Utilisation d'un Langage Formel Pour UML-RT: une Meta-Architecture

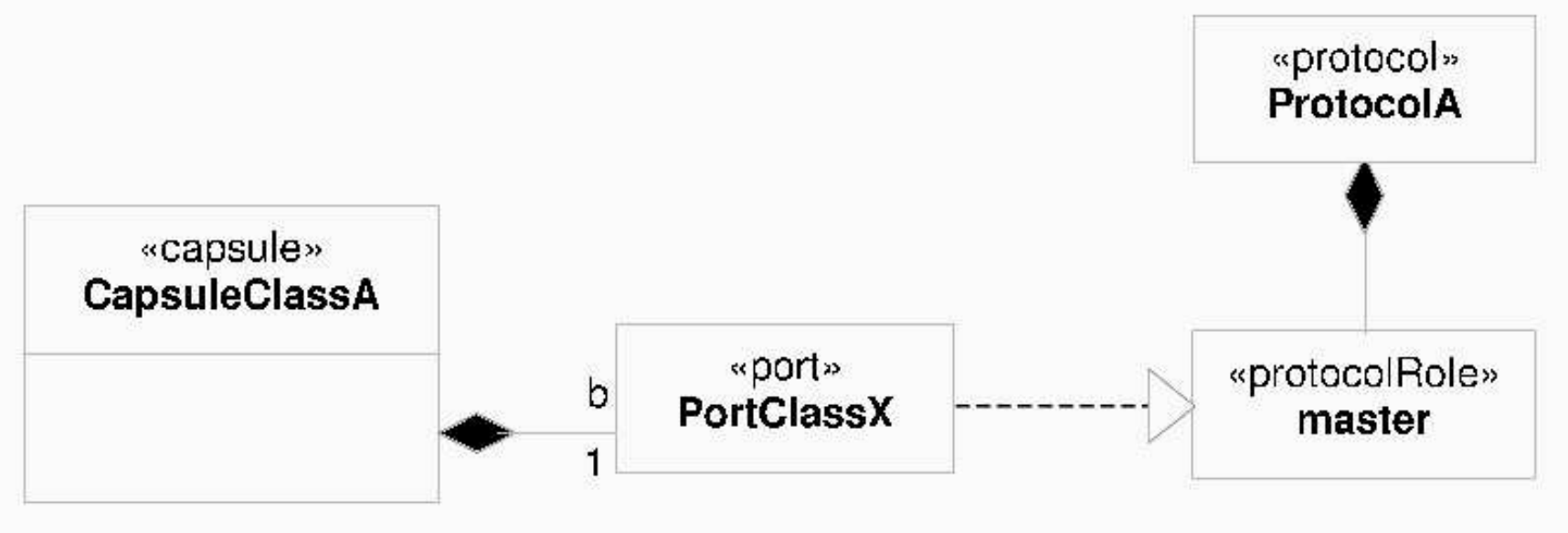
Arnaud Bailly **ENST**

18 Janvier 2001

Plan

1. Présentation de UML-RT. L'apport Proposé.
2. Une Meta-Architecture UML Pour *ArtOC*.
3. Spécification et Vérification avec *ArtOC*.
4. Avons-nous Résolu le Problème ?
5. Conclusion.

ROOM et UML-RT

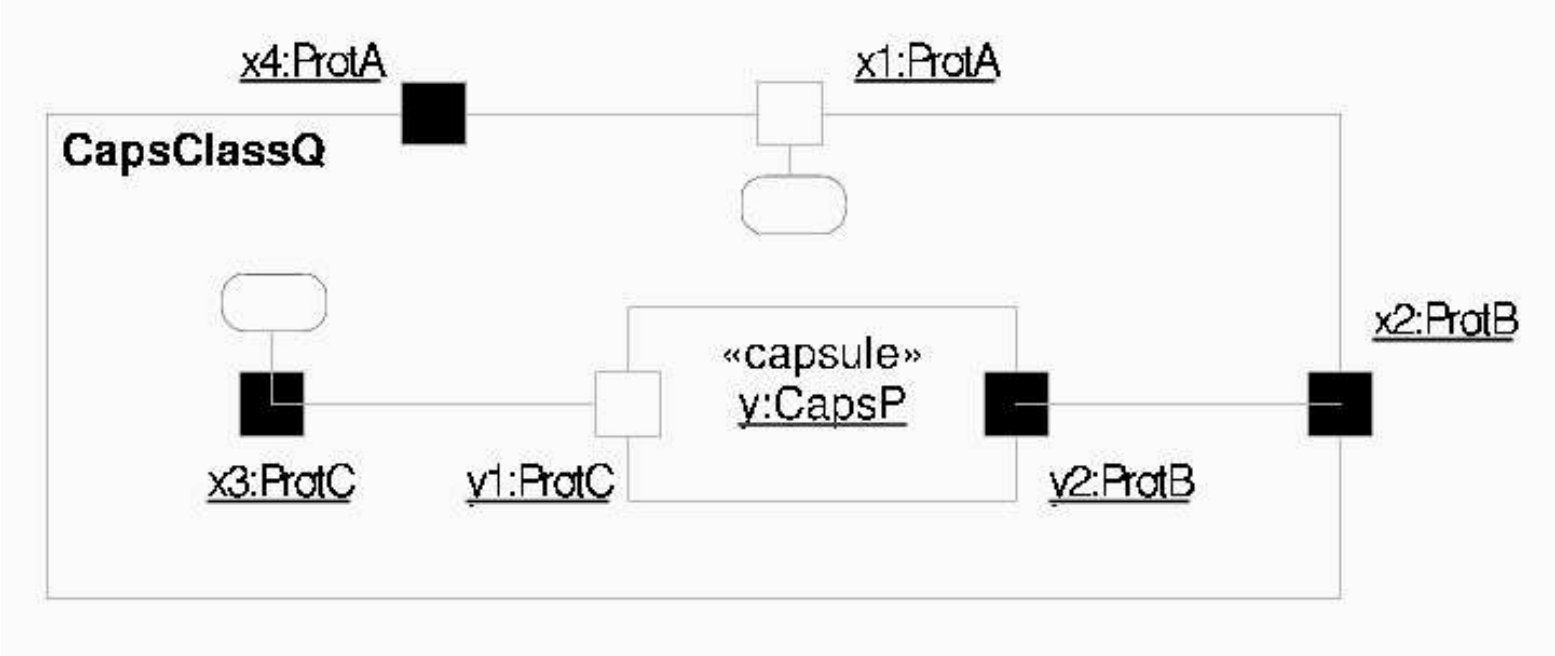


Un diagramme de classe.

Les entités principales sont:

- les capsules,
- les ports et protocoles,
- les connecteurs.

ROOM et UML-RT (2)



Un diagramme de collaboration.

- Les connecteurs sont les liens entre les ports.
- Ils sont complètement passifs,
- mais ils peuvent recevoir un comportement en utilisant des classes-associations.

L'apport Proposé

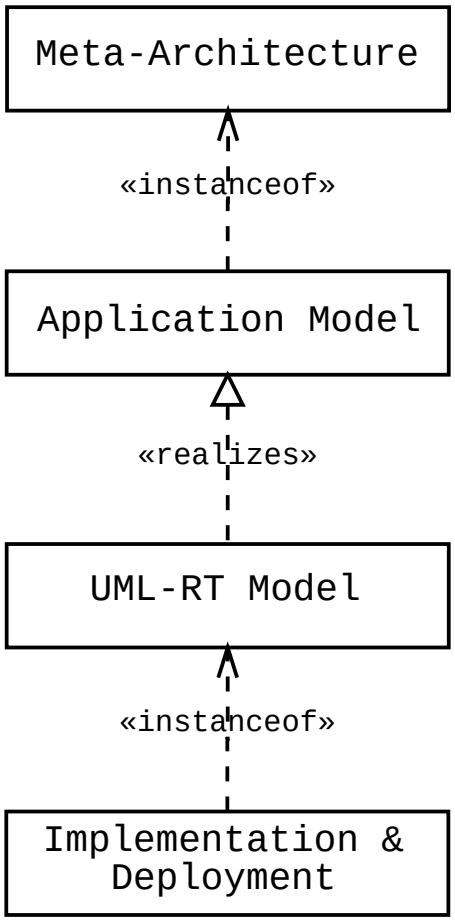
> [...] connectors [...] interconnect ports that play complementary roles in the protocol associated with the connector. [...] the protocol roles [...] have to be compatible with the protocol of the connector[1]. [...]
>
> [1] We will not discuss the rules of protocol compatibility here except that it is based on behavioral sufficiency.

- Nous avons résolu ce problème pour une certaine classe de connecteurs !

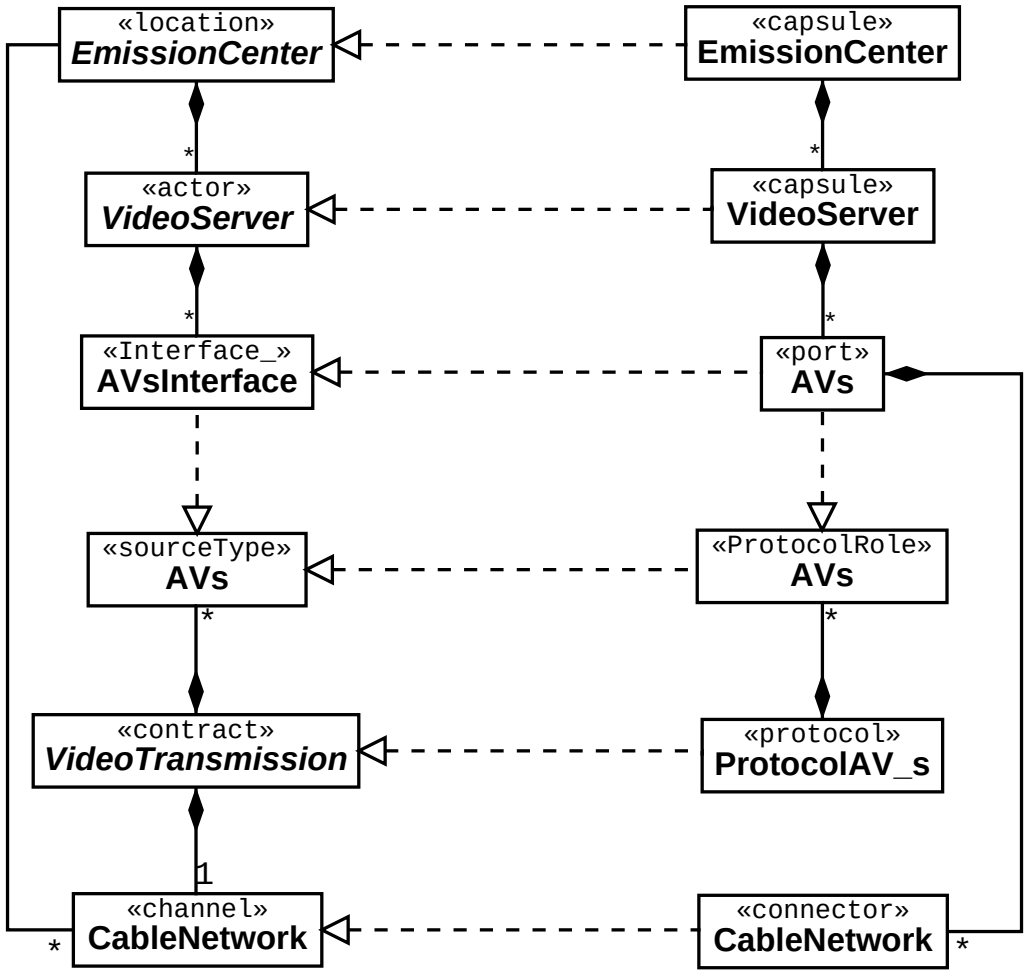
Plan

1. Présentation de UML-RT. L'apport Proposé.
2. Une Meta-Architecture UML Pour *ArtOC*.
3. Spécification et Vérification avec *ArtOC*.
4. Avons-nous Résolu le Problème ?
5. Conclusion.

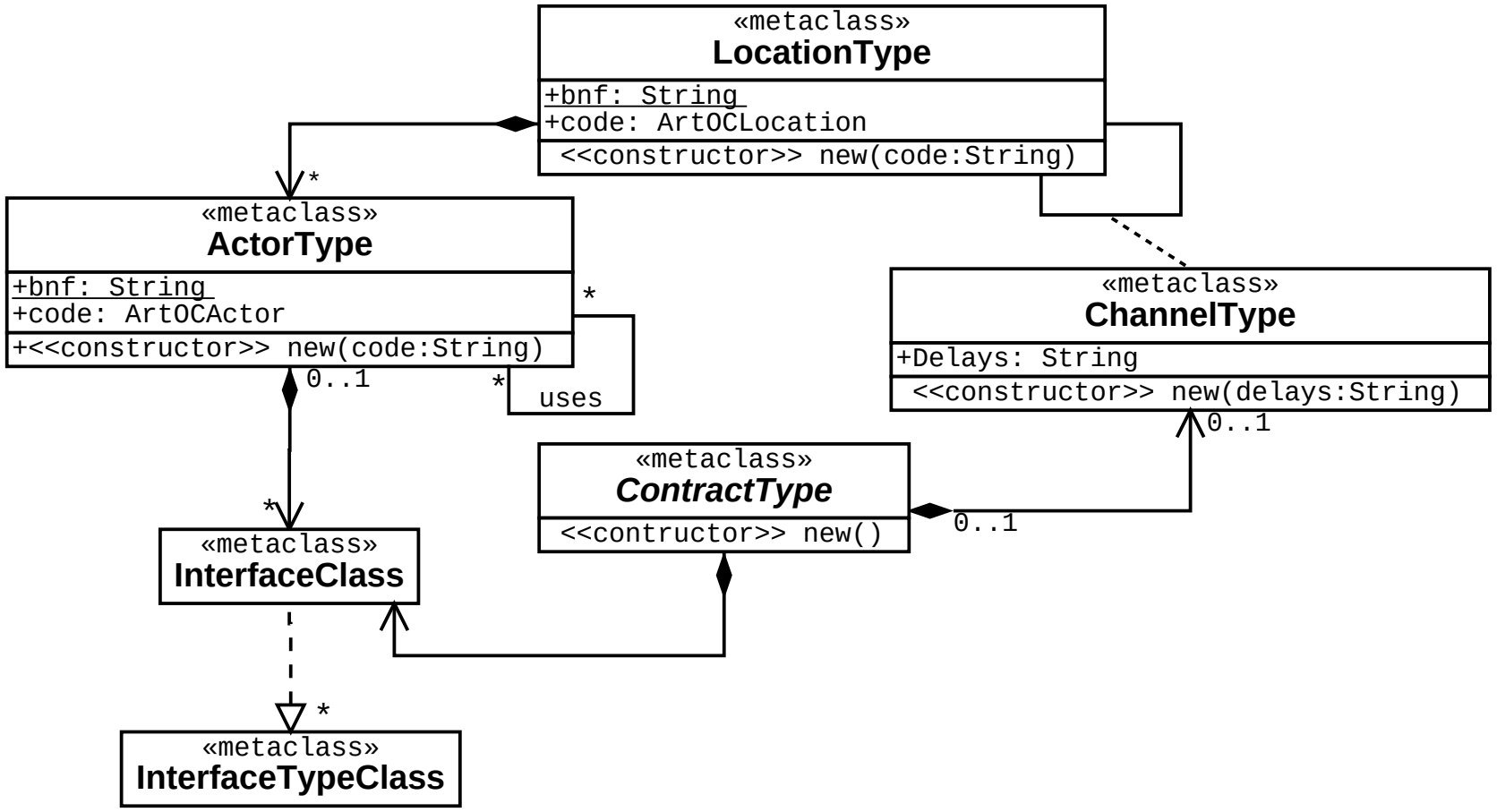
La «Pile des Modèles»



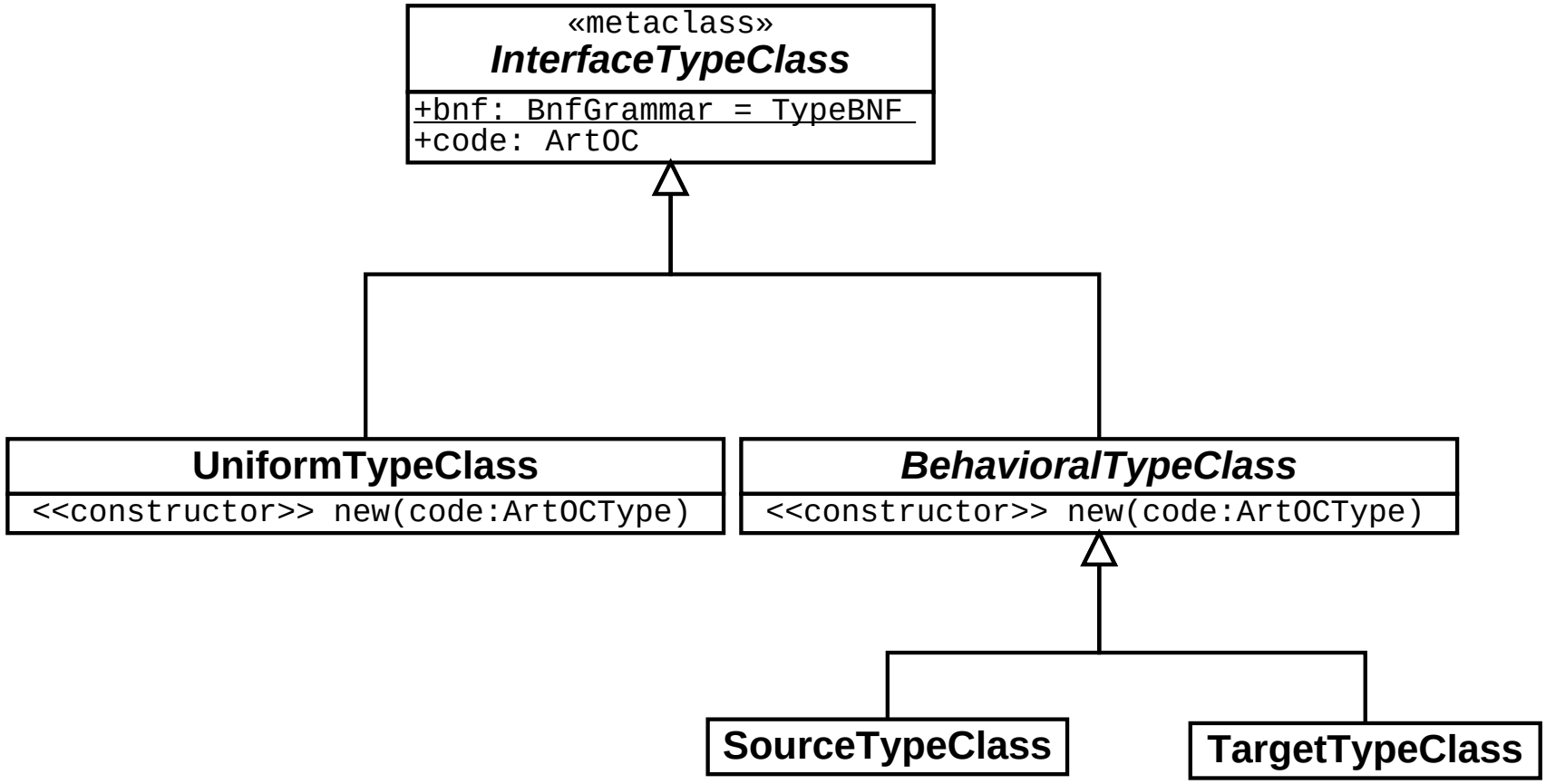
Isomorphie des Concepts et Réalisation



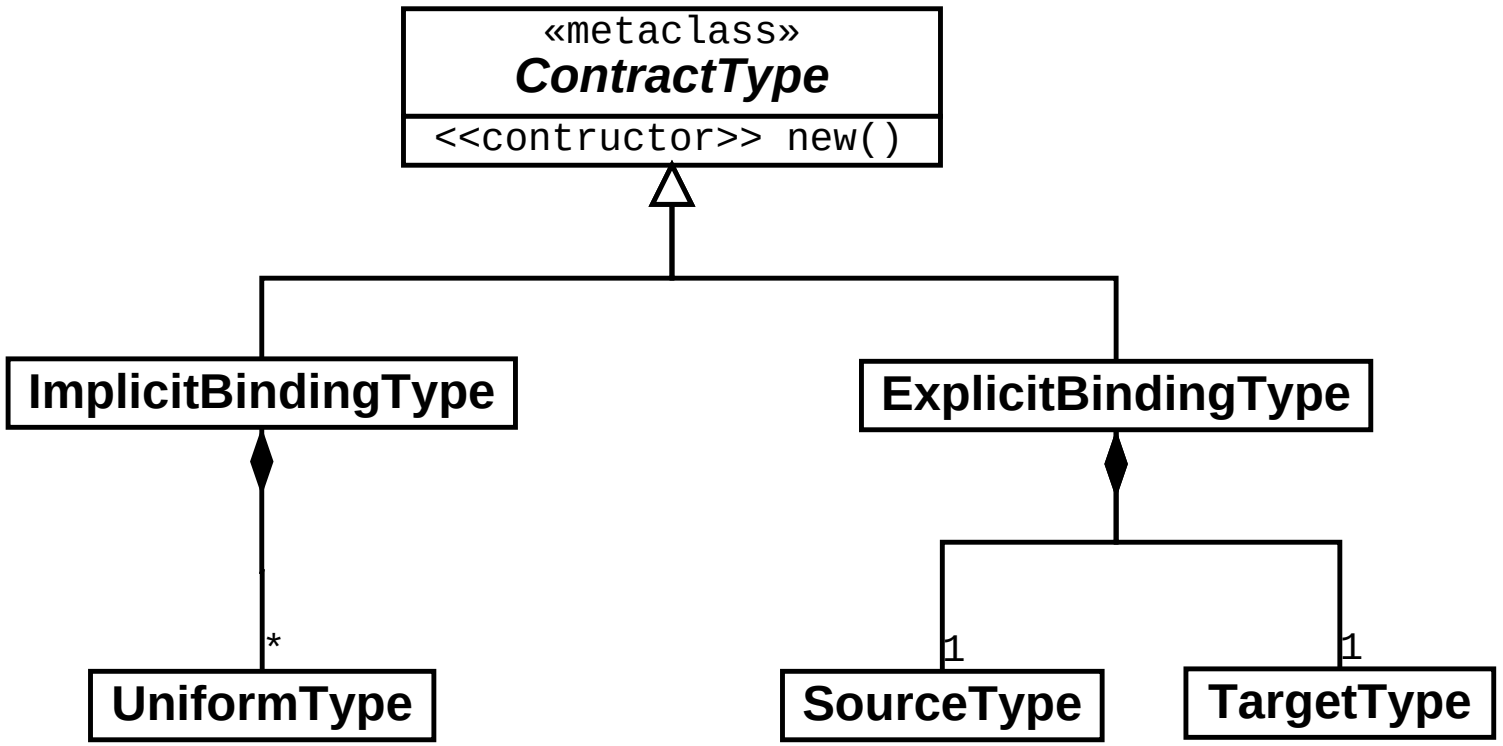
Une Meta-Architecture



La Hiérarchie des Types

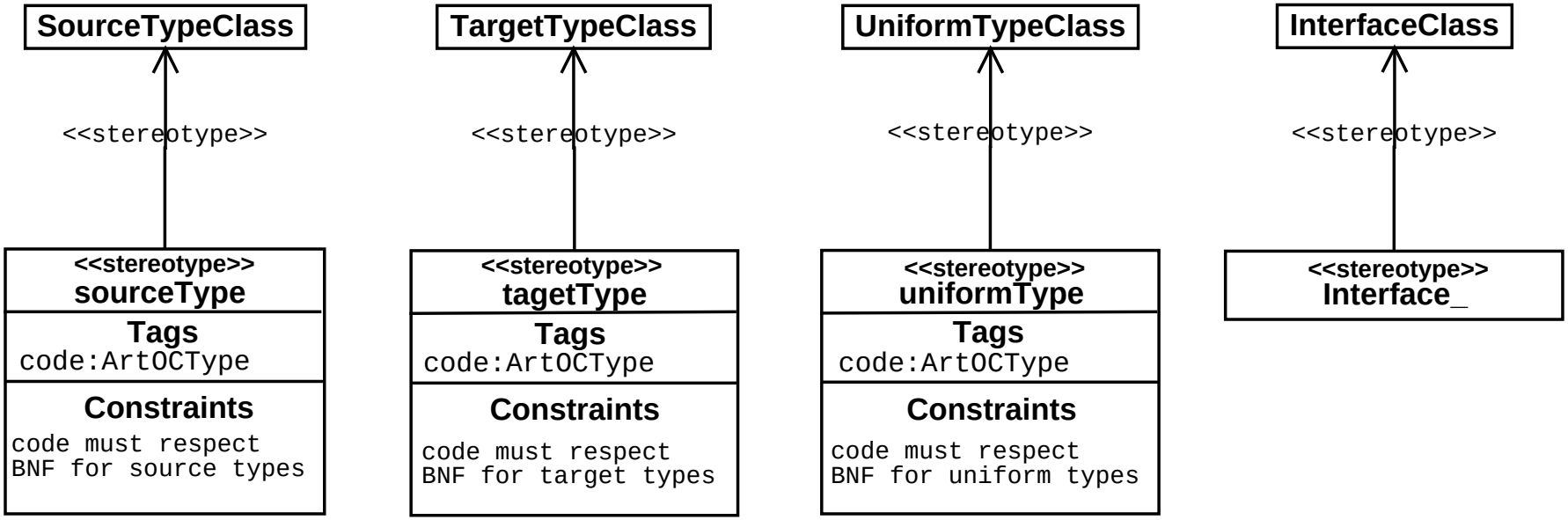


La Hiérarchie des Contrats

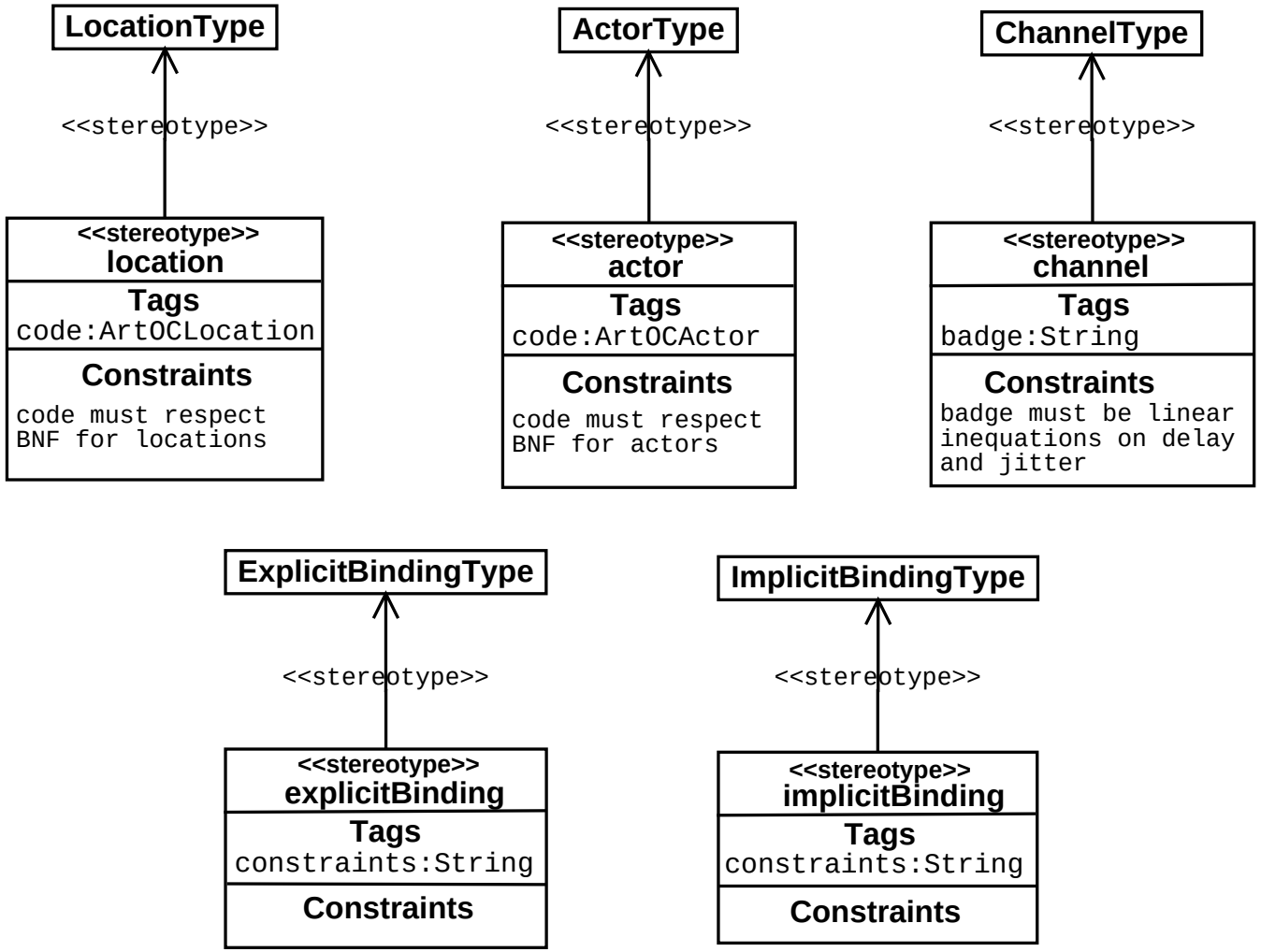


Contrainte OCL: une liaison implicite ne peut recevoir qu'un seul serveur.

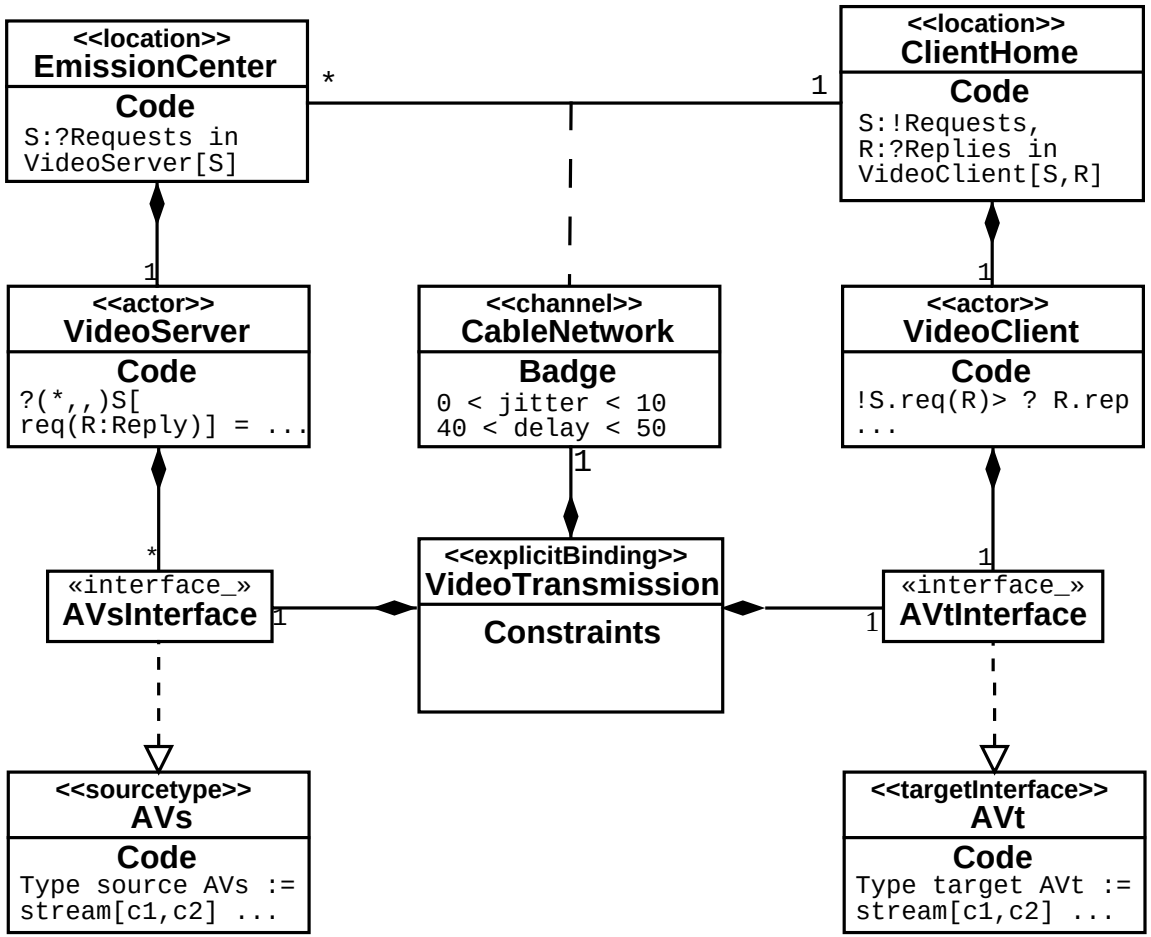
Stéréotypons ... (1)



Stéréotypons ... (2)



Un Exemple: «Video à la Demande»



Plan

1. Présentation de UML-RT. L'apport Proposé.
2. Une Meta-Architecture UML Pour *ArtOC*.
3. Spécification et Vérification avec *ArtOC*.
4. Avons-nous Résolu le Problème ?
5. Conclusion.

Objectifs

- Écriture aisée de spécifications comportementales avec contraintes de QoS reconfigurables.
- Vérification compositionnelle de:
  - la sûreté (safety),
  - la vivacité (liveness).
- Configurations extensibles dynamiquement (*ouvertes*).

On Résoud le Problème Suivant

- Deux spécifications temporisées d'entités de protocoles A et B.
- Canal de communication σ entre A et B:
  - temps de transmission T borné tel que les bornes sont *connues*,
  - file FIFO infinie sans pertes.
- On vérifie que tout signal émis par A sera reçu par B à son arrivée, et que tout signal émis par B sera reçu par A à son arrivée.

Méthode

On spécifie:

- Spécifications Comportementales = Acteurs Temporisés.
- Un acteur communique à travers des *interfaces comportementales temporisées*:
  - Les actions (envois/réceptions) qu'elles autorisent changent avec les actions déjà effectuées.
  - Les actions qu'elles autorisent changent avec l'écoulement du temps.

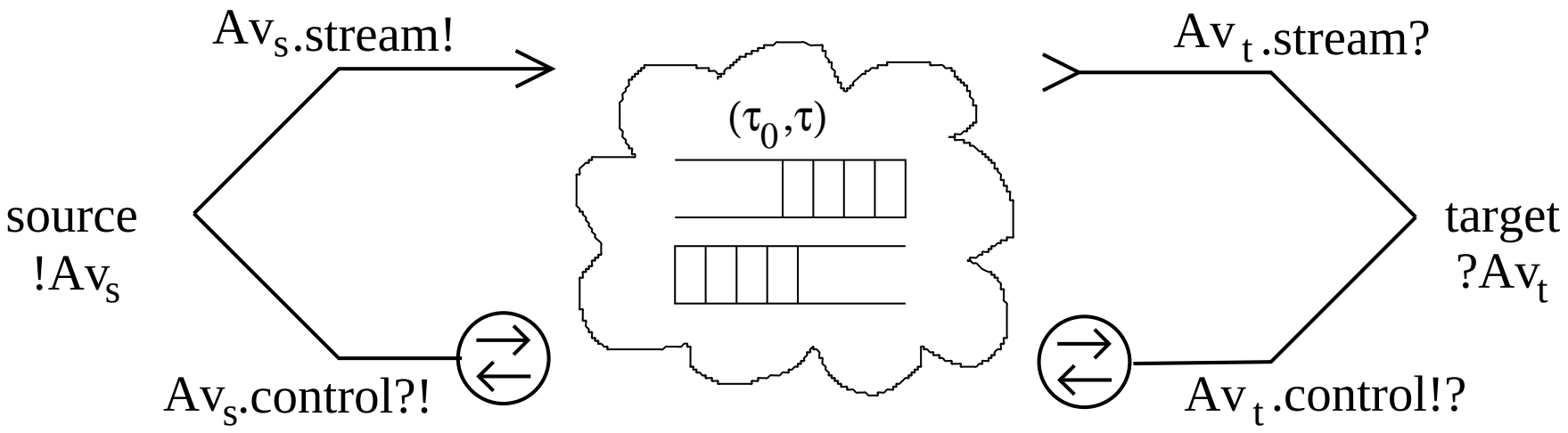
On vérifie:

- Les interfaces en vis-à-vis doivent être compatibles.
- Le *type comportemental* d'un acteur doit respecter les contraintes de *toutes* les interfaces qu'il possède.

Une Interface Comportementale

- Possède deux parties:
  - Une partie de flux (*stream*) unidirectionnelle,
  - Une partie de contrôle à sens variable.
- Possède un *type* qui décrit l'évolution de l'interface au cours du temps, et un rôle: *source* ou *cible*.
- Un type indique:
  - Les horloges que possède l'interface,
  - À un instant donné, les actions qui sont possibles via l'interface,
  - Quelle sera l'évolution de l'interface lors du déclenchement d'une action.

Exemple de Deux Interfaces



- La partie flot évolue indépendamment de la partie contrôle.
- La partie contrôle peut faire évoluer la partie flôt de manière déterministe.

Un Exemple de Type d'Interface: $\mathbf{AV_t}$

- Type de cible souhaitant recevoir de la vidéo et de l'audio synchronisés.
- La cible a 5 secondes pour demander le départ de la vidéo, puis doit attendre 5 secondes avant d'en demander l'arrêt. L'arrêt est effectif avant 1 seconde.

$$
\begin{array}{lll}
\text{Type target } AV_t(\tau_0, \tau) := & & \\
\quad \text{stream}[c_1, c_2] : & x_1 & = \emptyset \\
& x_2 & = c_1 \text{ in } [40 - 2\tau, 40]\ video() \text{ until } 40 \ ; \text{stream}.x_2(c_1 = 0) \\
& & + c_2 \text{ in } [100 - 2\tau, 100]\ audio() \text{ until } 100 \ ; \text{stream}.x_2(c_2 = 0) \\
\quad \text{control}[c_3] : & x_3 & = c_3 \text{ in } [0, 5000]\ !start(); \text{control}.x_4(c_3 = 0) \\
& x_4 & = c_3 \text{ in } [0, 5000]\ ?start\_ack(); (\text{stream}.x_2, \text{control}.x_5)(c_3 = 0) \\
& x_5 & = c_3 \text{ in } [5000, \infty]\ !stop(); \text{control}.x_6(c_3 = 0) \\
& x_6 & = c_3 \text{ in } [0, 1000]\ ?stop\_ack(); (\text{stream}.x_1, \text{control}.x_7)(c_3 = 0) \\
& x_7 & = \emptyset
\end{array}
$$

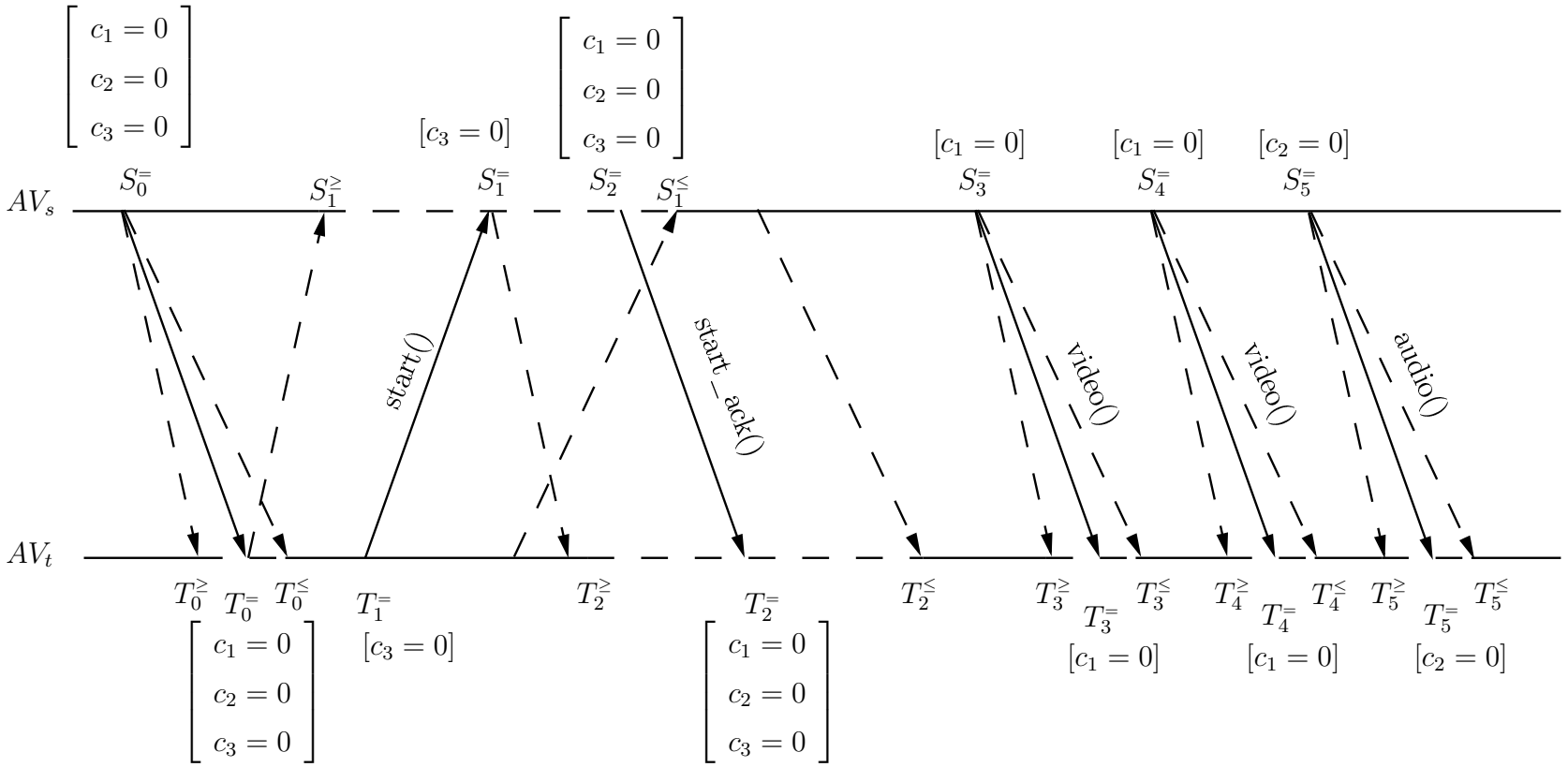
Un Type AV_s Compatible avec AV_t

$$
\begin{array}{lll}
\text{Type source } AV_s := & & \\
\text{stream}[c_1, c_2]: & x_1 & = \emptyset \\
& x_2 & = c_1 \text{ in } [40-\tau, 40-\tau]\ video() \text{ until } 40-\tau \ ; \text{stream}.x_3(c_1 = 0) \\
& & + c_2 \text{ in } [100-\tau, 100-\tau]\ audio() \text{ until } 100-\tau \ ; \text{stream}.x_4(c_2 = 0) \\
& x_3 & = c_1 \text{ in } [40, 40]\ video() \text{ until } 40 \ ; \text{stream}.x_3(c_1 = 0) \\
& & + c_2 \text{ in } [100-\tau, 100-\tau]\ audio() \text{ until } 100-\tau \ ; \text{stream}.x_5(c_2 = 0) \\
& x_4 & = c_1 \text{ in } [40-\tau, 40-\tau]\ video() \text{ until } 40-\tau \ ; \text{stream}.x_5(c_1 = 0) \\
& & + c_2 \text{ in } [100, 100]\ audio() \text{ until } 100 \ ; \text{stream}.x_4(c_2 = 0) \\
& x_5 & = c_1 \text{ in } [40, 40]\ video() \text{ until } 40 \ ; \text{stream}.x_5(c_1 = 0) \\
& & + c_2 \text{ in } [100, 100]\ audio() \text{ until } 100 \ ; \text{stream}.x_5(c_2 = 0) \\
\text{control}[c_3]: & x_6 & = c_3 \text{ in } [2\tau_0, 5000 + 2(\tau_0 + \tau)]\ ?start(); \text{control}.x_7(c_3 = 0) \\
& x_7 & = c_3 \text{ in } [0, 5000 - 2(\tau_0 + \tau)]\ !start\_ack(); (\text{stream}.x_2, \text{control}.x_8) \\
& & \ (c_3 = 0) \\
& x_8 & = c_3 \text{ in } [5000, \infty]\ ?stop(); \text{control}.x_9(c_3 = 0) \\
& x_9 & = c_3 \text{ in } [2\tau_0, 1000 - 2(\tau_0 + \tau)]\ !stop\_ack(); (\text{stream}.x_1, \text{control}.x_b) \\
& & \ (c_3 = 0) \\
& x_b & = \emptyset
\end{array}
$$

Contraintes de QoS et Principe d'Action

- QoS offerte: à travers les interfaces en émission.
  - Un acteur doit *émettre* à la cadence indiquée par les interfaces de rôle «client» qu'il possède.
- QoS requise: à travers les interfaces en réception.
  - Un acteur doit être capable de recevoir tous les messages pouvant être délivrés par les interfaces de rôle «serveur» qu'il possède.

Compatibilité des Types: Un Exemple



$\mathcal{A}$rtOC

- Calcul d'acteurs temporisés avec localités
- Temps discret ou continu
- Envoi de message, réception (bloquante ou non bloquante), délai, instanciation d'interfaces, création de thread concurrent (spawn), instanciation d'objets (become)
- doit autoriser:
  - La vérification la conformité des objets par rapport à leur interfaces.
  - L'instanciation correcte des interfaces par rapport à leur types (compatibles).

$\mathcal{A}$rtOC: un Exemple

```
Loc Video_On_Demand =
   S: Incoming_Req,
   Period_AV [ Chan:  AV_t ] =
      ?(40,,) Chan [stop = Stop[Chan]] > !Chan.video() >
      ?(20,,) Chan [stop = Stop[Chan]] > !Chan.audio() >
      ?(20,,) Chan [stop = Stop[Chan]] > !Chan.video() >
      ?(40,,) Chan [stop = Stop[Chan]] > !Chan.video() >
      ?(40,,) Chan [stop = Stop[Chan]] > !Chan.audio() > !Chan.video() >
      ?(40,,) Chan [stop = Stop[Chan]] > !Chan.video() >
      Period_AV [Chan] ,
   Stop [ Chan:  AV_t ] =
      !Chan.stop_ack() > 0 ,
   Video_Server [ S: Incoming_Req ] =
      ?(∞,,) S [ req (R: Reply) =
         create Video_Server [S] > new Chan:  AV_s >
         new Chan:  AV_t > !R.rep(Chan) > delay (2τ_0) >
         ?(5000+2τ,,) Chan [ start() = !Chan.start_ack() >
            ?(40-τ,,) Chan [stop = Stop[Chan]] > !Chan.video() >
            Period_AV[Chan]
         ]
      ] > 0
in
   Video_Server [S]
```

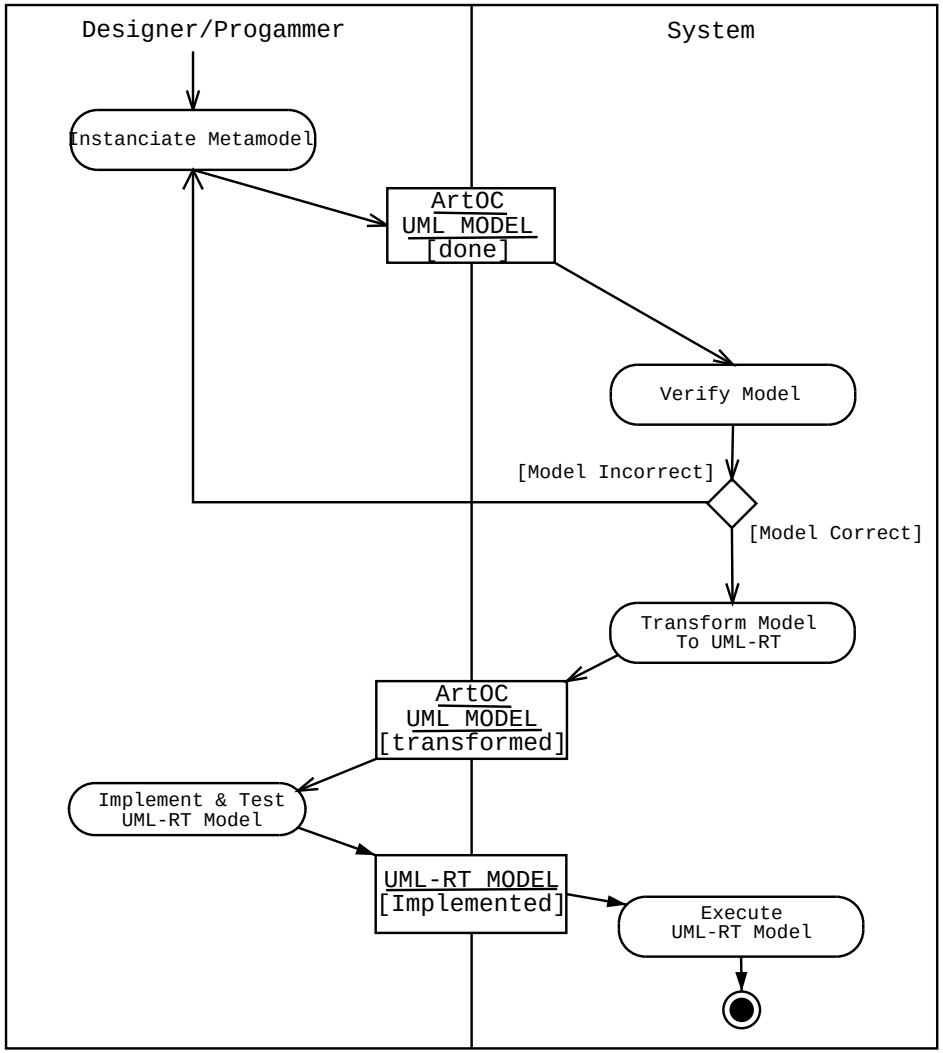
Travaux Existants

- Spécifications algébriques temporisées: extensions de CCS, CSP, LOTOS, etc.
- Langages synchrones et réactifs: Esterel, LUSTRE, Signal, etc.
- Langages de description de la QoS: QML, QL, etc.
- Synchoniseurs (Nielsen et Agha).
- Typage comportemental: à types explicite ou à inférence de types, pour interfaces uniformes ou non-uniformes (Nierstrasz, Puntigam, Nimour, Ravara, etc.).

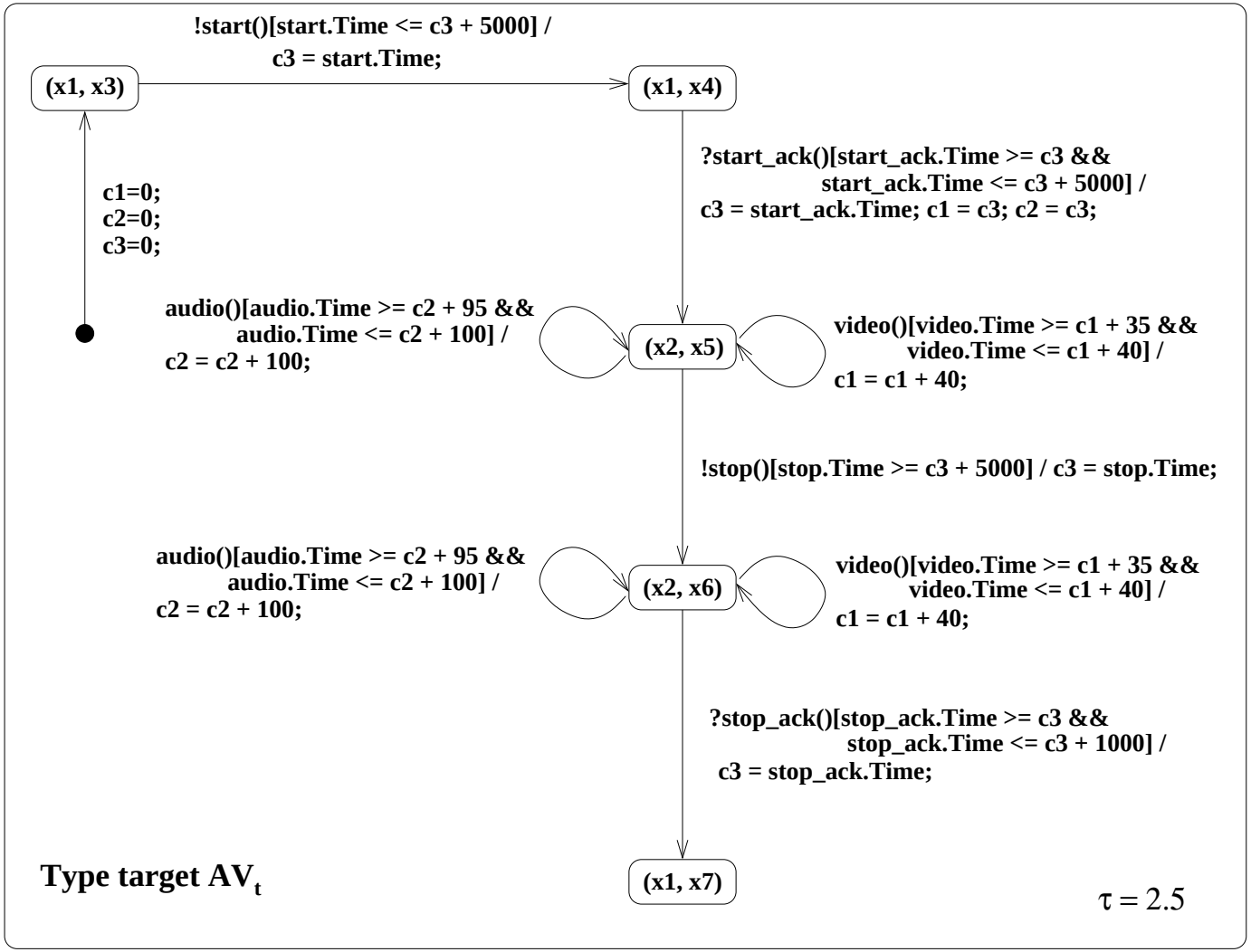
Plan

1. Présentation de UML-RT. L'apport Proposé.
2. Une Meta-Architecture UML Pour *ArtOC*.
3. Spécification et Vérification avec *ArtOC*.
4. Avons-nous Résolu le Problème ?
5. Conclusion.

Activités du Développeur



Un Diagramme d'États Pour le Type AV_t



Conclusion

- Définition de transformations.
- Sous-ensemble des spécifications UML-RT.
- Meilleure implémentation ?
- Problématique états finis / états infinis.
- Avez-vous des questions ?

Syntaxe pour les Types d'Interfaces

$$
\begin{aligned}
\rho &::= ! \mid ? \\
\text{Assignment} &::= c = c' \mid c = \nu \mid \text{Assignment, Assignment} \\
ec &::= c \text{ in } [\tau_{min}, \tau_{max}] \; ! \; m(\tilde{v}) \; ; \text{control}.x'(\text{Assignment}) \\
&\mid \; c \text{ in } [\tau_{min}, \tau_{max}] \; ? \; m(\tilde{v}) \; ; \text{control}.x'(\text{Assignment}) \\
&\mid \; c \text{ in } [\tau_{min}, \tau_{max}] \; \rho \; m(\tilde{v}) \; ; (\text{control}.x', \text{stream}.x'')(\text{Assignment}) \\
es &::= c \text{ in } [\tau_{min}, \tau_{max}] \; m(\tilde{v}) \; ; \text{stream}.x''(\text{Assignment}) \\
&\mid \; c \text{ in } [\tau_{min}, \tau_{max}] \; m(\tilde{v}) \text{ until } \tau_{max} \; ; \text{stream}.x''(\text{Assignment}) \\
\mathrm{E}_c &::= x = ec \mid \mathrm{E}_c + ec \mid \mathrm{E}_c \; \mathrm{E}_c . \\
\mathrm{E}_s &::= x = es \mid \mathrm{E}_s + es \mid \mathrm{E}_s \; \mathrm{E}_s \\
Private\_Role &::= \text{source} \mid \text{target} \\
Public\_Role &::= \text{client} \mid \text{server} \\
\text{Type\_Decl} &::= \text{Type } Private\_Role \; \mathrm{T} \; (\tau_0, \tau) := \quad \text{stream}[c_1 \ldots c_k] : \mathrm{E}_s , \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{control}[c_{k+1} \ldots c_n] : \mathrm{E}_c \\
&\mid \; \text{Type } Public\_Role \; \mathrm{T} \; := \Sigma m(\tilde{v})
\end{aligned}
$$

$\mathcal{A}rtOC$ (2)

C	::=	L	B	::=	0
	|	$C \mid C$		|	new $u : T > B$
L	::=	loc $LocId = Dec$ in E		|	$A[\tilde{u}]$
Dec	::=	$A[\tilde{v}] = B$		|	create $A[\tilde{u}] > B$
	|	$u : T$		|	$!\ u.m(\tilde{u}) > B$
	|	Dec, Dec		|	$?(\tau, @VarId, VarId) \sum_{i=1}^{n} R_i > B$
v	::=	$u : T$		|	delay $(\tau) > B$
E	::=	B	R	::=	$u[\ m_1(\tilde{v}_1) = B_1, \ldots$
	|	$B \mid B$			$\ldots, m_n(\tilde{v}_n) = B_n\ ]$