

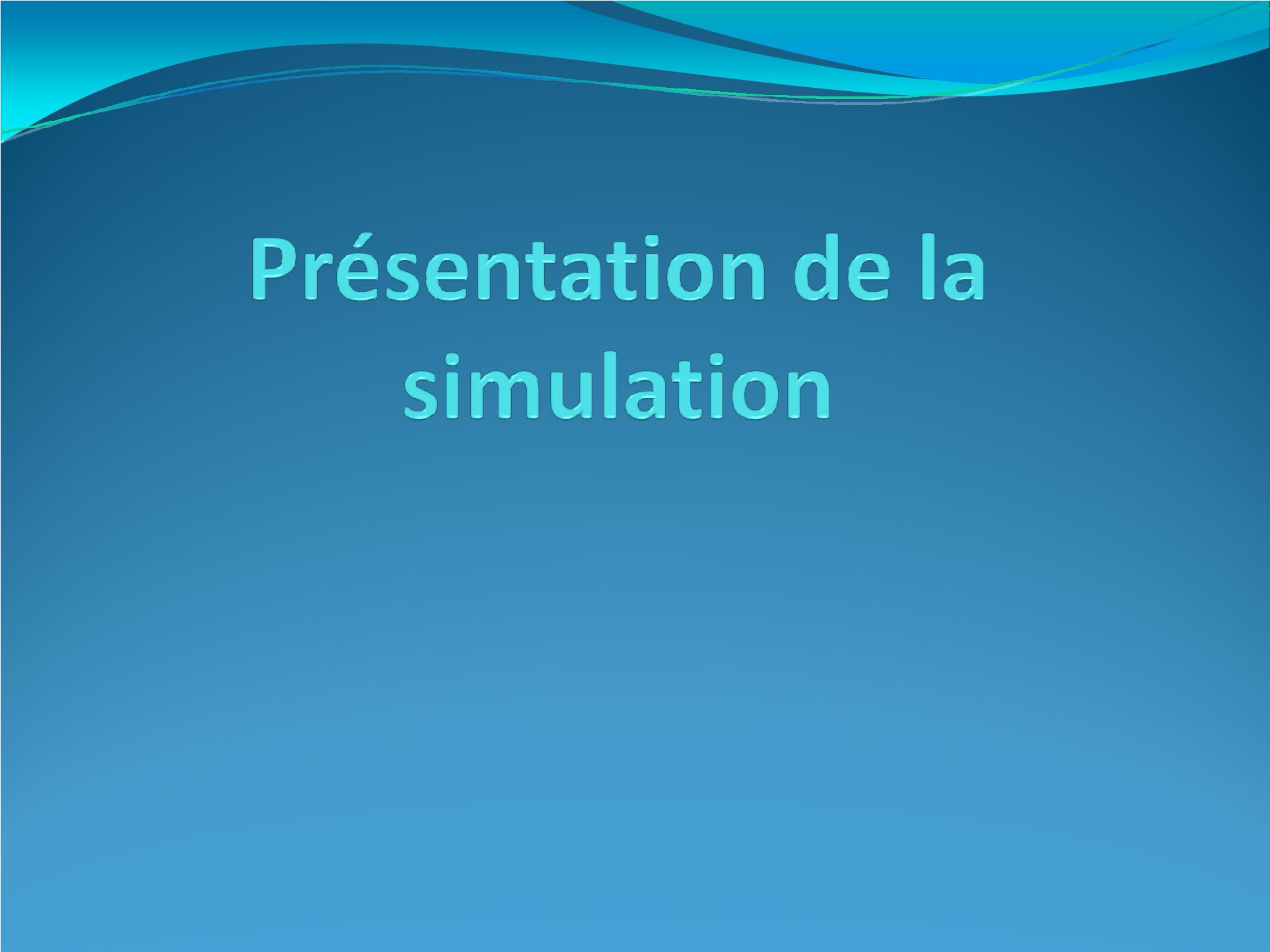
Simulation Distribuée d'un Système Embarqué

Plate-forme PRISE



Sommaire

- Présentation de la simulation
- Diagrammes de séquence HLA
- Couche d'appel à CERTI C++/Java
- Fonctionnalités des fédérés
- Conclusion

The background of the slide is a solid blue color. At the top, there are several wavy, horizontal lines in shades of blue and cyan, creating a sense of movement or a horizon line. The text is centered in the upper half of the slide.

Présentation de la simulation



Introduction

- Objectif: Simuler des systèmes embarqués sur avion de manière distribuée
- Pourquoi une simulation distribuée ?
 - Aujourd'hui, la complexité des appareils et des systèmes embarqués augmente, de même que le besoin en simulation
 - La puissance de plusieurs systèmes informatiques peut être mise à profit pour des simulations d'envergure

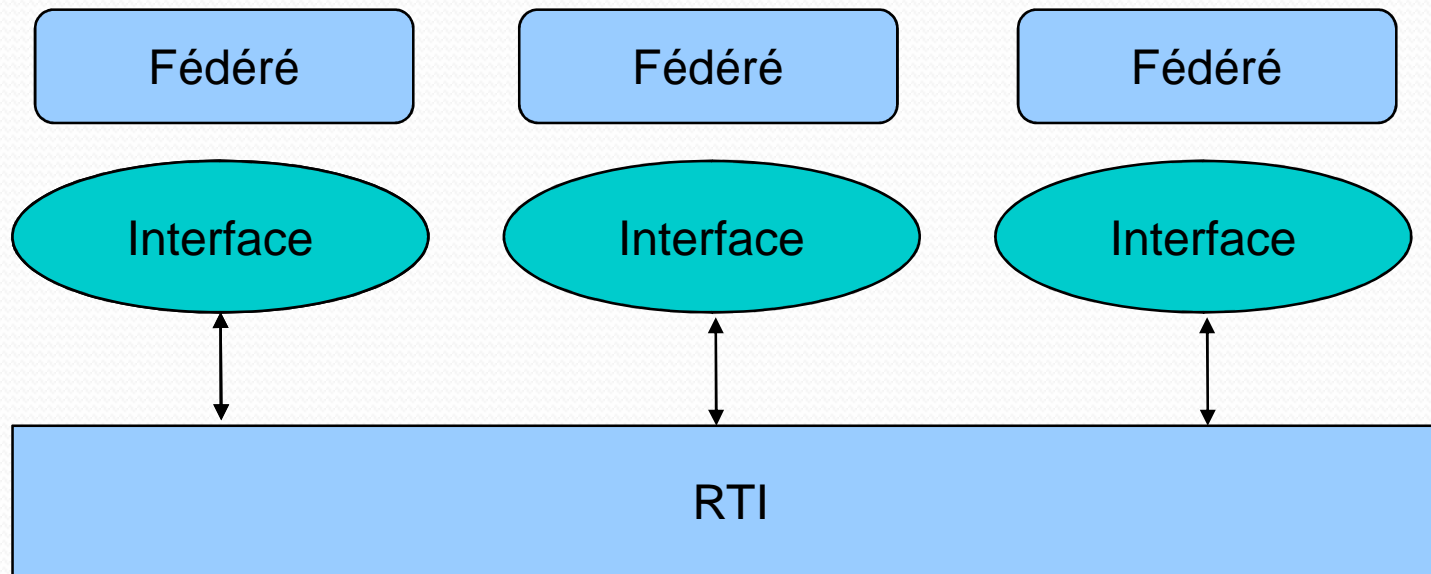
Introduction

- La norme HLA répond à ces besoins, en fournissant une base de commune de travail à un ensemble de processus simulés qui peuvent alors interagir entre eux
- La norme HLA répond à ces besoins, en fournissant une base de commune de travail à un ensemble de processus simulés qui peuvent alors interagir entre eux



Principe

- Distribuer des éléments de la simulation sur des fédérés, constituant une fédération



Objectifs

- Spécifier une fédération permettant la simulation d'un appareil en vol
- Mettre en oeuvre cette fédération sur les noeuds disponibles au DMIA
- ***Matériel disponible au DMIA :***
 - Deux stations HP permettant de faire tourner des simulateurs commerciaux de manière fluide
 - Deux ensemble Yoke-Throttle pour le contrôle de l'appareil simulé
 - Une plateforme de quatre stations RedHawk pour l'implémentation des fédérés

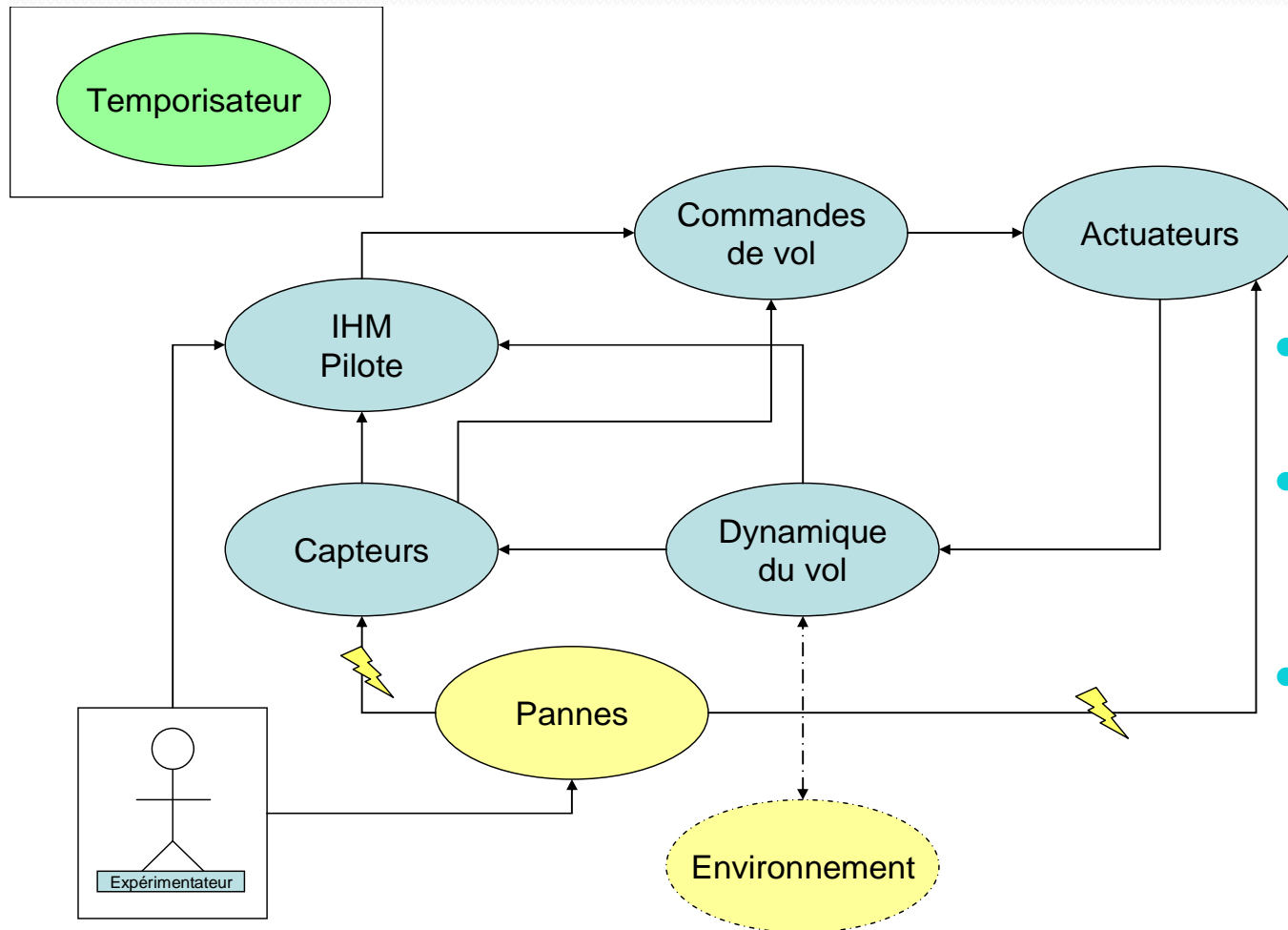




Mise en oeuvre

- *Simplifications:*
 - Pas de vol latéral
 - Dynamique du vol simplifiée
 - Environnement simplifié
 - Lois de commandes simplifiées et non « tunées »
 - Un seul avion simulé
- Permettre l'utilisation et l'amélioration future du projet de manière agréable et intuitive, en partant de bonnes bases.

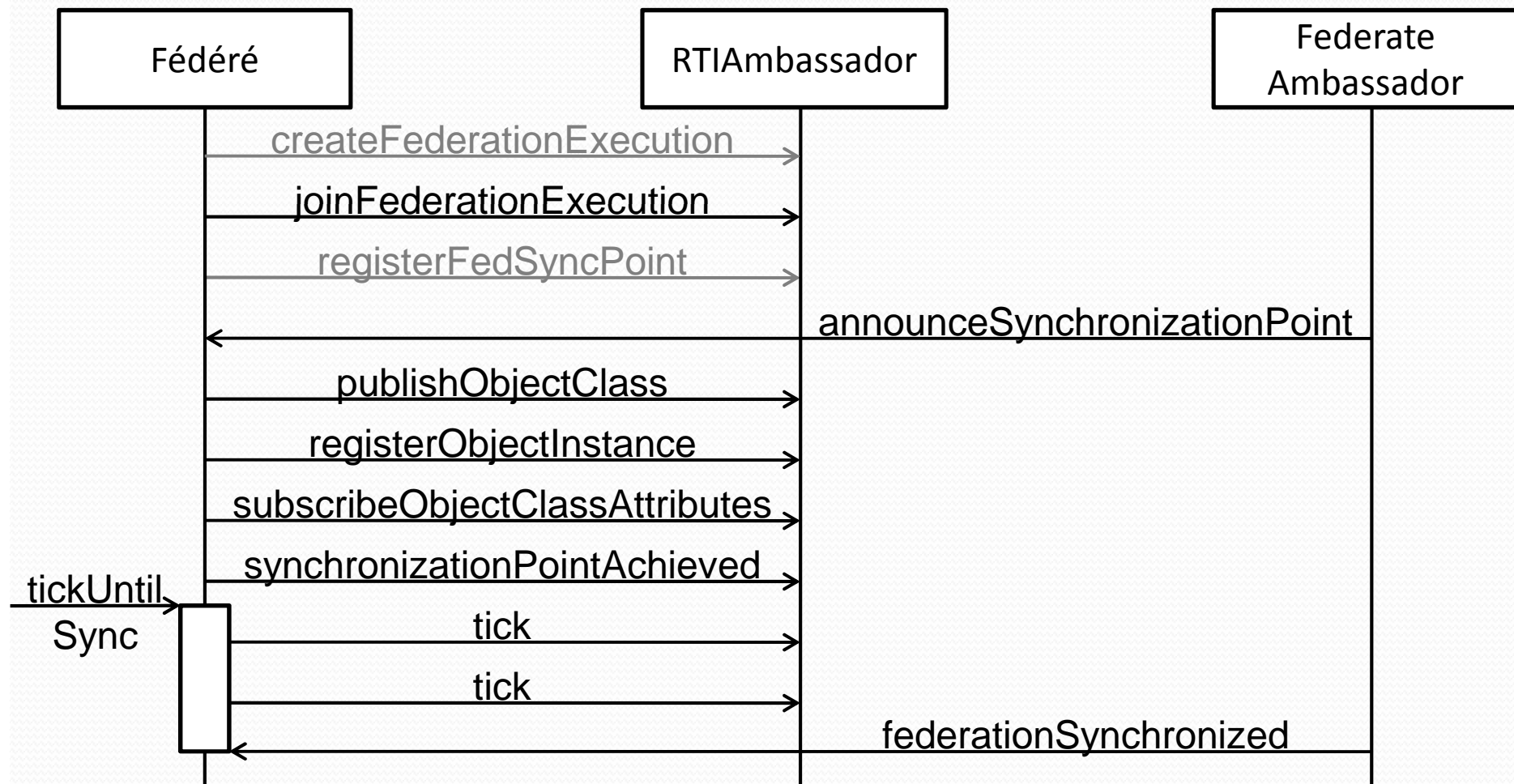
Architecture spécifiée



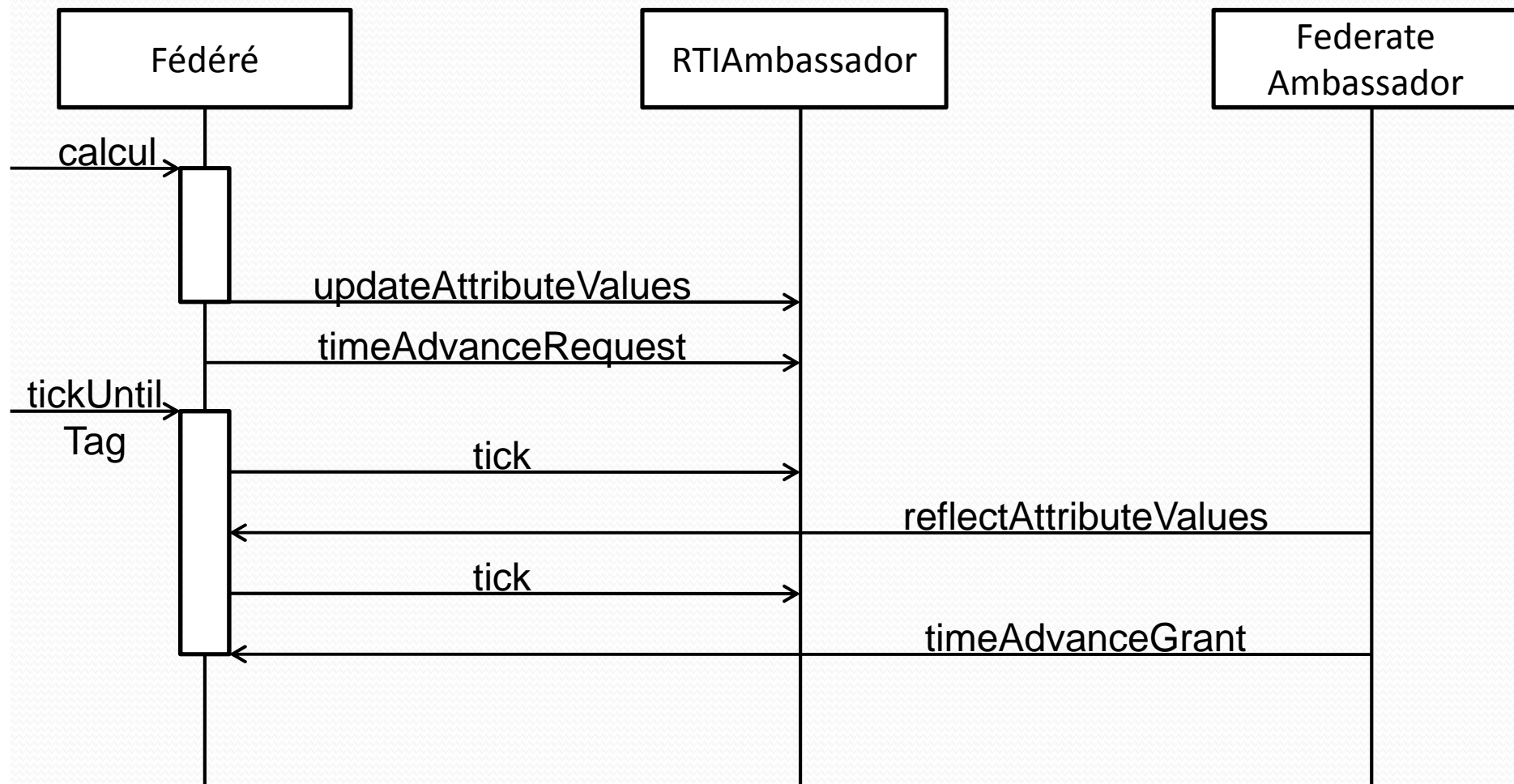
- **Le temps doit être distribué !**
- Certains fédérés seront instanciés plusieurs fois
- La configuration se fait via un fédéré à part

Diagramme de séquence HLA

Diagrammes de séquence HLA



Diagrammes de séquence HLA

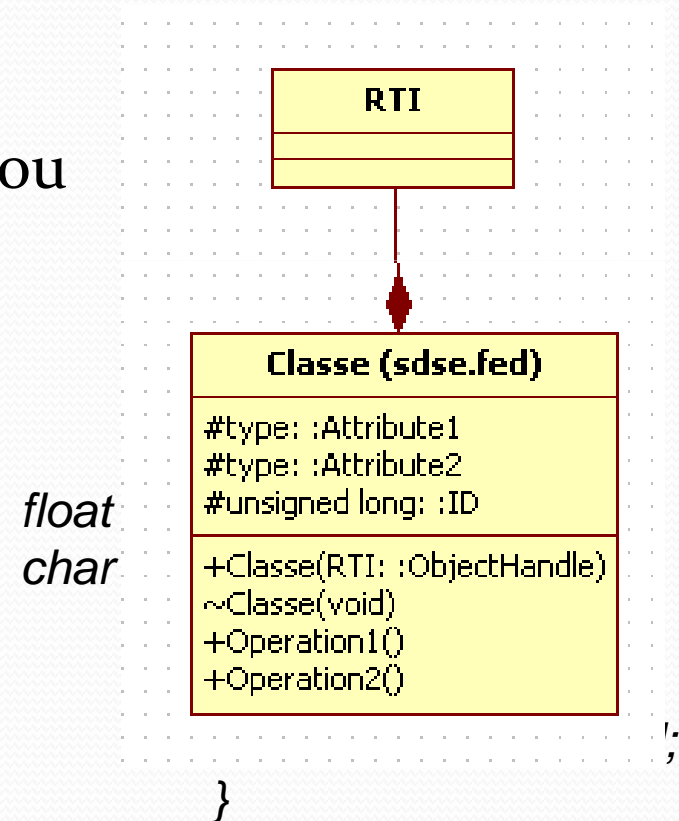


Couche d'appel à CERTI

C++ / Java

Couche d'appel CERTI : C++

- Fédérés inspirés du fédéré «billard»
- Deux types de fonction:
 - envoie de requêtes au RTIA
 - Les fonctions du FederateAmbassador
- Une classe pour chaque objet envoyé ou reçu:
 - Stocker le numéro d'instanciation
 - Plus compréhensible
 - Automatique
- Attention au format des données:
 - En c++: *char * a=(char *)&module;*
 - Problème entre c++ et java
 - On envoie tout dans le même ordre



Couche d'appel CERTI : C++

Les différents appels :

```
DynaFederateAmbassador();  
virtual ~DynaFederateAmbassador()throw (RTI::FederateInternalError);  
RTI::FederateHandle getHandle() const ;  
void recupererlespoignes();  
void publishPosition();  
void publishVitesse();  
void publishOrientation();  
void publishAttitude();  
void subscribePoussee();  
void subscribeAngle_Braquage();  
void synchronize(int );  
void resign(char *);  
void step(void);  
void join(char* ,char*);  
void tick();
```

On s'abonne ou l'on
souscrit très facilement à
de nouvelles classes:
« Générique »

Couche d'appel CERTI : C++

Les différents appels : (suite)

```
RTI::ObjectHandle registerPositionInstance(const char *);  
RTI::ObjectHandle registerOrientationInstance(const char *);  
RTI::ObjectHandle registerVitesseInstance(const char *);  
RTI::ObjectHandle registerAttitudeInstance(const char *);  
void declare_vitesse(char* );  
void declare_position(char* );  
void declare_orientation(char* );  
void declare_attitude(char* );  
void sendUpdatePosition(const RTI::FedTime&);  
void sendUpdateOrientation(const RTI::FedTime& );  
void sendUpdateVitesse(const RTI::FedTime&);  
void sendUpdateAttitude(const RTI::FedTime&);
```


Couche d'appel CERTI : C++

Les call-backs:

```
void discoverObjectInstance(RTI::ObjectHandle ,RTI::ObjectClassHandle ,const char *)
```

```
void announceSynchronizationPoint(const char *, const char *)
```

```
void federationSynchronized(const char *)
```

```
void reflectAttributeValues(RTI::ObjectHandle ,const RTI::AttributeHandleValuePairSet& ,  
    const RTI::FedTime& ,const char *,RTI::EventRetractionHandle )
```

```
void timeAdvanceGrant(const RTI::FedTime& )
```



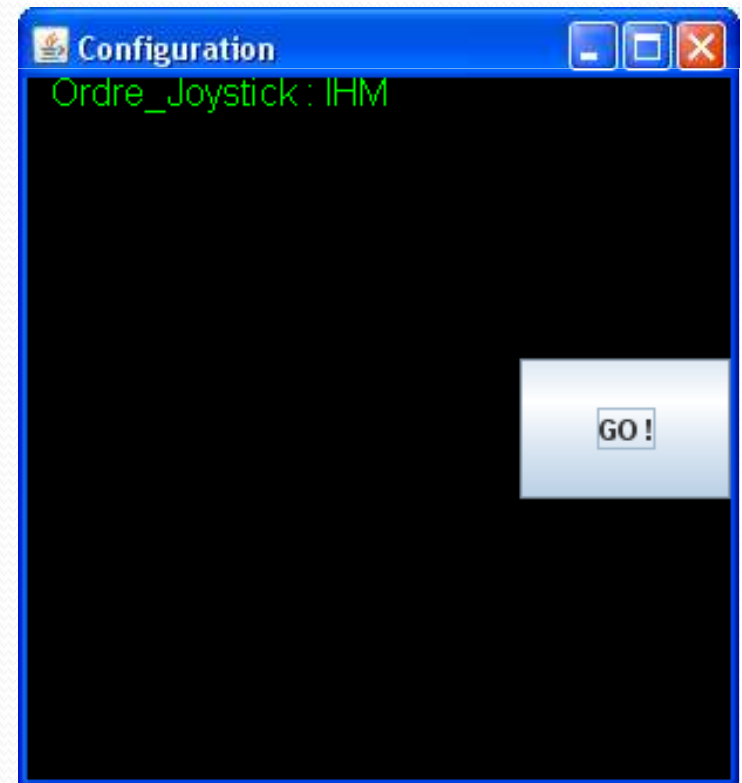
Couche d'appel CERTI : Java

- Deux classes de dialogue
 - GestionnaireHLA (envoi de requêtes au RTIA)
 - FederateAmbassador (NullFederateAmbassador)
- Interface générique pour récupérer les attributs : Poignée
 - void initPoignees(RTIambassador rtia
 - void addAttributes(AttributeHandleSet attributes)
 - boolean setInstance(int objectId, int instance);
 - boolean majAttributes(int instance, ReflectedAttributes attributes);

Fonctionnalité des fédérés

Fédéré Configurateur

- Crée un point de synchronisation
- Trace les connexions des fédérés à la simulation (par les instances)
- Permet de démarrer la simulation
- Régulateur du temps
- Configuré en XML



Fédéré Configurateur - Evolutions

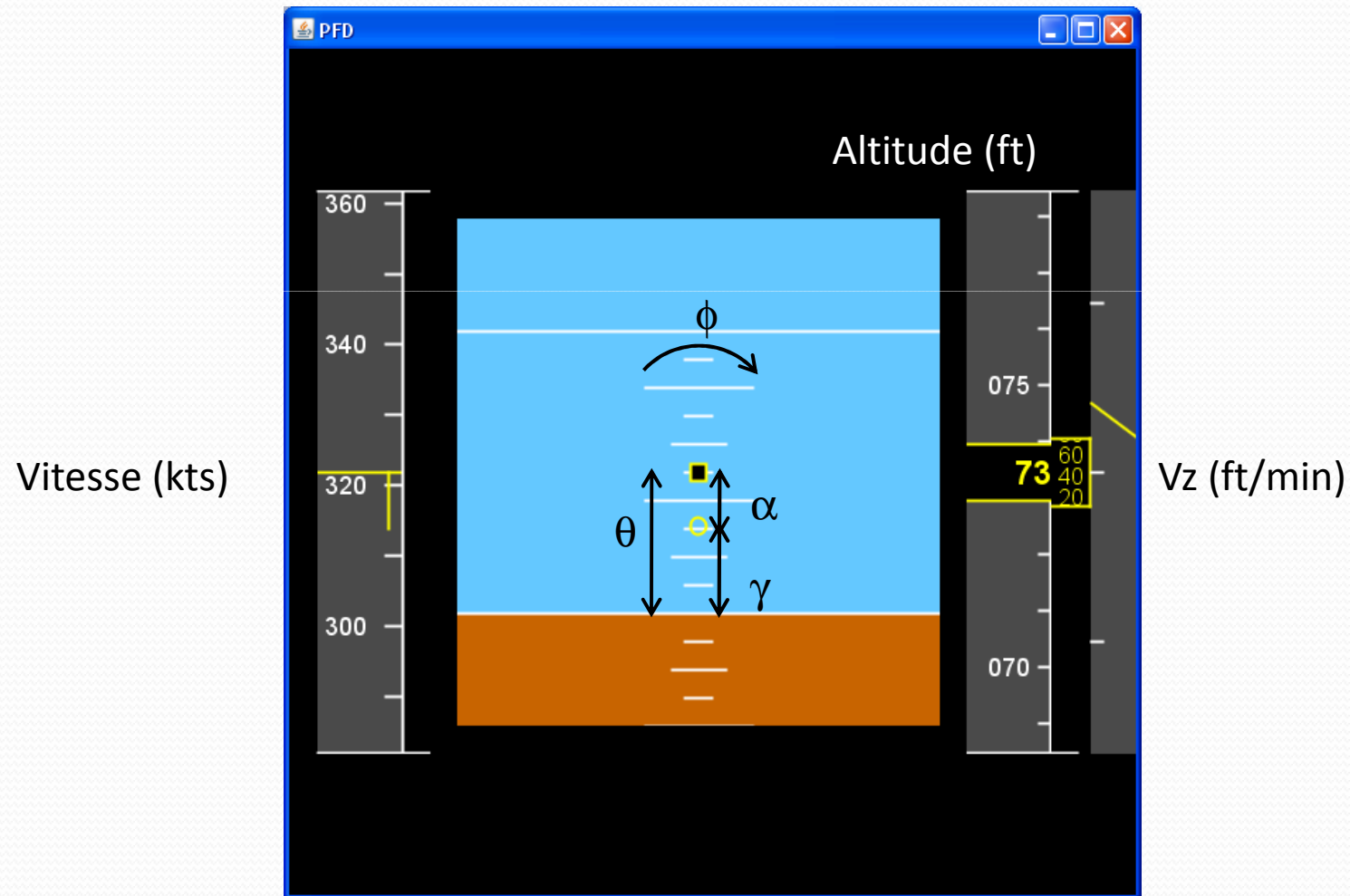
- Transmettre aux autres fédérés des éléments de configuration de la simulation
- Parser un fichier XML de description de la fédération pour recréer le fichier fed



Fédéré IHM - Fonctionnalités

- Afficher au pilote la situation de l'avion
 - Vitesse vraie
 - Assiette, ϕ (θ , ϕ)
 - Incidence (α)
 - Altitude, V_z
- Récupérer les ordres du pilote
 - Tangage, roulis, lacet, gaz
 - Configurable (fichier texte)
- Enregistrement / rejeu possible

Fédéré IHM





Fédéré IHM - Evolutions

- Terminer l'horizon artificiel et le bandeau vitesse verticale
- Ajouter un bandeau de cap (en bas)
- Ajouter un bandeau FMA (en haut)
- Ajouter un pilote automatique
- Développer une interface cockpit complète (EFIS Airbus)
- Ajouter un Flight Manager et un Navigation Display



Fédéré Commandes de vol

- Fonctionnalités
 - Traiter la consigne dans une « boîte » commande de vol
 - Transmettre la consigne aux actuateurs
- Etat Actuel
 - La commande de vol se fait via un PID
 - Les gains ne sont pas réglés, cela nécessite un gros travail
 - La boucle asservi la vitesse de tangage passée en entrée
- Evolutions
 - Calculer finement les gains de la commande de vol
 - Améliorer la commande de vol (Automatique)
 - On peut imaginer avoir plusieurs modes de commande, comme dans un avion réel



Fédéré Actuateurs

- Fonctionnalités
 - Traiter la consigne issue de la commande de vol
 - Transmettre sa position à la dynamique du vol
- Etat Actuel
 - L'actuateur suit la consigne suivant un filtre d'ordre 1...
 - ... et modélise ainsi un smart actuateur, chargé du suivi de consigne
 - Un seul actuateur pour tangage et gaz
 - Gestion des pannes
- Evolutions
 - Discriminer la loi selon l'actuateur
 - Instancier plusieurs actuateurs



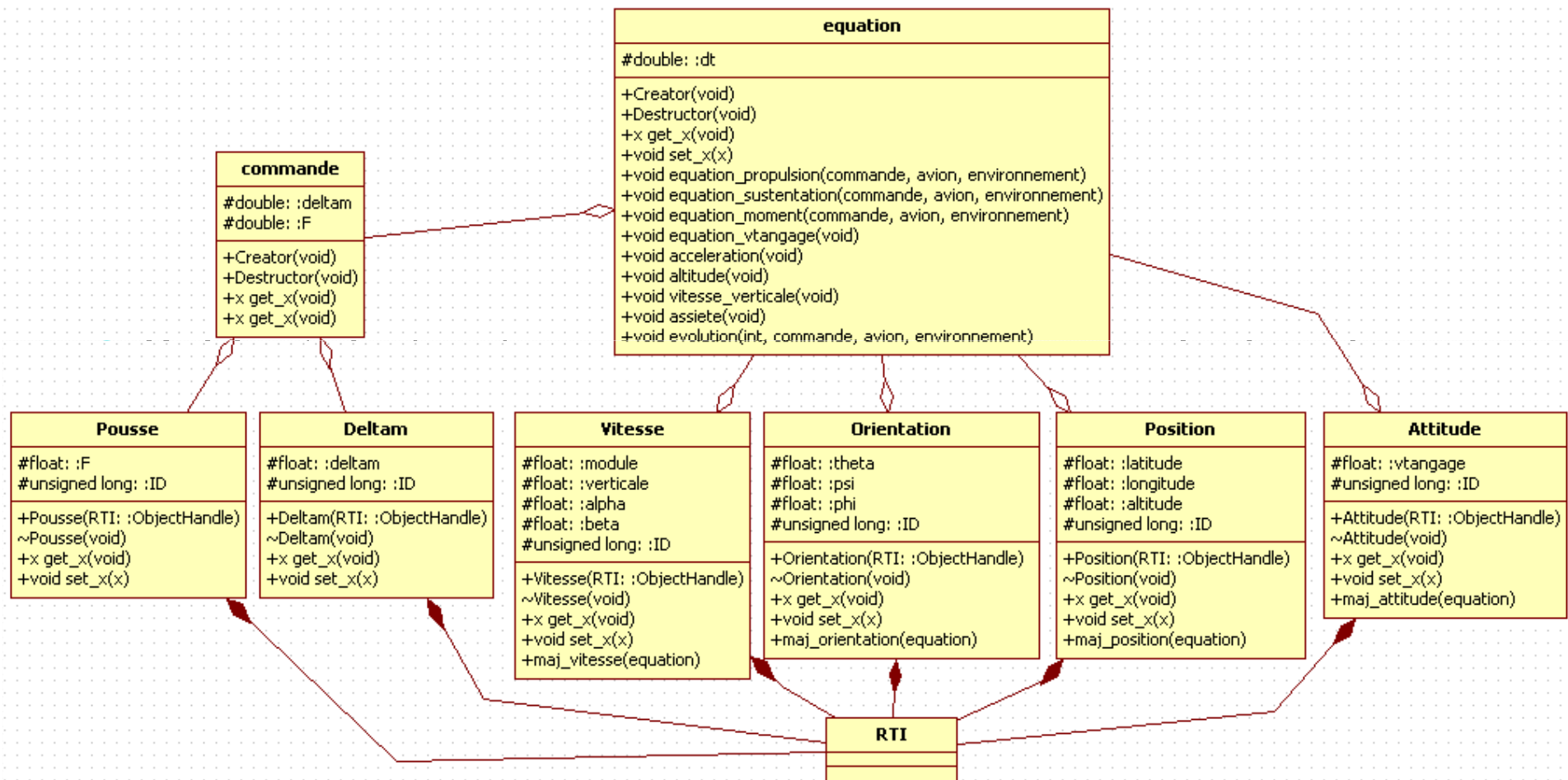
Fédéré Pannes

- IHM développée
- Non connecté à HLA
- Pannes à envisager
 - Moteur
 - Commande
- Evolution
 - Envoyer les pannes sous forme d'interaction

Fédéré dynamique du vol

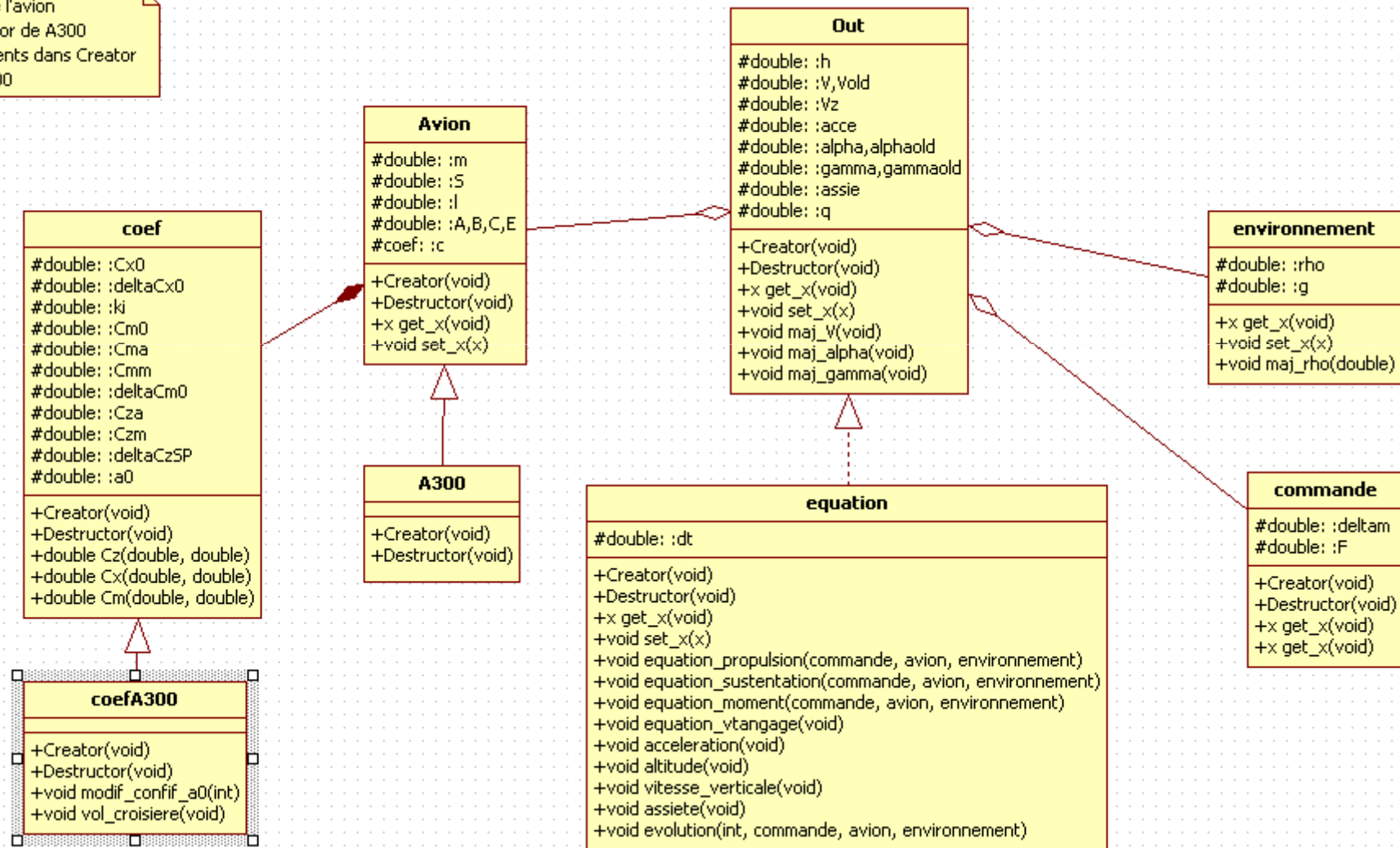
- Vol longitudinal
- 5 équations:
 - Équation de propulsion
 - Équation de sustentation
 - Équation des moments
 - Équation de la vitesse de tangage
 - Équation de la vitesse d'ascension
- Discrétisation des équations
- Simuler le fédéré environnement avec une classe
 - Décroissance linéaire de la densité avec l'altitude
- Calibrer les variables
- Attention au temps (le pilote est en temps réel)

Fédéré dynamique du vol



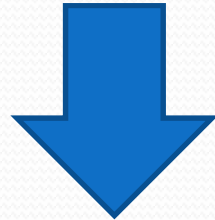
Fédéré dynamique du vol

On initialise l'avion
dans Creator de A300
les coefficients dans Creator
de coefA300



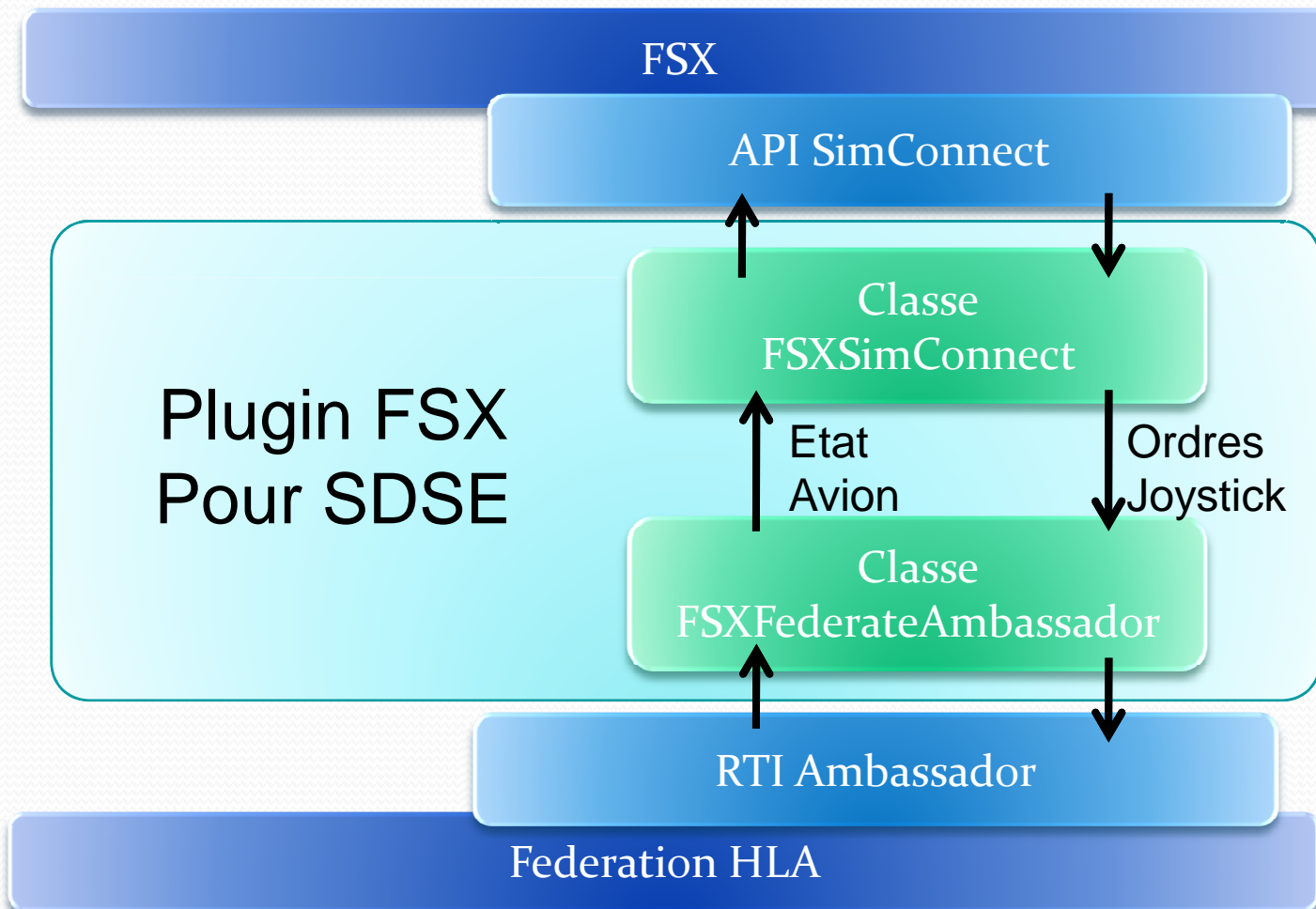
Fédéré dynamique du vol: Bilan

- Générique et modulable
- Protocole de communication avec HLA valide
- Facile à complexifier

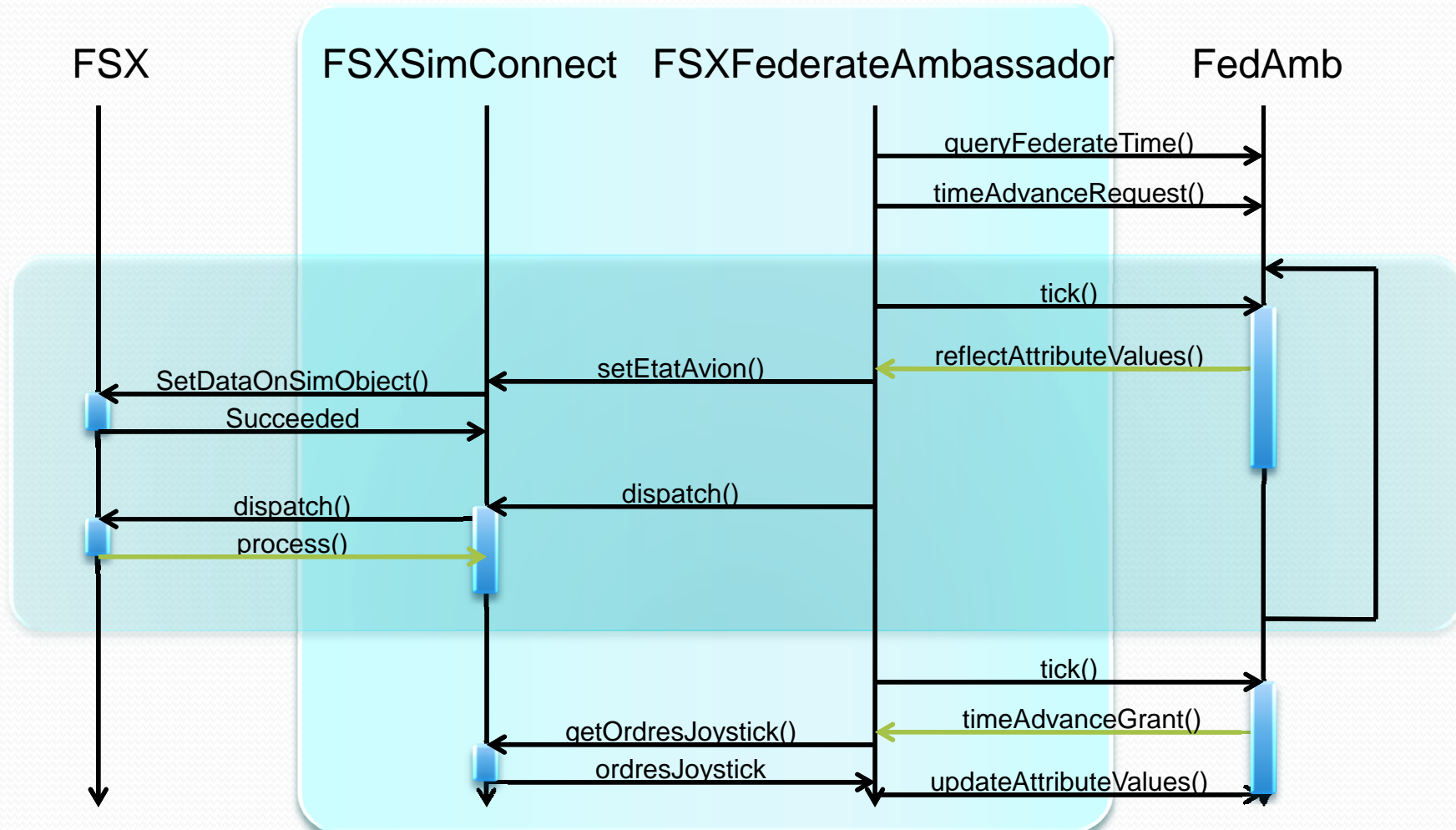


- Intégrer le vol latéral
- Résolution plus fine: discrétisation en Runge-Kutta
- Mieux calibrer les différentes constantes
- Interactions avec un fédéré environnement:
 - Turbulence
 - Variations locales d'incidence

Fédéré FSX : Organisation du plugin



Fédéré FSX : Séquence pour un pas



Fédéré FSX : API SimConnect

```
void FsxSimConnect::connector() {
    SimConnect_Open(&hSimConnect, "FSX SDSE", NULL, 0, 0, 0);
    SimConnect_AddToDataDefinition(hSimConnect, DEFINITION_ALTITUDE, "Plane Altitude",
                                                                    "meter");

    SimConnect_MapClientEventToSimEvent(hSimConnect,
                                         EVENT_JOYSTICK_TANGAGE, "SDSE.joystick0Yaxis");
    SimConnect_MapInputEventToClientEvent(hSimConnect, INPUT_1,
                                         "joystick:0:yaxis", EVENT_JOYSTICK_TANGAGE);
    SimConnect_AddClientEventToNotificationGroup(hSimConnect,
                                                GROUP_1, EVENT_JOYSTICK_TANGAGE, TRUE);
    SimConnect_SetInputGroupState(hSimConnect, INPUT_1, SIMCONNECT_STATE_OFF);
}

void FsxSimConnect::process(SIMCONNECT_RECV *pData, DWORD cbData){
    switch(pData->dwID){
        case SIMCONNECT_RECV_ID_EVENT:{
            SIMCONNECT_RECV_EVENT *evt = (SIMCONNECT_RECV_EVENT*)pData;
            switch(evt->uEventID){
                case EVENT_SIM_START:
                case EVENT_JOYSTICK_TANGAGE:
            }
            break;
        }
        case SIMCONNECT_RECV_ID_SIMOBJECT_DATA: {}
        case SIMCONNECT_RECV_ID_QUIT: {}
    }
}
```

Conclusion



Conclusion

- Une boucle de simulation complète + FSX
 - Encore beaucoup d'évolutions possible
 - Dialogue de fédérés C++ et Java
-
- Un projet très intéressant et très prenant
 - Merci pour votre aide