

# Modélisation d'objets et d'applications OpenMASK

**Y.Bekkers & B.Chanclou**  
**B.Lamarche & N.Robin**

# Plan

- **Objectifs**
- **Description**
  - **Conception**
  - **Utilisation**
- **État d'avancement**
- **Conclusion**

# Objectifs du projet

- **Utiliser des standards de programmation et accélérer le développement des objets**
- **Référencer et centraliser les objets pour les réutiliser**
- **Pour au final**
  - **Se concentrer sur « l'intelligence » des objets**
  - **Laisser la machine faire la partie « idiote »**
  - **Mettre en valeur la recherche et non le développement**

# Objectifs (Les outils)

- **Fournir des outils de gestion**
  - base d'objets
  - documentation
- **Fournir des outils de modélisation**
  - objets
  - communications entre objets
  - application
- **Fournir des outils de génération**
  - projet (code+makefile)
  - éditeur de configuration
  - documentation

# Description (Technologies utilisées)

- **Modélisation XML**
- **Plugins Eclipse**

**Ces deux technologies permettent**

- **de générer des éditeurs de modèles rapidement et efficacement**
- **d'utiliser des feuilles de style XSLT pour la génération du toute sorte de code**
- **Une grande facilité d'évolution et de maintenance**

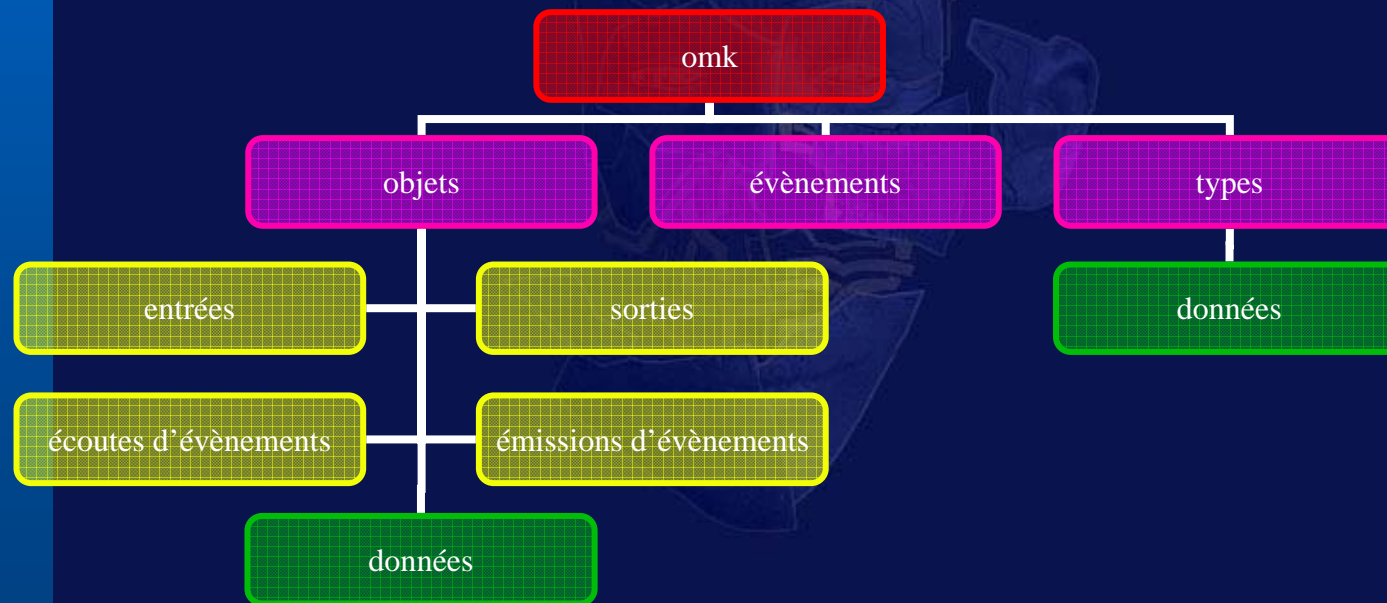
# Description (Conception)

## En gros

- On crée une description xsd du méta modèle (*i.e.* langage de description des objets OpenMASK)
- On génère un plugin pour Eclipse fournissant un éditeur du schéma xml du modèle
- On couple le tout à des générateurs de codes (sources, makefile, configuration,...)

# Description (Structure)

## Structure simplifiée du modèle



# Comment l'utiliser (Créer)

- **Création des objets OpenMASK**
  - définir des types de données (*PsType*)
  - définir des évènements (identifiant et type)
  - définir les objets (*PsSimulatedObject*)
- ajouter aux objets
  - des données
  - des entrées
  - des sorties
  - l'écoute d'évènements
  - l'émission d'évènements

# Comment l'utiliser (Générer)

- **Génération automatique du code C++**
  - identifiants d'évènements
  - types de données
  - objets (base + impl)
  - *main()* de l'application
- **Services fournis par les objets**
  - la documentation
  - la réinitialisation des données
  - la lecture de paramètres de configuration (données & connexions)
  - les accesseurs des types de données
  - les *call backs* d'évènements et le *compute()* des objets

# Comment l'utiliser (Compléter)

- **Complétion du code d'implémentation**

La génération automatique du code fournit le squelette de la classe d'implémentation des objets

- Écriture des *call backs* de chacun des évènements écoutés
- Écriture du *compute()*

⇒ **Compilation exécution**

# État d'avancement (C'est fait)

- **Le modèle d'objet incluant**
  - les entrées/sorties
  - les données
  - les évènements
- **Les plugins Eclipse**
  - documentation
  - accès à la base de données
  - éditeur du modèle d'application OpenMASK
  - générateur du code c++ de l'application
- **L'application de génération de code**
  - ligne de commande intégrable dans un makefile

## État d'avancement (C'est en cours)

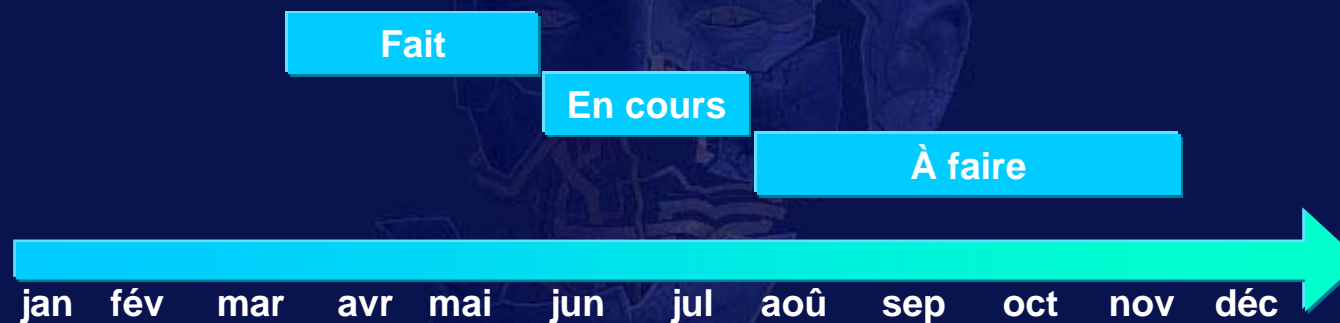
- **La gestion des modèles dans la base de données**
- **Le plugin de l'éditeur de fichier de configuration**
- **Le loader xml du fichier de configuration associé**

# État d'avancement (C'est à faire)

- **Compléter le modèle et le code généré avec de nouveaux services**
  - la visualisation
  - l'interactivité
  - ...?
- **Les générateurs pour**
  - Les projets : Visual, KDevelop, Makefile, ...?
- **La cohabitation dans la base des objets du code généré et édité ...?**

# État d'avancement (Calendrier)

- **En gros**



# Conclusion

- **L'utilisateur est heureux**
  - Il peut plus facilement concevoir la structure de ses objets
  - Il a beaucoup moins de code à écrire
  - Il peut réutiliser ce que d'autres ont fait car
    - il y a une documentation
    - le format des objets est plus standard
  - Il peut proposer ses objets aux autres sans se donner plus de travail

# Conclusion (Questions, remarques)

- **Si vous avez des questions ou remarques**
  - **Yves.Bekkers@irisa.fr**
  - **Benoit.Chanclou@irisa.fr**