Tinap : Modèle et infrastructure d'exécution orienté composant pour applications multi-tâches à contraintes temps réel souples et embarquées

Frédéric Loiret

Directeur : L. Duchien
Encadrants : D. Servat, L. Seinturier
Cea-List / **Lise**
Ustl / Lifl / Inria Lille - Nord Europe / **Adam**

26 mai 2008

Contexte

1. Systèmes embarqués temps réel
2. Paradigme composant

Systèmes embarqués temps réel

- **Système Embarqué**
  - Hétérogénéité (logiciel/matériel)
  - Ressources limitées (processeur, mémoire, énergie)
  - Sûreté de fonctionnement

- **Temps réel**
  - Contraintes temporelles
  - Criticité / Prédictabilité

Paradigme composant

1. **CBSE**
   - Abstraction des détails d'implantation
   - Réutilisation (*COTS*)
   - Infrastructure d'exécution (*conteneur*)
2. **Langages de Description d'Architectures (ADL)**
   - Composabilité
   - Langages spécifiques à un domaine (DSL)
3. **Support pour la séparation des préoccupations**
   - Horizontale
   - Verticale

Constat relativement au domaine

Volonté de rattraper les avancées issues du génie logiciel

- Les besoins en terme de fonctionnalités augmentent,
- Les besoins se diversifient, frontière de plus en plus floue
  - Entre applications à prépondérance fonctionnelle vs. prépondérance contrôle/commande
  - Entre besoins intra-applicatif contraints par le temps vs. non contraints

Constat relativement au domaine

Volonté de rattraper les avancées issues du génie logiciel

- Les besoins en terme de fonctionnalités augmentent,
- Les besoins se diversifient, frontière de plus en plus floue
  - Entre applications à prépondérance fonctionnelle vs. prépondérance contrôle/commande
  - Entre besoins intra-applicatif contraints par le temps vs. non contraints

Frein à l'adoption du paradigme composant :

- Être en mesure de raisonner sur les couches sous-jacentes
  - Applicatifs fortement dépendants des « couches basses »
- Abstractions tenaces employées dans le domaine
  - Conception « orientée activité »

Une forte dynamique de recherche

Nombreuses initiatives

- ARTIST2, *Real Time Component Cluster*
- COMES'08 / Progress
- AS 195 CNRS (« Composants et Architectures temps réel »)
- R&D : FT, STM

Une forte dynamique de recherche

Nombreuses initiatives

- ARTIST2, *Real Time Component Cluster*
- COMES'08 / Progress
- AS 195 CNRS (« Composants et Architectures temps réel »)
- R&D : FT, STM

- Composants et ADL issus du domaine [AADL, CLARA, SAVECCM]
- Plate-formes d'exécution [CORBA RT, COMPARE]
- Modélisation [AUTOSAR, MARTE]

- Pas de réel concensus, ni de modèle émergent

Contribution : Objectifs Fixés

Restrictions sur le domaine ciblé

- Exclusivement le logiciel,
- Sur mono-processeur
- Faciliter la prédictabilité, non l'assurer

Contribution : Objectifs Fixés

Restrictions sur le domaine ciblé

- Exclusivement le logiciel,
- Sur mono-processeur
- Faciliter la prédictabilité, non l'assurer

Objectifs

1. Conception « **orientée fonctionnalité** »,

Contribution : Objectifs Fixés

Restrictions sur le domaine ciblé

- Exclusivement le logiciel,
- Sur mono-processeur
- Faciliter la prédictabilité, non l'assurer

Objectifs

1. Conception « **orientée fonctionnalité** »,
2. Expérimenter le paradigme composant verticalement, pour assurer
   - Une **traçabilité** entre les différents niveaux d'abstraction
   - Un **cadre méthodologique** et des outils **unifiés**
   - Un **contrôle fin** des fonctionnalités nécessaires

Contribution : Objectifs Fixés

Restrictions sur le domaine ciblé

- Exclusivement le logiciel,
- Sur mono-processeur
- Faciliter la prédictabilité, non l'assurer

Objectifs

1. Conception « **orientée fonctionnalité** »,
2. Expérimenter le paradigme composant verticalement, pour assurer
   - Une **traçabilité** entre les différents niveaux d'abstraction
   - Un **cadre méthodologique** et des outils **unifiés**
   - Un **contrôle fin** des fonctionnalités nécessaires
3. Un applicatif « orientée fonctionnalité »,

Contribution : Objectifs Fixés

Restrictions sur le domaine ciblé

- Exclusivement le logiciel,
- Sur mono-processeur
- Faciliter la prédictabilité, non l'assurer

Objectifs

1. Conception « **orientée fonctionnalité** »,
2. Expérimenter le paradigme composant verticalement, pour assurer
   - Une **traçabilité** entre les différents niveaux d'abstraction
   - Un **cadre méthodologique** et des outils **unifiés**
   - Un **contrôle fin** des fonctionnalités nécessaires
3. Un applicatif « orientée fonctionnalité », mais assurer le pont vers une représentation « orientée activité »
   - Se rapprocher des abstractions courantes du domaine (outils, formalismes)

Contribution : besoins

Disposer d'un **espace de conception orientée composant**,

Contribution : besoins

Disposer d'un **espace de conception orientée composant**,

- Capturant les préoccupations du domaine :
  - Aspects multi-tâches et réactifs,
  - Contraintes temporelles,
  - Patrons de communication et de synchronisation, etc

Contribution : besoins

Disposer d'un **espace de conception orientée composant**,

- Capturant les préoccupations du domaine :
  - Aspects multi-tâches et réactifs,
  - Contraintes temporelles,
  - Patrons de communication et de synchronisation, etc

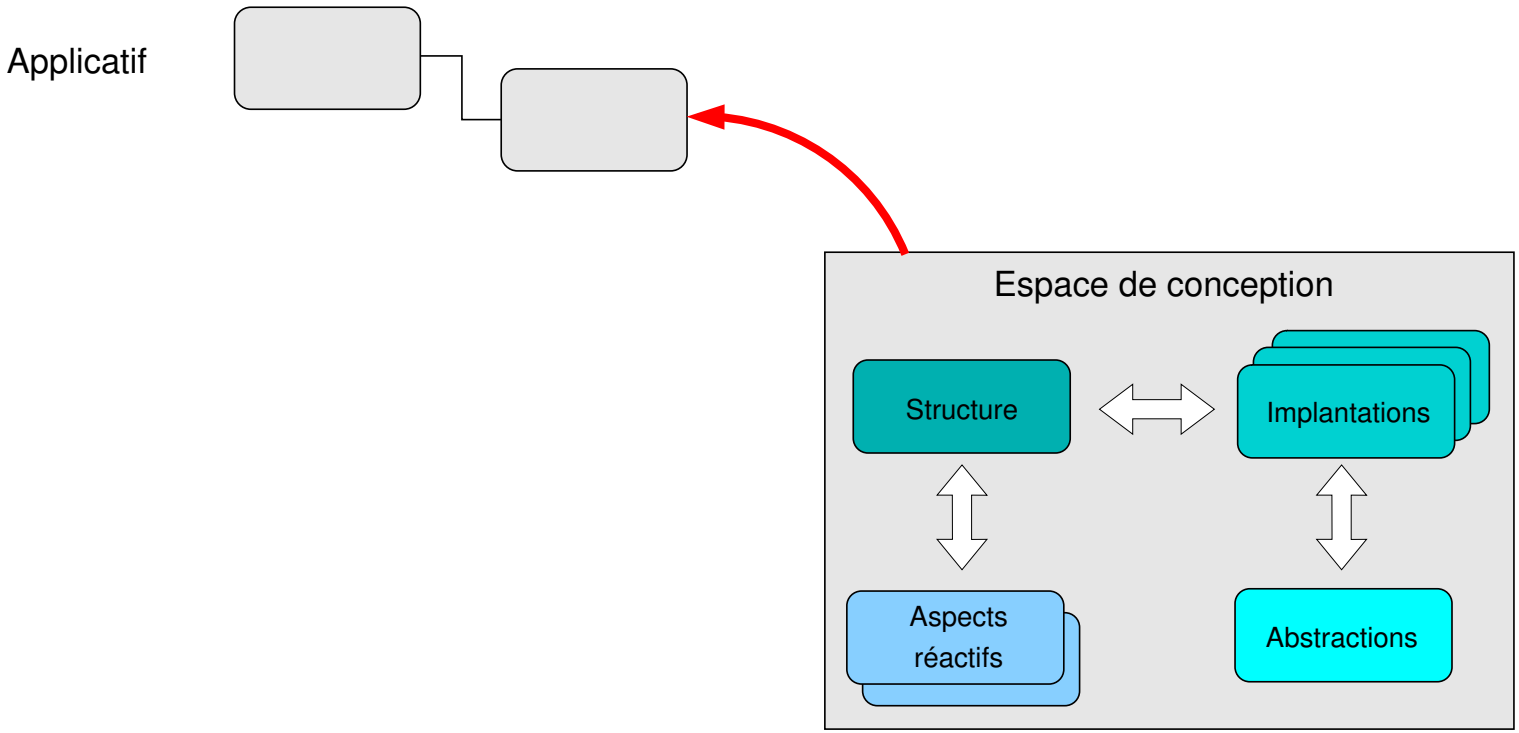
- Utilisable à tous niveaux d'abstraction
  - ADLs du domaine trop spécifiques et contraints
  - Approches composants classiques non adaptés

Contribution : besoins

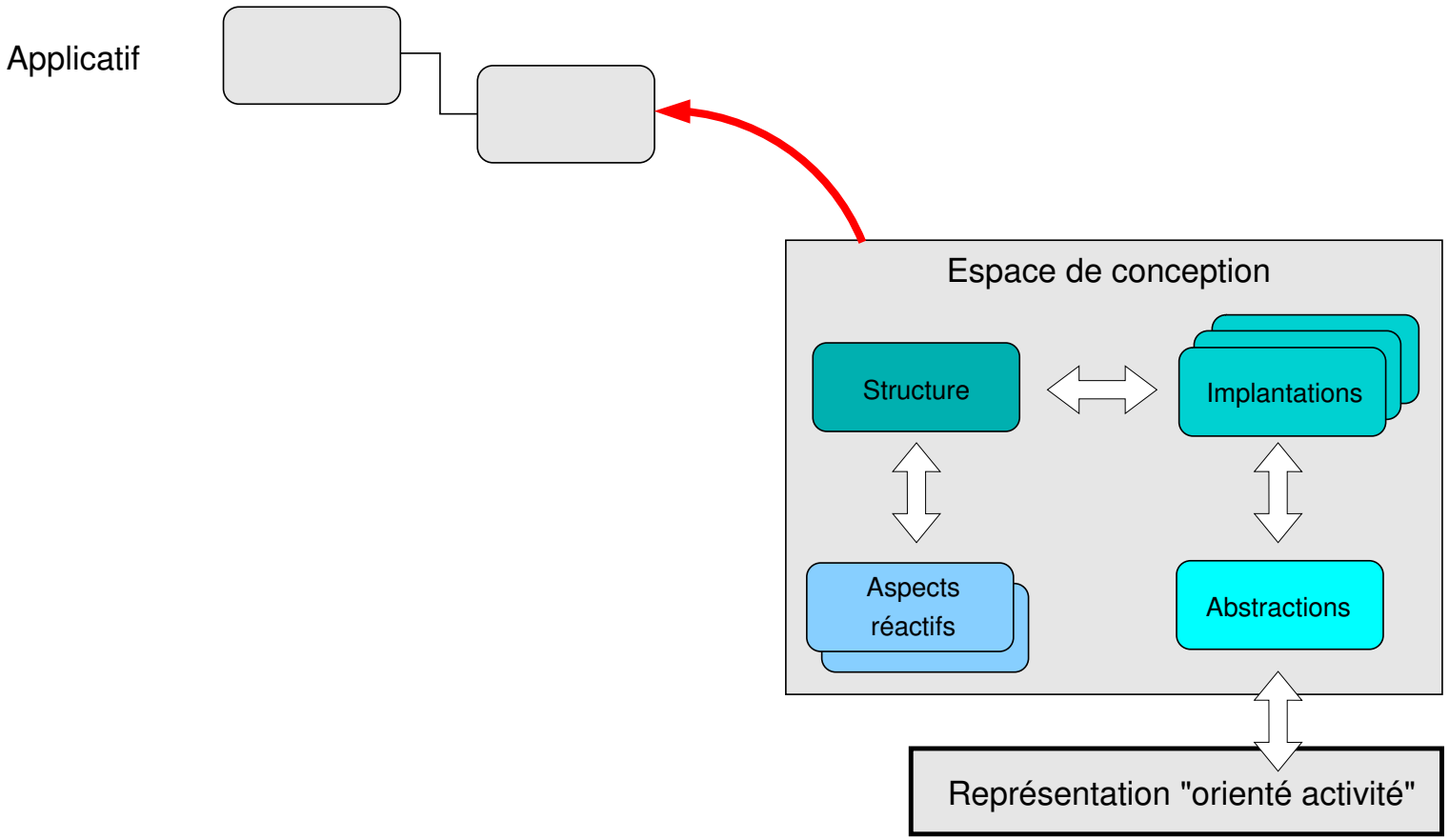
Disposer d'un **espace de conception orientée composant**,

- Capturant les préoccupations du domaine :
  - Aspects multi-tâches et réactifs,
  - Contraintes temporelles,
  - Patrons de communication et de synchronisation, etc
- Utilisable à tous niveaux d'abstraction
  - ADLs du domaine trop spécifiques et contraints
  - Approches composants classiques non adaptés
- Fournissant les abstractions nécessaires de l'implantation
  - Pour raisonner sur les aspects multi-tâches et concurrents
  - Tout en assurant la réutilisabilité

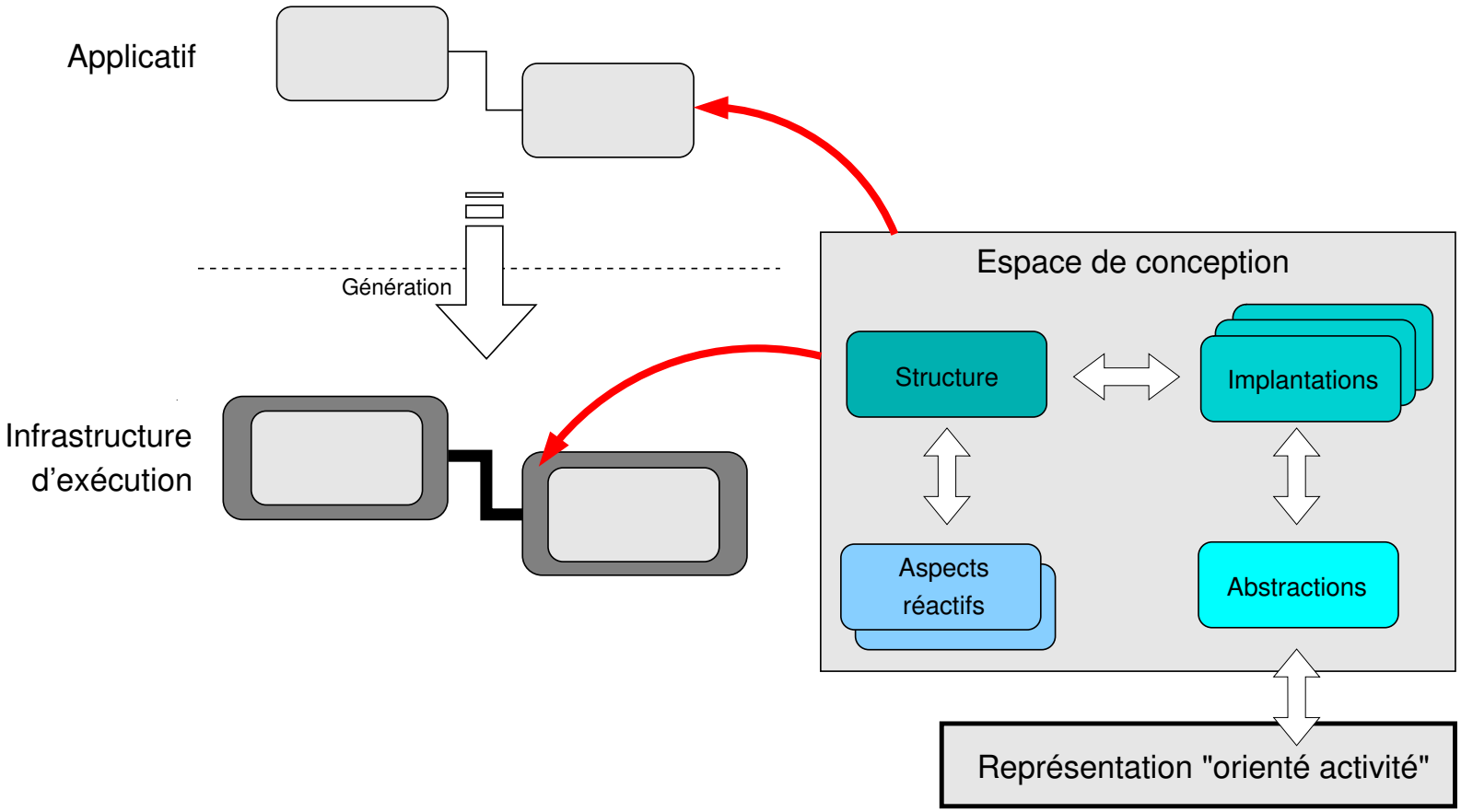
Vue d'ensemble de Tinap



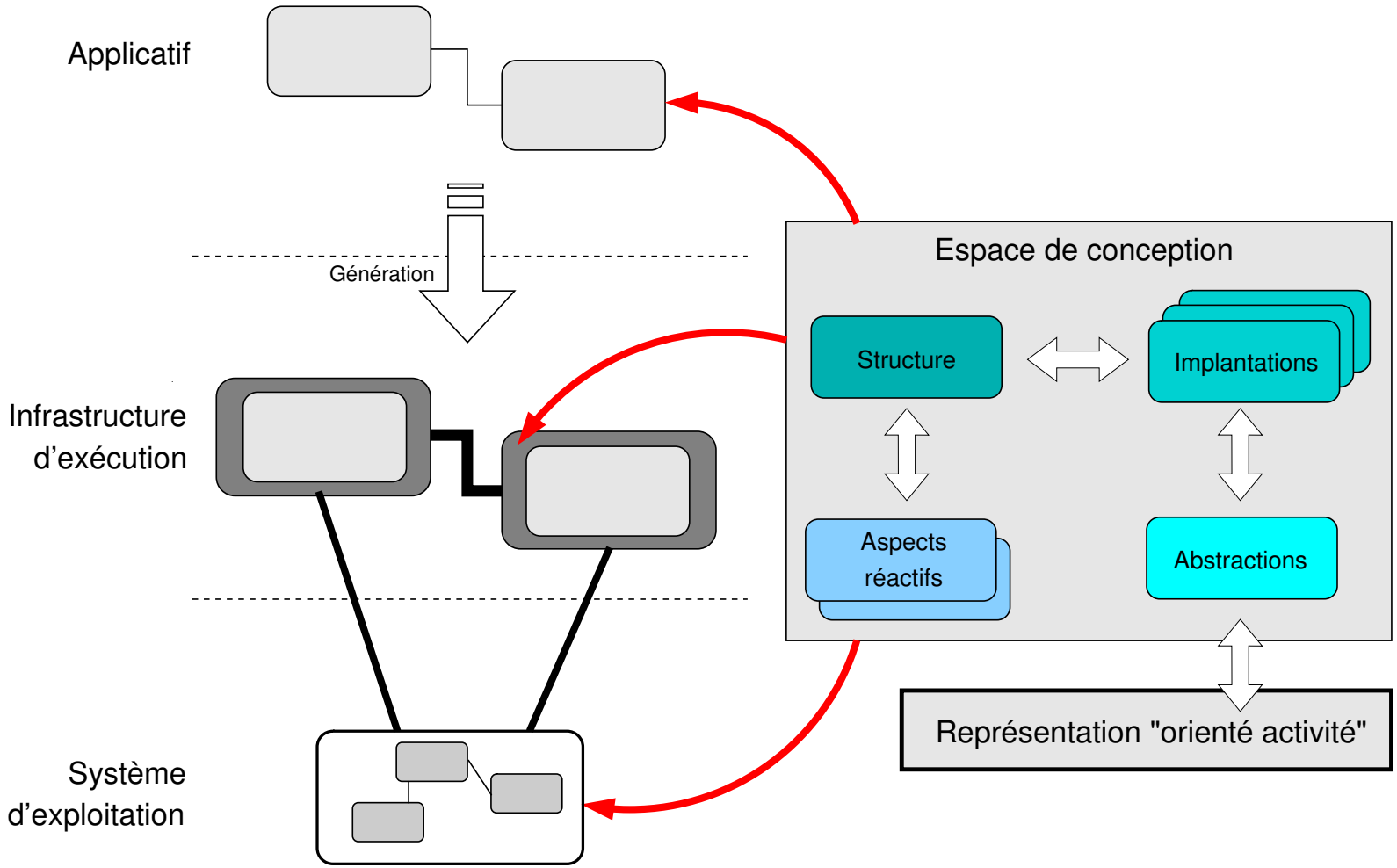
Vue d'ensemble de Tinap



Vue d'ensemble de Tinap



Vue d'ensemble de Tinap



Plan

Plan

Pré-conditions à la définition de l'espace de conception

- « **Propriétés générales** » des approches composant
  - Interfaces / Composabilité / Encapsulation hiérarchique
- Unité d'**encapsulation fonctionnelle et d'exécution arbitraire**

Pré-conditions à la définition de l'espace de conception

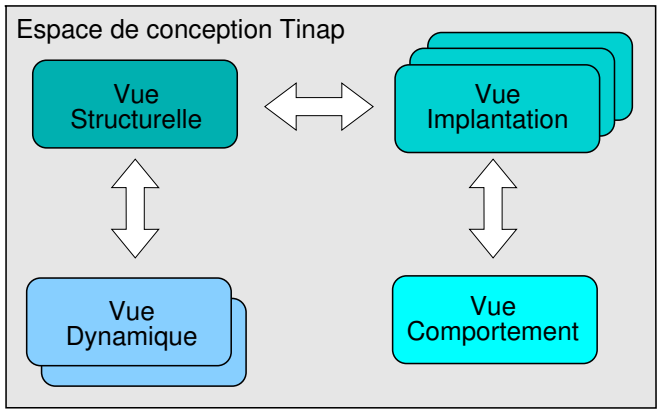
- « **Propriétés générales** » des approches composant
  - Interfaces / Composabilité / Encapsulation hiérarchique
- Unité d'**encapsulation fonctionnelle et d'exécution arbitraire**
- **Architecture logicielle** à « l'*épicentre* » du cycle de conception
  - Préoccupations spécifiques du domaine
  - Expressions de propriétés non-fonctionnelles

Pré-conditions à la définition de l'espace de conception

- « **Propriétés générales** » des approches composant
  - Interfaces / Composabilité / Encapsulation hiérarchique
- Unité d'**encapsulation fonctionnelle et d'exécution arbitraire**
- **Architecture logicielle** à « l'*épicentre* » du cycle de conception
  - Préoccupations spécifiques du domaine
  - Expressions de propriétés non-fonctionnelles
- « **Interfaces riches** »
  - Abstraire les propriétés de l'implantation
  - Raisonner sur les aspects comportementaux, multi-tâches et concurrents

Pré-conditions à la définition de l'espace de conception

- « **Propriétés générales** » des approches composant
  - Interfaces / Composabilité / Encapsulation hiérarchique
- Unité d'**encapsulation fonctionnelle et d'exécution arbitraire**
- **Architecture logicielle** à « l'*épicentre* » du cycle de conception
  - Préoccupations spécifiques du domaine
  - Expressions de propriétés non-fonctionnelles
- « **Interfaces riches** »
  - Abstraire les propriétés de l'implantation
  - Raisonner sur les aspects comportementaux, multi-tâches et concurrents

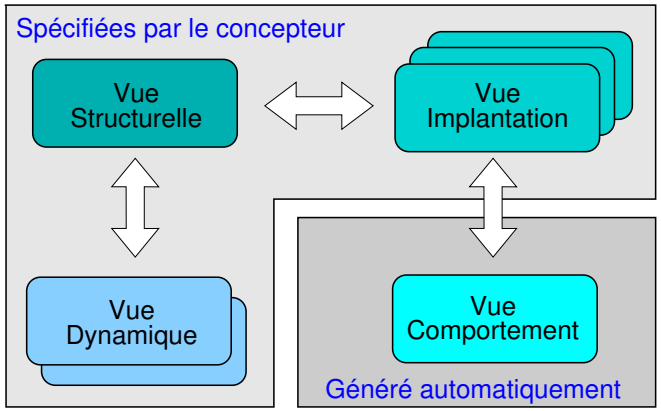


Espace de conception spécifié en quatre vues

Pré-conditions à la définition de l'espace de conception

- « **Propriétés générales** » des approches composant
  - Interfaces / Composabilité / Encapsulation hiérarchique
- Unité d'**encapsulation fonctionnelle et d'exécution arbitraire**

- **Architecture logicielle** à « l'*épicentre* » du cycle de conception
  - Préoccupations spécifiques du domaine
  - Expressions de propriétés non-fonctionnelles
- « **Interfaces riches** »
  - Abstraire les propriétés de l'implantation
  - Raisonner sur les aspects comportementaux, multi-tâches et concurrents

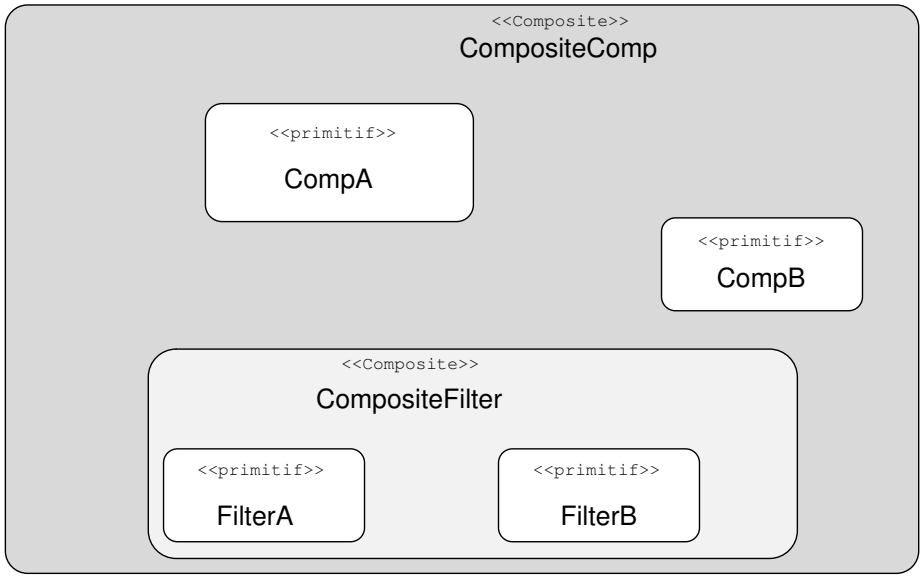


Espace de conception spécifié en quatre vues

Vue Structurelle

Caractéristiques

- **Composants** : primitifs, composites



Vue Structurelle

Caractéristiques

- **Composants** : primitifs, composites
- **Natures d'interfaces**
  - De services
  - De flots
    - Données / Événements
  - D'attributs

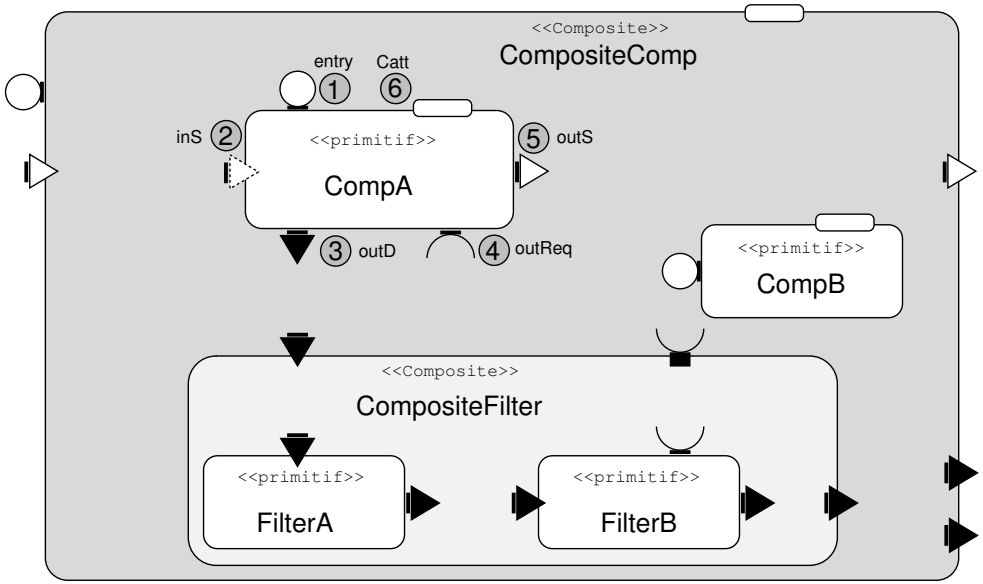
Service fourni

Service requis

Flot de Données

Flot d'Evénements

Attributs



Vue Structurelle

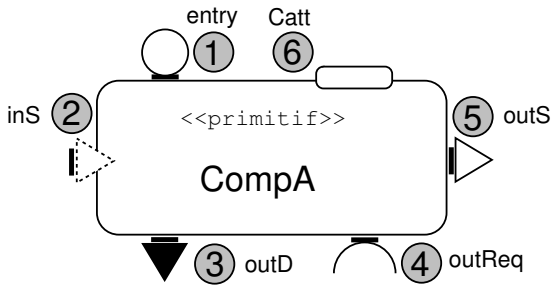
Caractéristiques

- **Composants** : primitifs, composites
- **Natures d'interfaces**
  - De services
  - De flots
    - Données / Événements
  - D'attributs
  - **Typées**

Interfaces d'entrée

1. entry
ServiceInterface `provItf`
  void mA (...);
  void mB (...);

2. inS
EventInterface `inSignItf`
  inNotifyEvt;



Interfaces de sortie

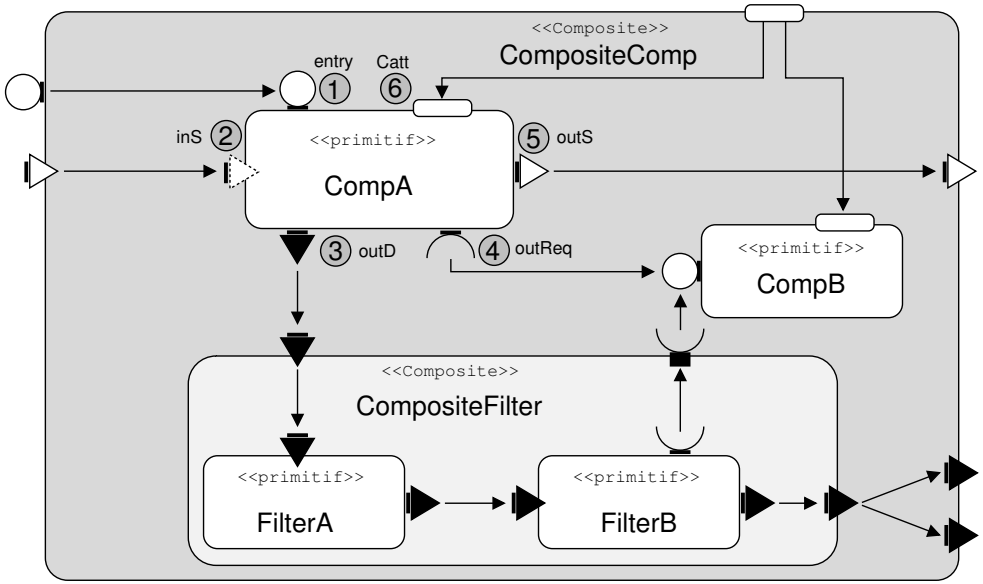
3. outD
DataInterface `outDataItf`
  MyStructType s;

4. outReq
ServiceInterface `reqItf`
  int mCa (...);
  String mCb (...);

Vue Structurelle

Caractéristiques

- **Composants** : primitifs, composites
- **Liaisons**
  - Horizontales ou déléguées
  - Selon les natures d'interfaces
  - Interfaces multi-liées
- **Natures d'interfaces**
  - De services
  - De flots
    - Données / Événements
  - D'attributs
  - **Typées**



Vue Dynamique

Motivations

- L'architecture logicielle offre de bonnes abstractions
  - Pour caractériser les préoccupations / patrons de conception nécessaires
  - L'encapsulation hiérarchique $\Rightarrow$ notion de domaine
- **Une approche descriptive pour personnaliser l'architecture** ($\approx$ DSL)

Vue Dynamique

Motivations

- L'architecture logicielle offre de bonnes abstractions
  - Pour caractériser les préoccupations / patrons de conception nécessaires
  - L'encapsulation hiérarchique $\Rightarrow$ notion de domaine
- **Une approche descriptive pour personnaliser l'architecture** ($\approx$ DSL)

TINAP : aspects liés à la concurrence

1. Aspects multi-tâches, règles d'activation
2. Modèles de communication et de synchronisation
3. Protection de sections critiques (granularité composant)

Vue Dynamique : **descripteurs**

Descripteurs s'appliquant aux composants

- **Activity** *Descriptor*
  - Spécifie les activités concurrentes (*Composants Actifs* vs. *Passifs*)
  - mono-actif / multi-actif
  - Avec opérateur logique
  - Sporadique, périodique
- **Protected** *Descriptor*
  - Protection des accès concurrents (*Composants Protégés*)

Vue Dynamique : **descripteurs**

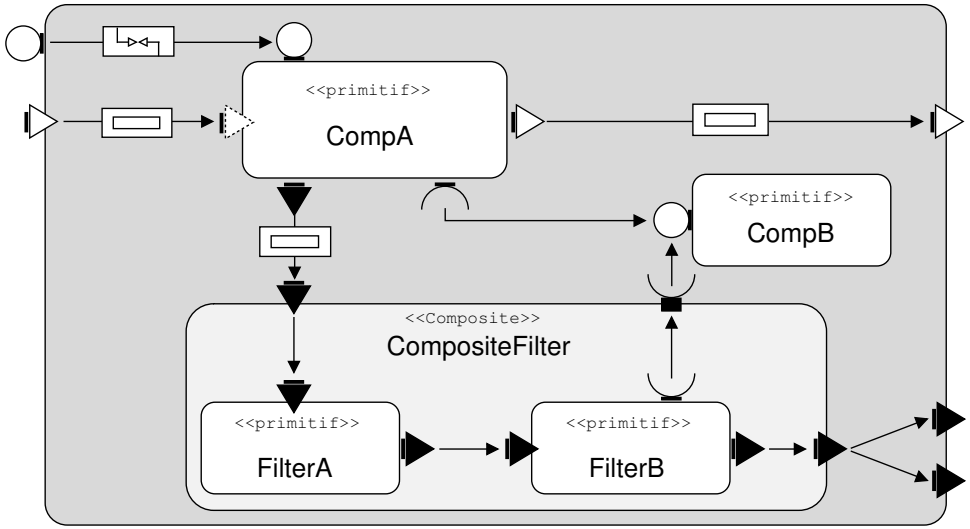
Descripteurs s'appliquant aux composants

- **Activity** *Descriptor*
  - Spécifie les activités concurrentes (*Composants Actifs* vs. *Passifs*)
  - mono-actif / multi-actif
  - Avec opérateur logique
  - Sporadique, périodique
- **Protected** *Descriptor*
  - Protection des accès concurrents (*Composants Protégés*)

Descripteurs s'appliquant aux liaisons

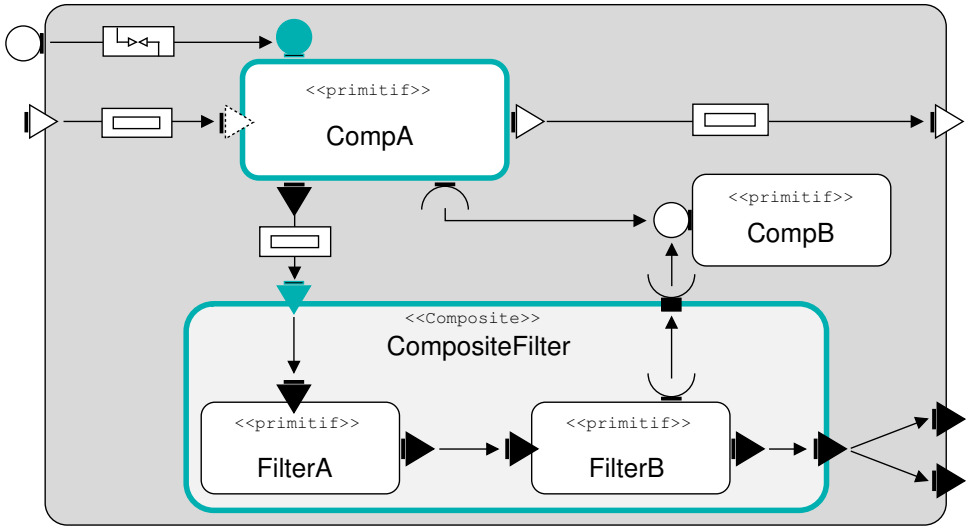
- **Binding** *Descriptor* ($\Rightarrow$ADL CLARA)
  - Modèles d'interaction et de synchronisation communément employés
- **Echéances temporelles**

Vue Dynamique : illustration



« Personnalisation » de l'architecture

Vue Dynamique : illustration



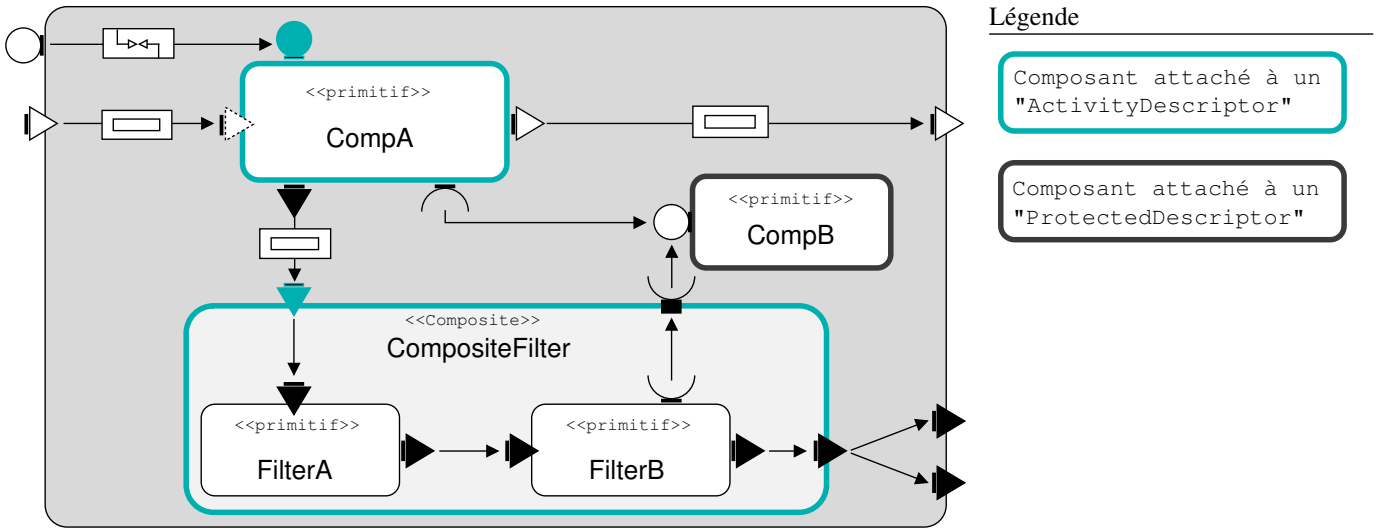
Légende

Composant attaché à un "ActivityDescriptor"

Composants Actifs

- Spécification des interfaces d'activation
- Configuration

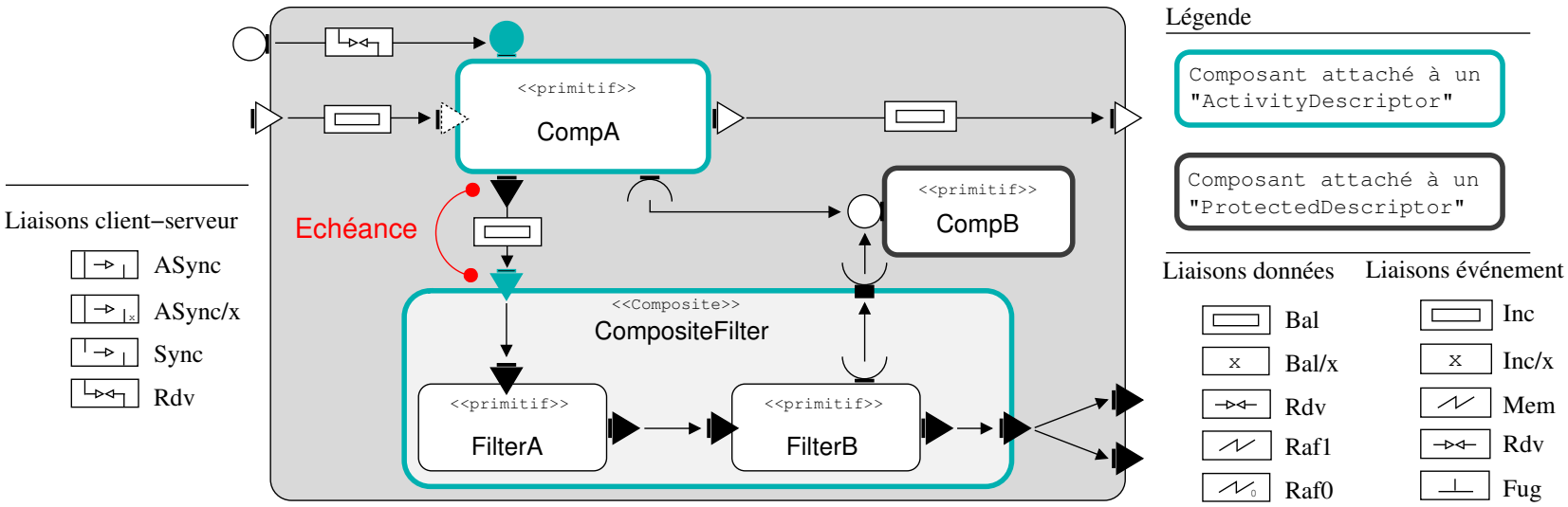
Vue Dynamique : illustration



Composants Protégés

- Configuration

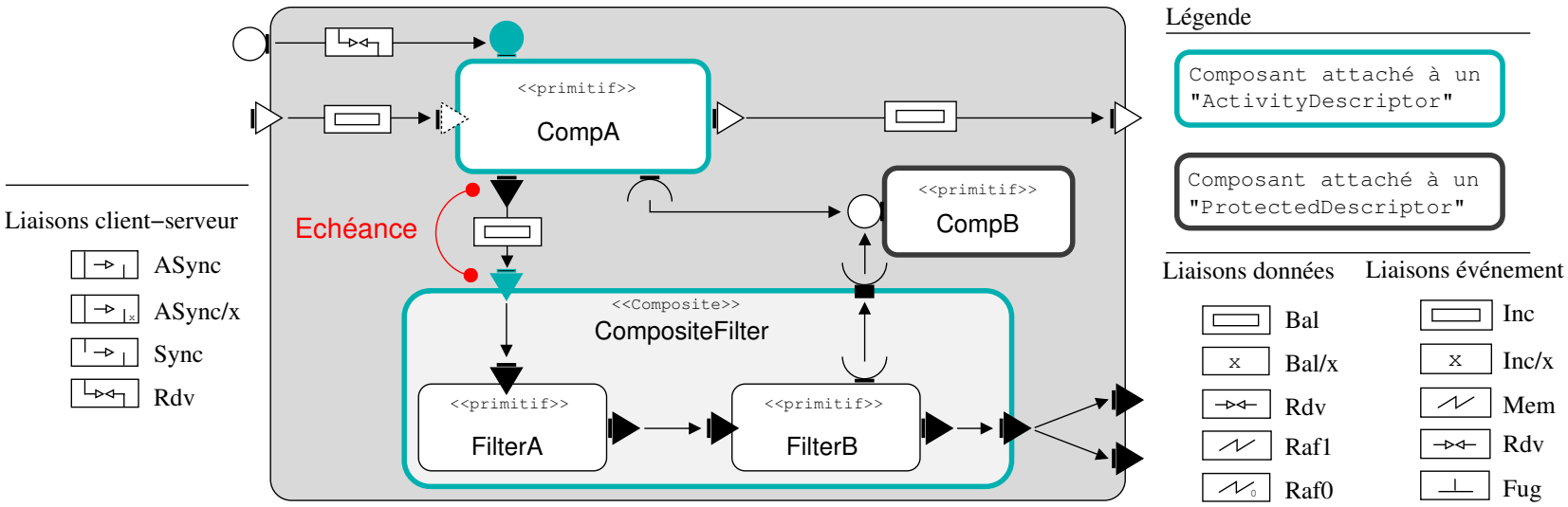
Vue Dynamique : illustration



Descripteurs de liaisons

- Spécification des modèles de communication
- Echéances temporelles

Vue Dynamique : illustration



Ensemble de contraintes sur l'application des descripteurs

Implantation des composants primitifs

Motivations

- Se focaliser exclusivement sur le comportement métier
- **Encapsulation fonctionnelle arbitraire**
- Assurer la **traçabilité entre implantation et concepts architecturaux**
- Implantations en C

Implantation des composants primitifs

Motivations

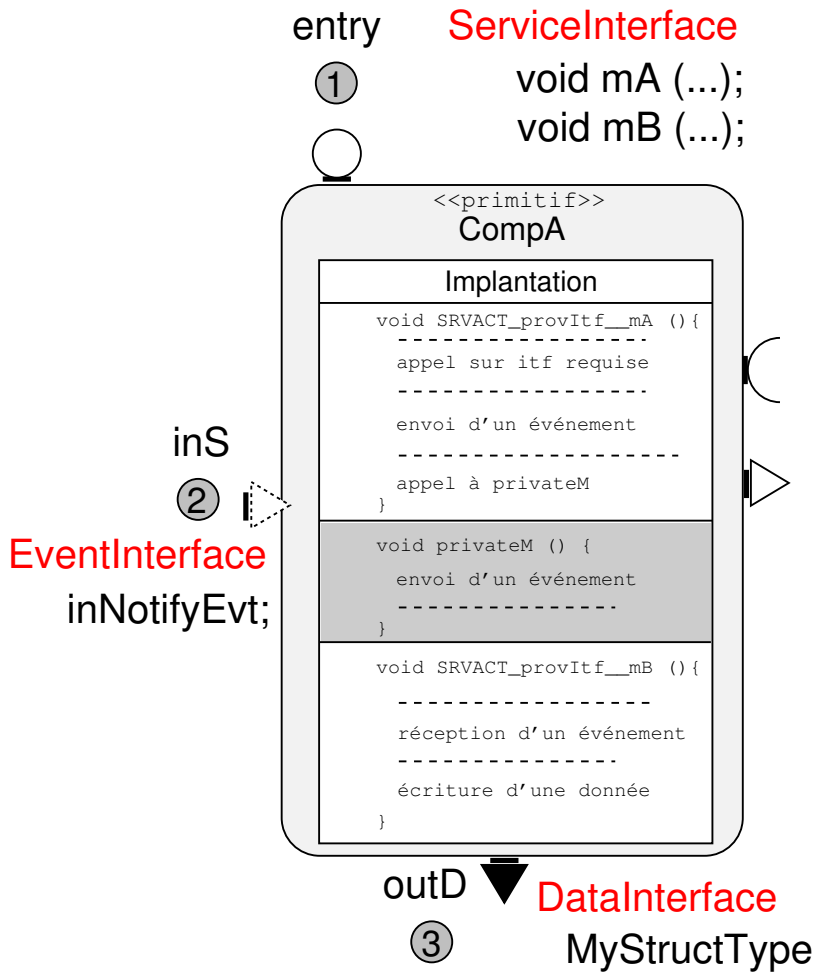
- Se focaliser exclusivement sur le comportement métier
- **Encapsulation fonctionnelle arbitraire**
- Assurer la **traçabilité entre implantation et concepts architecturaux**
- Implantations en C

Implantations des primitifs

- Mots clefs selon la nature des interfaces
- Conventions de nommage / codage ($\Rightarrow$THINK)

Exemple d'implantation

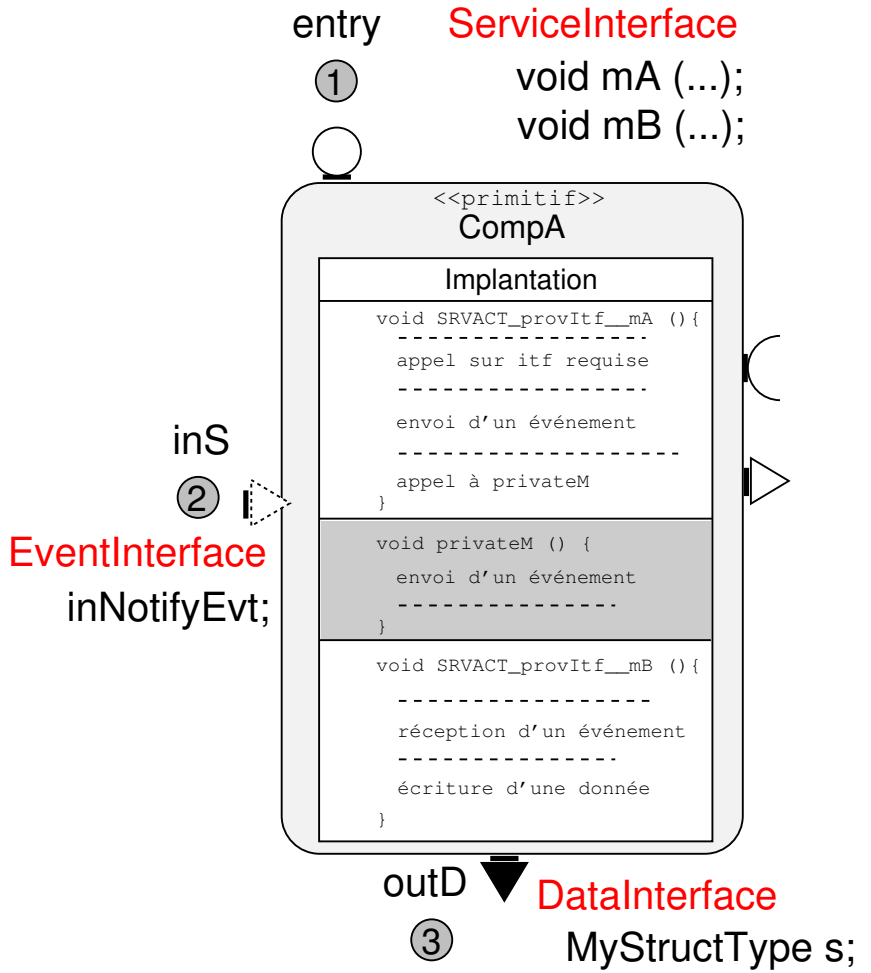
```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier

      PRV_privateM ();
}
void PRV_privateM() {

      if (ATT_b == TRUE)
        CLTSEND_outS__EventB();
      else
        ######  Bloc de code métier
}
// Point d'entrée de séquence de mB
void SRVACT_entry__mB (...) {

      ######  Bloc de code métier

      CLTCONSUME_inS__InNotifyEvt();
      ######  Bloc de code métier

      CLTWRITE_outD__s (...);
}
```



Exemple d'implantation

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier

      PRV_privateM ();
}
void PRV_privateM() {

      if (ATT_b == TRUE)
        CLTSEND_outS__EventB();
      else
        ######  Bloc de code métier
}
```

```
// Point d'entrée de séquence de mB
void SRVACT_entry__mB (...) {

      ######  Bloc de code métier

      CLTCONSUME_inS__InNotifyEvt();
      ######  Bloc de code métier

      CLTWRITE_outD__s (...);
}
```



Vue Comportement (de niveau composant)

Motivations

- Fournir une abstraction du comportement des composants
- Externaliser certaines caractéristiques de l'implantation

Vue Comportement (de niveau composant)

Motivations

- Fournir une abstraction du comportement des composants
- Externaliser certaines caractéristiques de l'implantation

Représentation statique sous forme d'automate

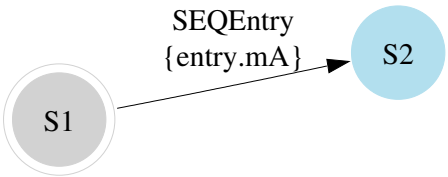
- Informations :
  1. Traces d'exécution des interactions avec l'environnement
  2. Blocs de code séquentiels (entourés par les structures de contrôle)
  3. Accès aux variables partagées du composant
- Généré automatiquement

Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
```

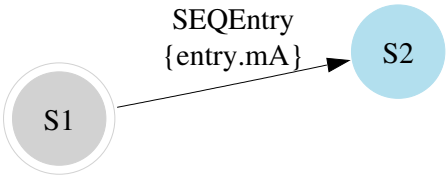
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
```



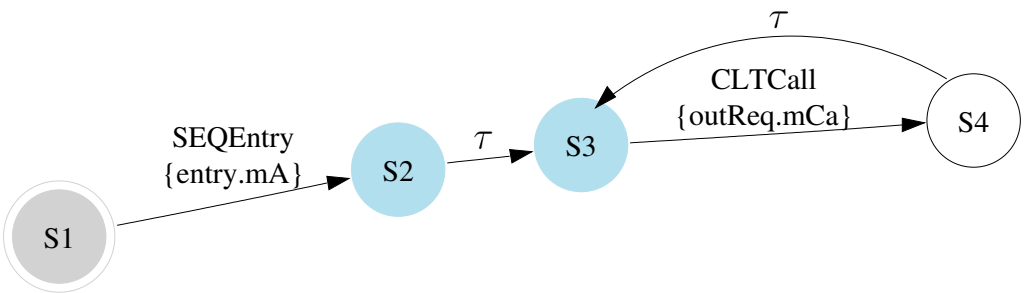
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }
```



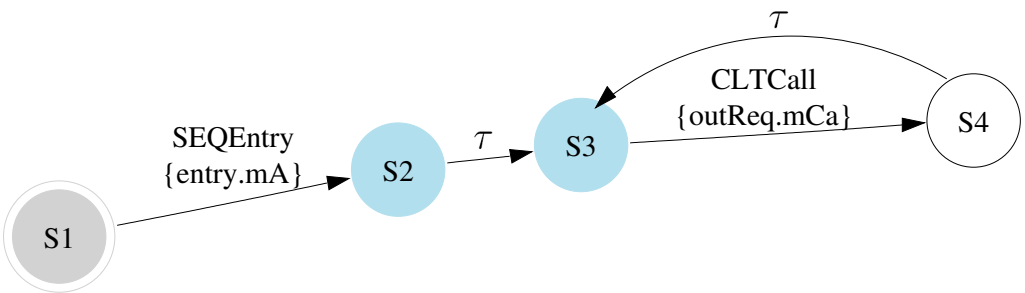
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }
```



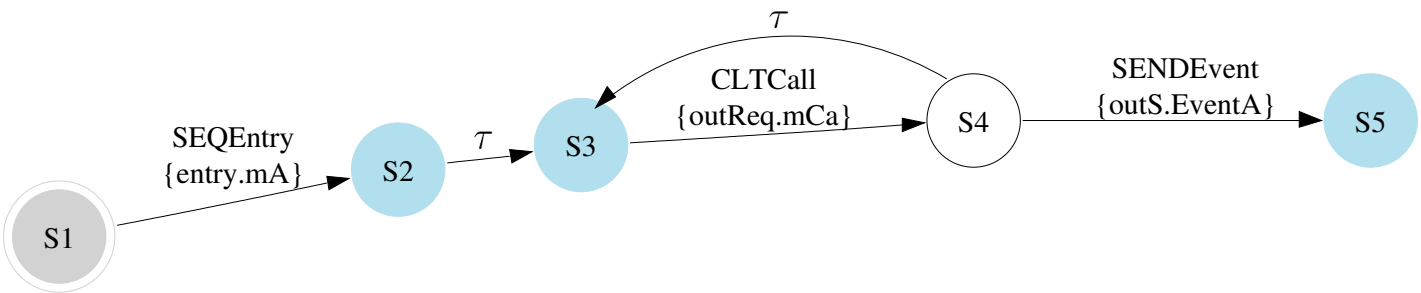
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier
```



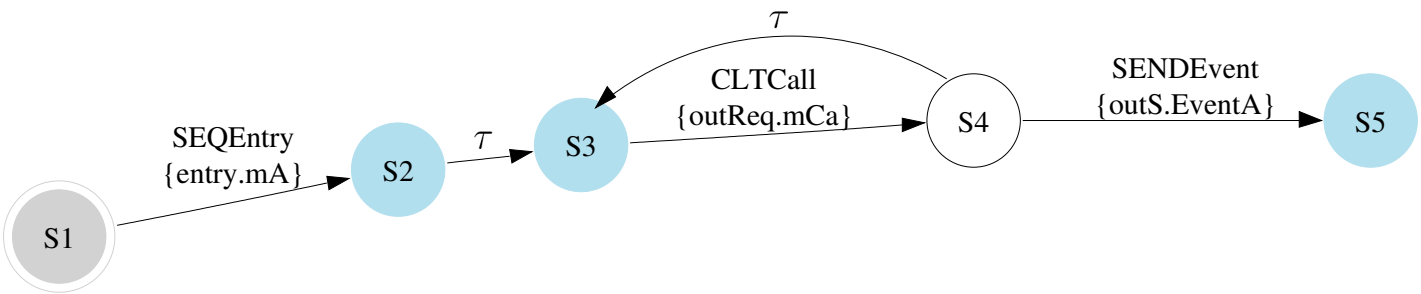
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier
```



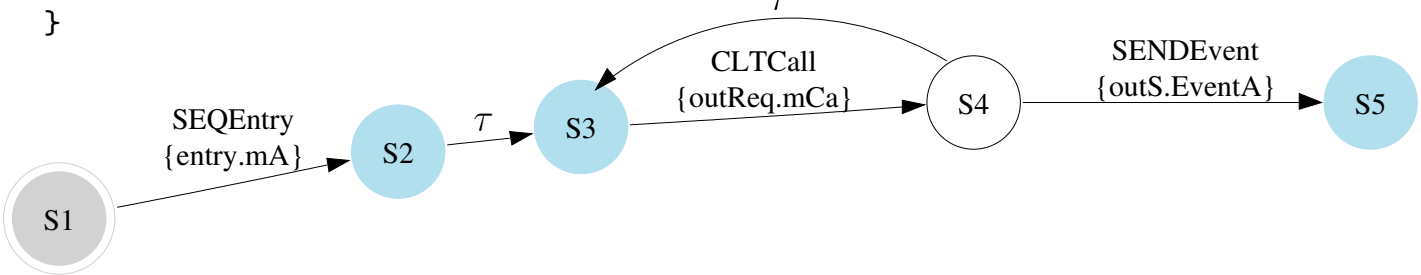
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier

      PRV_privateM ();
```



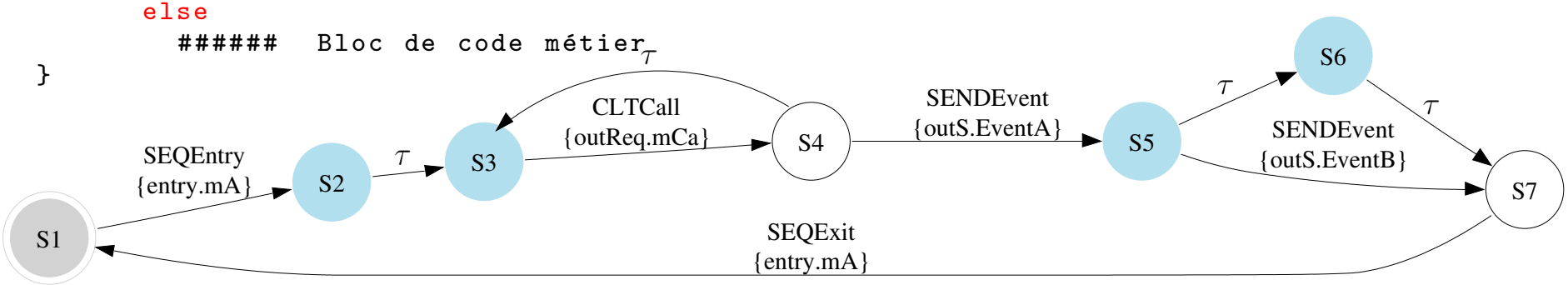
Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier

      PRV_privateM ();
}
void PRV_privateM() {

      if (ATT_b == TRUE)
        CLTSEND_outS__EventB();
      else
        ######  Bloc de code métier
}
```



Vue Comportement, illustration

```
// Point d'entrée de séquence mA
void SRVACT_entry__mA (...) {

  ######  Bloc de code métier
      for (...) {

        CLT_outReq__mCa (...);
      }

      CLTSEND_outS__EventA();
      ######  Bloc de code métier

      PRV_privateM ();
}
void PRV_privateM() {

      if (ATT_b == TRUE)
        CLTSEND_outS__EventB();
      else
        ######  Bloc de code métier
}
```



Bilan sur l'espace de conception Tinap

- Un processus de conception « orientée fonctionnalité »
- Un support de schémas d'interaction symétrique et asymétrique
- Découplage des aspects structurels des préoccupations du domaine
  - Différentes « vues dynamiques » superposables sur une même vue structurelle
  - Implantations exclusivement centrées sur le métier, en C
- Traçabilité entre implantations et interfaces de la vue structurelle
- Externaliser les caractéristiques de l'implantation nécessaires pour raisonner sur le comportement (« interfaces riches »)

Bilan sur l'espace de conception Tinap

- Un processus de conception « orientée fonctionnalité »
- Un support de schémas d'interaction symétrique et asymétrique
- Découplage des aspects structurels des préoccupations du domaine
  - Différentes « vues dynamiques » superposables sur une même vue structurelle
  - Implantations exclusivement centrées sur le métier, en C
- Traçabilité entre implantations et interfaces de la vue structurelle
- Externaliser les caractéristiques de l'implantation nécessaires pour raisonner sur le comportement (« interfaces riches »)

Vue activité

- **Applicatif Tinap :**
  - Assemblage de composants actifs, passifs, protégés
  - $\neq$ Modèle centré sur les tâches

Vue activité

- **Applicatif Tinap :**
  - Assemblage de composants actifs, passifs, protégés
  - $\neq$ Modèle centré sur les tâches
- Comportement de l'application orthogonal à l'architecture

Vue activité

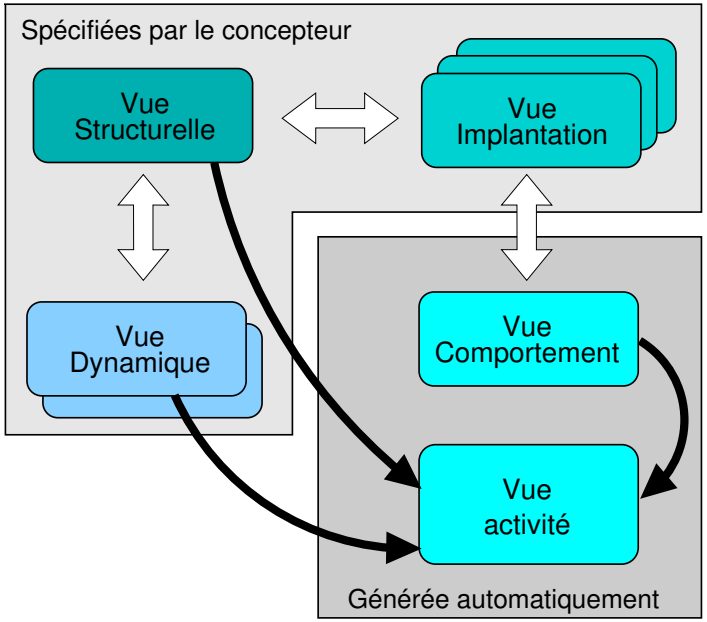
- **Applicatif Tinap :**
  - Assemblage de composants actifs, passifs, protégés
  - $\neq$ Modèle centré sur les tâches
- Comportement de l'application orthogonal à l'architecture

Motivations

- Représenter la nature concurrente et réactive de l'application
  - Faciliter la compréhension de l'application en cours de conception
  - Réduire l'écart entre abstractions Tinap et « modèle de tâche »
    - Perspectives d'analyses
- Générée automatiquement

Vue activité

- **Applicatif Tinap :**
  - Assemblage de composants actifs, passifs, protégés
  - $\neq$ Modèle centré sur les tâches
- Comportement de l'application orthogonal à l'architecture



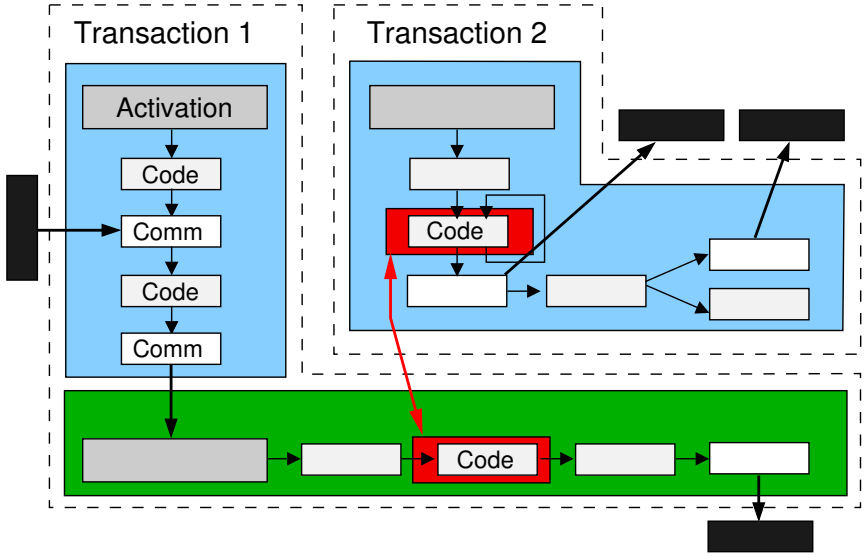
Motivations

- Représenter la nature concurrente et réactive de l'application
  - Faciliter la compréhension de l'application en cours de conception
  - Réduire l'écart entre abstractions Tinap et « modèle de tâche »
    - Perspectives d'analyses
- Générée automatiquement

Abstractions manipulées

Une représentation qui réifie

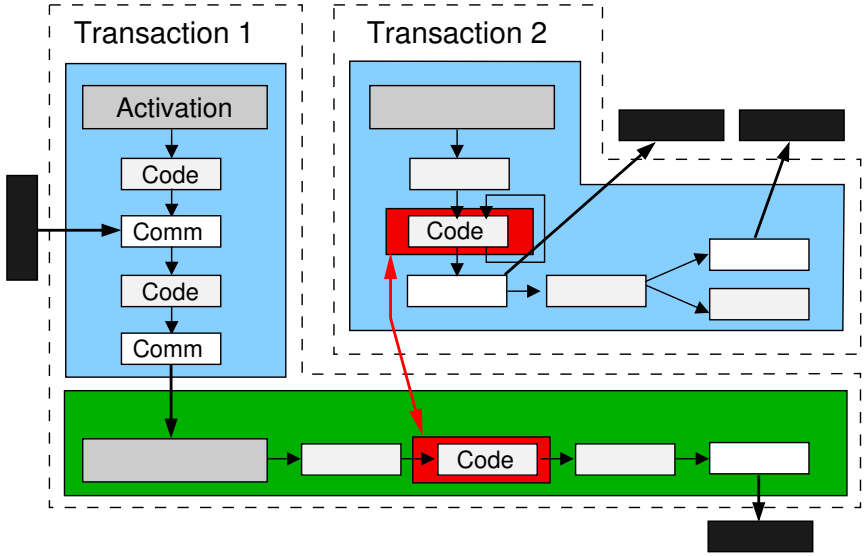
- **Les tâches**, se caractérisant par



Abstractions manipulées

Une représentation qui réifie

- **Les tâches**, se caractérisant par
  - Leurs règles d'activation
  - Un séquencement :
    1. De blocs de code séquentiels perçus comme unité d'ordonnancement
    2. De synchronisations avec les autres tâches et/ou l'environnement

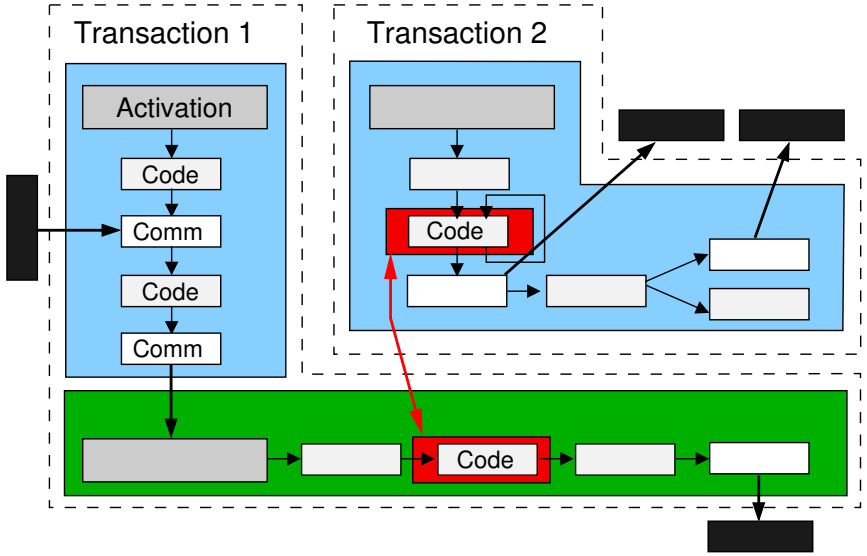


Abstractions manipulées

Une représentation qui réifie

- **Les tâches**, se caractérisant par
  - Leurs règles d'activation
  - Un séquencement :
    1. De blocs de code séquentiels perçus comme unité d'ordonnancement
    2. De synchronisations avec les autres tâches et/ou l'environnement

- **Les transactions**
  - Flot d'exécution de bout-en-bout
  - Liens de causalité entre tâches

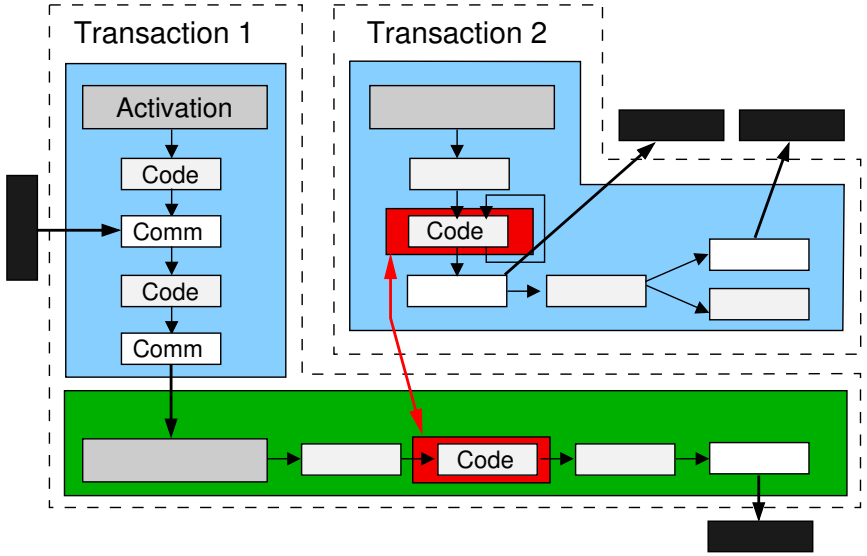


Abstractions manipulées

Une représentation qui réifie

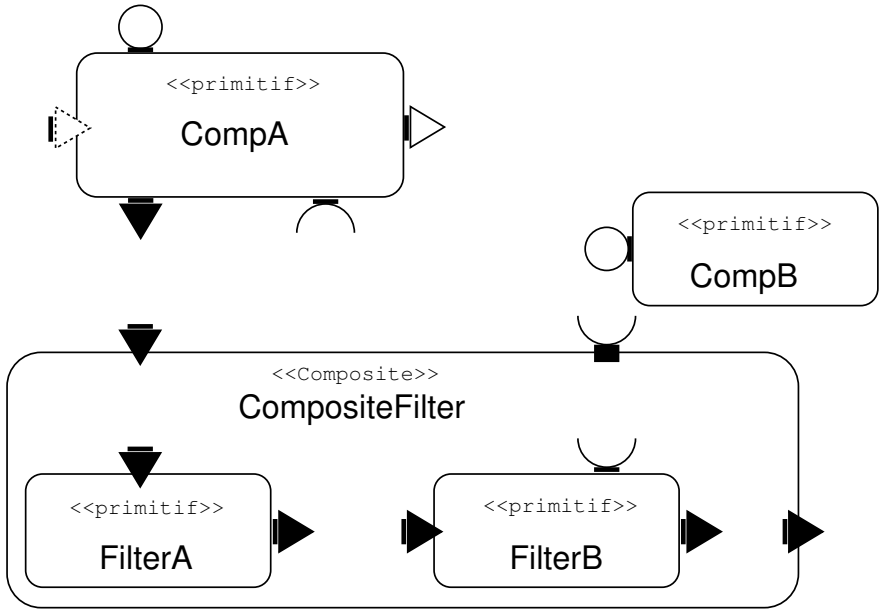
- **Les tâches**, se caractérisant par
  - Leurs règles d'activation
  - Un séquencement :
    1. De blocs de code séquentiels perçus comme unité d'ordonnancement
    2. De synchronisations avec les autres tâches et/ou l'environnement

- **Les transactions**
  - Flot d'exécution de bout-en-bout
  - Liens de causalité entre tâches
- **Les dépendances entre transactions**
  - Ressources partagées



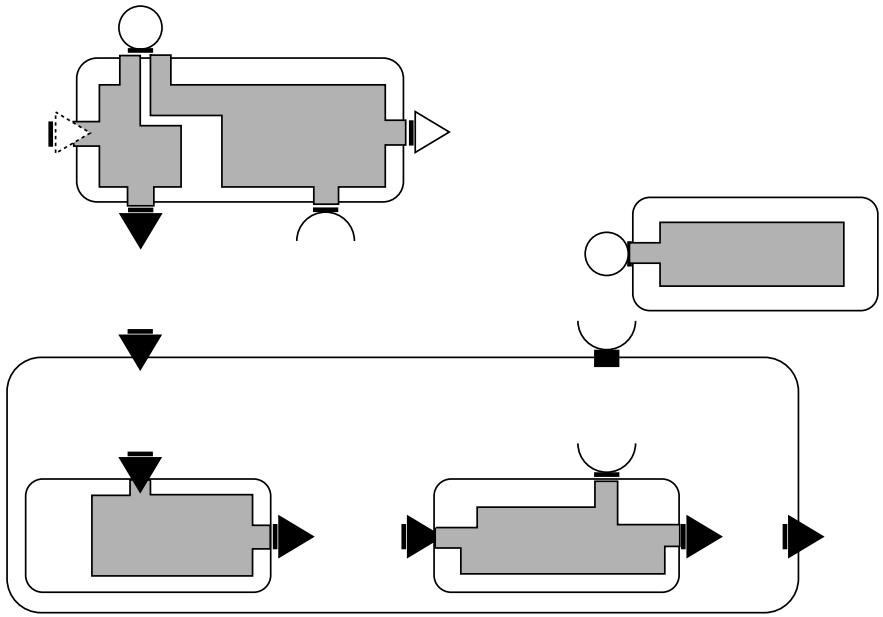
Vue activité : Processus de génération

À partir des composants primitifs



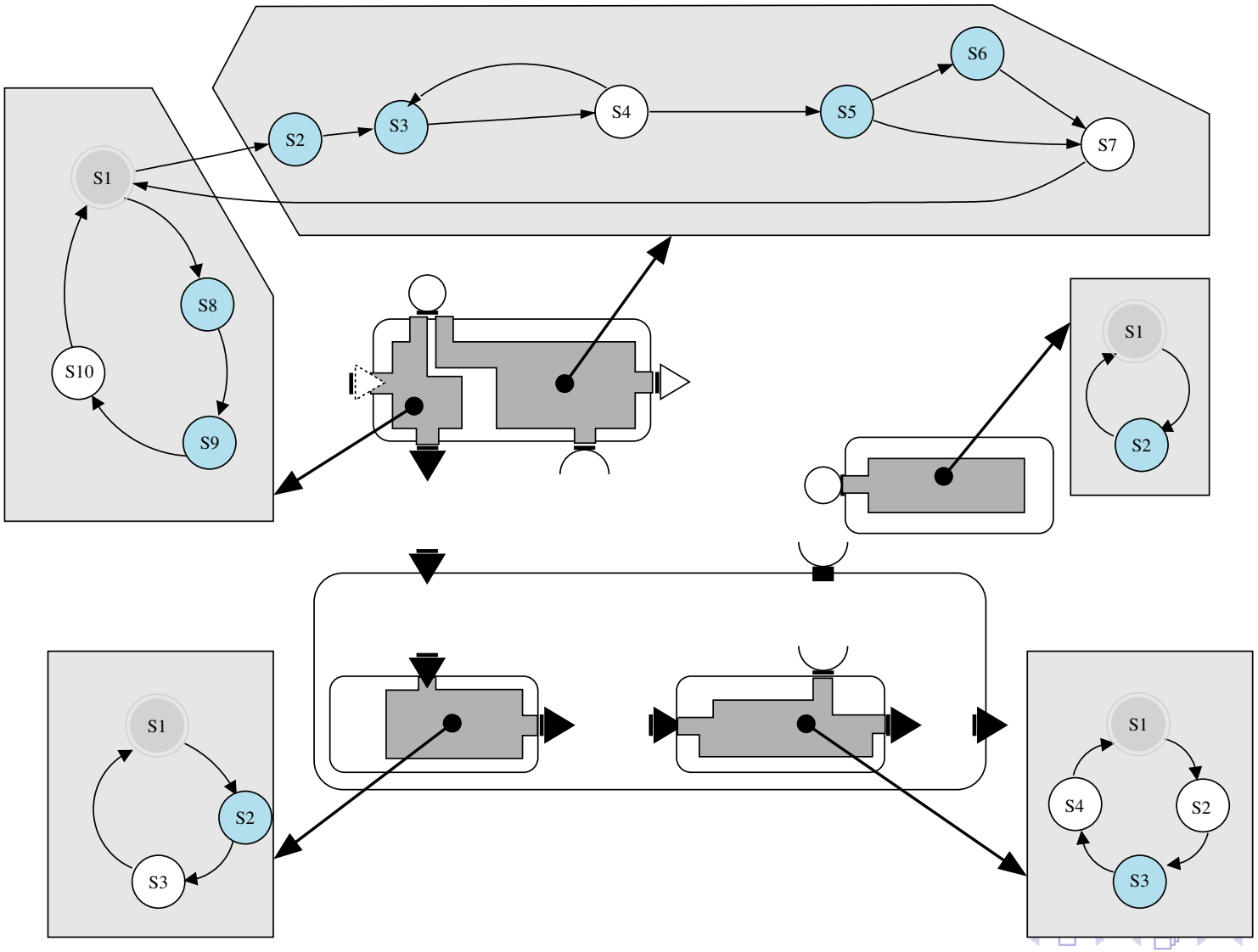
Vue activité : Processus de génération

« **Vue comportement** » : chemins d'exécution de chaque composant,



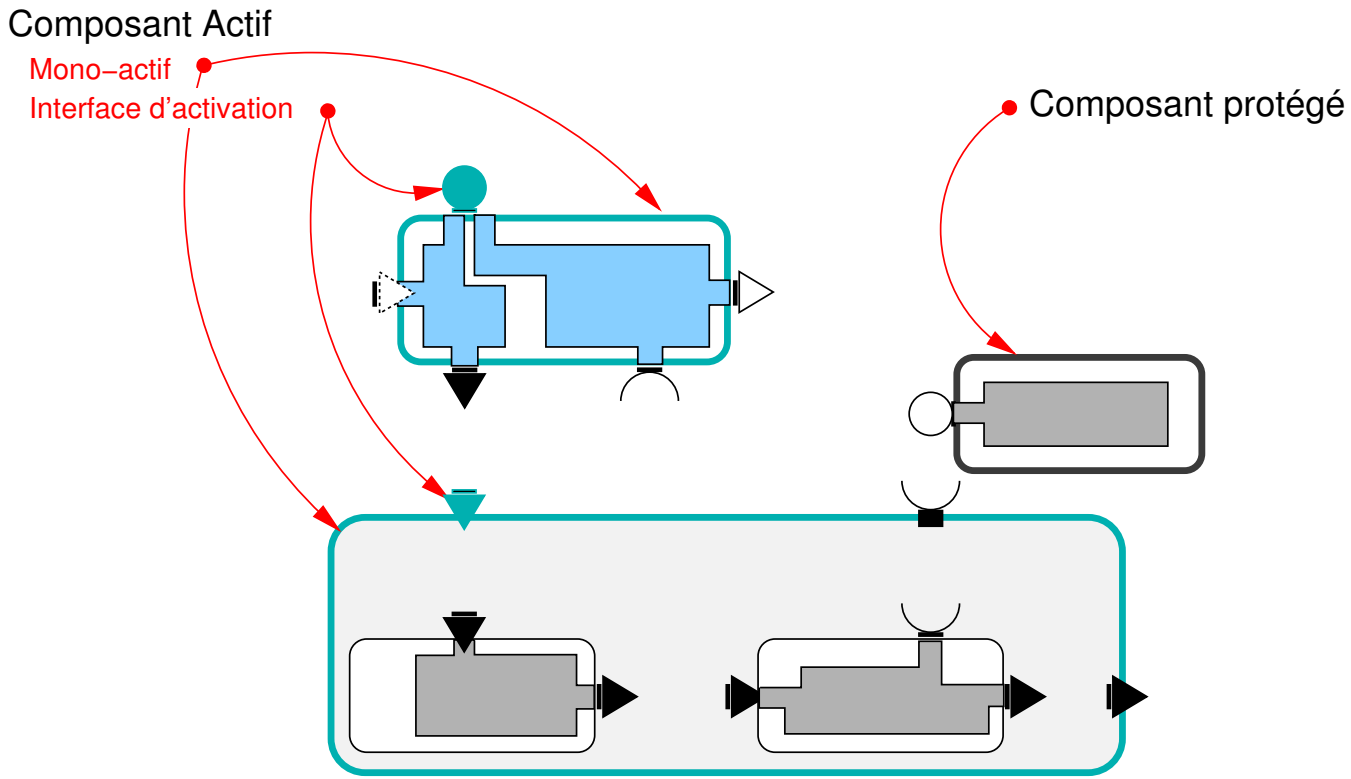
Vue activité : Processus de génération

Et de leurs **comportements**



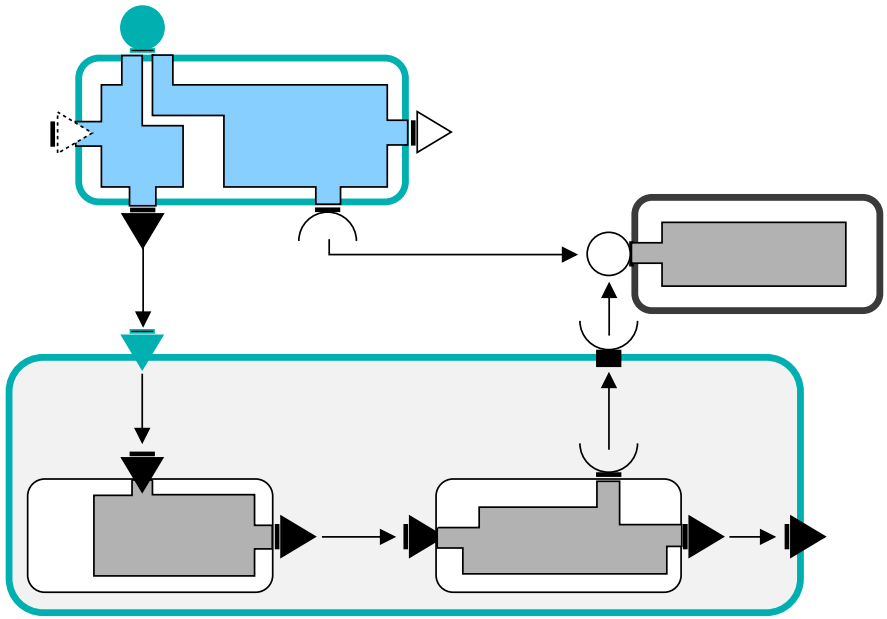
Vue activité : Processus de génération

À partir de la « **vue dynamique** » : connaissance de l'allocation des tâches



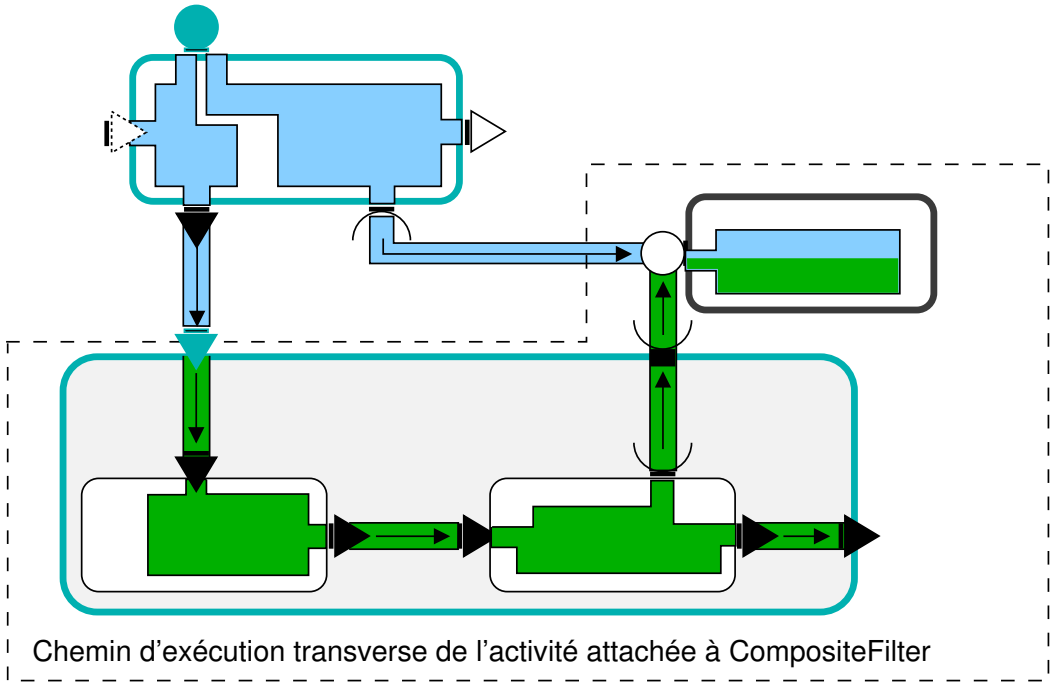
Vue activité : Processus de génération

À partir des interactions entre les composants



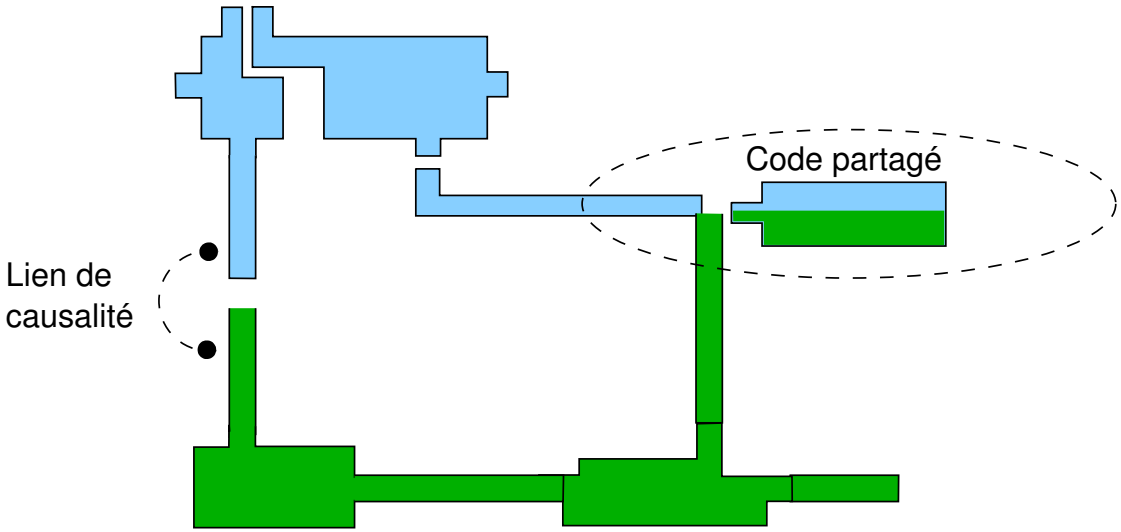
Vue activité : Processus de génération

Extraction des **chemins d'exécution transverses à l'architecture**



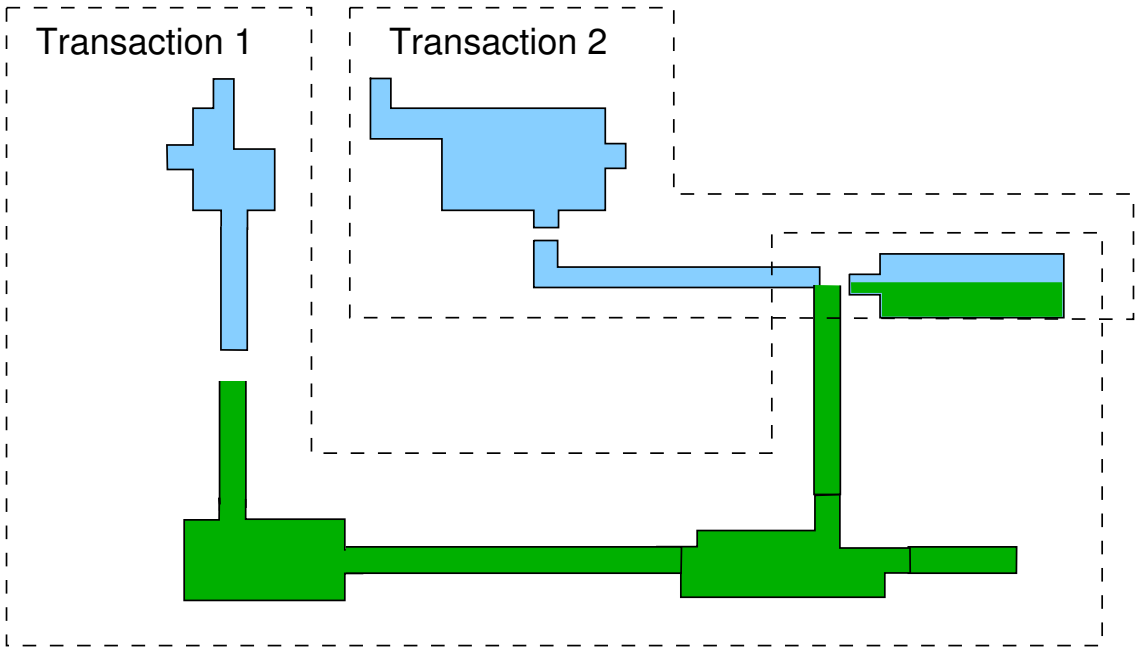
Vue activité : Processus de génération

Liens de causalité entre tâches / partages de ressources



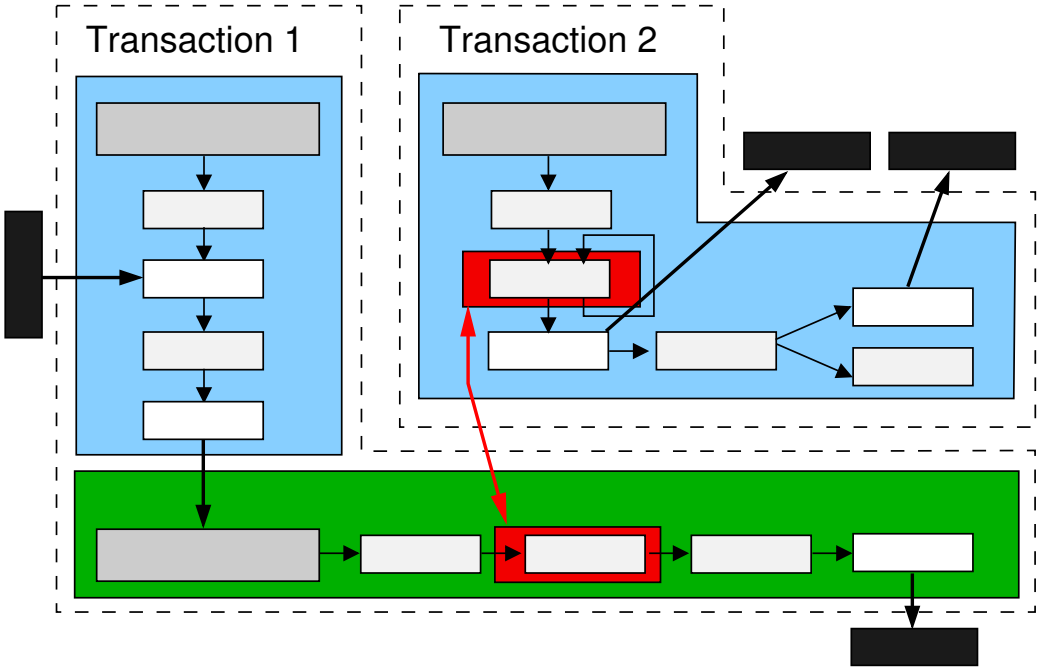
Vue activité : Processus de génération

Obtention d'une « **vue transactionnelle** »

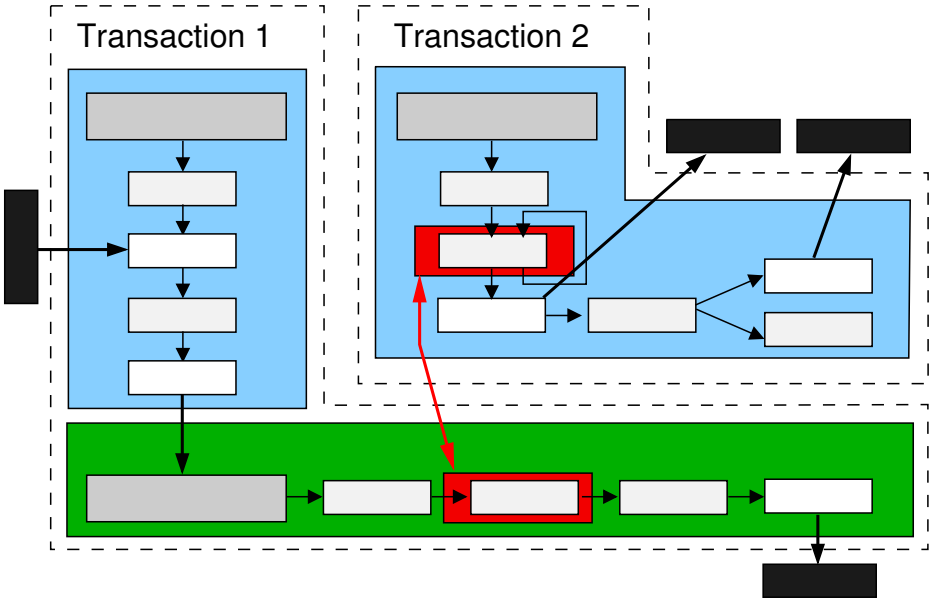


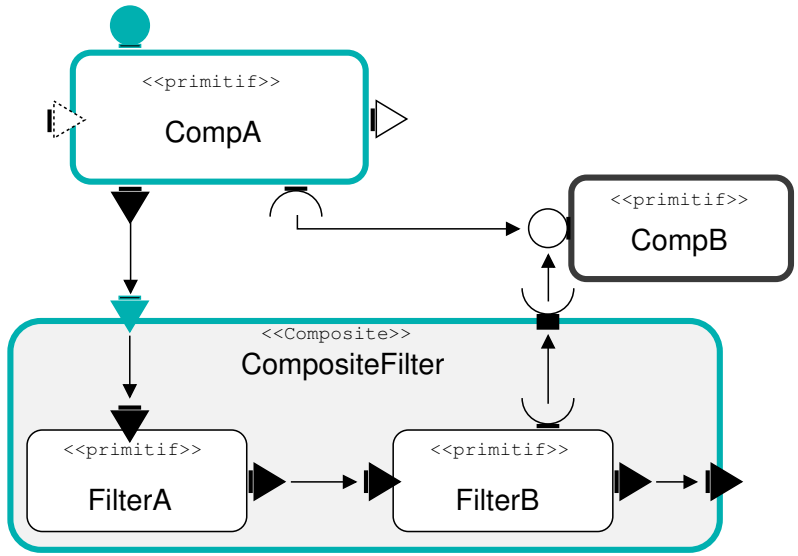
Vue activité : Processus de génération

Obtention de la Vue activité



Vue activité : traçabilité

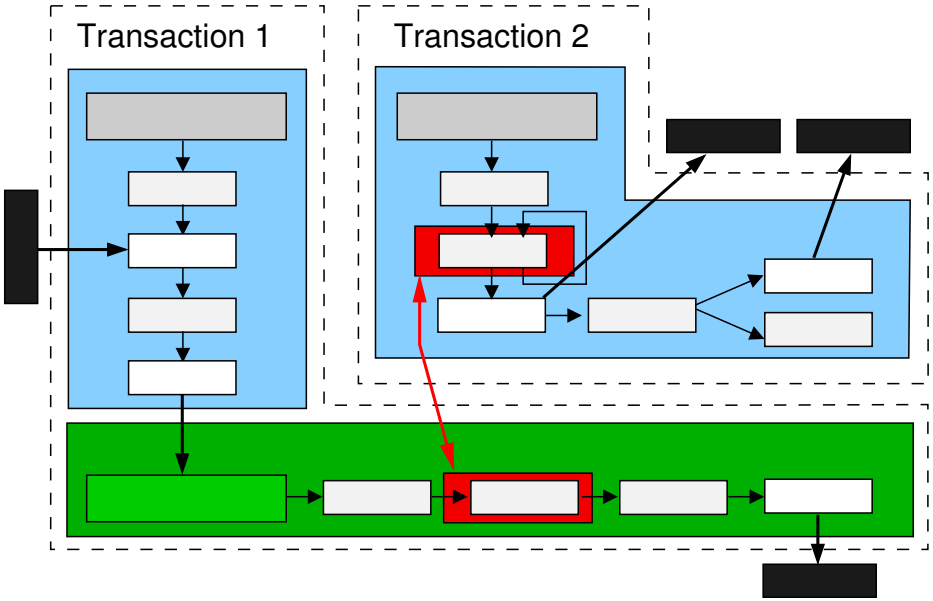


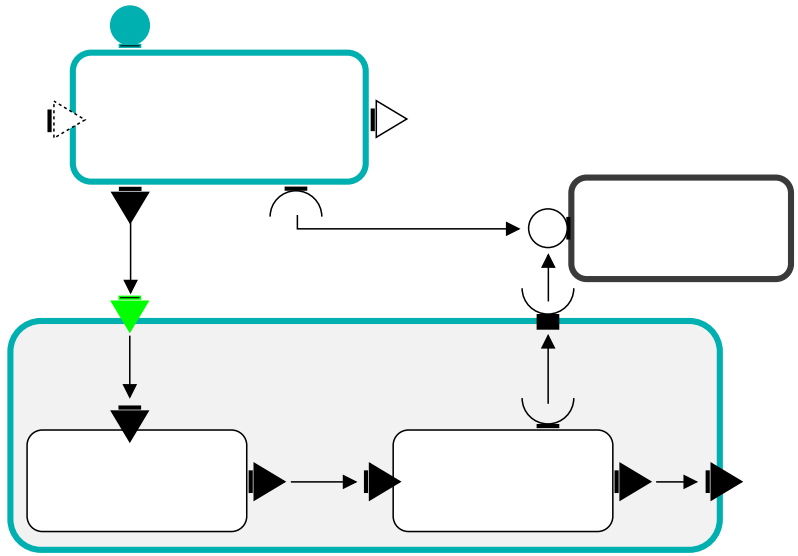


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité

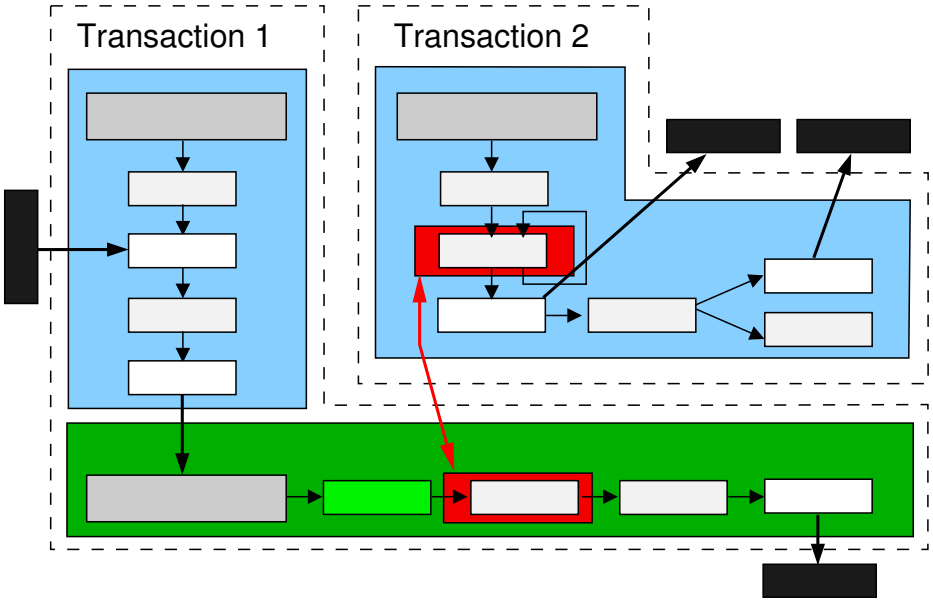


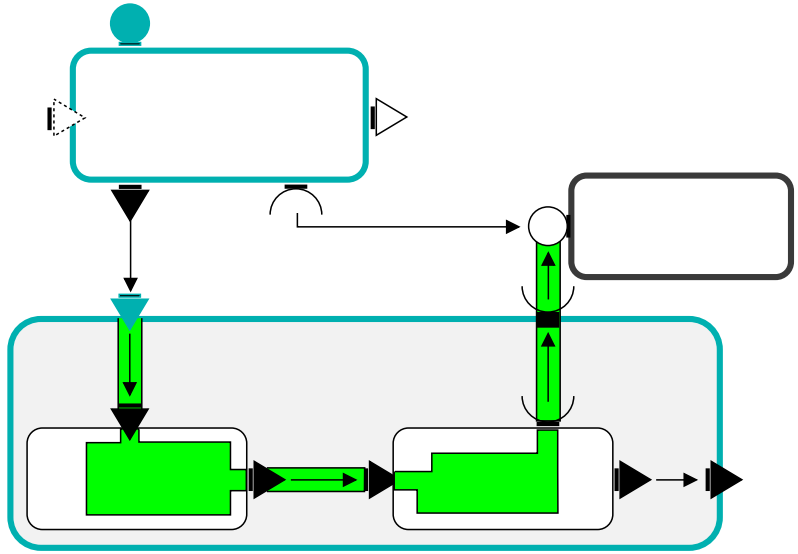


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité

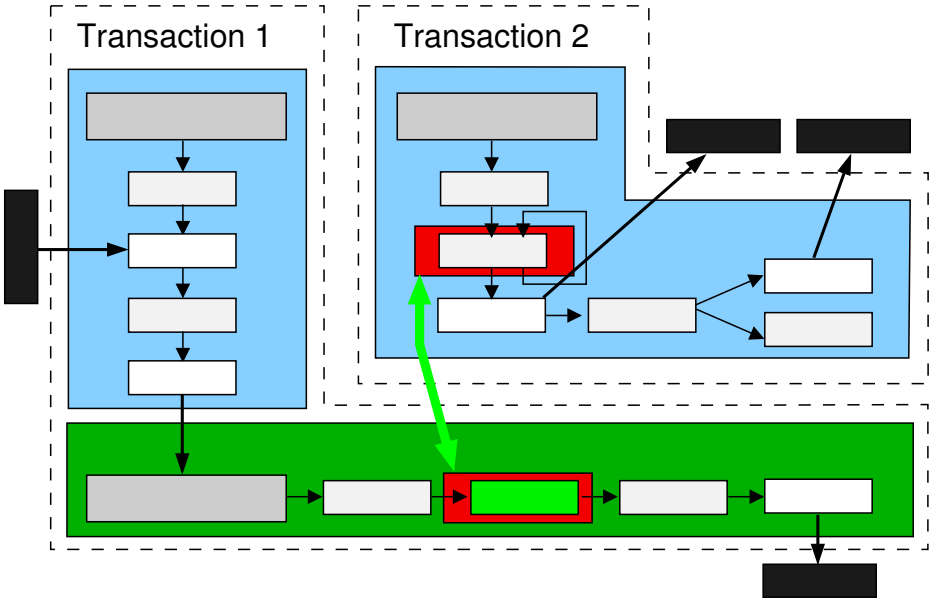


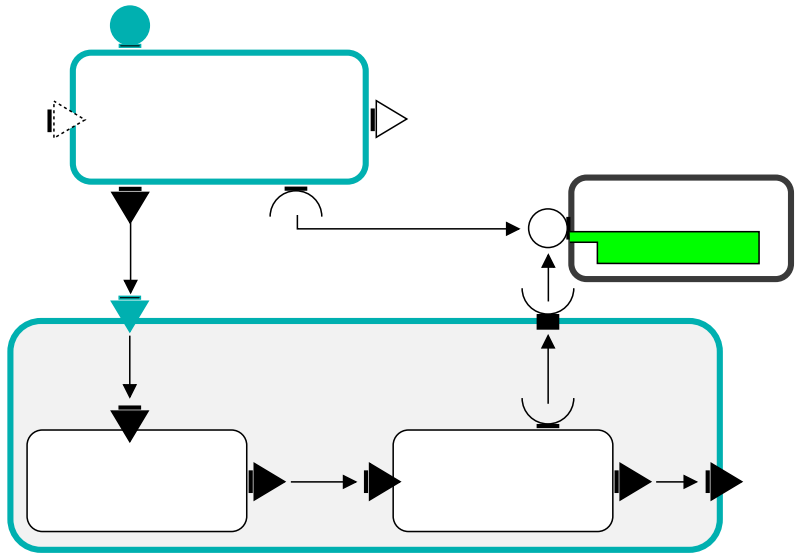


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité

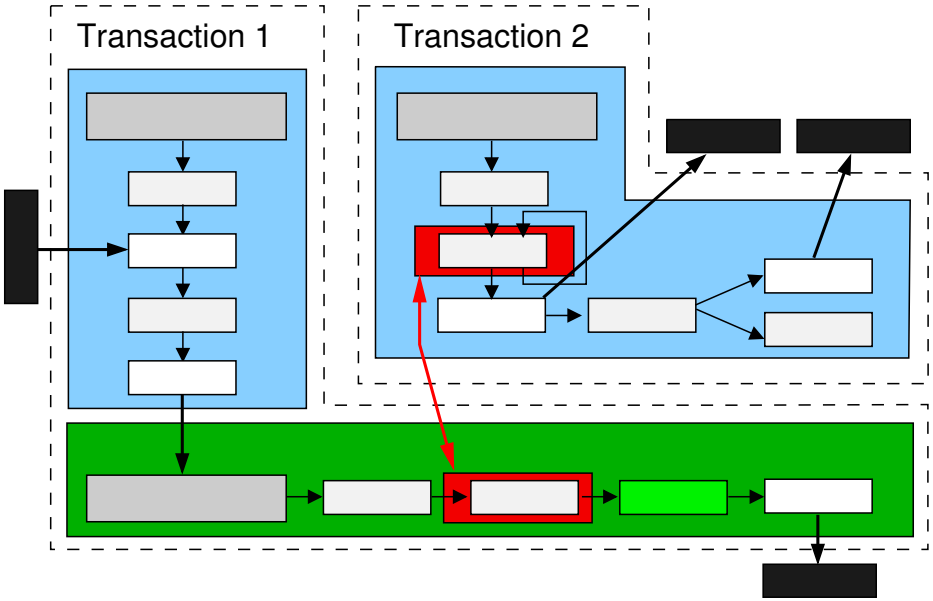


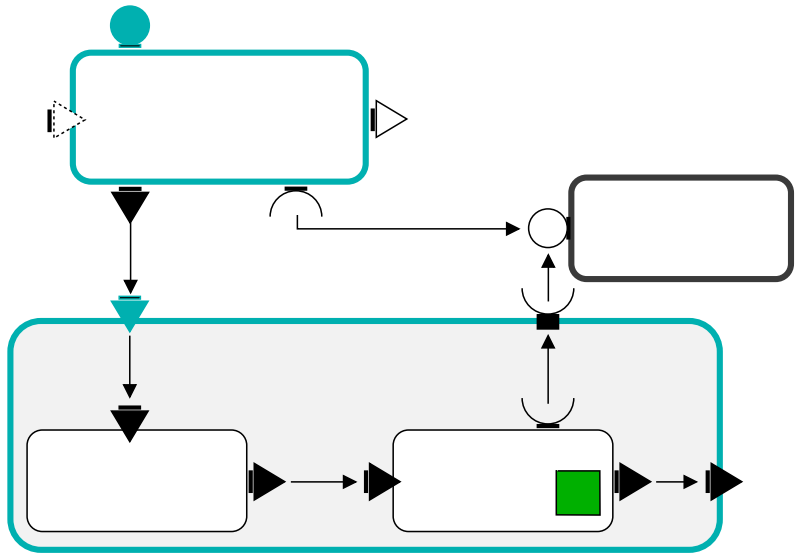


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité

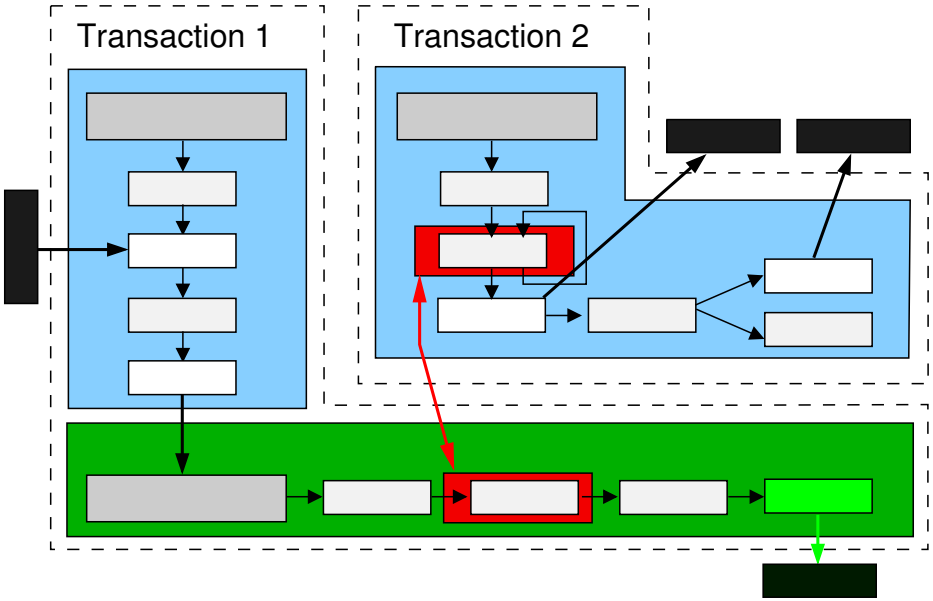


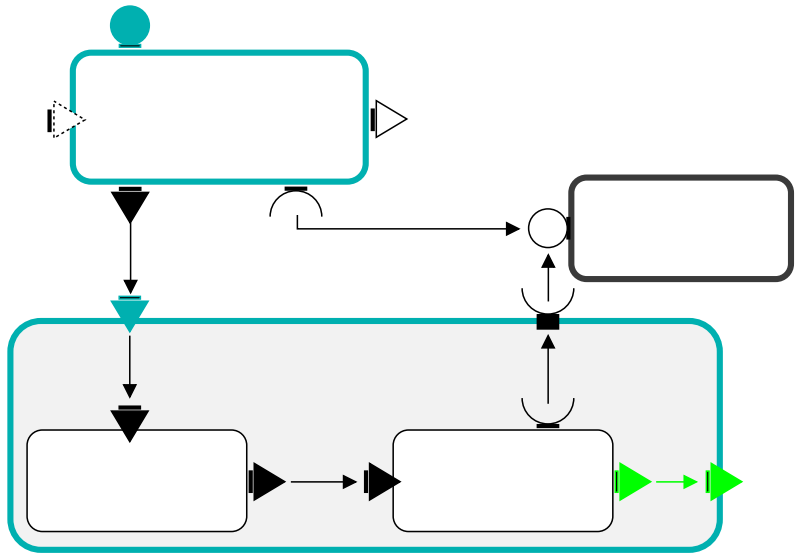


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité

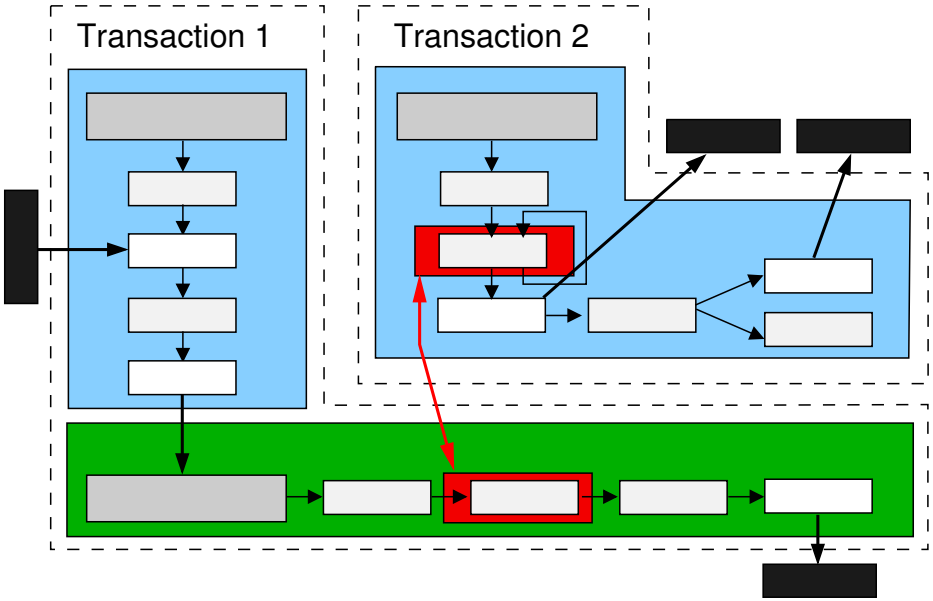


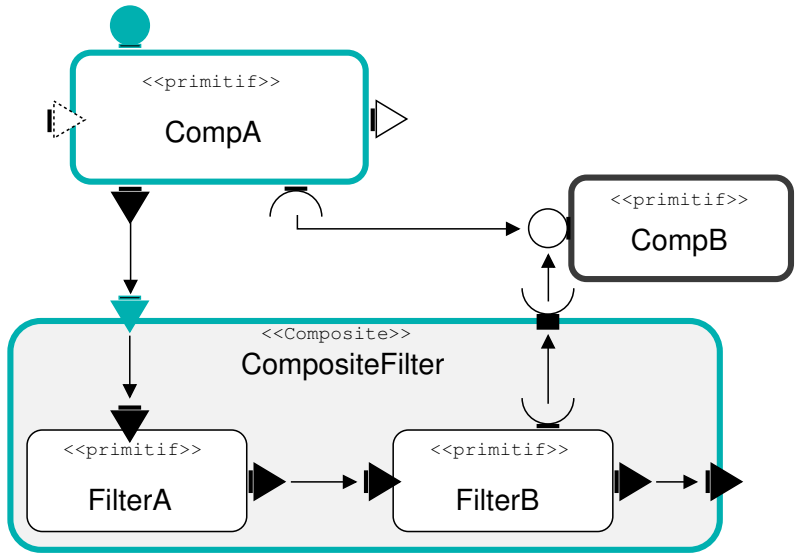


Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Vue activité : traçabilité





Vue activité $\Longleftrightarrow$ Architecture Tinap

- Deux représentations complémentaires de l'application
- Traçabilité assurée entre les deux vues (méta-modélisation)

Bilan sur la « vue activité »

- Unités de réutilisation Tinap : composants couplés à leurs « interfaces riches »
- Construction de la « vue activité » à partir de l'ensemble des informations et abstractions Tinap

Intérêts

- Facilite la compréhension de l'application en cours de conception
- Fournit un support pour définir des contraintes temporelles de bout-en-bout
- Favorise la réutilisabilité
- Réduit l'écart entre espace de conception Tinap et espace de conception centré sur un modèle de tâche
  - Perspective pour un support outillé manipulant ces abstractions

Bilan sur la « vue activité »

- Unités de réutilisation Tinap : composants couplés à leurs « interfaces riches »
- Construction de la « vue activité » à partir de l'ensemble des informations et abstractions Tinap

Intérêts

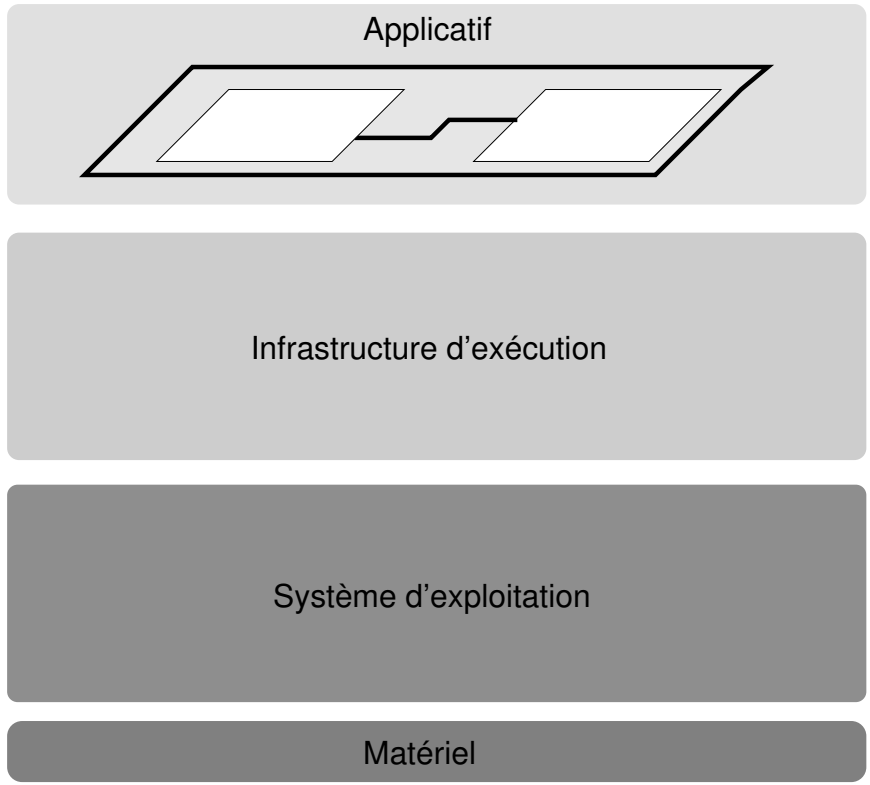
- Facilite la compréhension de l'application en cours de conception
- Fournit un support pour définir des contraintes temporelles de bout-en-bout
- Favorise la réutilisabilité
- Réduit l'écart entre espace de conception Tinap et espace de conception centré sur un modèle de tâche
  - Perspective pour un support outillé manipulant ces abstractions

Plan

Vers une « architecturisation » de l'infrastructure

Constat

- Conception monolithique des couches sous-jacentes
- Dépendances entre « domaines de fonctionnalité » non spécifiées



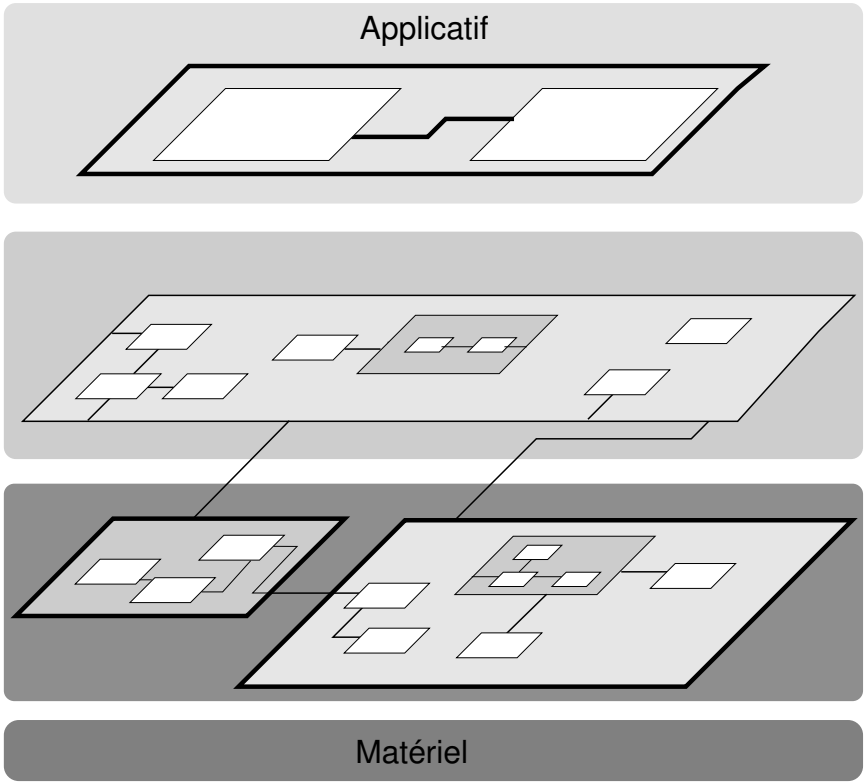
Vers une « architecturisation » de l'infrastructure

Constat

- Conception monolithique des couches sous-jacentes
- Dépendances entre « domaines de fonctionnalité » non spécifiées

Paradigme composant à tous niveaux

- Composabilité inter et intra domaines
- Espace de conception commun



Vers une « architecturisation » de l'infrastructure

Constat

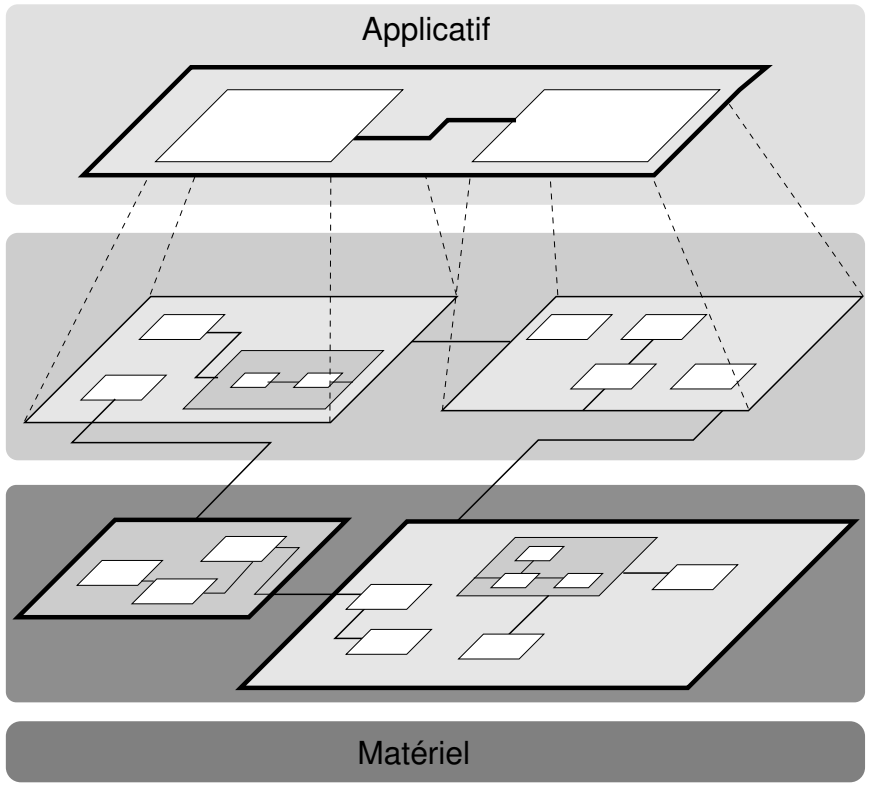
- Conception monolithique des couches sous-jacentes
- Dépendances entre « domaines de fonctionnalité » non spécifiées

Paradigme composant à tous niveaux

- Composabilité inter et intra domaines
- Espace de conception commun

Patron de conception membrane

- Démarche générative
- Exploitation du caractère ouvert de la membrane FRACTAL



Vers une « architecturisation » de l'infrastructure

Constat

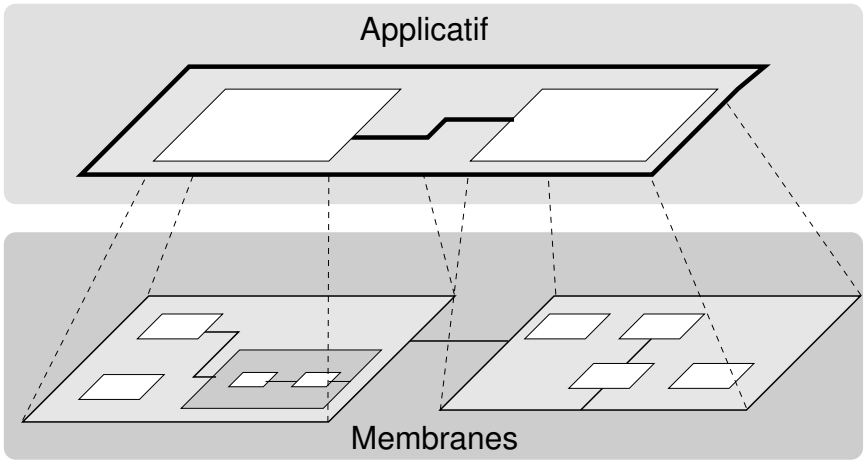
- Conception monolithique des couches sous-jacentes
- Dépendances entre « domaines de fonctionnalité » non spécifiées

Paradigme composant à tous niveaux

- Composabilité inter et intra domaines
- Espace de conception commun

Patron de conception membrane

- Démarche générative
- Exploitation du caractère ouvert de la membrane FRACTAL

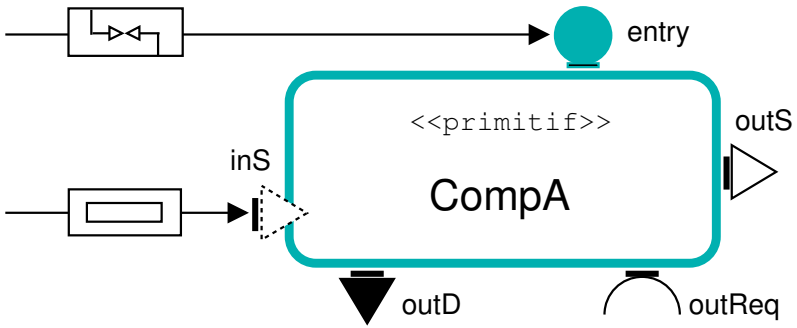


- Fonctionnalités superposées
- Programmable et générée en fonction des besoins intra-applicatifs
- Couche de virtualisation

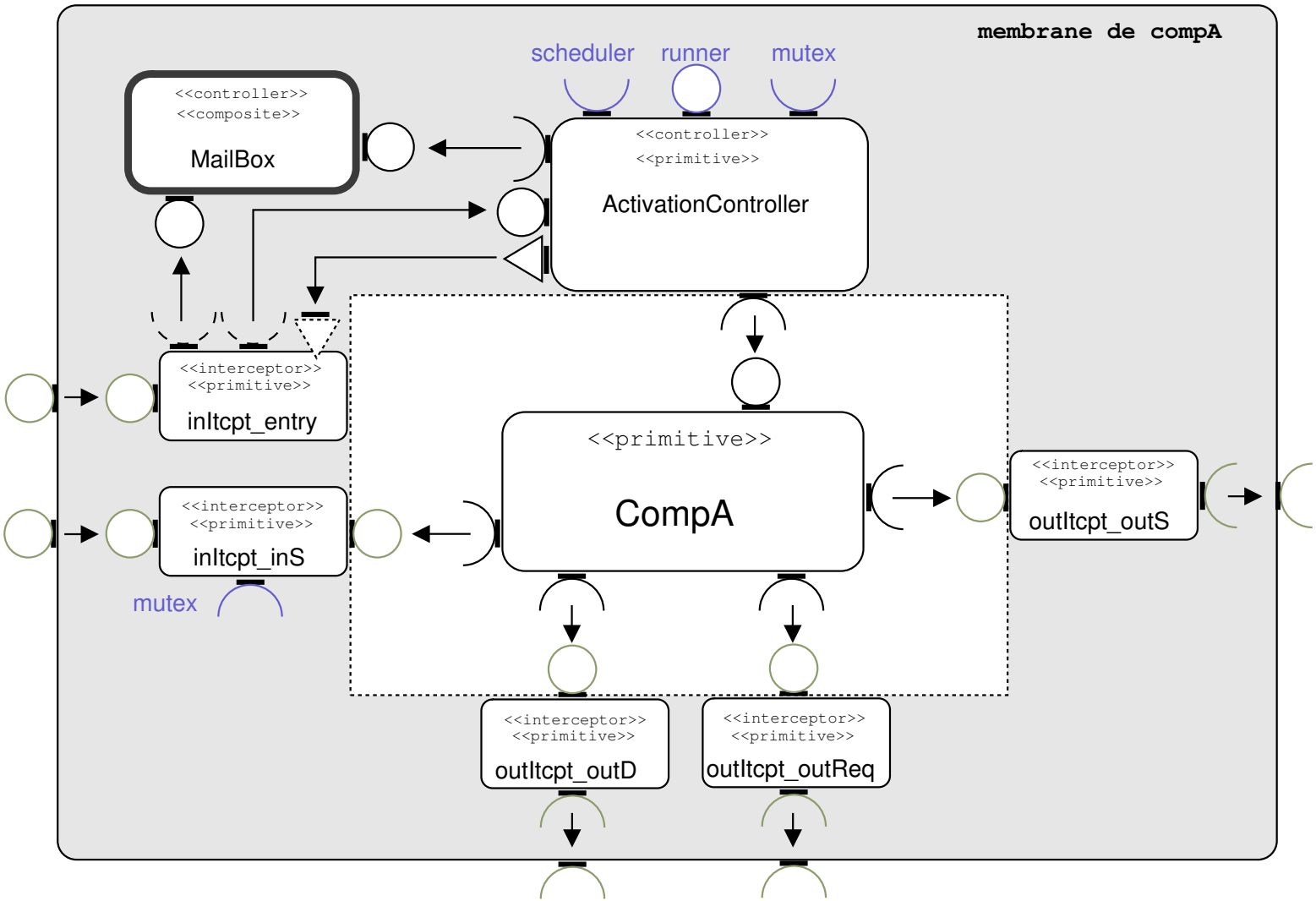
Membranes TINAP

- **Réification des dépendances** entre métier et infrastructure
- **Contrôleurs et intercepteurs** implantent les concepts de l'espace de conception TINAP
  - Logique des communications
    - Gestion des contraintes temporelles
    - Mécanismes de bufferisation / copies
    - Liaisons multiples
  - Logique des descripteurs attachés aux composants
    - Logique des activations des composants actifs
    - Protection des composants protégés
  - L'interfaçage avec les autres domaines de fonctionnalités (OS)

Exemple de membrane TINAP

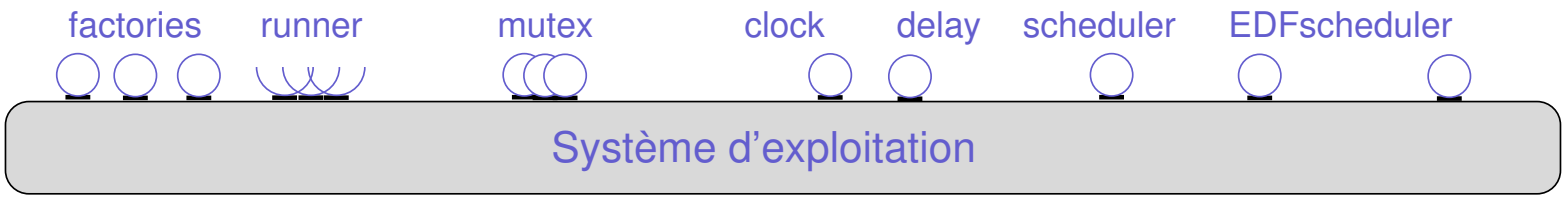


Exemple de membrane TINAP



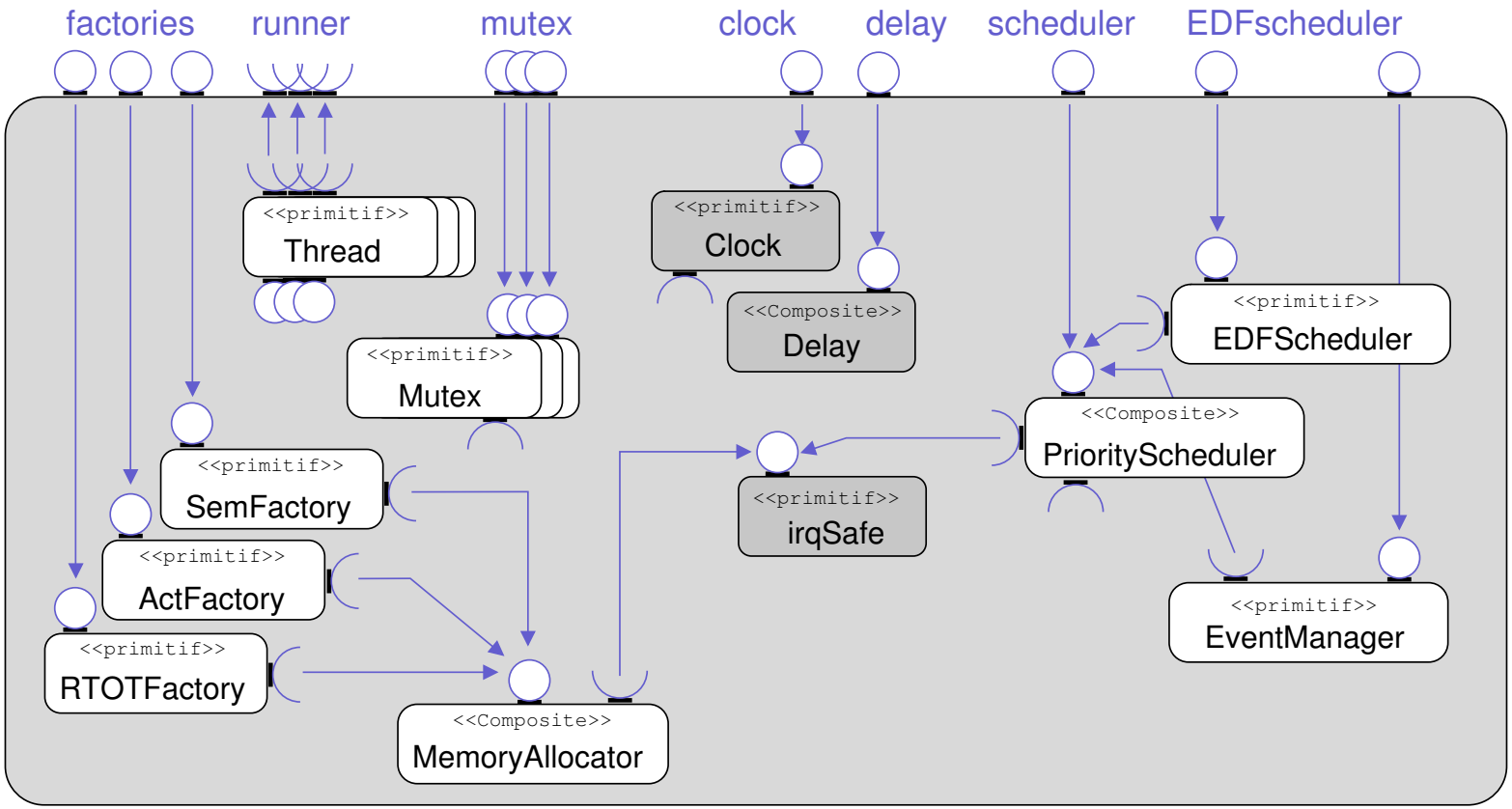
Domaine de fonctionnalités de l'OS

- Services élémentaires requis par l'infrastructure
- Utilisation de THINK/ KORTEX
- Extensions nécessaires à Tinap



Domaine de fonctionnalités de l'OS

- Services élémentaires requis par l'infrastructure
- Utilisation de THINK/ KORTEX
- Extensions nécessaires à Tinap



Bilan sur l'infrastructure

- Une approche générative
  - Pour la mise en œuvre des patrons de conception du domaine
  - Pour contrôler finement le strict nécessaire requis par l'applicatif

Bilan sur l'infrastructure

- Une approche générative
  - Pour la mise en œuvre des patrons de conception du domaine
  - Pour contrôler finement le strict nécessaire requis par l'applicatif
- Exploitation d'un ensemble de concepts communs pour la conception d'un système de bout-en-bout
- Assurer la composabilité verticale (et donc la traçabilité) entre applicatif, infrastructure et système d'exploitation

Bilan sur l'infrastructure

- Une approche générative
  - Pour la mise en œuvre des patrons de conception du domaine
  - Pour contrôler finement le strict nécessaire requis par l'applicatif
- Exploitation d'un ensemble de concepts communs pour la conception d'un système de bout-en-bout
- Assurer la composabilité verticale (et donc la traçabilité) entre applicatif, infrastructure et système d'exploitation
- Composants réifiés à l'exécution : continuum entre espace de conception et infrastructure à l'exécution

Plan

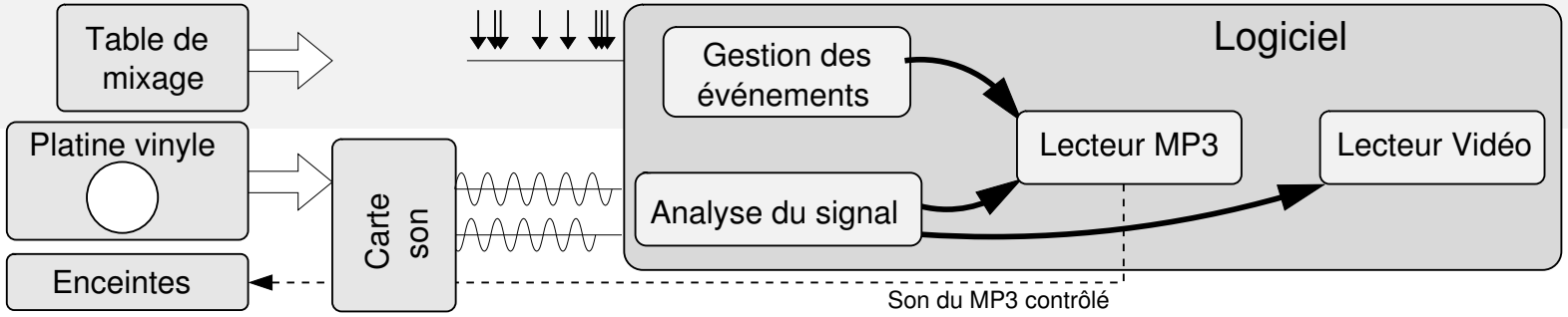
Objectifs

DeckX : contrôleur de flux multimédias

- Application essentiellement orientée flot de donnée et de contrôle
- Echanges de flux audio entre composants
- **Expérimentation des concepts applicatifs de Tinap**

TINAPisation Accord (CEA LISE)

- Réingénierie d'un modèle d'exécution
- **Expérimentation du patron de conception membrane**

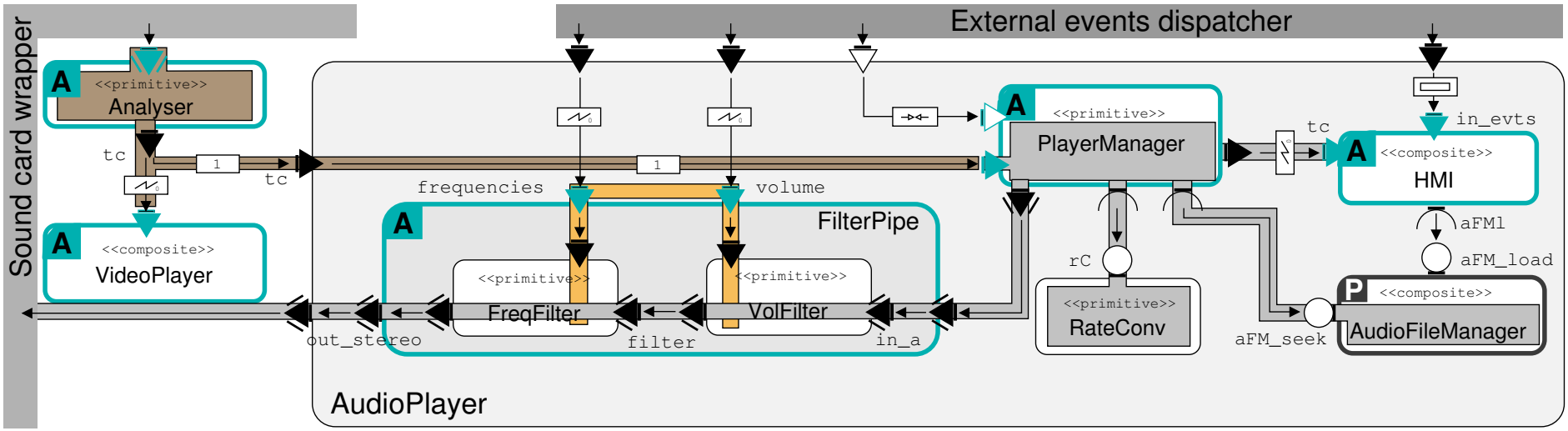


- Utilisation d'un disque vinyle comme contrôleur
- Un signal modulé encodant un *timecode* est gravé sur le disque
  - Sens de lecture, vitesse, position absolue du bras
- Le *timecode* est extrait par le logiciel
- Puis exploité pour contrôler un lecteur audio, vidéo

Vidéo de démo

Conception Tinap de DeckX

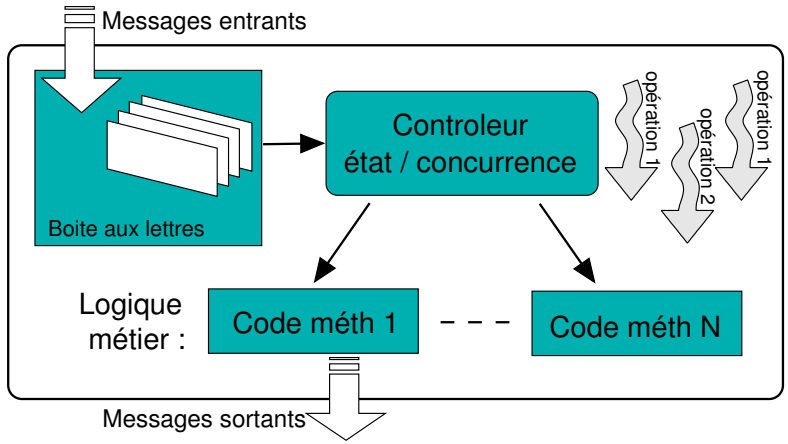
- Exploitation des descripteurs de contrôle définis
- Extension spécifique au domaine : **interfaces de flux audio**
  - **Typées** pour caractériser le flux
  - Interactions générées
- Caractéristiques des applications ciblées par notre approche
  - Prépondérance fonctionnelle, multi-tâches
  - Aspects de contrôle / Contraintes temporelles



Modèle d'exécution ACCORD

- Une plate-forme de modélisation et d'exécution d'application TR (CEA-LIST/LISE)

- Conçue par un **ensemble d'Objets Temps Réel (RTO)**
  - Un moniteur multi-tâches pour exécuter les services fournis
  - Activation par passage de messages ou signaux
  - Allocation : un thread par invocation
  - Contraintes sur les exécutions des opérations
    - Annotations temporelles
    - Contraintes d'états
    - Contraintes d'accès

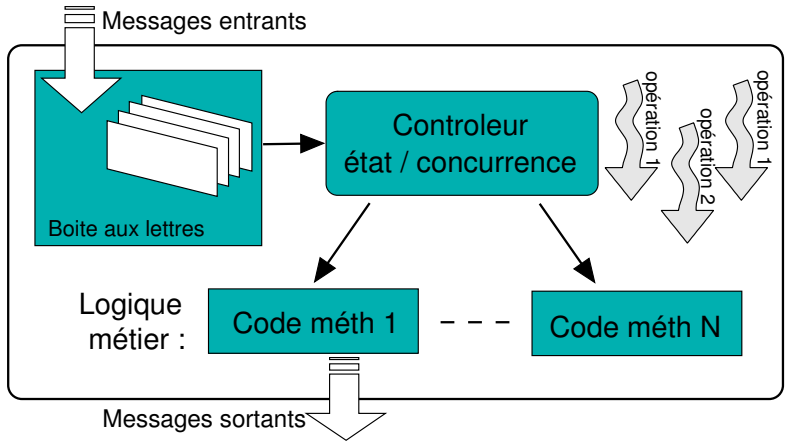


Modèle d'exécution ACCORD

- Une plate-forme de modélisation et d'exécution d'application TR (CEA-LIST/LISE)
- Conçue par un **ensemble d'Objets Temps Réel (RTO)**
  - Un moniteur multi-tâches pour exécuter les services fournis
  - Activation par passage de messages ou signaux
  - Allocation : un thread par invocation
  - Contraintes sur les exécutions des opérations
    - Annotations temporelles
    - Contraintes d'états
    - Contraintes d'accès

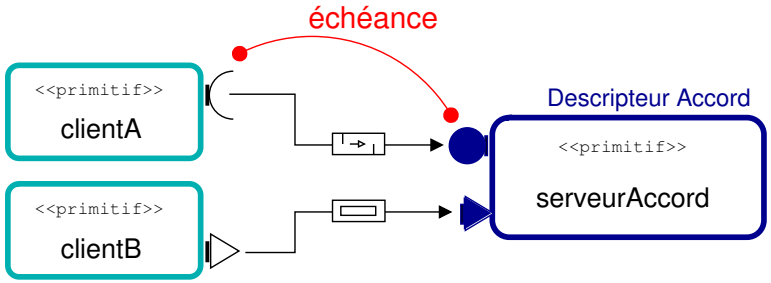
Ingénierie de Accord au sein de TINAP

- Descripteur de contrôle ACCORD
- Modèle d'exécution implanté au sein des membranes



TINAPisation de ACCORD

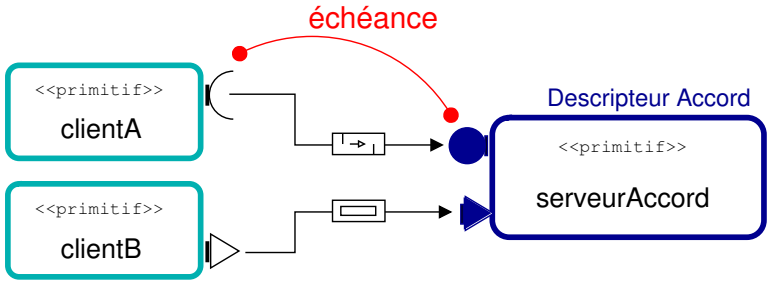
Niveau applicatif



Descriteur de contrôle ACCORD s'appliquant aux composants

TINAPisation de ACCORD

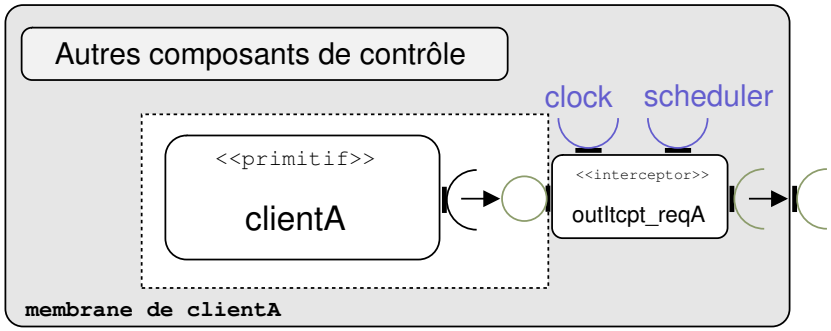
Niveau applicatif



Niveau infrastructure

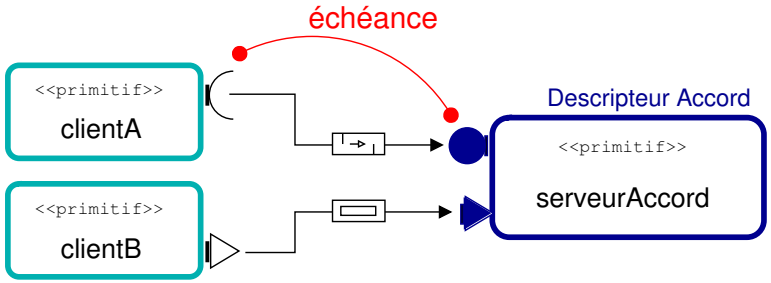
Interception côté client

Initialisation des messages à échéances



TINAPisation de ACCORD

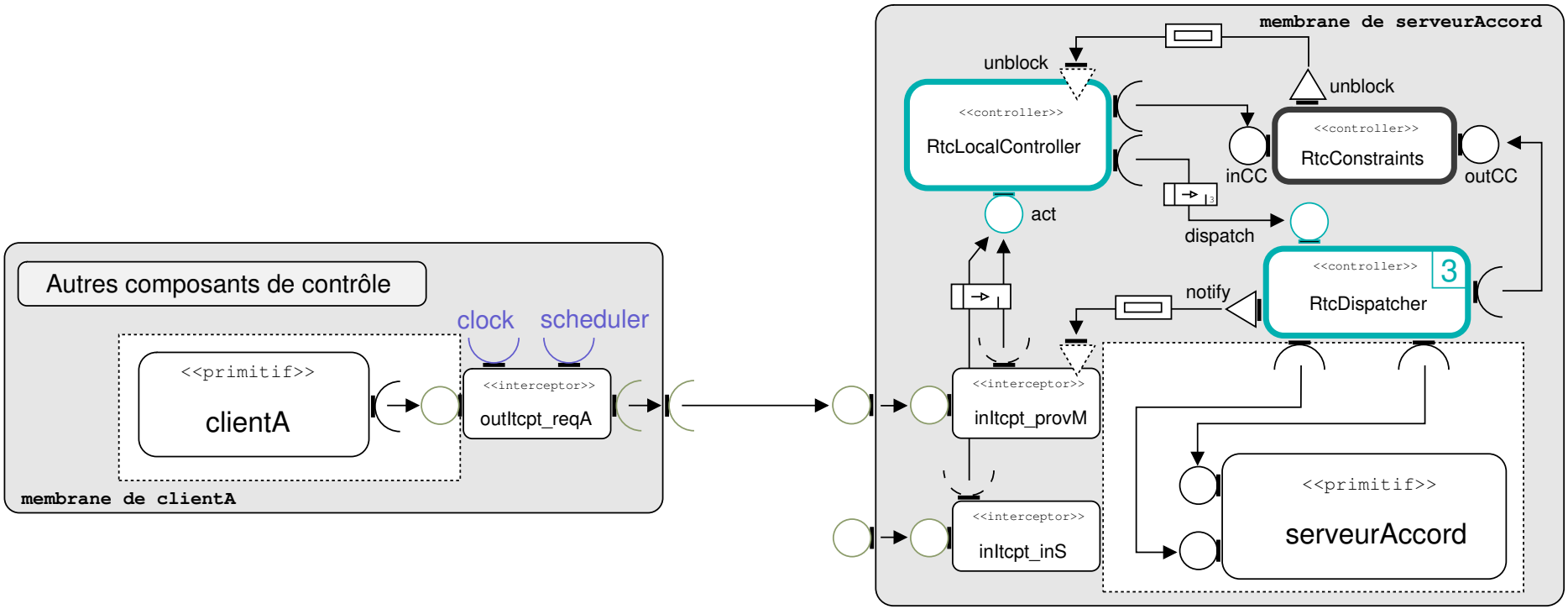
Niveau applicatif



Infrastructure côté serveur

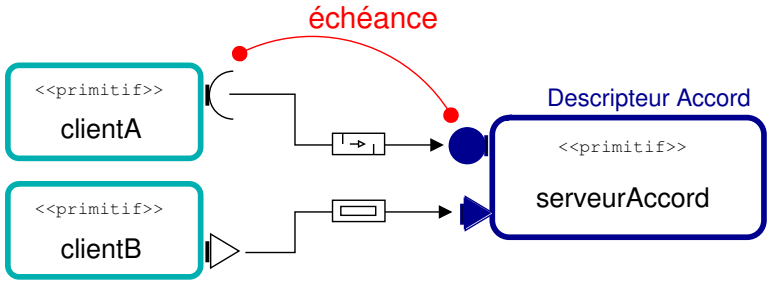
Mise en œuvre du modèle d'exécution ACCORD

Niveau infrastructure



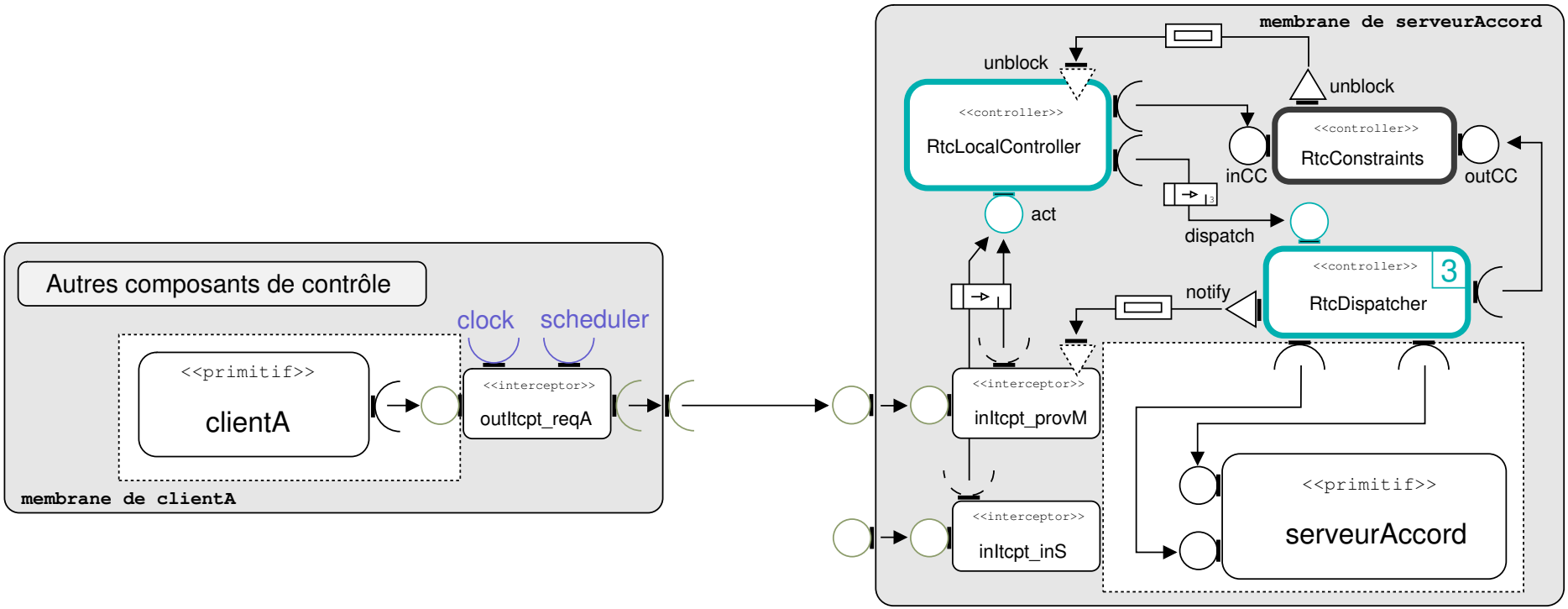
TINAPisation de ACCORD

Niveau applicatif



Exécution sur PowerPC

Niveau infrastructure



Évaluations

- Meilleure compréhension de l'applicatif
  - Externalisation des préoccupations du domaine
  - Vue activité
- Implantations centrées sur le fonctionnel
  - En C, dans « l'espace langage de programmation »
- Bonne couverture des abstractions Tinap pour la conception d'applications multi-tâches à contraintes temps réel souples (études de cas représentatives)
- Preuve de faisabilité pour la conception de modèles d'exécution non-triviaux

Plan

Conclusions

- La spécification d'un **espace de conception orientée composant**
  - Capturant les préoccupations du domaine ciblé
  - Proposant les abstractions nécessaires pour caractériser finement le comportement
  - Utilisable à tous niveaux du système

Conclusions

- La spécification d'un **espace de conception orientée composant**
  - Capturant les préoccupations du domaine ciblé
  - Proposant les abstractions nécessaires pour caractériser finement le comportement
  - Utilisable à tous niveaux du système
- **L'exploitation des abstractions Tinap** pour représenter une « vue orientée activité », proche des abstractions usuelles du domaine du temps réel

Conclusions

- La spécification d'un **espace de conception orientée composant**
  - Capturant les préoccupations du domaine ciblé
  - Proposant les abstractions nécessaires pour caractériser finement le comportement
  - Utilisable à tous niveaux du système
- **L'exploitation des abstractions Tinap** pour représenter une « vue orientée activité », proche des abstractions usuelles du domaine du temps réel
- **L'expérimentation du paradigme composant à tous niveaux**
  - Offre un cadre conceptuel unifié (*structure* / *comportement* / *abstractions*)
  - Exploite la composition verticale
  - Assure une traçabilité accrue entre niveaux d'abstraction
  - Permet de générer l'infrastructure nécessaire
  - Assure un contrôle fin sur les fonctionnalités à embarquer

Conclusions

- La spécification d'un **espace de conception orientée composant**
  - Capturant les préoccupations du domaine ciblé
  - Proposant les abstractions nécessaires pour caractériser finement le comportement
  - Utilisable à tous niveaux du système
- **L'exploitation des abstractions Tinap** pour représenter une « vue orientée activité », proche des abstractions usuelles du domaine du temps réel
- **L'expérimentation du paradigme composant à tous niveaux**
  - Offre un cadre conceptuel unifié (*structure* / *comportement* / *abstractions*)
  - Exploite la composition verticale
  - Assure une traçabilité accrue entre niveaux d'abstraction
  - Permet de générer l'infrastructure nécessaire
  - Assure un contrôle fin sur les fonctionnalités à embarquer
- Vers le support de nouveaux outils, **continuum assuré par Tinap** entre :
  - AST C $\longrightarrow$ Vue comportement $\longrightarrow$ Vue structurelle $\longrightarrow$ Vue activité

Perspectives à court terme

- **Paradigme de conception membrane**
  - Spécifier le processus de génération des membranes de manière générale
  - Interfaçage avec la chaîne d'outils sous-jacente

Perspectives à court terme

- **Paradigme de conception membrane**
  - Spécifier le processus de génération des membranes de manière générale
  - Interfaçage avec la chaîne d'outils sous-jacente
- **Componentisation d'un OS temps réel**
  - Componentisation des structures de données très entrelacées
  - Annuler le coût de la componentisation
    - Tester les optimisations mises en œuvre dans THINK
    - Nouvelles optimisations : fusion de code

Perspectives à court terme

- **Paradigme de conception membrane**
  - Spécifier le processus de génération des membranes de manière générale
  - Interfaçage avec la chaîne d'outils sous-jacente
- **Componentisation d'un OS temps réel**
  - Componentisation des structures de données très entrelacées
  - Annuler le coût de la componentisation
    - Tester les optimisations mises en œuvre dans THINK
    - Nouvelles optimisations : fusion de code
- **Analyses non-fonctionnelles**
  - Assurer le pont entre « vue activité » Tinap et outils d'analyse (rapprochement avec MARTE)
  - Caractériser le sous-ensemble de l'espace de conception Tinap compatible avec les formalismes d'entrée

Perspectives à plus long terme

- Support de la reconfigurabilité des concepts Tinap de haut-niveau
- Concepts à concrétiser à l'exécution ?
- Support de différents MoCC

La fin.