ÉTS

Présentation du mémoire

Modélisation des systèmes avioniques avec AADL

Présenté par: *Rim BEN DHAOU*

Sous la direction de: *Mr. Roger CHAMPAGNE*

Codirecteur: *Mr. Yann-Gaël GUÉHÉNEUC*

Plan

- Introduction
- Informations générales
- Expérimentations
- Evaluation et limites de AADL
- Modélisation des données (DFDL)
- Conclusion et perspective

INTRODUCTION

- Mise en contexte
- Questions de recherche
- Objectifs
- Méthodologie

Les systèmes avioniques

Systèmes de communication

Utilisés entre l'avion et la terre, entre avions,
ou entre différents éléments de l'avion (capteurs, sondes...)

Systèmes de génération et distribution électrique

Systèmes de navigation

Le pilote automatique

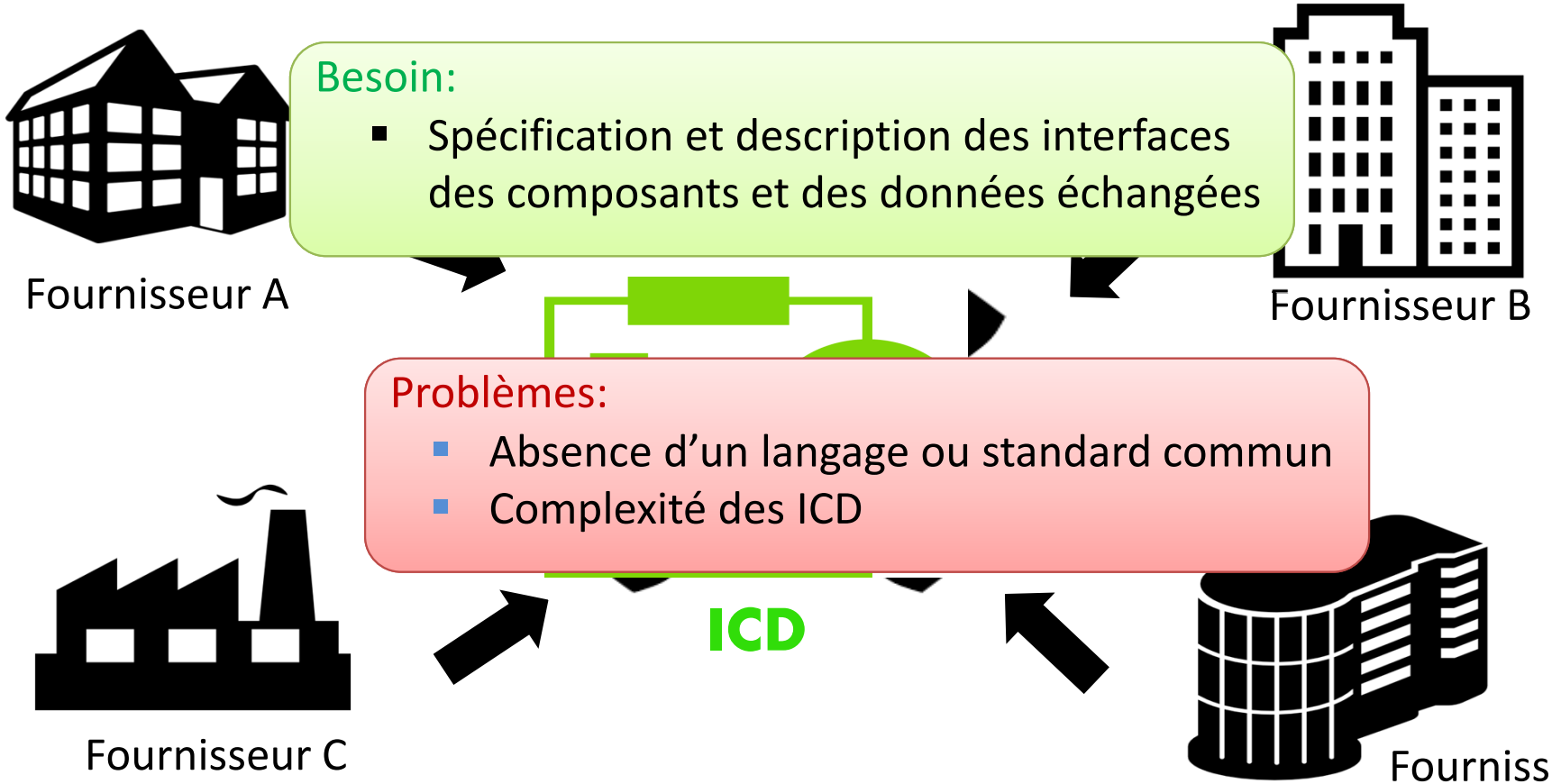
Système d'atterrissage

Radars

Les contrôles de vol à commande électrique et électronique

Les instruments de navigation, de contrôles moteur, et de paramètres de vol

Source : (Moir I. *et al.*, 2013)


Besoin:
Spécification et description des interfaces des composants et des données échangées
Fournisseur A
Fournisseur B
Problèmes:
Absence d'un langage ou standard commun
Complexité des ICD
ICD
Fournisseur C
Fournisseur D

Contexte du projet

- Projet AVIO-506 : Interface Control Document (ICD) Management
  - Trouver un standard/langage commun pour les ICD
  - Réduire les coûts de conception
  - Intégrer et valider les équipements avioniques embarqués décrits par les ICD

Partenaires

- Le Consortium de Recherche et d'Innovation en Aérospatiale au Québec (CRIAC)
- Le Conseil de Recherche en Sciences Naturelles et en Génie (CRSNG)
- Esterline CMC Electronics
- Solutions Isonéo

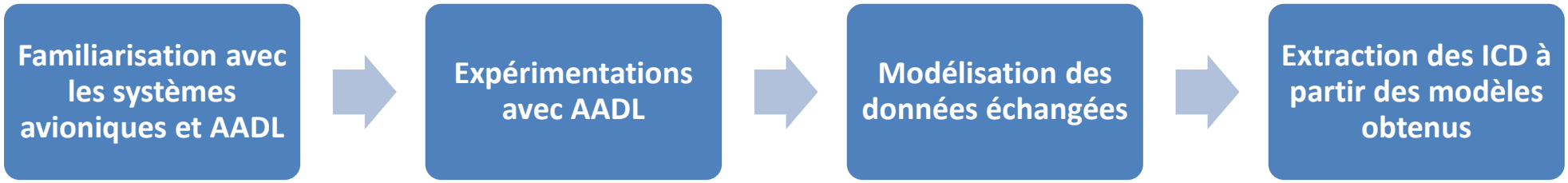
Motivations

- Trouver une suite d'outils libres pour la gestion des ICD
- Explorer le langage AADL (Architecture Analysis and Design Language)
- Profiter d'une suite d'outils existants et libres autour de AADL pour capturer des informations d'interfaces des systèmes avioniques

Questions de recherche

- Est-ce que AADL et son écosystème d'outils libres nous permettent de modéliser les systèmes avioniques et les informations à inclure dans des ICD?
- Est-ce que les outils libres basés sur AADL permettent de valider et vérifier la conformité des modèles aux normes des systèmes avioniques ?
- Est-ce qu'on peut extraire les informations pertinentes reliées aux interfaces des modèles générés?

Démarche


Familiarisation avec les systèmes avioniques et AADL
Expérimentations avec AADL
Modélisation des données échangées
Extraction des ICD à partir des modèles obtenus

INFORMATIONS GÉNÉRALES

- Systèmes avioniques
- Standards avioniques
- AADL

Familiarisation avec les systèmes avioniques et AADL

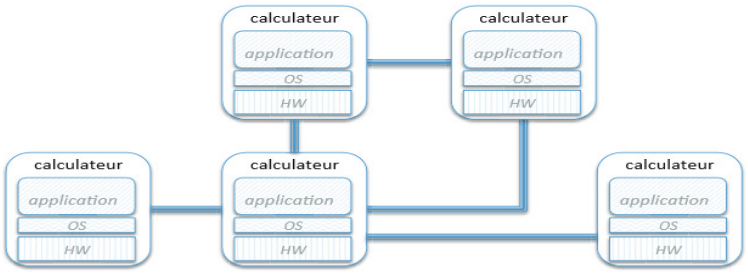
Familiarisation avec les systèmes avioniques et AADL → **Expérimentations avec AADL** → **Modélisation des données échangées** → **Extraction des ICD à partir des modèles obtenus**

- Systèmes avioniques (architecture, normes et standards)
- AADL et son écosystème

Architecture fédérée vs. intégrée

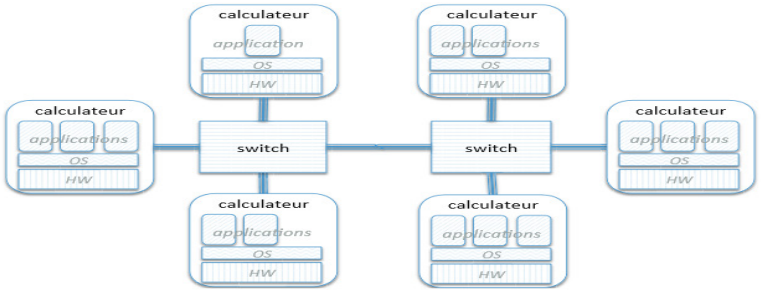
Nouvelle architecture IMA

Architecture fédérée



- Calculateur conçu pour une application
- Chaque calculateur possède ses propres ressources et plateforme matérielle
- Canaux de communication dédiés

Architecture intégrée



- Calculateurs génériques, configurables, exécutant plusieurs applications
- Moins de calculateurs
- Réseau de communication partagé pour la communication entre les calculateurs

Source : (Michael Lafaye, 2014), These : Modélisation de plate-forme avionique pour exploration de performance avance de phase

Standards avioniques pertinents

- ARINC 429
  - Bus de données
  - Acheminer des données entre calculateurs
  - Liaison directe unidirectionnelle (point à point)
  - Un seul émetteur, un ou plusieurs récepteurs
  - Données échangées codées sur 32 bits
- ARINC 653
  - Partitionnement spatial
    - chaque partition a ses propres ressources
  - Partitionnement temporel
    - chaque partition a un accès exclusif au processeur pendant un laps du temps
- ARINC 664
  - Réseau Ethernet *Avionics Full DupleX switched ethernet* (AFDX)
  - Réseau commuté et adapté aux besoins de communication entre les modules avioniques
  - Réseau redondant : double (ou multiple) transmission
  - Liaisons virtuelles (*Virtual Links*, VL)

AADL

- AADL: Architecture Analysis & Design Language
  - Langage standardisé par SAE (*Society of Automotive Engineers*)
  - Langage de modélisation architecturale orienté composants
  - Dédié initialement aux systèmes avioniques puis généralisé aux systèmes embarqués temps réel
  - Décrit les composants logiciels et matériels d'un système

AADL

- Exemple de composants

Composants exécutifs	Composants applicatifs	Composite
Processor, virtual processor	Process	System
Device	Data	
Memory	Subprogram, subprogram group	
Bus, virtual bus	Thread, thread group	

AADL

- La description d'un composant est divisée en deux niveaux (type et implémentation)
- Éléments d'interfaces (ports de communications)
  - Déclarées dans la section ***features*** du composant
  - Un port peut être spécifié par sa direction (in, out , in out) et son type (data, event, event data)
- Les flux
- Les propriétés
- Les annexes

Familiarisation avec les systèmes avioniques et AADL

Expérimentations avec AADL

- Les informations reliées aux interfaces
- Étude de cas (ADIRU)
- Validation du modèle
- Évaluation de l'expérimentation
- Limites de AADL

Modélisation des données échangées

Extraction des ICD à partir des modèles obtenus

Collecte d'informations d'interfaces

Architecture fédérée

- Les interfaces sont les points de liaison physique entre les différents équipements
  - Interconnexions et les flux de données
  - Le nombre de ports, leurs caractéristiques physiques
  - Données échangées (mot ARINC 429) et leurs structures

Architecture intégrée

- Les interfaces physiques ne sont pas pertinentes
  - **Connexions virtuelles**
  - **Les données échangées**

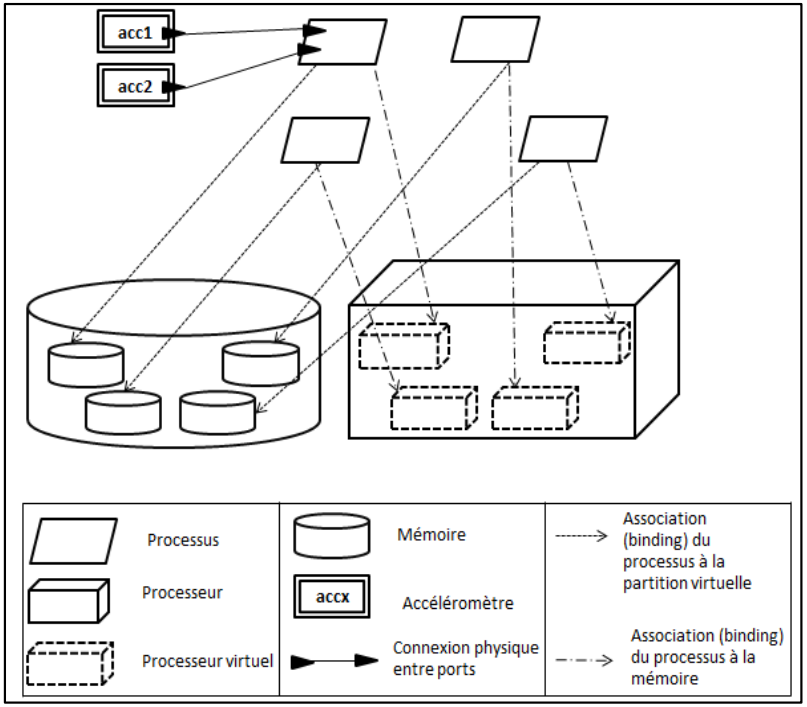
Étude de cas

- Langage de modélisation:
  - AADL V2
    - Annexe ARINC653
- Outil de modélisation:
  - OSATE2
- Exemple à modéliser: ADIRU dans u
  - Inertial Reference System
  - Air Data Reference

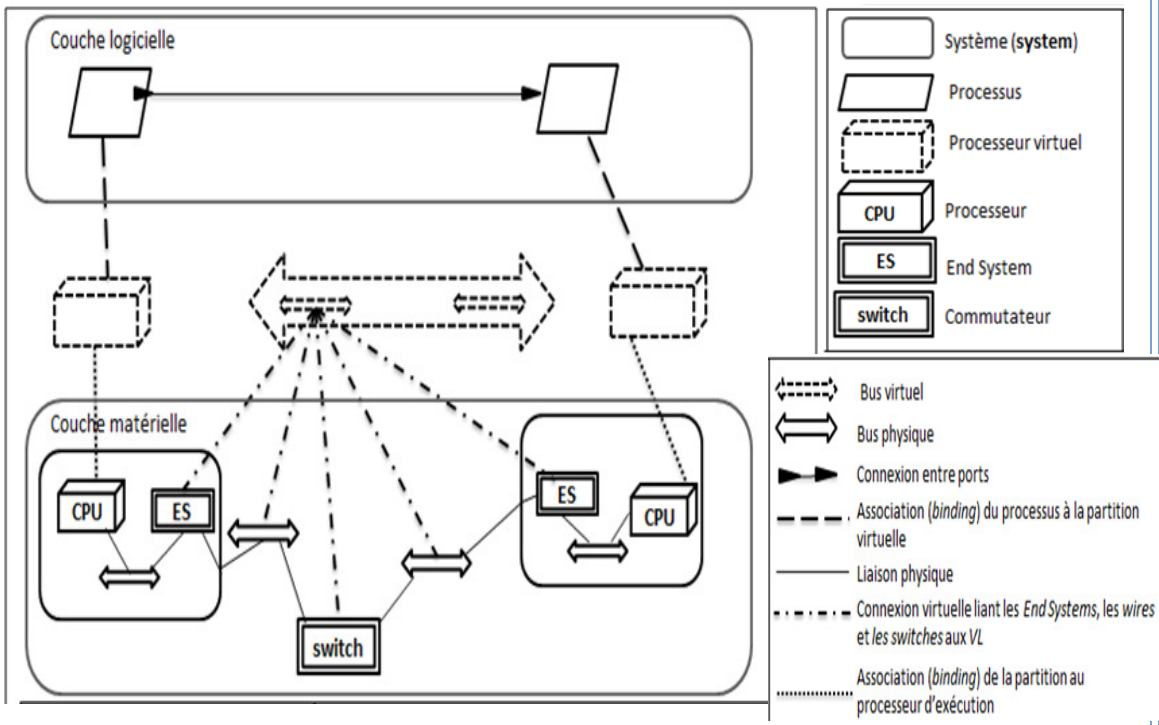
Motivation:
Modèle AADL de
(Hugues et
Delange, 2015)

Sources d'inspiration

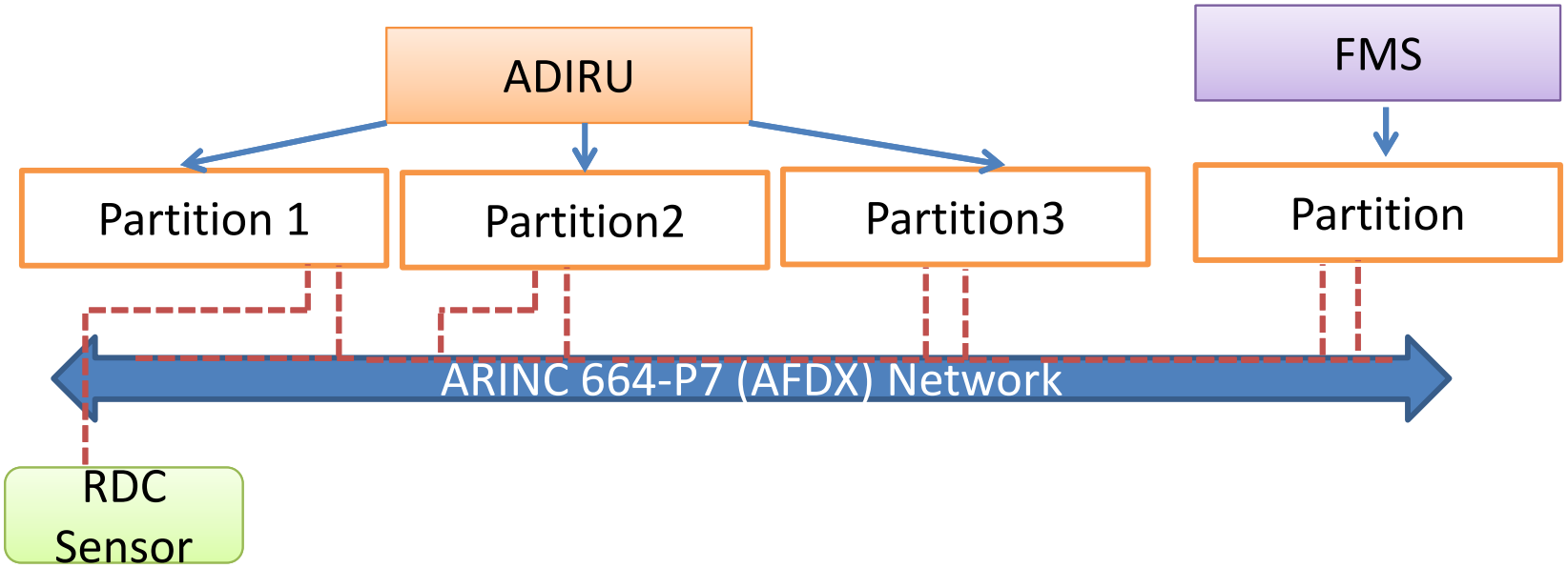
- Modèle Hugues et Delange



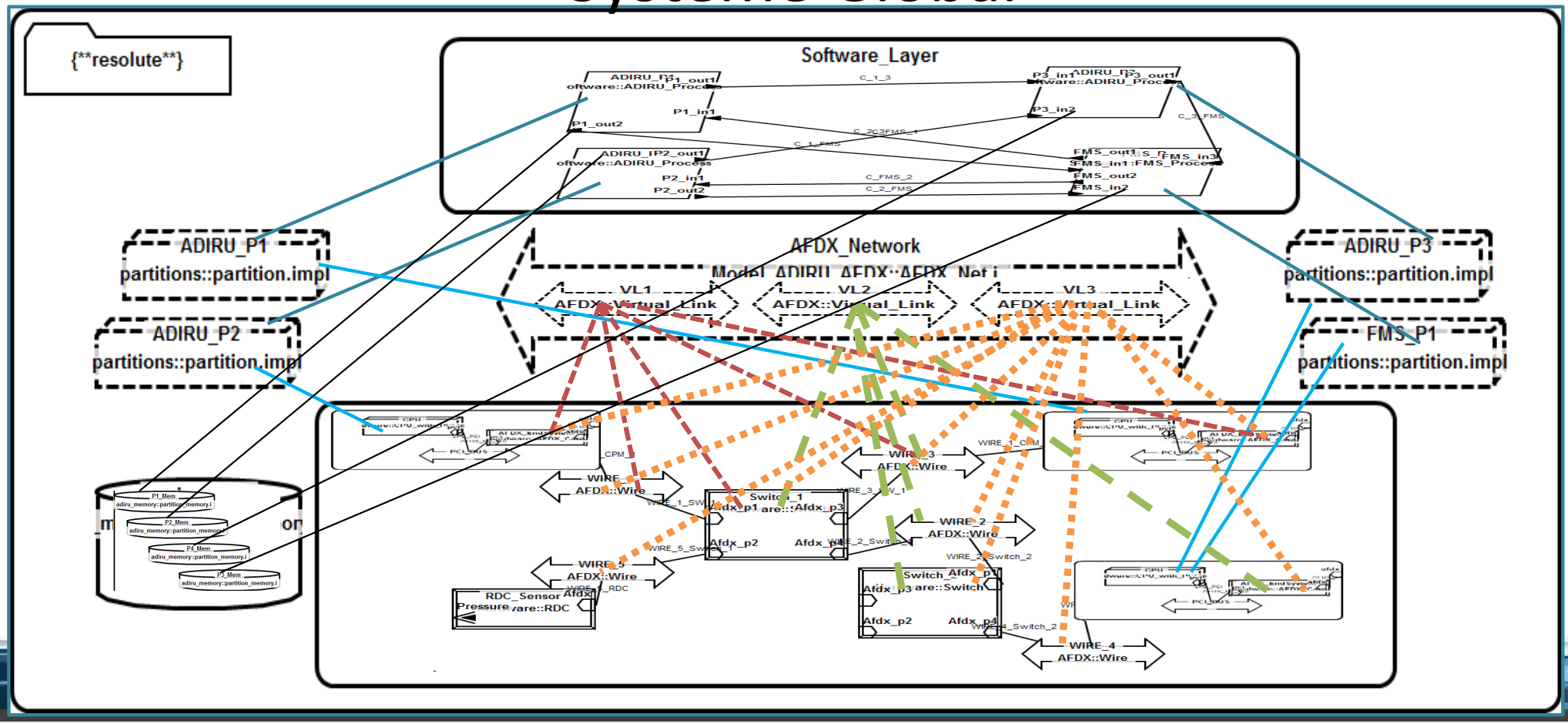
- Modèle Khoroshilov



Étude de cas: ADIRU

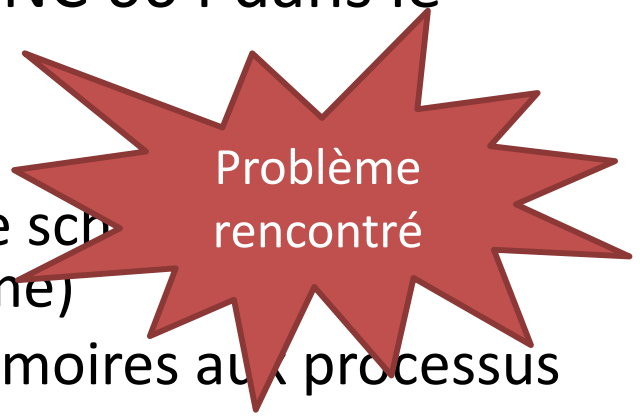


Système Global



Modèle ADIRU

- Présenter les aspects ARINC 653 et ARINC 664 dans le même modèle
- Ajouter les propriétés ARINC 653
  - Partitionnement temporel (propriétés de sch
    (Module_Schedule, Module_Major_Frame)
  - Partitionnement spatial : associer les mémoires aux processus
- Ajouter les propriétés ARINC 664
  - Les propriétés des VL (les tables de routage, ..)
  - Les propriétés des switch (BAG, Lmax, SkewMax)



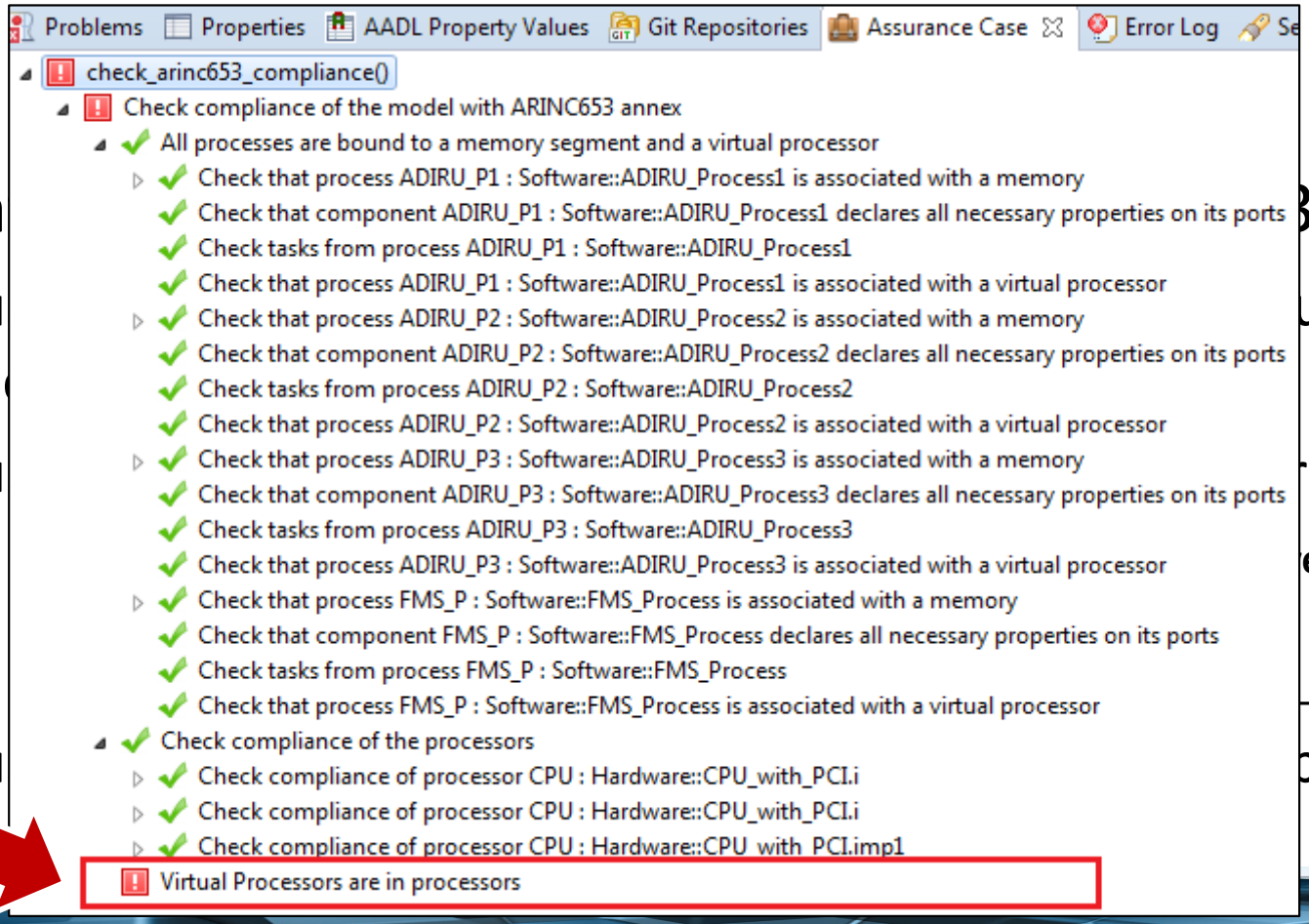
Problème

- Incompatibilité entre ces deux modèles
- Au niveau de la présentation des partitions
  - Modèle 1: les partitions sont déclarées comme des sous composants du processeur physique
  - Modèle 2: les partitions sont déclarées comme des composants autonomes en dehors des processeurs

Validation du modèle

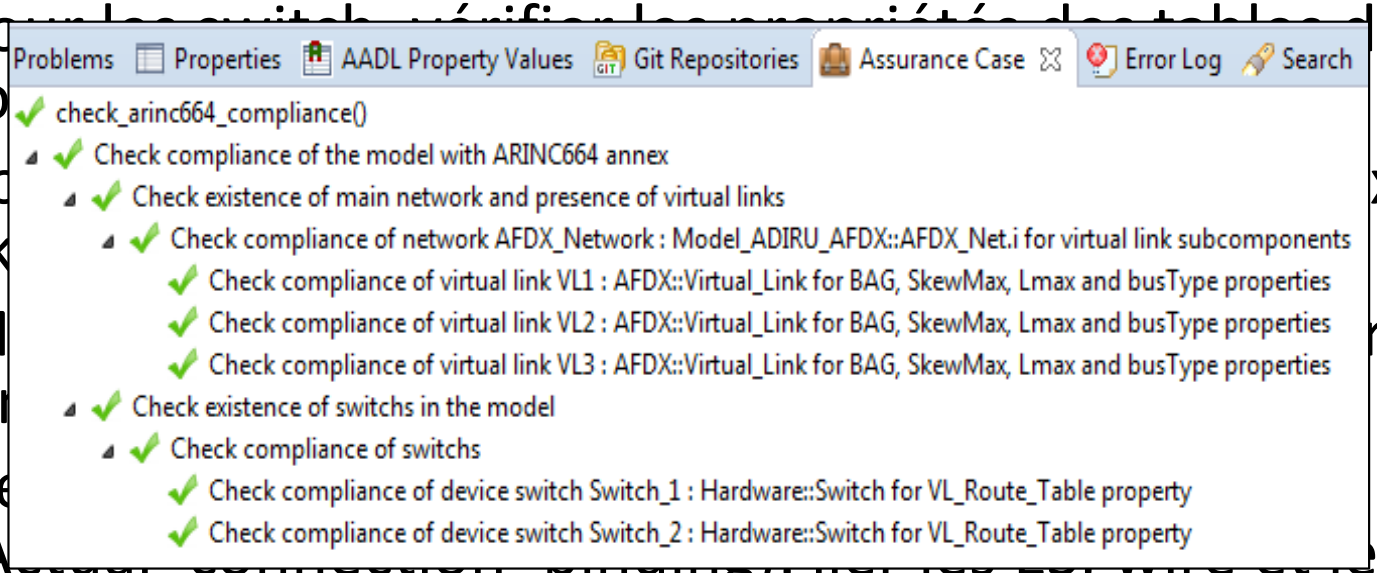
- Validation de la conformité aux normes
  - ARINC 653 (AADL Lib)
  - ARINC 664
- Outil utilisé pour la vérification
  - RESOLUTE

- Valida
  - ✓ Tou
    pro
  - ✓ Tou
  - ✓ Tou



Validation du modèle

- Validation de la conformité à la norme ARINC 664
  - ✓ Pour les switch, vérifier les propriétés des tables de ro
  - ✓ Po x, Sk
  - ✓ M rtual Li
  - ✓ De s switch aux VL



Évaluation des capacités de modélisation d'AADL

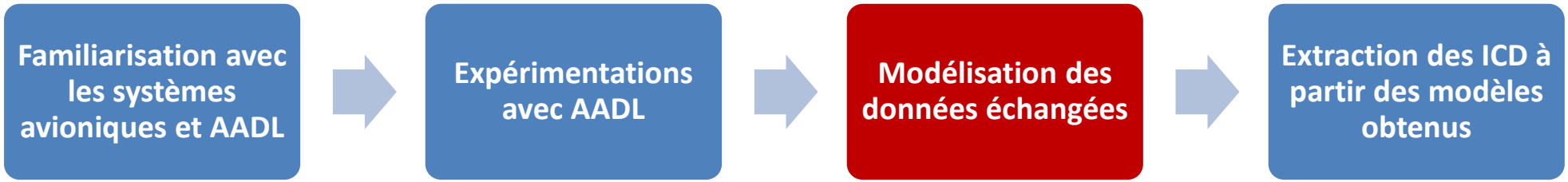
- AADL nous a permis de :
  - ✓ Modéliser les concepts d'un module IMA
    - Usage de l'annexe ARINC653
    - Description des partitions et des caractéristiques d'ordonnancement
    - Description des ports et de leur caractéristiques
    - Description des threads et les caractéristiques d'exécution
  - ✓ Modéliser le réseaux AFDX
    - Présenter les VL, les switch,
    - Utiliser les propriétés AFDX (décrire les caractéristiques des VL: BAG, Lmax, Lmin, SkewMax …)

Problèmes

- Limites:
  - AADL est limité dans la modélisation des données échangées
  - Les données sont représentées sous forme de '*data*' (simple) sans aucune information sur leur structure interne

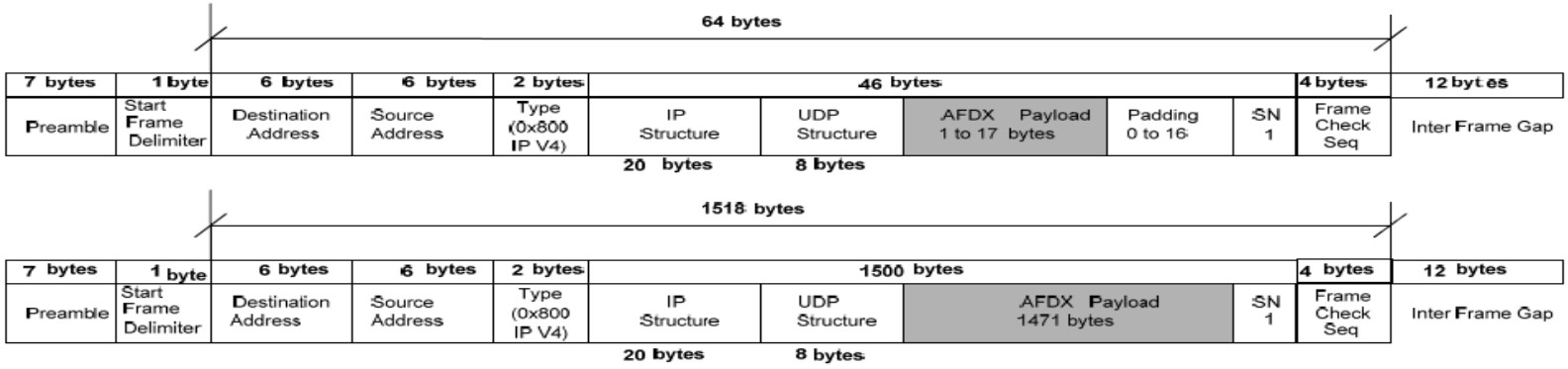
→ Besoin d'un langage spécifique pour la description des données

Modélisation des données échangées



Trame AFDX

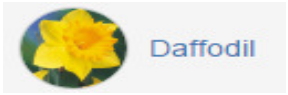
- La description des messages AFDX
  - Comment présenter la structure d'une trame AFDX



Source: ARINC specification 664 P7

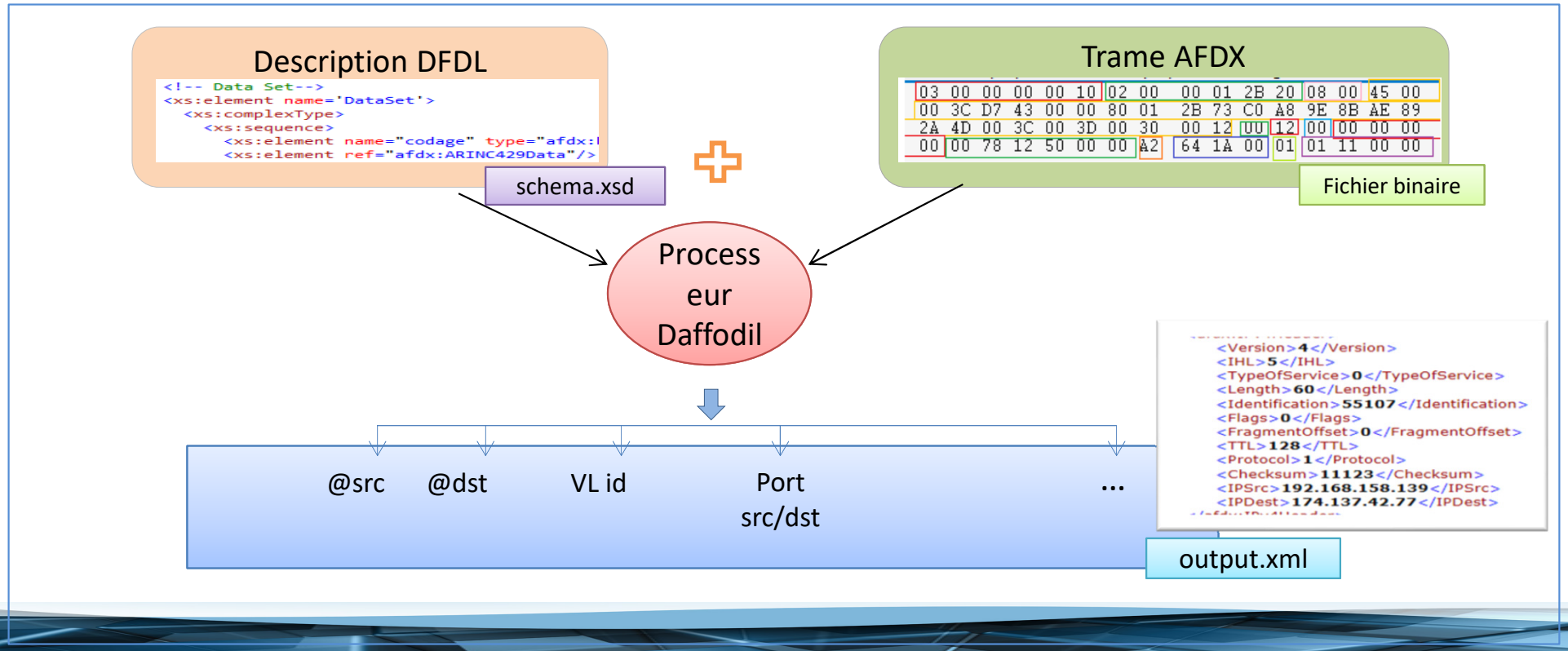
DFDL

- Langage de description de format de données
- W3C XML schema type system (schémas) + annotation DFDL
- Outil utilisé dans notre projet



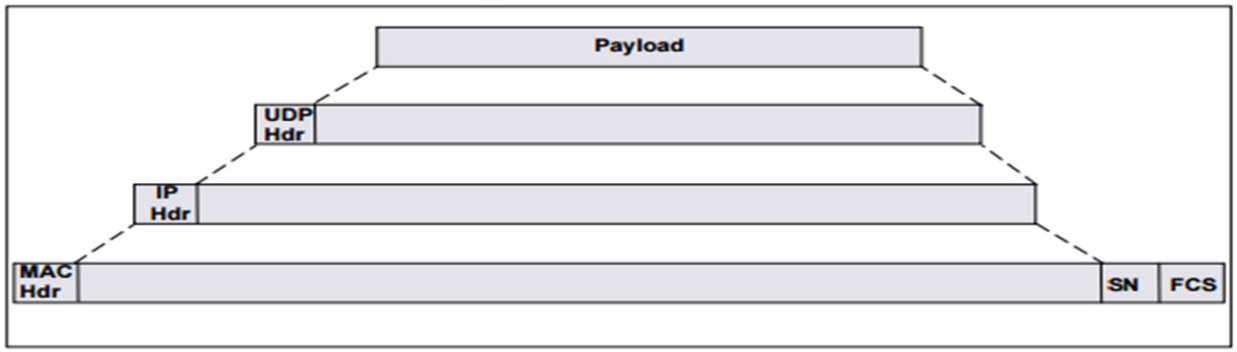
  - Daffodil (open source DFDL)
  - Sa spécification est créée par : Open Grid Forum

DFDL


Description DFDL
<!-- Data Set-->
<xs:element name='DataSet'>
<xs:complexType>
<xs:sequence>
<xs:element name="codage" type="afdx:
<xs:element ref="afdx:ARINC429Data"/>
schema.xsd
Trame AFDX
03 00 00 00 00 10 02 00 00 01 2B 20 08 00 45 00
00 3C D7 43 00 00 80 01 2B 73 C0 A8 9E 8B AE 89
2A 4D 00 3C 00 3D 00 30 00 12 00 12 00 00 00 00
00 00 78 12 50 00 00 A2 64 1A 00 01 01 11 00 00
Fichier binaire
Process eur Daffodil
@src
@dst
VL id
Port src/dst
...
<Version>4</Version>
<IHL>5</IHL>
<TypeOfService>0</TypeOfService>
<Length>60</Length>
<Identification>55107</Identification>
<Flags>0</Flags>
<FragmentOffset>0</FragmentOffset>
<TTL>128</TTL>
<Protocol>1</Protocol>
<Checksum>11123</Checksum>
<IPSrc>192.168.158.139</IPSrc>
<IPDest>174.137.42.77</IPDest>
output.xml

DFDL

- Les couches du protocole AFDX



Trame AFDX

14 bytes	46 .. 1500 bytes	4 bytes
MAC Header	Ethernet Payload	FCS

```
<!-- AFDX Frame -->
<xs:element name="AFDXFrame">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="afdx:MacHeader" />
      <xs:element ref="afdx:EthernetPayload" />
      <xs:element name="FCS" type="afdx:hexByte" dfdl:length="4"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Daffodil releases exemple: https://opensource.ncsa.illinois.edu/confluence/display/DFDL/Examples

DFDL

- Mac Header

- Adresse destination : identifie le Virtual Link (multicast)
- Adresse Source: doit être une adresse unicast (End system Id)

6 bytes	6 bytes	2 bytes
Adresse destination	Adresse source	Type

IPv4 0x0800
16 bits

Constant field 0000 0011 0000 0000 0000 0000 0000 0000	Virt
32 bits	16

Interface Id	0000 0
3 bits	5 bits

```
<!-- MacHeader -->
<xs:element name="MacHeader">
  <xs:complexType>
    <xs:sequence>
      <xs:element name='MacDestAddr'>
        <xs:complexType>
          <xs:sequence>
            <xs:element name='MacDestAddrCstField' type="xs:hexBinary" dfdl:lengthKind="explicit" dfdl:length="4" dfdl:len
              <xs:annotation>
                <xs:appinfo source="http://www.ogf.org/dfdl/">
                  <dfdl:assert message="First thirty two bits of MAC destination address must be a constant field" test="{
                </xs:appinfo>
              </xs:annotation>
            </xs:element>
            <xs:element name='MacDestAddrVirtualLinkId' type="afdx:bit" dfdl:length="16" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <!--xs:element name='MacSrcAddr' type="afdx:hexByte" dfdl:length="6" /-->
      <xs:element name='MacSrcAddrGrp' >
        <xs:complexType>
        <xs:sequence>
          <xs:element name='MacSrcAddrCstField' type="xs:hexBinary" dfdl:lengthKind="explicit" dfdl:length="3" dfdl:length
            <!--TODO check this field-->
            <!--xs:annotation>
```

DFDL

- Ethernet Payload

MAC Header	IP Header	IP Payload	FCS
14 bytes	20 bytes	26 .. 1480	4 bytes

Version	IHL	Type of service	Total length	Fragmentation Id	Control flag	Fragment offset	Time to live	Protocol	Header checksum	IP src adr	IP dest adr
4	4	8	16	16	3	13	8	8	16	32	32

```
<xs:element name="EthernetPayload">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="afdx:IPv4Header" />
      <xs:element ref="afdx:IPv4Payload" />
      <xs:element name="SeqNumber" type="a
    </xs:sequence>
  </xs:complexType>
```

```
<xs:element name='IPv4Header'>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Version" type="afdx:bit" dfdl:length="4" />
      <xs:sequence>
        <xs:annotation>
          <xs:appinfo source="http://www.ogf.org/dfdl/">
            <dfdl:assert message="Version must be 4" test="{ (xs:unsignedInt(./Version) eq 4) }" />
          </xs:appinfo>
        </xs:annotation>
      </xs:sequence>
      <xs:element name="IHL" type="afdx:bit" dfdl:length="4" />
      <xs:element name="TypeOfService" type="afdx:bit" dfdl:length="8" />
      <xs:element name="Length" type="afdx:bit" dfdl:length="16" />
      <xs:element name="Identification" type="afdx:bit" dfdl:length="16" />
      <xs:element name="Flags" type="afdx:bit" dfdl:length="3" />
      <xs:element name="FragmentOffset" type="afdx:bit" dfdl:length="13" />
      <xs:element name="TTL" type="afdx:bit" dfdl:length="8" />
      <xs:element name="Protocol" type="afdx:bit" dfdl:length="8" />
      <xs:element name="Checksum" type="afdx:bit" dfdl:length="16" />
      <xs:sequence dfdl:hiddenGroupRef="afdx:IPSrcGrp"/>
      <xs:element name="IPSrc" type="xs:string" dfdl:inputValueCalc="{ fn:concat(../IPSrcByte[1], '.', ../IPSrcByte[2],
      <xs:sequence dfdl:hiddenGroupRef="afdx:IPDestGrp"/>
      <xs:element name="IPDest" type="xs:string" dfdl:inputValueCalc="{ fn:concat(../IPDestByte[1], '.', ../IPDestByte[2
    </xs:sequence>
  </xs:complexType>
```

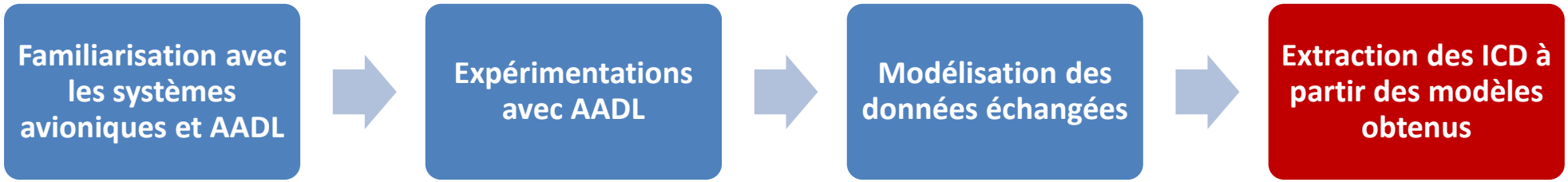
Intégration AADL et DFDL

```
property set ARINC664 is

--Name of the DFDL Frame Schema File
Frame_Schema : aadlstring applies to (port,data);

--Name of the AFDX Frame File
Frame_AFDX : aadlstring applies to (port,data);

end ARINC664;
```

- ssociées aux ports qui
  es
- Convention de no
  - Nom du fichier de
    (schémas XSD)
  - Nom du fichier de la trame AFDX (Fichier.afdx)

```
-------------------------------------------------------------
--- 2.1 ADIRU Process 1
-------------------------------------------------------------
  PROCESS ADIRU_Process1
    FEATURES
      P1_in1 : IN DATA PORT;
      P1_out1 : OUT DATA PORT;
      P1_out2 : OUT DATA PORT DataType::Pressure
      { ARINC664::Frame_Schema => "schema_DFDL.xsd";
        ARINC664::Frame_AFDX   => "frame_AFDX.afdx"; };
  END ADIRU_Process1;
```

Extraction des ICD

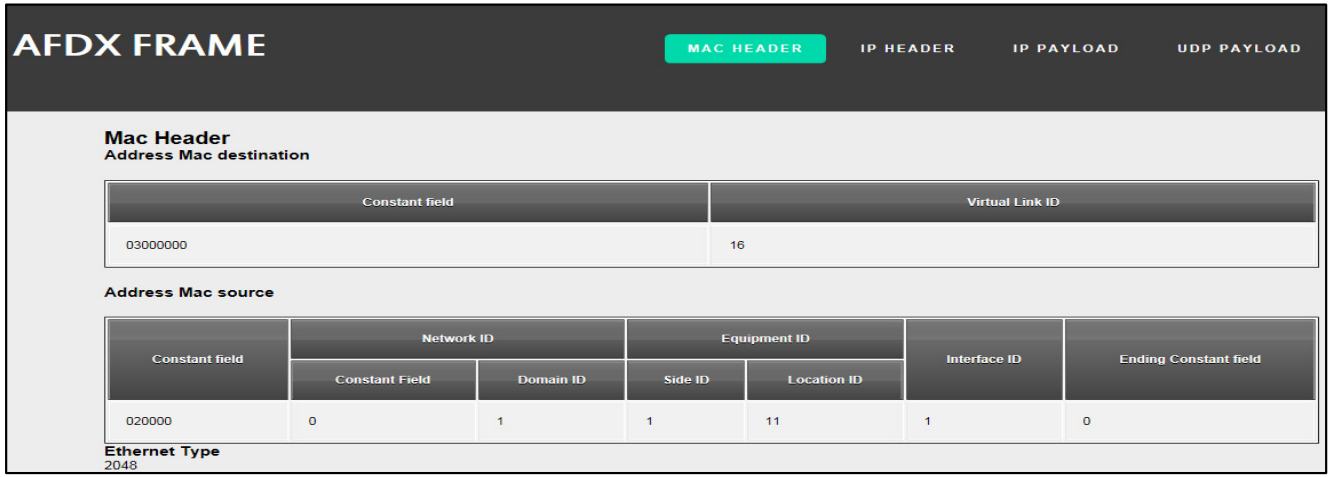


Extraction des ICD

- À partir des modèles XML
- XSLT (Extensible Stylesheet Language Transformations)
- Transformation du contenu XML en HTML
- Sous forme tabulaire (très utilisée par les intégrateurs)

Résultat

- Exemple de trame AFDX



AFDX FRAME

MAC HEADER | IP HEADER | IP PAYLOAD | UDP PAYLOAD

Mac Header

Address Mac destination

Constant field	Virtual Link ID
03000000	16

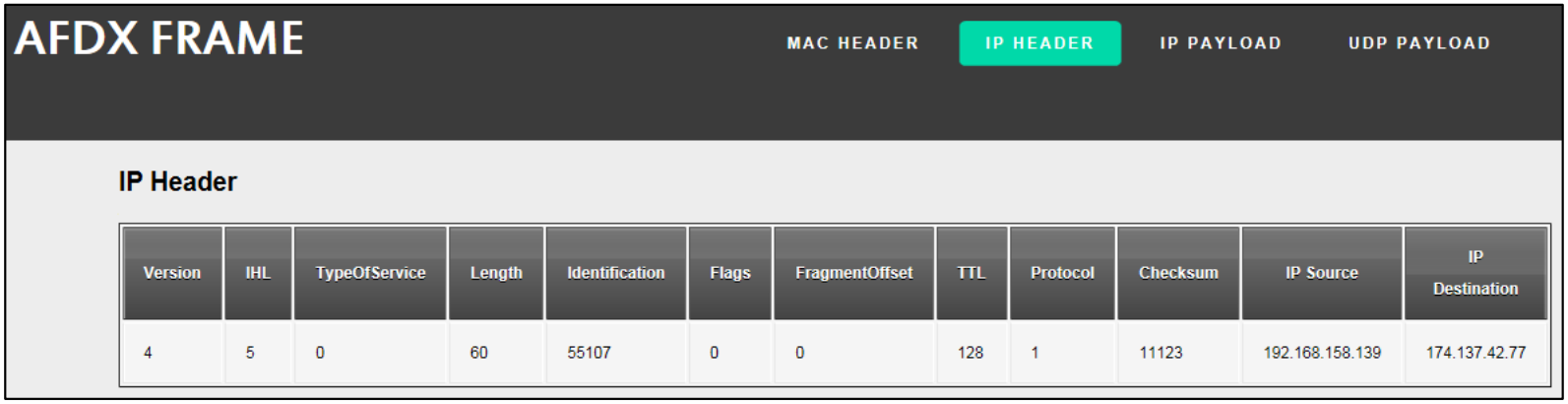
Address Mac source

Constant field	Network ID		Equipment ID		Interface ID	Ending Constant field
	Constant Field	Domain ID	Side ID	Location ID		
020000	0	1	1	11	1	0

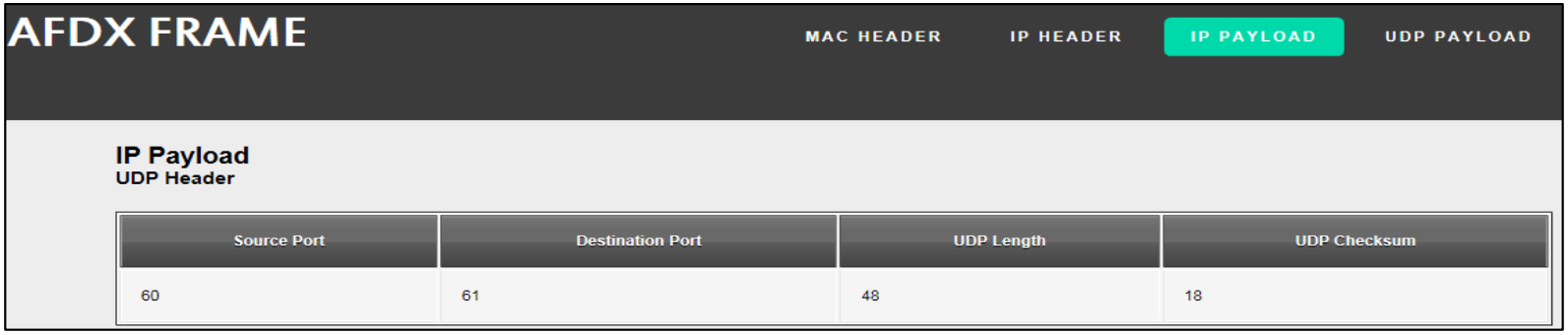
Ethernet Type

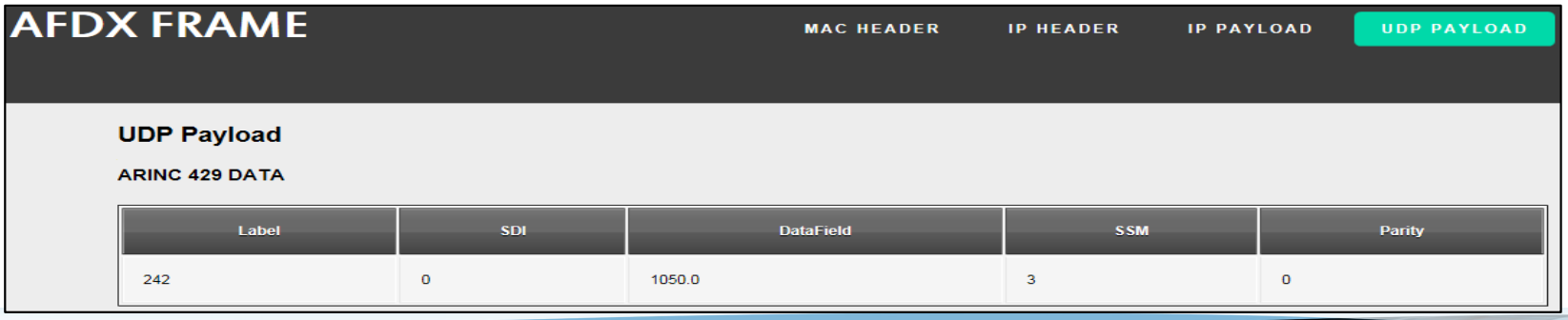
2048

Résultat


AFDX FRAME
MAC HEADER
IP HEADER
IP PAYLOAD
UDP PAYLOAD
IP Header
Version
IHL
TypeOfService
Length
Identification
Flags
FragmentOffset
TTL
Protocol
Checksum
IP Source
IP Destination
4
5
0
60
55107
0
0
128
1
11123
192.168.158.139
174.137.42.77

Résultat


AFDX FRAME
MAC HEADER
IP HEADER
IP PAYLOAD
UDP PAYLOAD
IP Payload
UDP Header
Source Port
Destination Port
UDP Length
UDP Checksum
60
61
48
18


AFDX FRAME
MAC HEADER
IP HEADER
IP PAYLOAD
UDP PAYLOAD
UDP Payload
ARINC 429 DATA
Label
SDI
DataField
SSM
Parity
242
0
1050.0
3
0

Évaluation des expérimentations

- Efficacité de DFDL dans la description des structures de données (trame ADFX) dans les deux sens (parsing et unparsing)
- Complémenter AADL avec la description du format de données (DFDL)
- On obtient l'architecture générale du sous-système, les partitions, les ports (E/S) + la description des messages échangés

Conclusion

- AADL
  - ✓ Modéliser les différents composants d'un système IMA
  - ✓ Validation par RESOLUTE (peut être exploiter dans la validation de l'interopérabilité des sous-systèmes)
  - ✗ Limites dans la modélisation des données
- DFDL
  - ✓ Modélisation des données (exemple : trames AFDX)

Perspectives

- Valider notre solution avec les partenaires industriels pour vérifier sa pertinence en termes de temps de réalisation et de coûts
- Diversifier les cas d'exemples des sous-systèmes avioniques pour en sortir avec une confirmation plus solide
- Exploiter RESOLUTE dans la validation de l'intégrations des sous-systèmes
- Améliorer l'intégration entre la description DFDL et le modèle

Merci pour votre attention

Références

- (AEEC., 2009) AEEC., 2009a. ARINC-664-P7: Aircraft data network part 7 avionics full-duplex switched Ethernet network. Aeronautical Radio.
- (AEEC., 2010) AEEC., 2010. ARINC 653: Avionics Application software standard interface. Aeronautical Radio.
- (AEEC., 2012) AEEC., 2012. ARINC 429P1-18: Digital Information Transfer System (DITS) part 1 functional description, electrical interfaces, label assignments and word formats. Aeronautical Radio.
- (Hugues, J., & Delange, J., 2015) Hugues, J., & Delange, J. (2015). Modeling and Analyzing IMA Architectures with AADL, From Modeling to Safety Evaluation and Code Generation: A Case-Study.
- (Khoroshilov A., 2012) Khoroshilov, A., Albitskiy, D., Koverninskiy, I., Olshanskiy, M., Petrenko, A., & Ugnenko, A. (2012). AADL-based toolset for IMA system design and integration. *SAE International Journal of Aerospace*, *5*(2012-01-2146), 294-299.
- (Moir, I *et al*., 2013) Moir, I., Seabridge, A., & Jukes, M. (2013). Civil avionics systems. John Wiley & Sons.
- (Feiler, P. H., & Gluch, D. P., 2012). Feiler, P. H., & Gluch, D. P. (2012). Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language. Addison-Wesley.

Objectifs

- Déterminer les informations d'interfaces nécessaires et pertinentes à extraire pour chacun des deux types principaux d'architecture en avionique (fédérée, modulaire)
- Modéliser avec AADL un cas d'équipement avionique dans un contexte modulaire (IMA)
  - Valider la conformité des modèles obtenus aux standards avioniques (principalement ceux d'ARINC)
  - Analyser les capacités et les limites d'AADL quant à la modélisation des interfaces
- Extraire les ICD