Slow Control et Acquisition en Ada

Eric Legay – Nicolas Dosme

CSNSM

5ème Journée Informatique IN2P3 – DAPNIA 2006

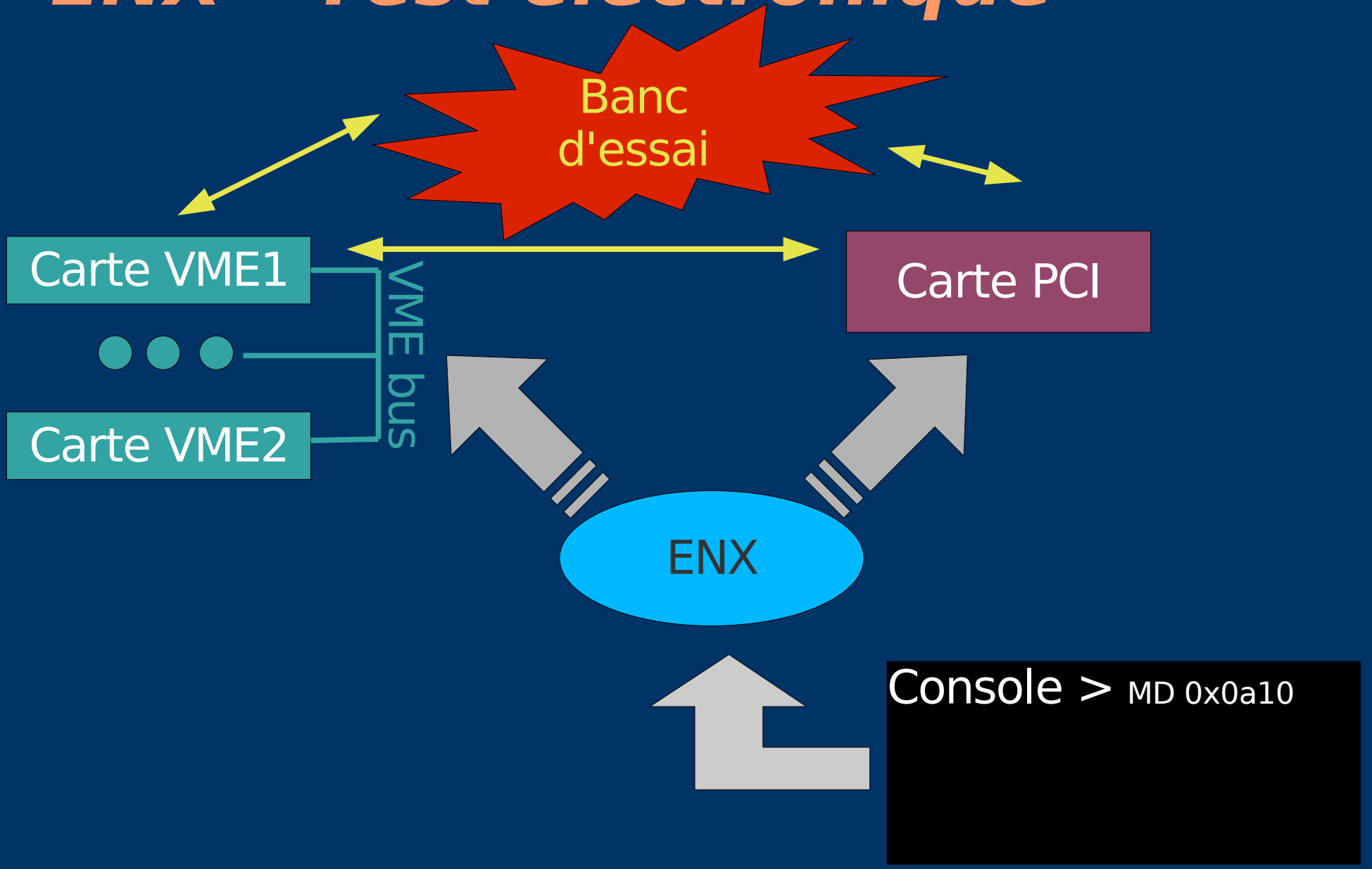
CSNSM – Informatique

- 2 ingénieurs développement
- Développement interne:
  - Mis a jour d'une microsonde isotopique
  - Déploiement d'un outil de scanning
  - Aide au service électronique
- Développement externe:
  - AGATA
  - NARVAL

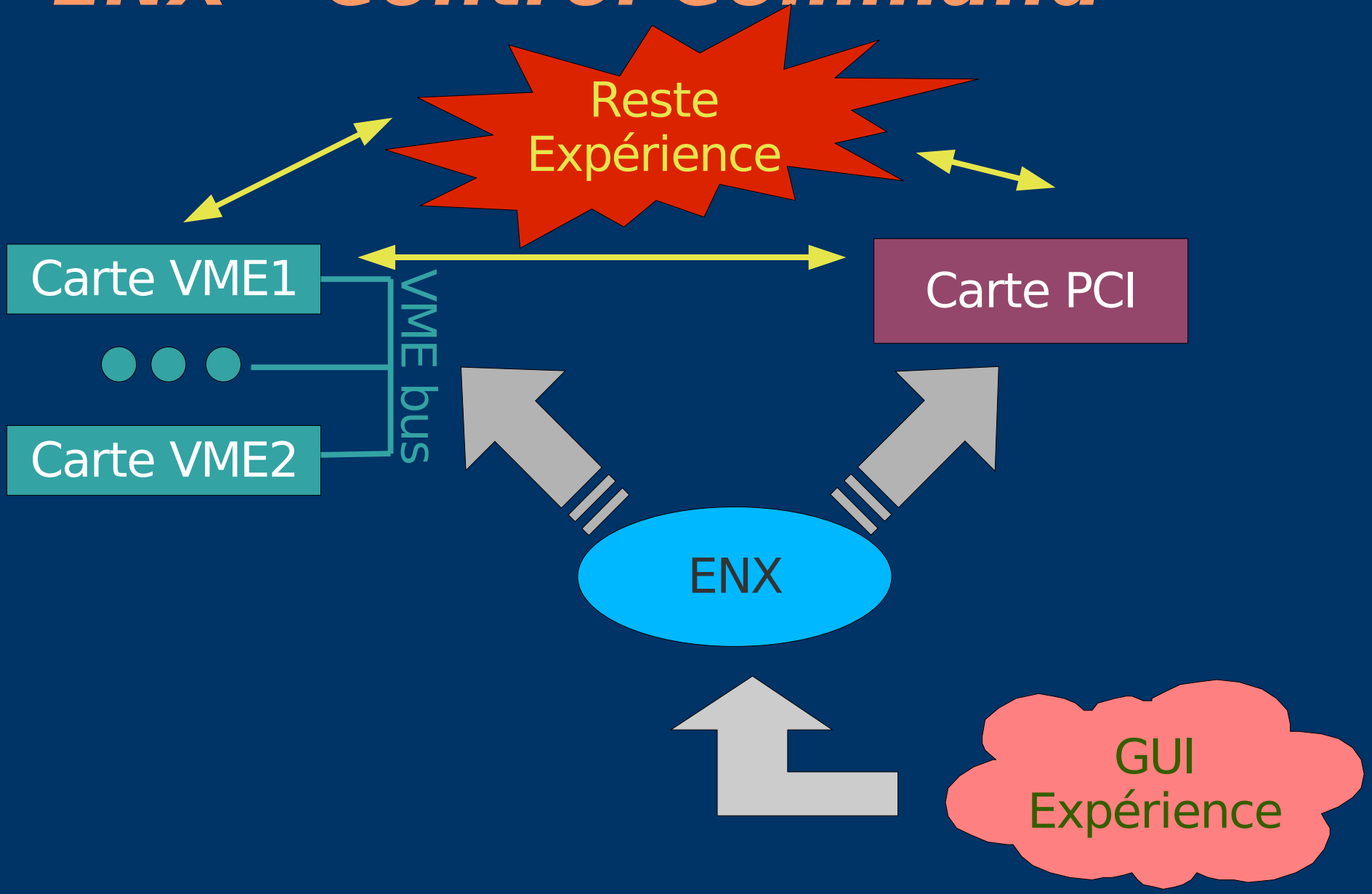
Exemples de projets

- Développements
  - ENX – Interface avec l'électronique
    - Besoins
    - Solution
  - NARVAL – Système d'acquisition

ENX – Test électronique



ENX – Control Command

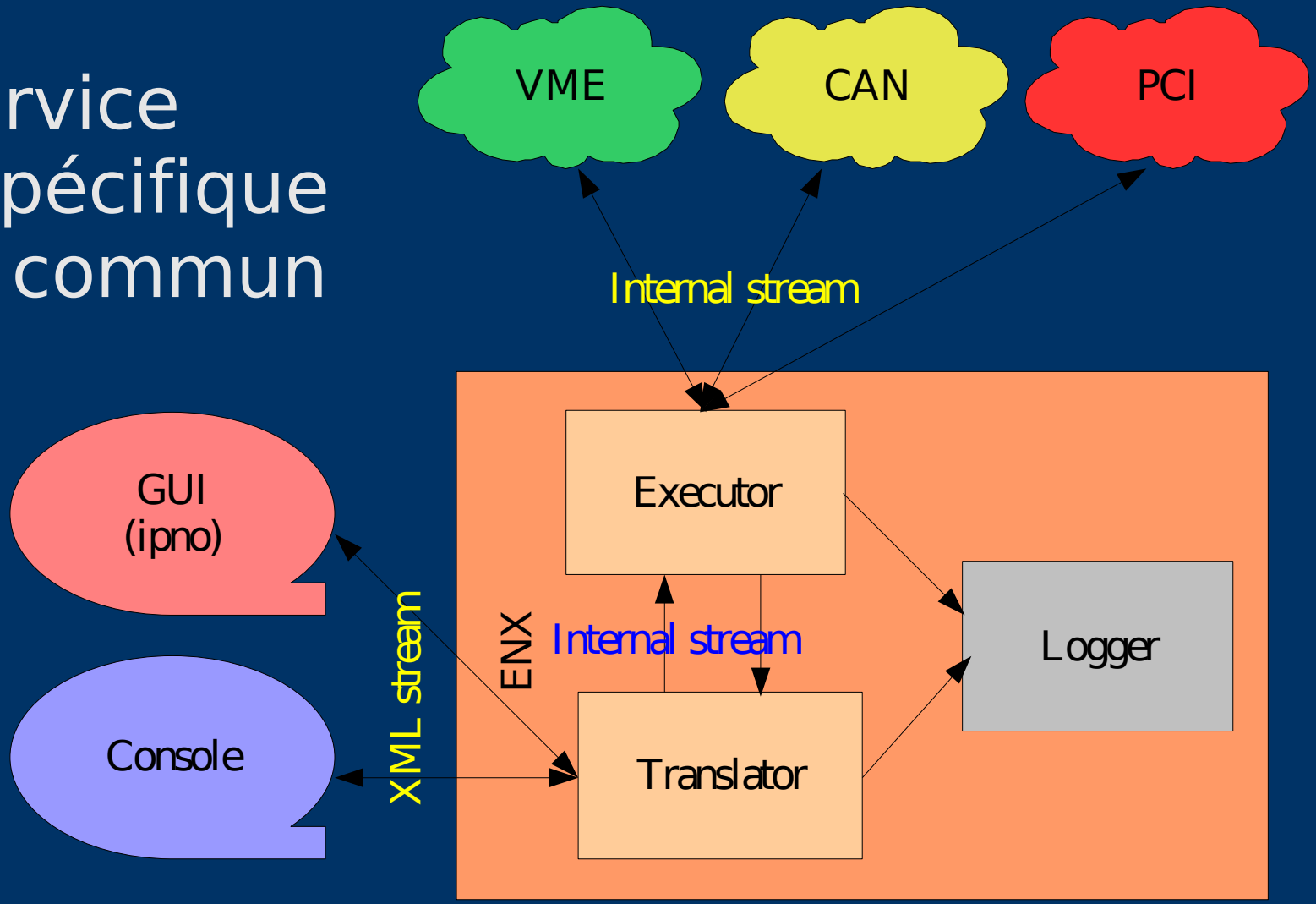


ENX – Spécifications techniques

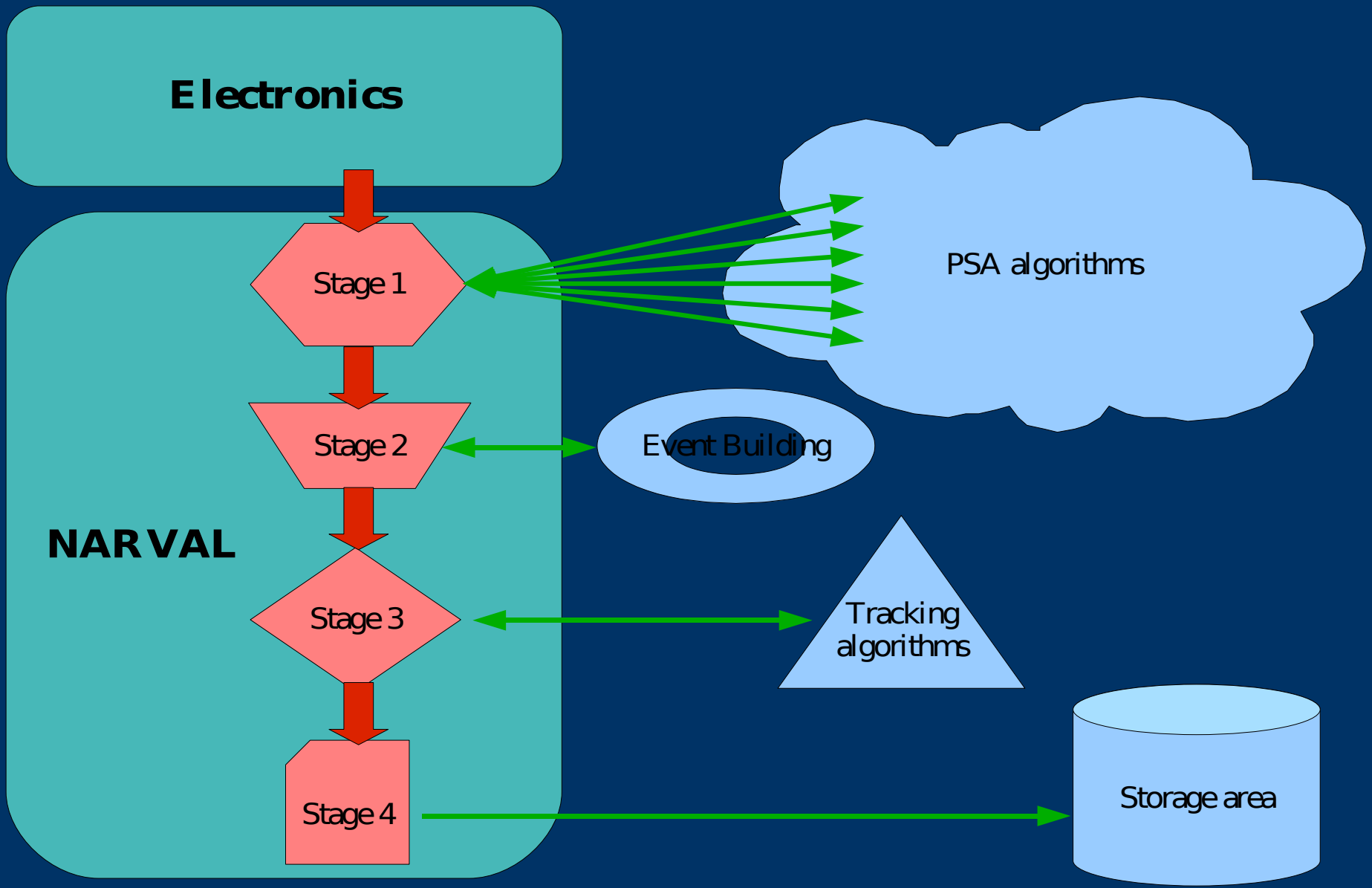
- Besoin commun:
  - Pas de contraintes de temps fortes
  - Adressage bus différents (CAN, VME, ...)
  - Travail coopératif entre différents bus
- Spécifique *Test*
  - Répétition commandes
- Spécifique *Control Command*
  - Automatisation de série d'actions

ENX - Solution en développement

- Web Service
- Client spécifique
- Drivers commun



NARVAL – Exemple AGATA



Ada

- Ada
  - La norme
  - Le langage
  - Exemples
    - Manipulation de donnée bas niveau
    - Boucles

Ada – La norme

- Première Norme ISO : 1983
  - dernier standard : 1995
    - POO
    - Programmation distribuée
  - dernière révision : 2005
    - Interface à la Java
    - Notation post-fixé
  - Norme gratuite
- Compilateur
  - Conforme à la norme
  - GNAT basé sur GCC – Source disponible

Ada – Le langage

- Idées:
  - Facilité de lecture privilégié sur l'écriture
  - Robustesse du code
  - Avoir un langage indépendant du compilateur et de sa version
- Techniques:
  - Langage très fortement typé
  - Taches gérées en natif dans le langage
  - Langage orienté objet
  - Programmation distribué en natif

Ada - Besoins pour nos développements

- Serveur service web
  - Outil de déploiement automatique créé par M. Pascal Obry (France Télécom)
- Interface avec des modules externes
  - Interface prévue avec C, C++, Fortran, Cobol
- Code portable et robuste
  - GNAT est porté sur la majorité des architectures actuelles
  - Robustesse d'un langage utilisé dans les centrales nucléaires et les avions

Ada - Exemple : Registre

```
type Reg_Status is
   record
      Busy : Boolean;
      LAM : Boolean;
      Dmd : Boolean;
      Error : Boolean;
      MSG : Boolean;
      DErr : Boolean;
      Rdy : Boolean;
      ByPass : Boolean;
   end record;
```

Ada - Exemple : Registre

```
type Reg_Status is
   record
      Busy : Boolean;
      ...
      ByPass : Boolean;
   end record;
```

```
for Reg_Status use
   record
      Busy at 0 range 0 .. 0;
      LAM at 0 range 1 .. 1;
      Dmd at 0 range 2 .. 2;
      Error at 0 range 3 .. 3;
      MSG at 0 range 4 .. 4;
      DErr at 0 range 5 .. 5;
      Rdy at 0 range 6 .. 6;
      ByPass at 0 range 7 .. 7;
   end record;
```

Ada - Exemple : Registre

```
type Reg_Status is
  record
    Busy : Boolean;
    ...
    ByPass : Boolean;
  end record;
```

```
for Reg_Status use
  record
    Busy at 0 range 0 .. 0;
    ...
    ByPass at 0 range 7 .. 7;
  end record;
```

```
function Is_Busy (Reg:Unsigned_8) return boolean is
  function To_Reg_Status is new
   Ada.Unchecked_Conversion(Unsigned_8,Reg_status);
begin
  return To_Reg_Status(Reg).Busy;
end Is_Busy;
```

Ada - Exemple : for

```
type tableau is range 0 .. Calcule(i);
...
Tab1, Tab2: tableau;
...

B_1:
  for I in tab1'RANGE loop
    for J in tab2'RANGE loop
      --  traitement...
      exit B_1 when Condition;
      --  traitement...
    end loop;
  end loop B_1;
```

- Déclaration d'un type Borne calculé dynamiquement

Ada - Exemple : for

```
type tableau is range 0 ..
Calcule(i);
...
Tab1, Tab2: tableau;
...

B_1:
  for I in tab1'RANGE loop
    for J in tab2'RANGE loop
      --  traitement...
      exit B_1 when Condition;
      --  traitement...
    end loop;
  end loop B_1;
```

- Parcours de 2 tableaux

Ada - Exemple : for

```
type tableau is range 0 ..
Calcule(i);
...
Tab1, Tab2: tableau;
...

B_1:
  for I in tab1'RANGE loop
    for J in tab2'RANGE loop
      --  traitement...
      exit B_1 when Condition;
      --  traitement...
    end loop;
  end loop B_1;
```

- Sortie de boucle Possible au milieu du traitement.
Sortie sur condition

- Etiquette permettant de sortir des deux boucles

Conclusion

- Pourquoi Ada?
  - Raison historique
  - Robustesse
  - Facilité de relecture du code
- Pourquoi pas Ada?
  - Communauté IN2P3 peu importante
    (mais grandissante : GANIL, CSNSM, IPNO, LPC Caen, ...)
  - Environnement de travail
- A venir:
  - 2nde formation Ada IN2P3