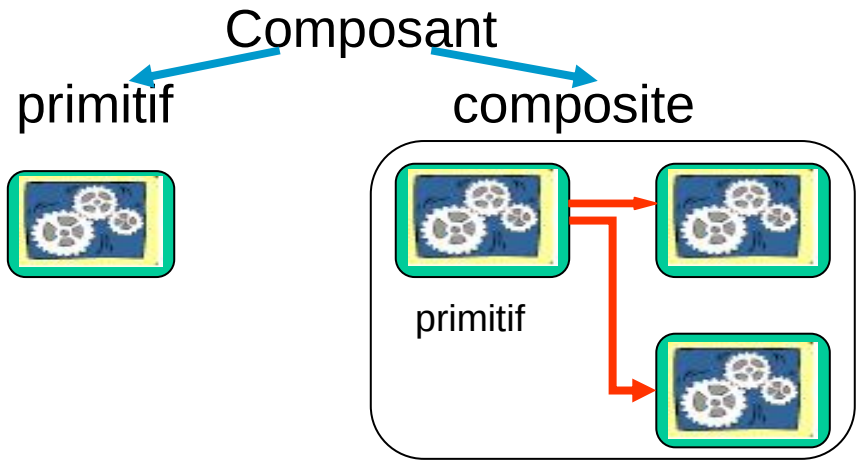
2. Description d'architecture d'un CBS non suffisante pour la modélisation

→ doit être complétée par des **informations de l'implémentation**.

3. Classes de couleurs de base modélisent les entités de données ou les entités actives des composants.

Etape 1 : Modélisation

Modélisation des composants



Modélisation des composants primitifs

1. Analyser le code source, fixer un niveau de détail, modéliser les activités.

2. Pour toute activité liée à l'appel d'une interface, modéliser les **interfaces**
→ dépend des **interactions possibles** entre composants.

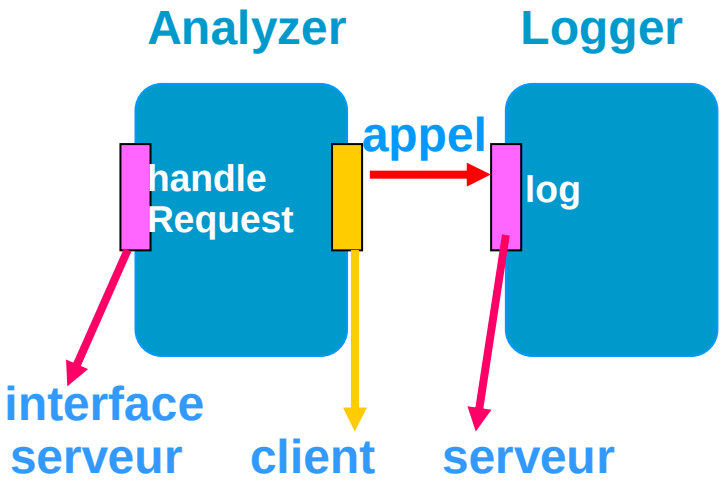
- Invocation de service (synchrone)
- Communication par notification/réception d'évènements (asynchrone)

Etape 1 : Modélisation

Invocation de service (synchrone)

entre interface **client** (invoquant une requête) et interface **serveur** (traitant la requête).

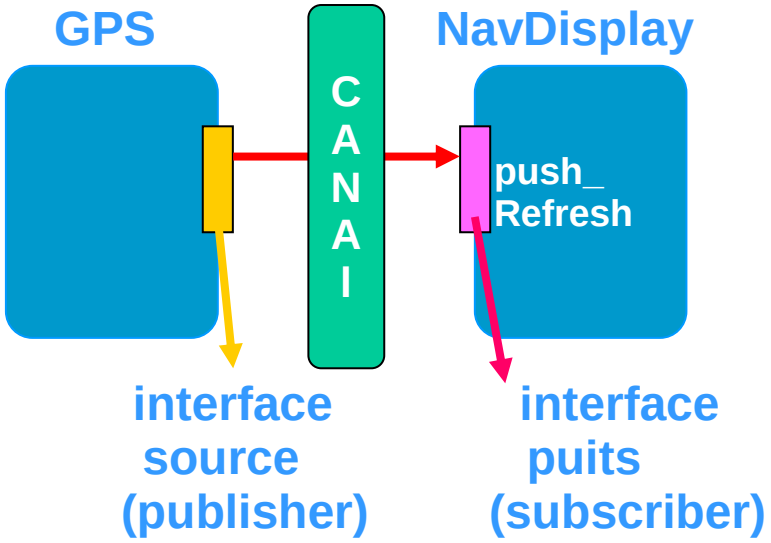
```
public class RequestAnalyzer implements RequestHandler
{
  private Logger l;
  private RequestHandler rh;
  // functional concern
  public void handleRequest (Request r) throws IOException
  { r.in = new InputStreamReader(r.s.getInputStream());
    r.out = new PrintStream(r.s.getOutputStream());
    String rq = new LineNumberReader(r.in).readLine();
    l.log(rq);
    if (rq.startsWith("GET "))
    { r.url = rq.substring(5, rq.indexOf(' ', 4));
      rh.handleRequest(r); }
    r.out.close();
    r.s.close();
    }
  }
```



Etape 1 : Modélisation

Communication par évènements (asynchrone)
entre interface **source** (envoyant un événement) et une ou plusieurs interfaces **puits** (recevant et traitant l'évènement).

```
Class NavDisplay_Executor: public virtual NavDisplay, public virtual CORBA:: LocalObject
{
  public:
   virtual void push_Refresh (tick *ev)
   {
      this->refresh_reading();
      }
   virtual void refresh_reading(void)
   {
      position_var cur = this->context_->
                              get_connection_GPSLocation();
      long coord = cur->get_pos();
      };
  };
```



- Possibilité de médiation des événements à travers un **canal d'événements** (réception et routage des évènements vers les souscripteurs, enregistement des souscriptions, filtrage d'évènements suivant des définitions de classes)

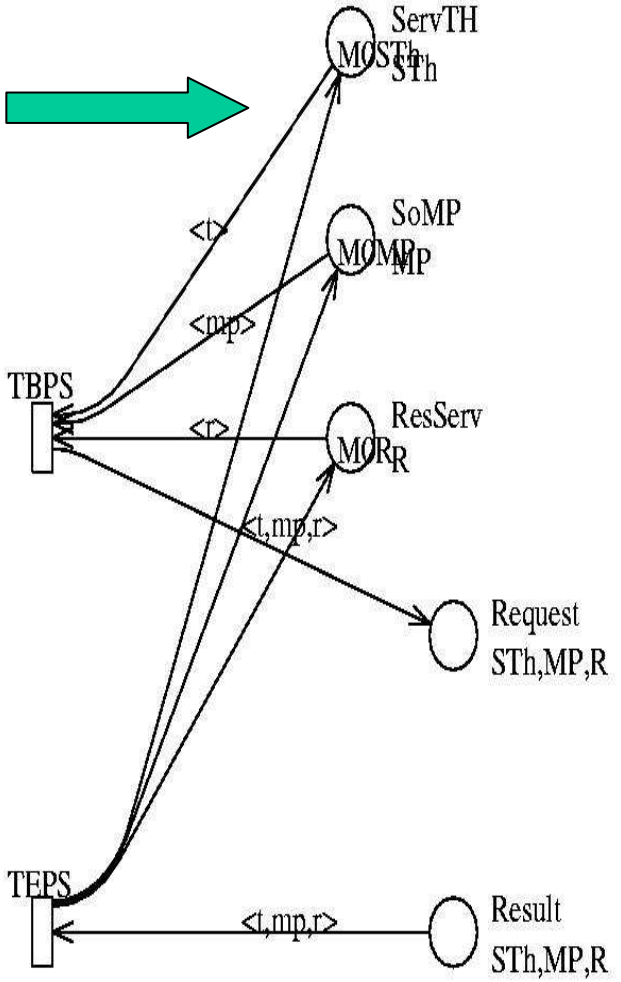
Mapping rules (1)

Invocation de service

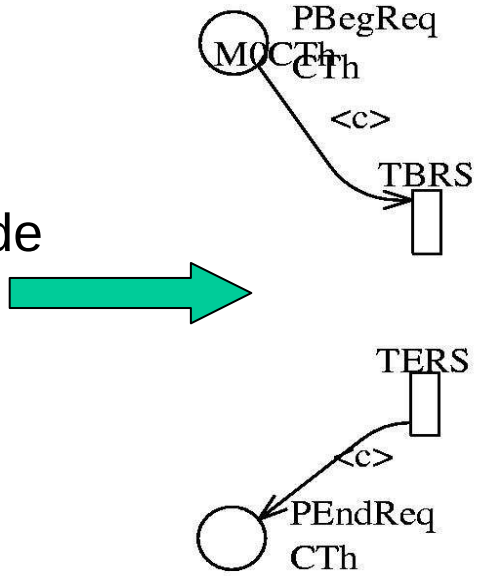
■ **Mapping rule 1:**

1. Modèle d'une **interface serveur** identifiée par un ensemble STh de threads serveur, et offrant un ensemble MP de méthodes avec leurs paramètres, utilisant éventuellement des ressources d'un ensemble R /

STh, MP, R : classes de couleurs de base



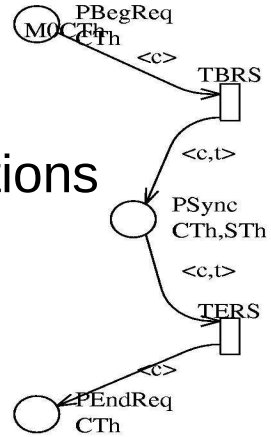
2. Modèle d'une **interface client** identifiée par un ensemble CTh de couleurs modélisant les threads clients.



Mapping rules (2)

Mapping rule 2 :

CC-SWN d'une interface client : ajouter une place entre les transitions de requête *TBRS* et *TERS*.



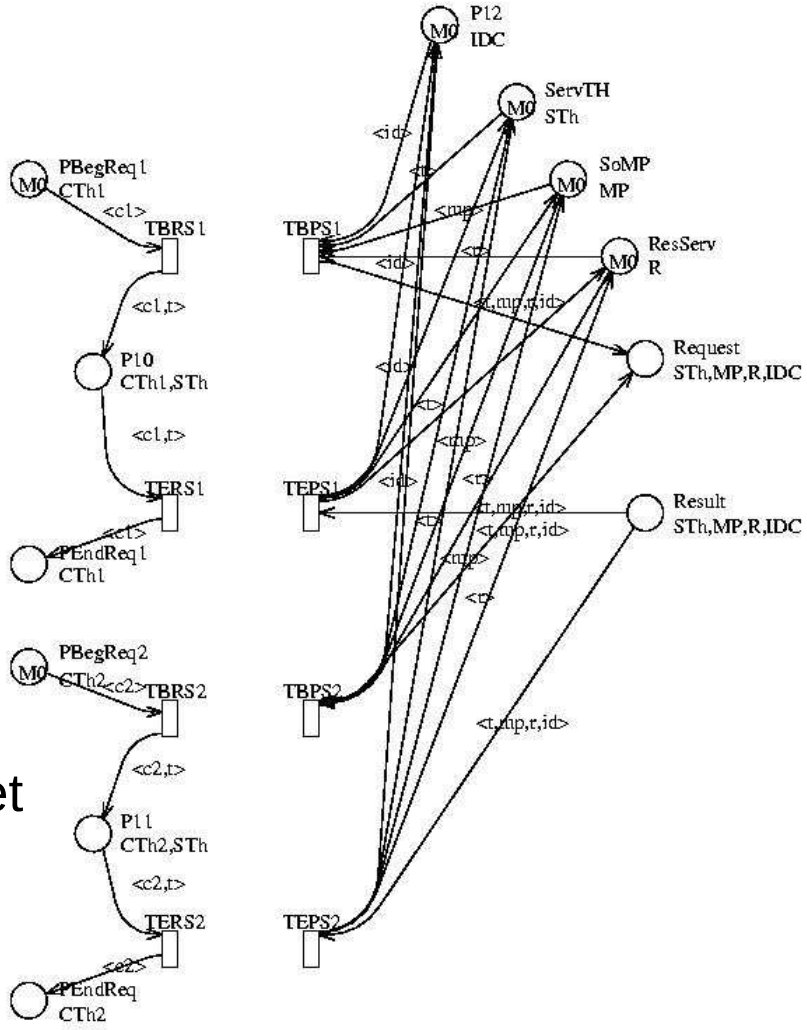
Cas de plusieurs interfaces client liées à la même interface serveur d'un composant

Mapping rule 3 :

Interface serveur avec plusieurs clients :

(i) Dupliquer les transitions d'entrée/sortie *TBPS* et *TEPS* autant de fois que d'interfaces client connectées.

(ii) Une classe de couleur IDC pour distinguer entre différent composants clients.

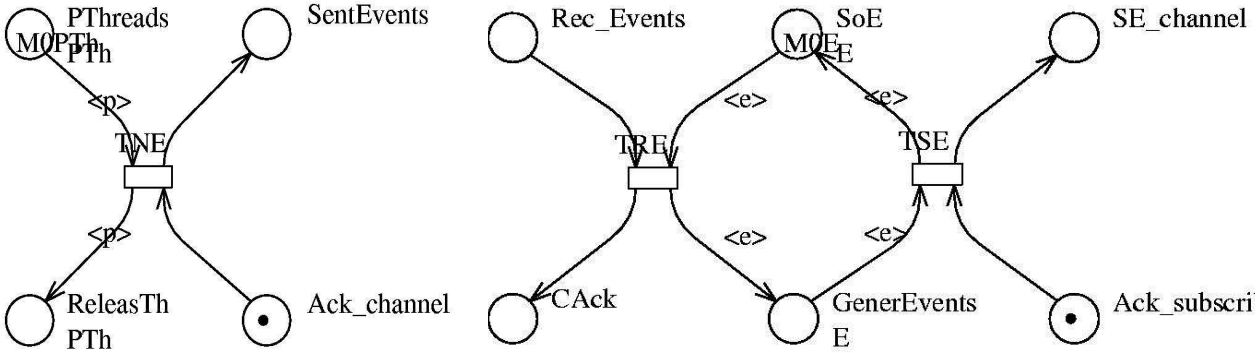


Mapping rules (3)

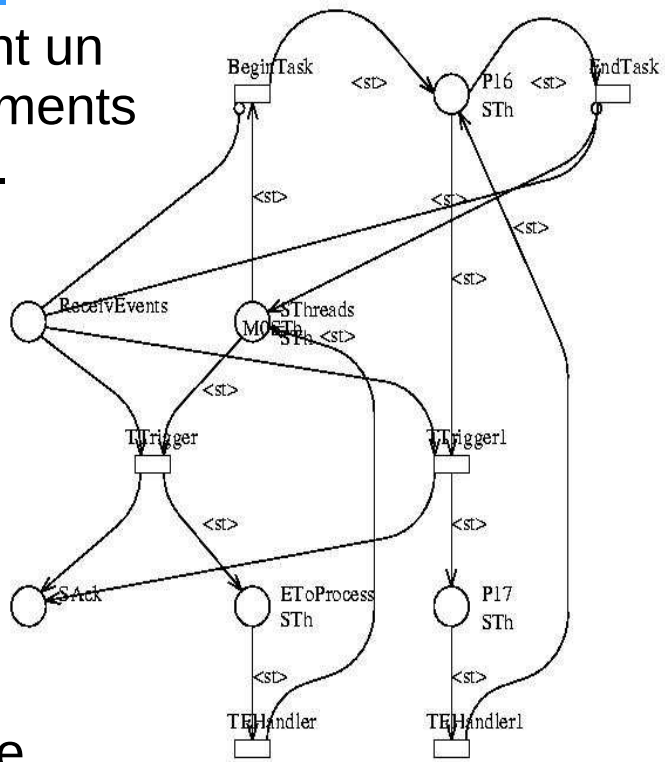
Communication par événement (spécification CORBA)

■Mapping rule 4 :

1. Modèle d'une interface **source (publisher)**, identifiée par un ensemble PTh de threads publisher,



2. Modèle d'un **canal d'événements** gérant un ensemble E d'évènements d'un type spécifique.



3. Modèle d'une **interface puits (subscriber)**, identifiée par un ensemble STh de threads subscriber :

partie locale + partie traitement événement (handler)

Mapping rules (4)

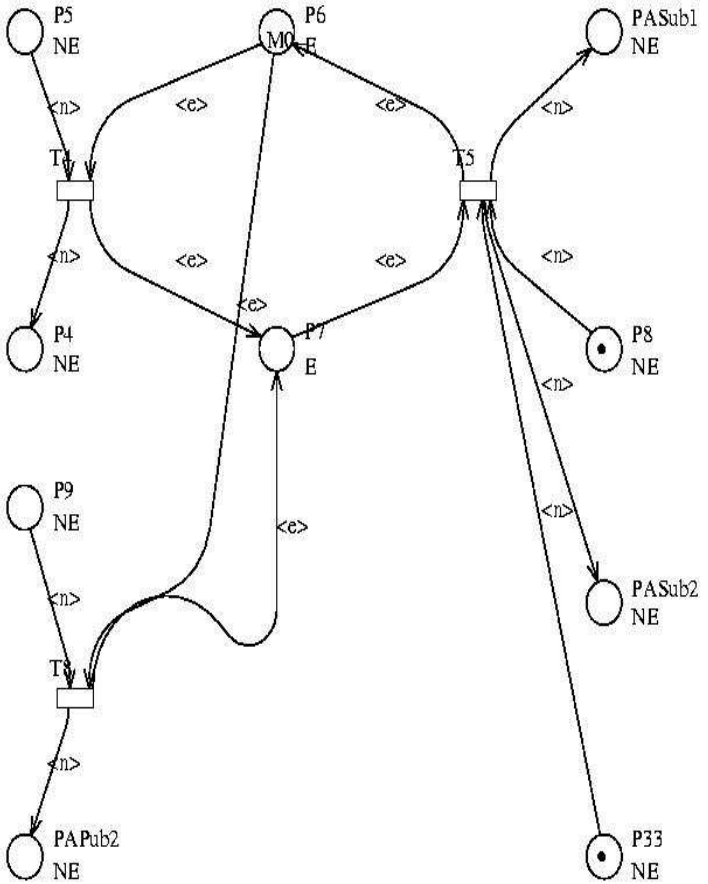
Cas de plusieurs interfaces sources et puits liées au même canal d'événements

- **Mapping rule 5 :**

CC-SWN d'un canal d'évènements connecté à :

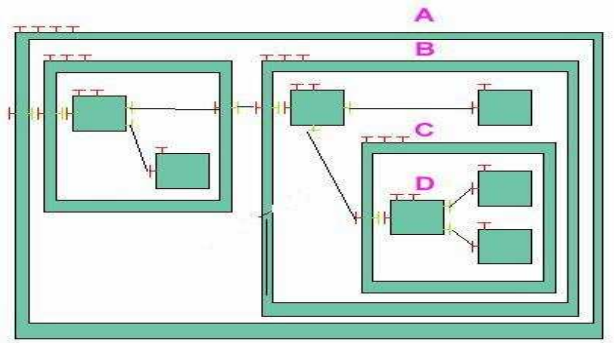
(i) plusieurs sources (publishers) → Dupliquer la transition *TRE* et ses arcs et places d'entrée/sortie *Rec_Events* et *CAck*, autant de fois que de publishers.

(ii) plusieurs puits (subscribers) → Dupliquer les places *SE_channel* et *Ack_subscribe* et leurs arcs autant de fois que de subscribers.



Etape 1 : Modélisation

Modélisation des composants composites et du G-SWN



Algorithme Construction du G-SWN

Soit N le nombre de niveaux du CBS.

1. Modéliser les composants primitifs du niveau 0.

2. Pour (i=1; i<N; i++)

   Pour chaque composite C d'un niveau i :

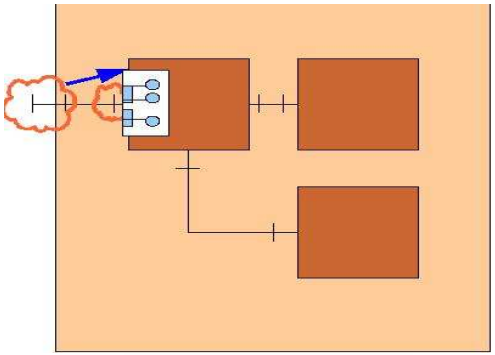
   (i) Assembler les composants du niveau i-1 :

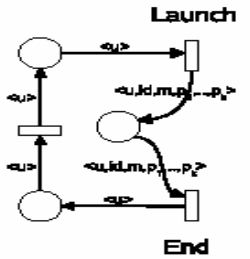
   - Pour chaque couple de sous-composants connectés via une invocation de service, fusionner les transitions (TBRS,TBPS) et (TERS,TEPS) correspondantes.
   - Pour chaque couple de sous-composants connectés via une interaction événement, fusionner les places du publisher et canal d'événements (SentEvents, Rec_Events) et (Ack_channel,CAck), et les places du canal d'évènements et du subscriber (SE_channel, ReceivEvents) et (Ack_subscriber,SAck).

   (ii) Chaque interface non connectée d'un sous-composant est considéré comme une interface externe de C.

   - Modéliser les composants primitifs du niveau i.

3. Fermer les interfaces externes du composant du plus niveau par un réseau de Petri fermant. Le **G-SWN** est obtenu.





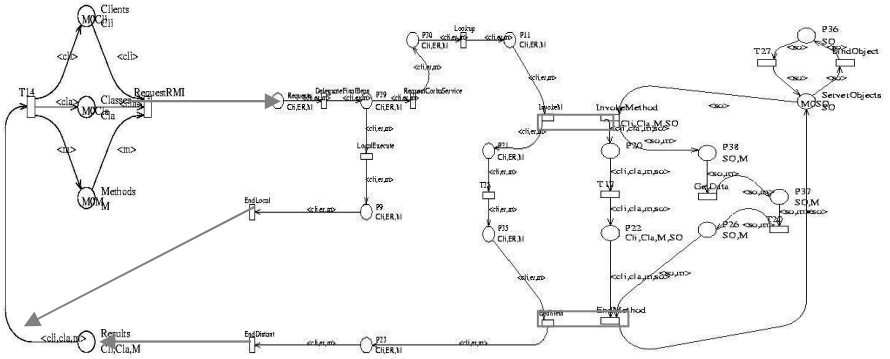
Réseau de Petri fermant

Etape 2: Analyse de Performances

■ **Notre objectif : Eviter** l'analyse du SWN global et profiter les propriétés de compositionalité du CBS pour réduire les coûts de calcul et de mémoire de l'analyse des performances.

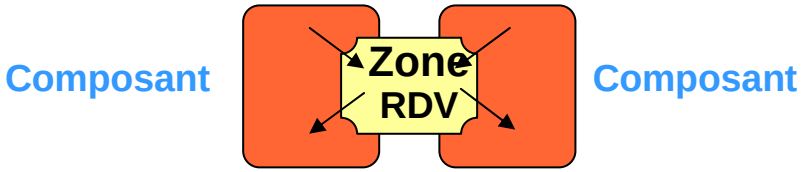
■ **Méthode d'analyse structurée** (Delamare et al.2003a,b, Moraeux & Haddad 96) :

Décomposer un SWN global en sous-réseaux (décomposition synchrone ou asynchrone)



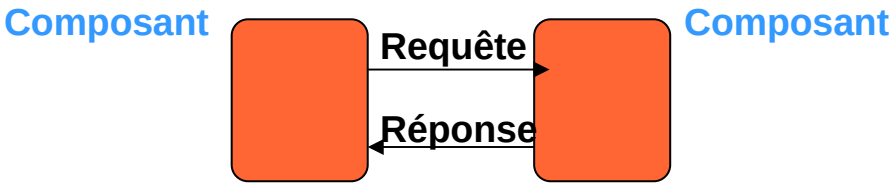
Décomposition synchrone

modélise une synchronisation complexe de type "Rendez-vous" entre deux SWNs



Décomposition asynchrone

= envoi/réception de message



Etape 2: Analyse de Performances

- **Méthode d'analyse structurée** (Delamare et al.2003a,b) :

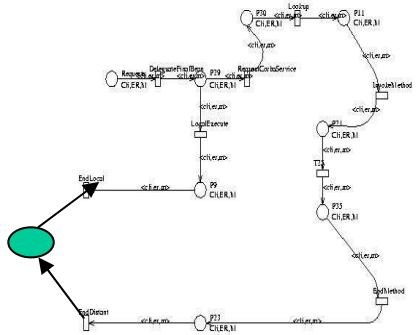
  - Décomposer un SWN global en sous-réseaux (décomposition synchrone ou asynchrone)

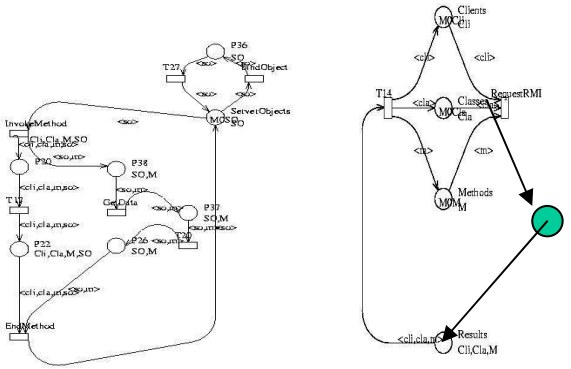
  - Etendre les sous-réseaux avec des "parties" agrégeant les interactions avec les autres.

  - Calculer le graphe d'accessibilité symbolique (SRG) de chaque sous-réseau.

  - Utiliser les SRGs pour dériver une représentation tensorielle du générateur de la chaîne de Markov du SWN global, pour calculer les indices de performance

  - Application sous certaines conditions





Etape 2: Analyse de Performances

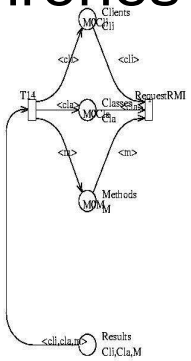
Extension et Adaptation de la méthode structurée aux CBS :

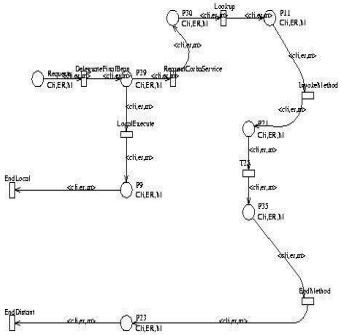
a nécessité l'étude de trois problèmes :

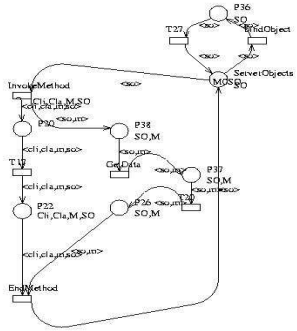
1. Composer les modèles de composants au lieu de décomposer un SWN global → Définition d'interfaces de CC-SWNs
2. Ramener une interconnexion de composants en une composition synchrone ou asynchrone de SWNs :

   Invocation de service → composition synchrone de CC-SWNs

   Communication par event → composition asynchrone de CC-SWNs
3. Etudier l'impact d'avoir des compositions synchrones et asynchrones au sein de même modèle global.





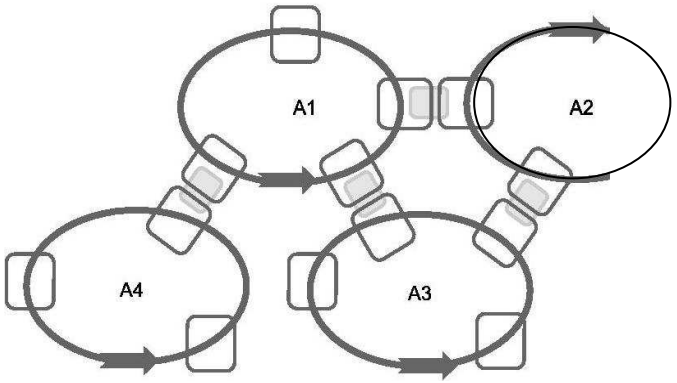


Etape 2: Analyse de Performances

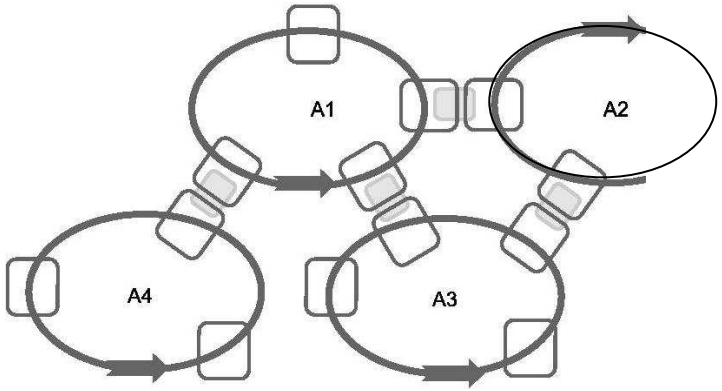
3. Etudier l'impact d'avoir des compositions synchrones et asynchrones au sein de même modèle global →

Le mixage d'interactions synchrone et asynchrone viole parfois les conditions d'analyse de composition synchrone ou asynchrone.

Composition non multisynchronisée
- **application possible**



Composition multisynchronisée pas possible..



Définition des conditions nécessaires pour permettre une analyse structurée

Application aux CBS Fractal

Modèle Fractal

- Développé au sein du Consortium **ObjectWeb** par France Telecom R&D et l'INRIA.

- Modèle de composant **hiérarchique général** utilisable dans tout domaine.

- **Plusieurs platformes :**

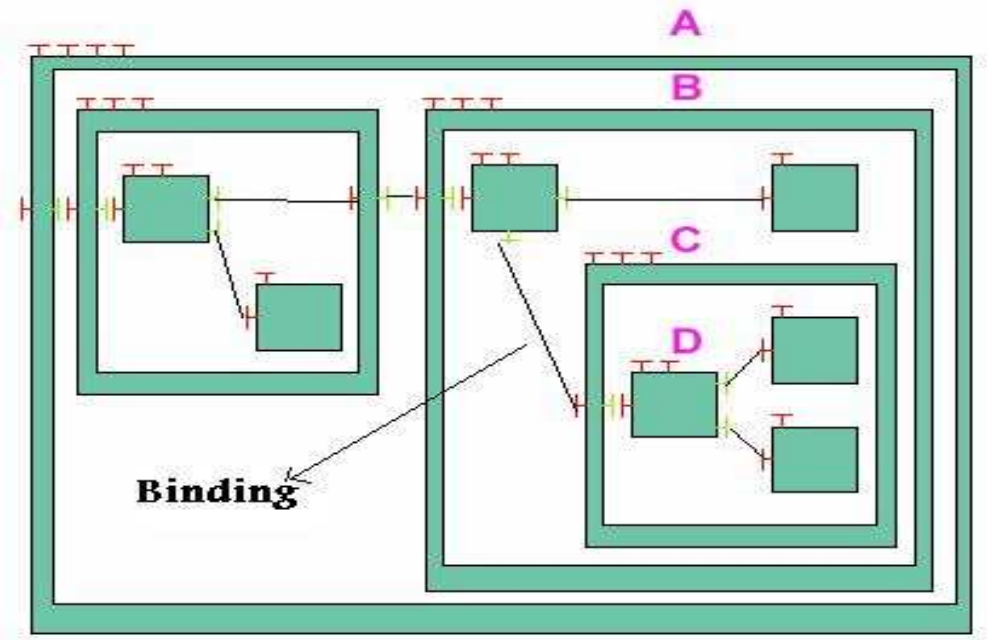
  Julia, AoKell, Proactive: en Java

  Think : en C

  FracTalk : en SmallTalk

  FracNet : en .Net

  …



Le modèle Fractal

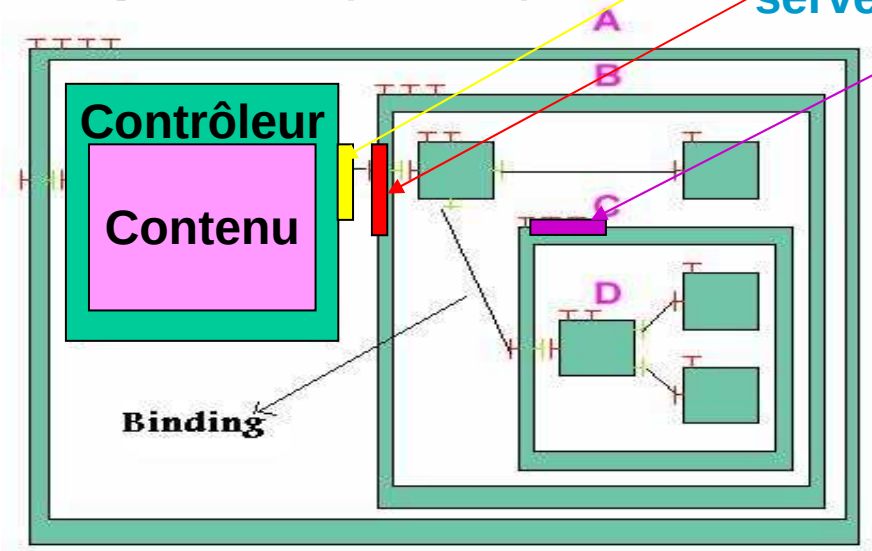
■ **Composant**= Entité d'exécution exhibant une structure récursive avec des capacités réflexives.

constitué de :

- **Contenu:** nombre fini de sous-composants (récursif).
  - **Primitifs** (D)
  - **Composite** (A,B,C)
- **Contrôleur (membrane)**
  pour monitoring, reconfiguration

Interfaces

- **Fonctionnelle**
  - client
  - server
- **Contrôle**
  - cycle de vie
  - configuration



Liaisons (Bindings)

Connexions orientées entre composants de type invocation de service

Analyse de Performance des CBS Fractal

Instanciation de notre méthode aux CBS développés sur l'implémentation Fractal Julia

Invocation de service implémentée par des appels synchrones de méthode

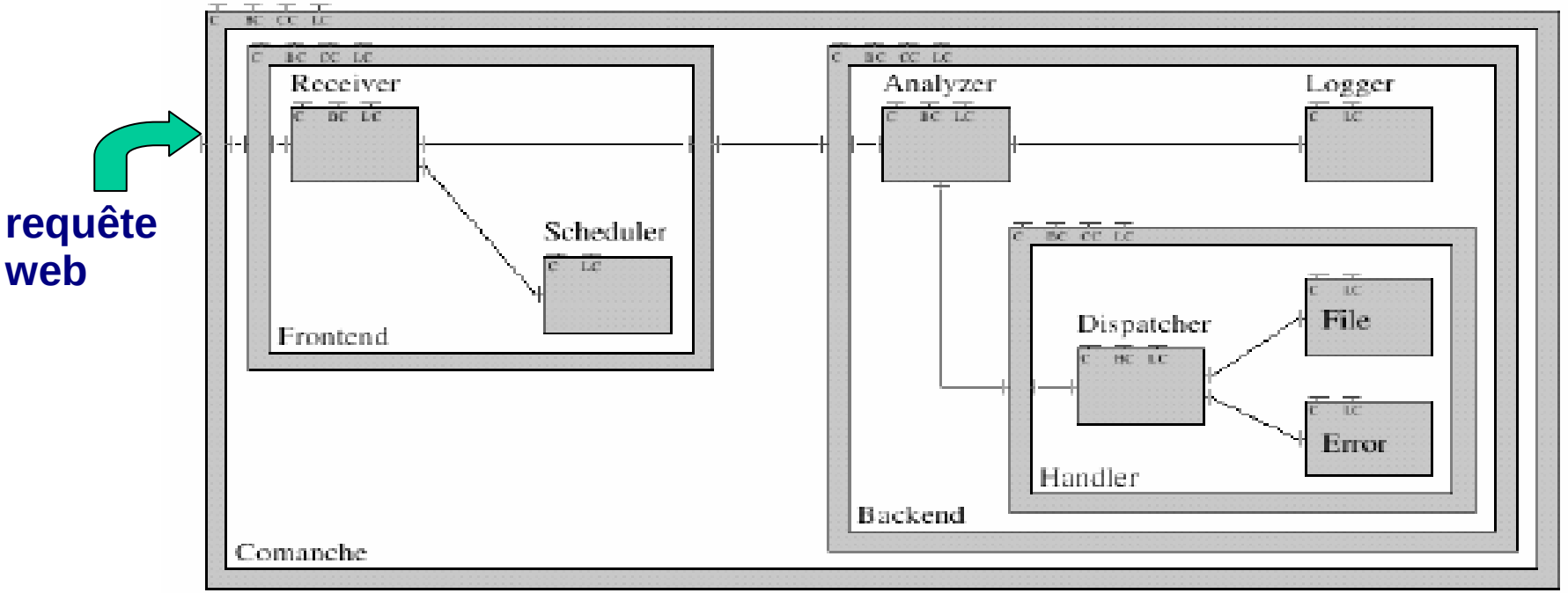
❖ **Modèle global = composition synchrone de modèles SWNs**

❖ **Condition d'application** : Une entité d'un sous-réseau qui peut être synchronisée avec un autre sous-réseau (liée à une entité de cet autre sous-réseau durant des actions communes) **doît être :**

- soit non synchronisée,

- soit synchronisée ave uniquement une seule entité d'un autre sous-réseau.

Illustration : Serveur Commanche



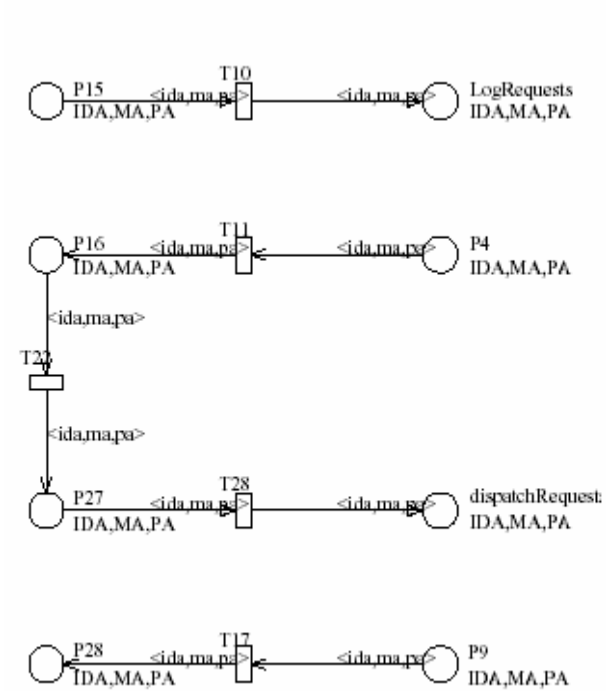
- Couleurs modélisant les requêtes HTTP, les threads schedulés, les requêtes de journal (log), les requêtes de fichiers, ...

- **Indices de performance à calculer ?**
  - Temps de réponse pour une requête web
  - Débit, nombre moyen de requêtes satisfaites/rejetées ...

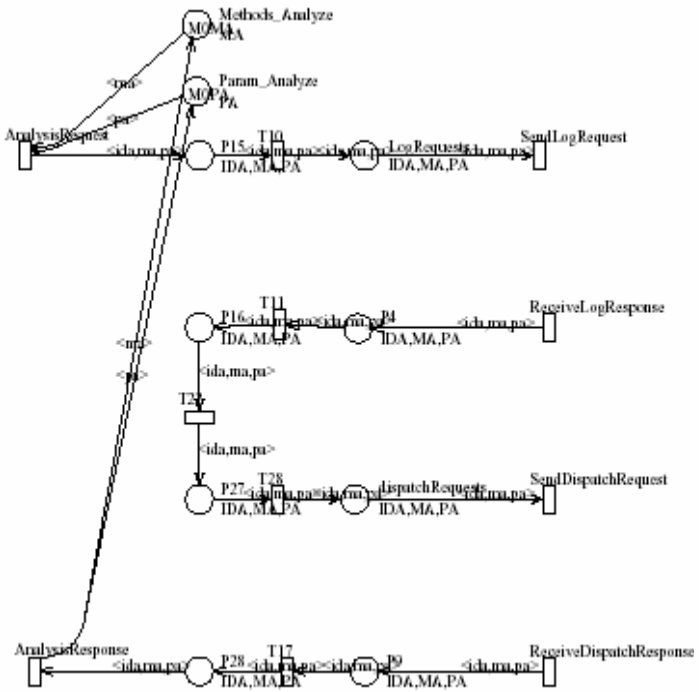
Application de la modélisation

Le composant *analyseur* de l'application Commanche

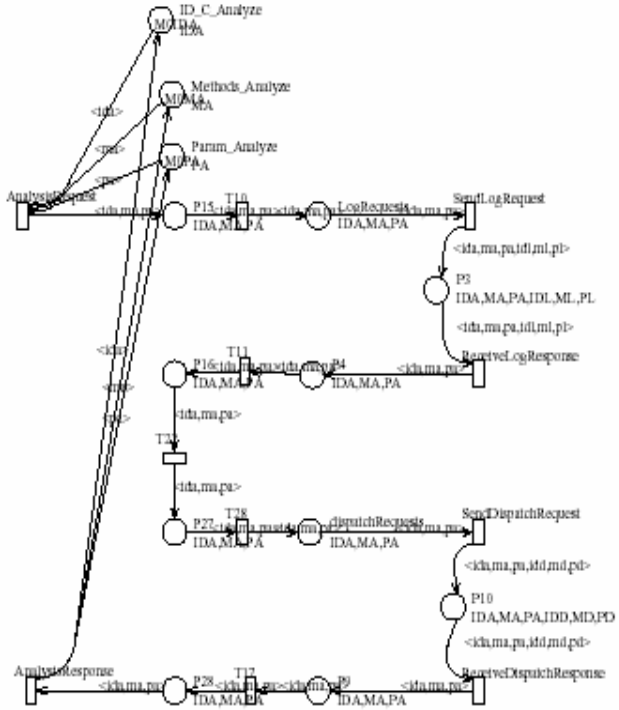
SWN



C-SWN

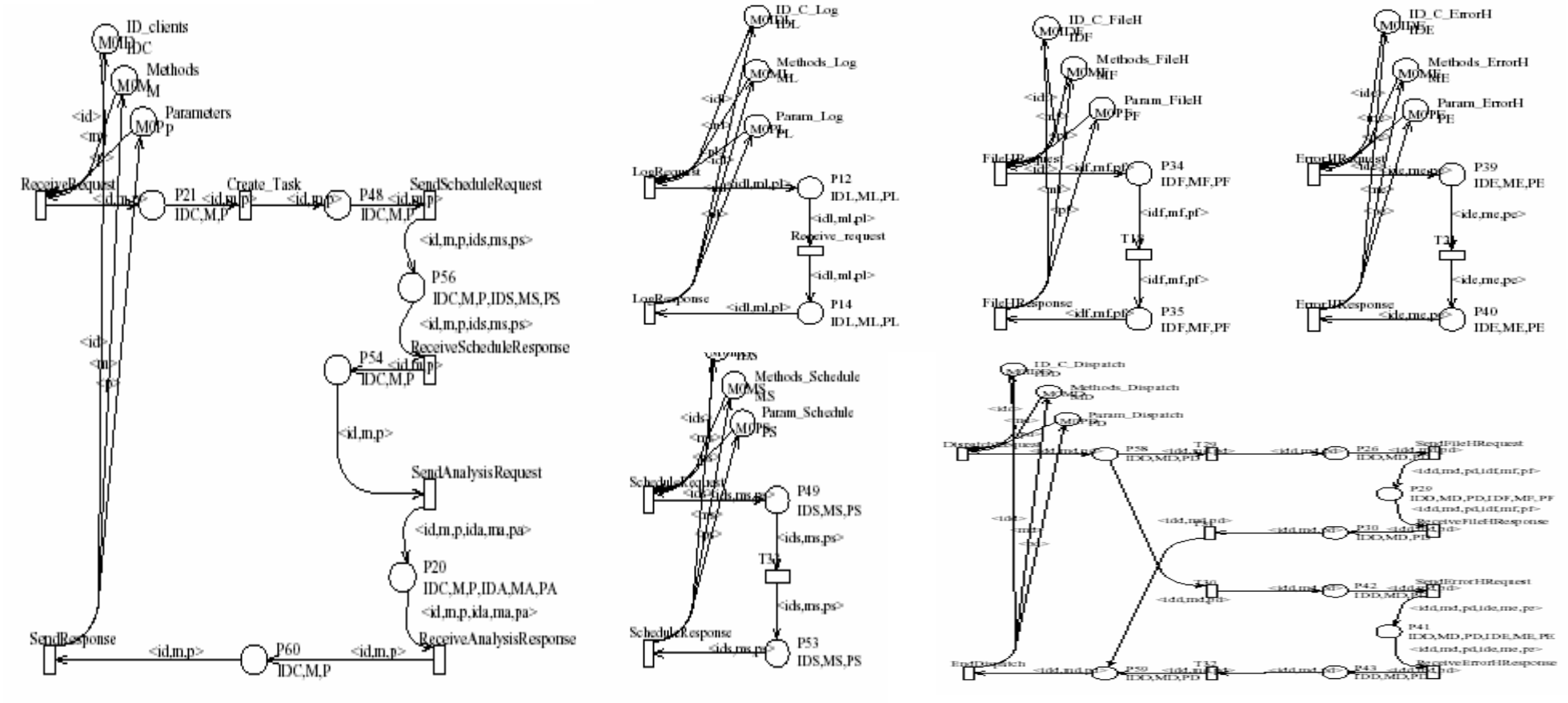


CC-SWN



Application de la modélisation

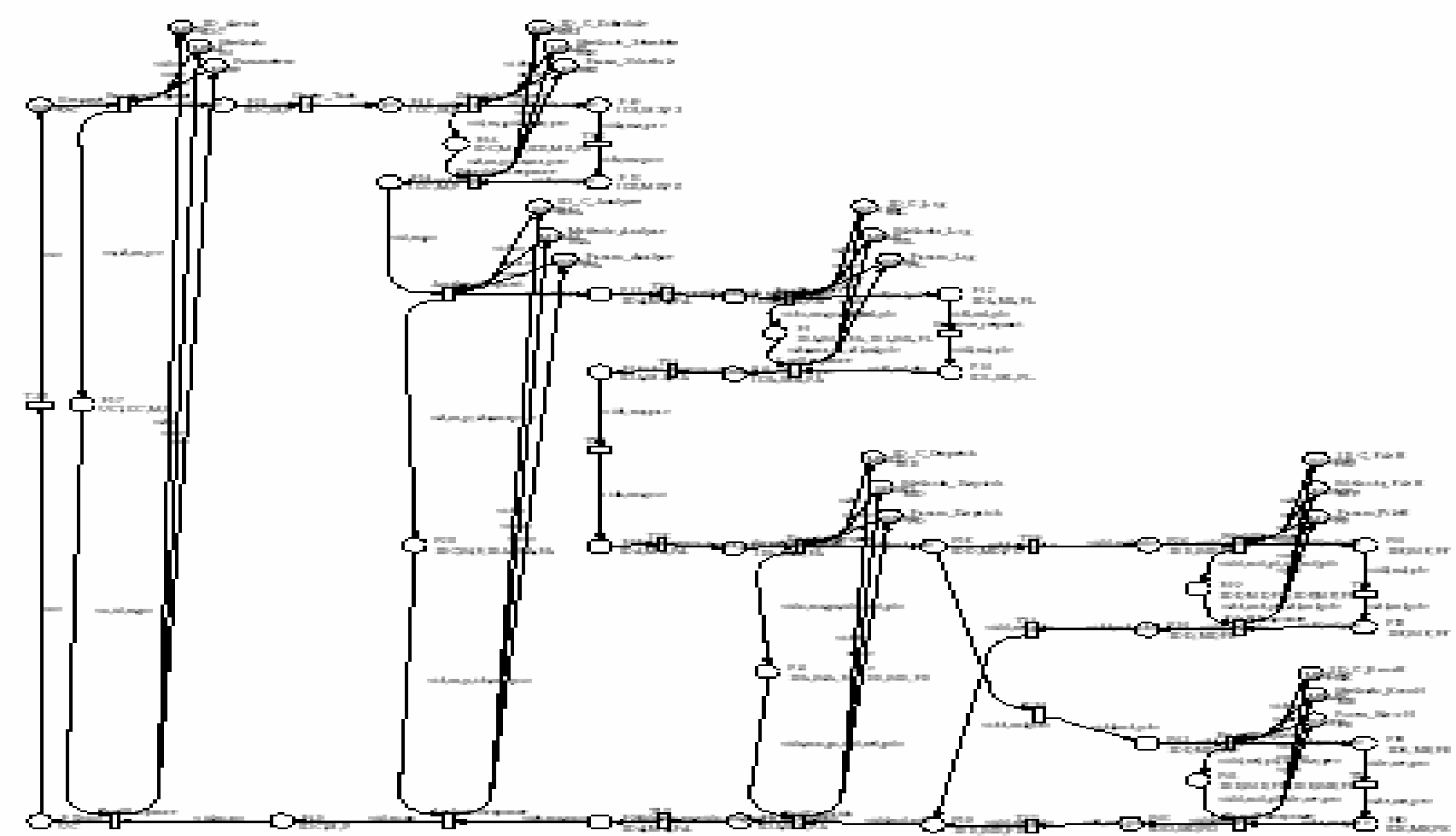
CC-SWNs des autres composants de l'application Commanche



Receiver (à gauche), Logger, File handler et Error handler (les trois modèles de droite en haut), Scheduler (milieu en bas), dispatcher (à droite en bas)

Génération du G-SWN

Le G-SWN de l'application Commanche



Application de l'analyse

- Conditions de la méthode structurée **satisfaites. Pas besoin de fusionner des** composants.
- **Station de test :**

  Station Suse linux 9.2 (1.7GHz Pentium IV CPU, 512 MB de mémoire centrale).

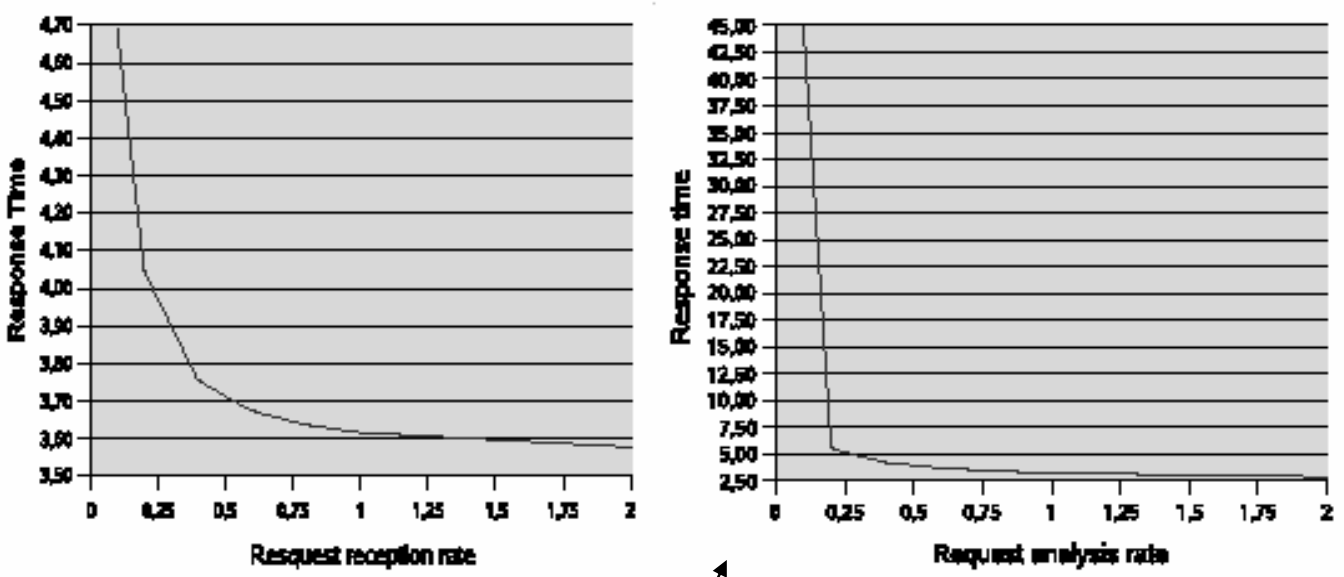
  Outils utilisés : CompSWN (calcul compositionel), GreatSPN sur le G-SWN pour comparaison des résultats.

- **Time and memory savings of our method**

Config	NbSymb	NbOrdin.	TimeGreat(s)	TimeComp(s)	MémGreat (B)	MémComp (B)
Cf1	82	35144	4	0	402	1484
Cf2	136	239392	5	0	510	1652
Cf3	271	16139264	12	0	780	2072
Cf4	406	2392068	1191	1	6305	5336
Cf5	784	24279944	4919	1	7095	6008
Cf6	1540	3656635680	-	2	-	7368
Cf7	3430	3113239552	-	3	-	10728
Cf8	7210	999926785	-	22	-	17448

Application de l'analyse

- **Temps de réponse par rapport à plusieurs paramètres :**
  - charge induite par les requêtes client,
  - vitesse de traitement des requêtes,
  - taux de recherche des données (accès aux fichiers) et taux d'erreurs.



- Taux de réception des requêtes client ↗ ⇒ Meilleur temps de réponse
- Vitesse de traitement des requêtes ↗ ⇒ Temps de réponse réduit

Conclusion et travaux futurs

- Présentation d'une nouvelle méthodologie pour l'analyse structurée des performances des CBS.

  Application aux CBS Fractal **Julia**.

- Résultats obtenus intéressants concernant le temps de calcul et espace mémoire utilisé.

- **Travaux futurs**

  - Automatiser l'extraction d'information à partir de la description d'un CBS Fractal et CCM pour une définition directe des interfaces des CC-SWNs.

  - Modéliser les propriétés de reconfiguration des CBS et vérification de leurs comportements.

Merci pour votre attention ...

Questions ?