DE LA RECHERCHE À L'INDUSTRIE

ROADMAP ARCANE

05 mars 2020

Stéphane De Chaisemartin – Gilles Grospellier

Plan

Déroulé

- **Retour d'expérience**
  - Etats des lieux
  - Pratique de développements et d'intégration
  - Echanges entre les équipes IFPEN et CEA
  - Difficultés et axes d'amélioration

- **Evolution d'Arcane au cours de l'avenant n°4 (2017-2020)**
  - Gestion du maillage
  - Invasion d'Alien
  - Nouvelle structure Arccon, Arccore, Axlstar

- **Directions de travail pour le futur d'Arcane, préparation du prochain avenant (2021-2024)**
  - Adaptation aux nouvelles architectures matérielles
  - Outils d'aide au développement
  - Aspects Génie Logiciel
  - Mise en Open Source

Retour d’expérience

Retour d'expérience

Etats des lieux

- **Arcane existe depuis 2000**
- **Collaboration CEA/IFPEN depuis 2007**
  - Quatre avenants de collaboration (n°4 2017-2020)

- **Utilisation au CEA**
  - Livraison uniquement sur Linux (CentOS 7 actuellement, CentOS 8 courant 2020)
  - Utilisation sur 2 logiciels de calcul
    - Entre 10 et 15 développeurs chacun
    - En général quelques centaines de processus MPI
    - Tests jusqu'à 16000 cœurs sur de vrais cas tests
  - En cours d'évaluation par le CEA/DEN (voir exposé spécifique)

- **Utilisation à l'IFPEN**
  - Utilisation dans 6 logiciels de calcul
    - Trois calculateurs branchés dans des codes industriels diffusés par notre filiale BeicipFranlab
      - Logiciels TemisFlow™ et DionisosFlow™
    - Deux calculateurs diffusés via des contrats de collaboration avec des industriels (JIP)
      - Logiciels CooresFlow™et Cats
  - Livraisons sur Linux (CentOS 6 et 7) et Windows (7 et 10)

Pratique de développements et d'intégration

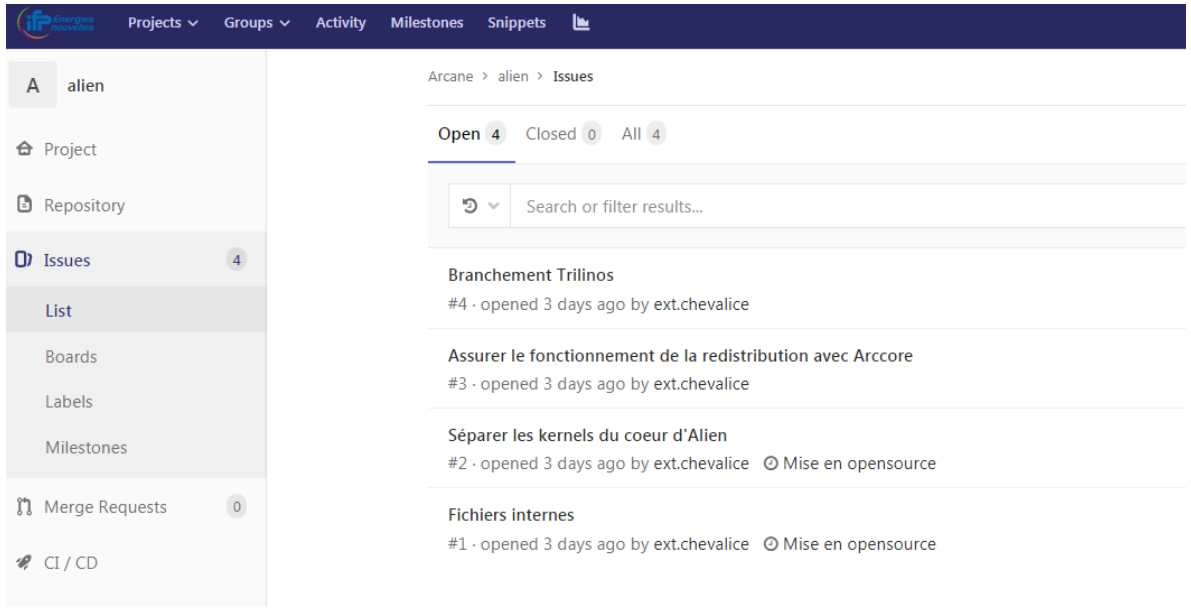
- **CEA**
  - Déploiement des bibliothèques système (MPI, compilateurs, débuggers, …) via Easybuild
  - Déploiement manuel par les développeurs de chaque outil/bibliothèque (solveur linéaires, python, hdf5, metis, mono, …)
  - En cours, déploiement de l'environnement développeur via 'spack' (LLNL)
  - Utilisation de l'outil Trollnet pour l'intégration, la vérification et la validation des codes
    - 7500 cas séquentiels, 6000 cas parallèles joués quotidiennement
    - Comparaison bit à bit ou via des courbes par rapport à une référence.

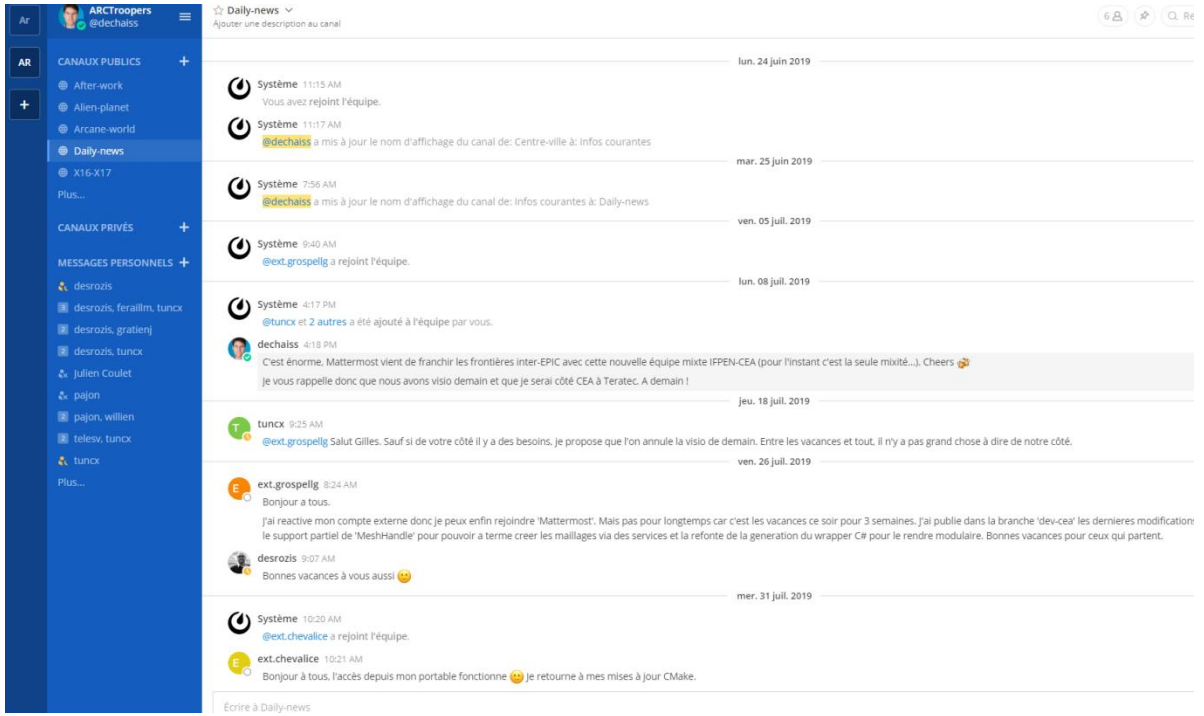
- **IFPEN**
  - Déploiement sous Linux via Easybuild
  - Outils DailyTest pour intégration continue : suivi des commits, basés sur validateurs, Linux uniquement
  - Utilisation de Jenkins
    - Test Windows et Linux
    - Livraison de version Windows et Linux
  - Utilitaire de mise à jour des .arc pour l'intégration continue

Retour d'expérience

Echanges entre les équipes IFPEN et CEA

- Visio tous les 15 jours
- Journée de travail à Teratec tous les 15 jours
- Migration des sources sur un serveur Gitlab commun
- Utilisation de l'outil d'échange Mattermost (type slack)





Difficultés et axes d'amélioration

- **Nécessité de faire évoluer la plateforme en restant compatible avec l'existant**
  - Mode compatibilité durant un certain temps pour que les applications puissent évoluer
  - Maintenir plusieurs versions de Arcane nécessite beaucoup de ressources
  - => Nécessité pour les utilisateurs d'utiliser la dernière version si possible et de supprimer toutes les méthodes signalées comme obsolètes.

- **Nécessité de s'adapter aux nouvelles architectures**

- **Constat sur le développement: difficile de tester en production les évolutions**
  - pas d'accès au code source des logiciels utilisateurs CEA/IFPEN
  - pas le même environnement (Windows/Linux)

- **En 20 ans, les techniques de développement ont beaucoup évolué**
  - Méthodes agiles
  - Intégration continue
  - Livraison fréquente de versions
- **Il est nécessaire d'être plus réactif**

Evolution d’Arcane au cours de l’avenant n°4 (2017-2020)

Plan

Déroulé

- **Retour d'expérience**
  - Etats des lieux
  - Pratique de développements et d'intégration
  - Echanges entre les équipes IFPEN et CEA
  - Difficultés et axes d'amélioration

- **Evolution d'Arcane au cours de l'avenant n°4 (2017-2020)**
  - Gestion du maillage
  - Invasion d'Alien
  - Nouvelle structure Arccon, Arccore, Axlstar

- **Directions de travail pour le futur d'Arcane, préparation du prochain avenant (2017-2020)**
  - Adaptation aux nouvelles architectures matérielles
  - Outils d'aide au développement
  - Aspects Génie Logiciel
  - Mise en Open Source

Evolution d'Arcane au cours de l'avenant n°4 (2017-2020)

Gestion du maillage

- **Au départ, Arcane ne supportait que des maillages non structurés**
- **Au fil du temps, de nouveaux besoins ont émergé**
  - AMR
  - Maillages cartésiens
  - Maillage non conformes
  - Maillage génériques (DualNode, Link)
  - Sous-maillages

- **Ces différentes fonctionnalités ont été ajoutés à la structure existante**
  - Gestion du maillage plus complexe
  - Pas d'optimisation spécifique possible
  - Mélange entre types de maillages pas forcément possible (ex: maillages non conformes et génériques) mais pas explicitement interdit.

- **Nécessite de revoir les fondations du maillage**

Gestion du maillage : nouveaux mécanismes de création du maillage

- **Modifications:**
  - Le type de maillage créé sera dépendant du code
  - Le maillage ne sera plus créé avant les services/modules
- **Problème: le maillage (`IMesh*`) doit être créé avant le service/module l'utilisant**
- **Nouvelle class `MeshHandle` permettant de référencer un maillage pas encore créé**
- **Les services et modules doivent utiliser '`MeshHandle`' au lieu de '`IMesh`' dans le constructeur**

```
VariableCellReal m_cell_temperature; // Variable déclarée dans le .h

MyModule::MyModule(const ModuleBuildInfo& mbi)
: cell_temperature(VariableBuildInfo(mbi.mesh(),"CellTemperature"))          // AVANT
: cell_temperature(VariableBuildInfo(mbi.meshHandle(),"CellTemperature"))  // APRÈS
```

- **Changement dans le jeu de données: <maillages/maillage> au lieu de <maillage>**

- **Le fonctionnement actuel (<maillage> restera opérationnel)**

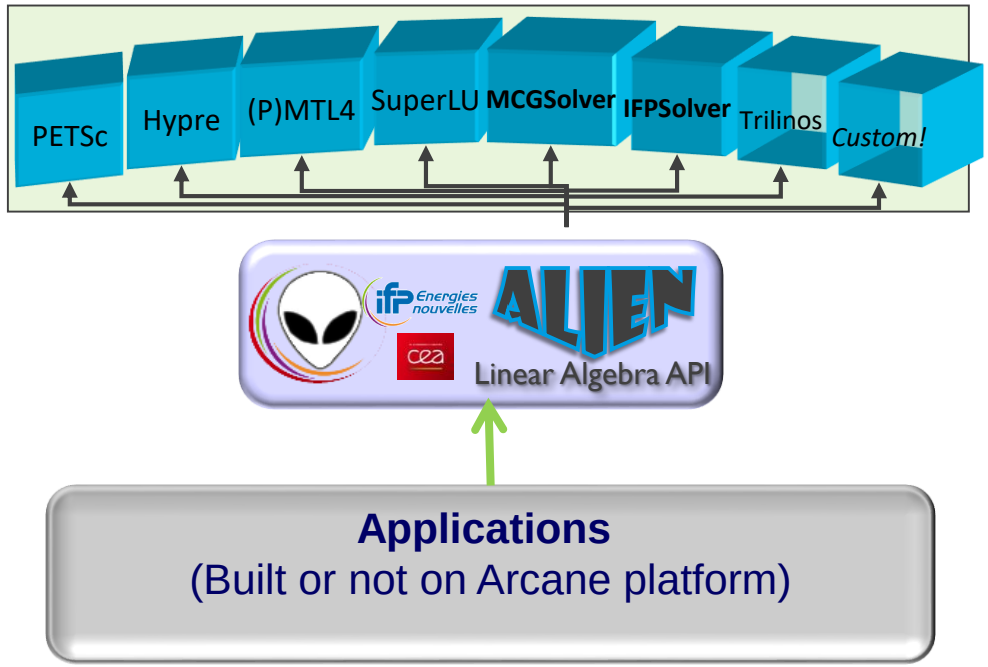
Evolution d'Arcane au cours de l'avenant n°4 (2017-2020)

Gestion du maillage : changement d'accès aux connectivités

- **Découplage des classes des entités (Cell, Node, …) de l'accès à la connectivité qui sera portée par le maillage:**
  - cell.node(0) => mesh.node(cell,0)
  - cell.face(0) => mesh.face(cell,0)
- **À terme, les classes de bases du maillage n'auront plus accès à la connectivité**
  - Des classes spécialisées par maillage pourront autorisé cet accès: par exemple une classe 'UnstructuredCell'
- **Réflexions en cours sur la possibilité d'avoir plusieurs représentation sur un maillage**
  - Par exemple, vue non structurée d'un maillage cartésien

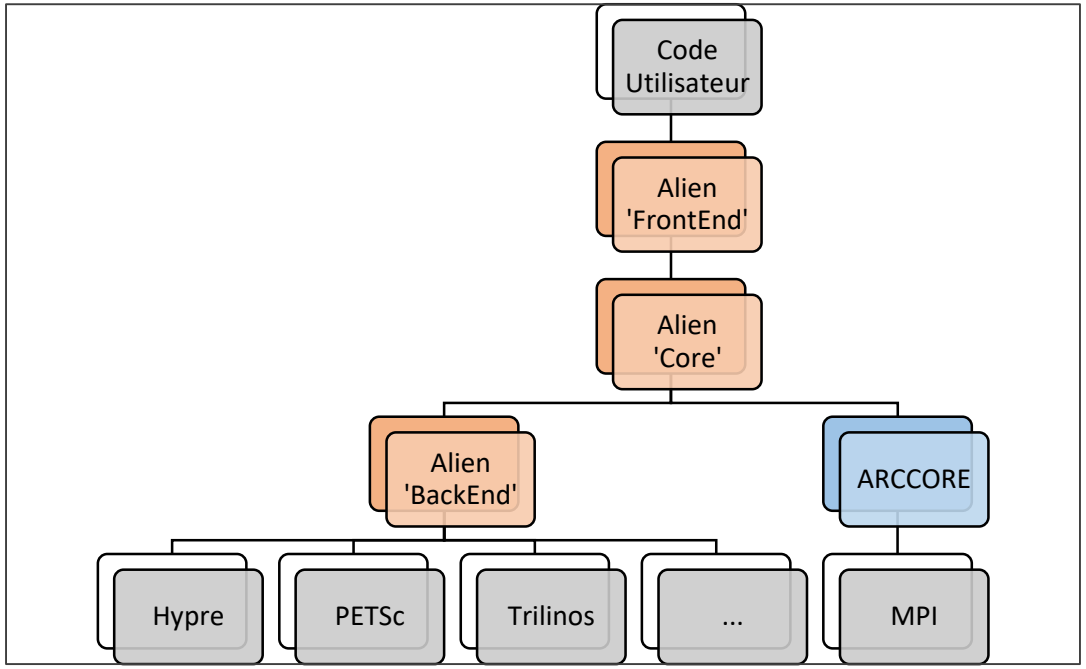
Invasion d'Alien : nouvelle interface d'algèbre linéaire

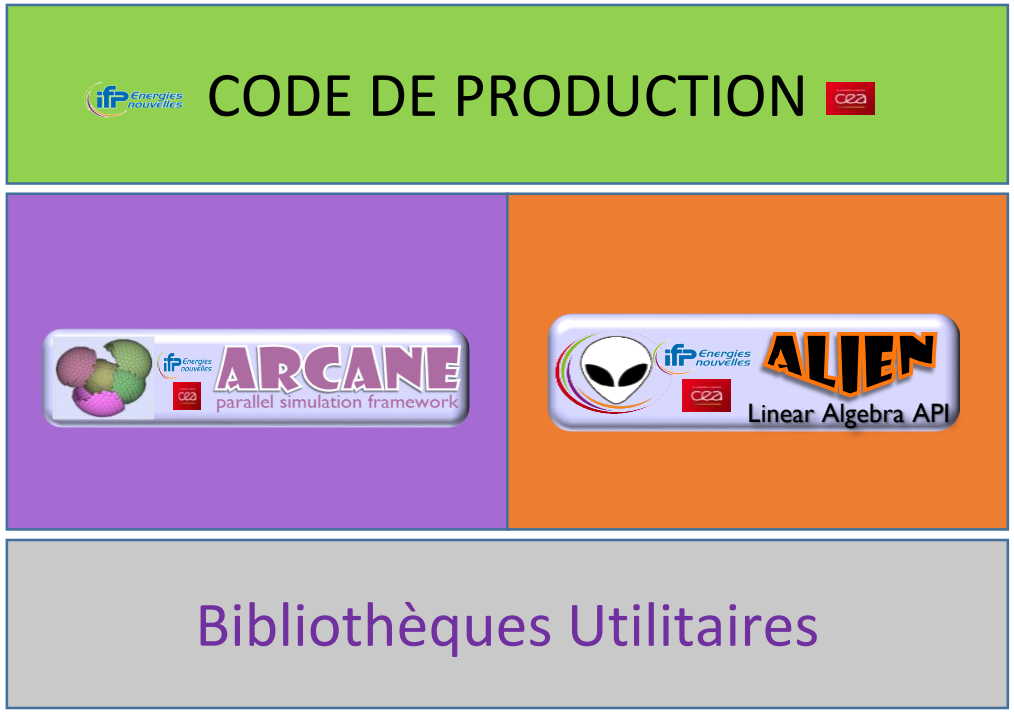
- **Une interface unique pour tous les solveurs linéaires**
  - Pour les applications basée sur **Arcane**
  - Ou pas…
- **Rationalisation des efforts pour brancher des solveurs linéaires dans une application**
  - Accès via un unique point d'entrée aux principales bibliothèques de la communauté
  - Mise à disposition en interne des derniers travaux HPC IFPEN sur les solveurs (GPU…)



Nouvelle structure Arccon, Arccore, Axlstar

- Mutualisation d'utilitaires
- Types de données
- Structures de bases du parallélisme (échange de message et vectorisation)
- Utilitaires d'affichage





Direction de travail pour le futur d'Arcane, préparation du prochain avenant (2021-2024)

Plan

Déroulé

- **Retour d'expérience**
  - Etats des lieux
  - Pratique de développements et d'intégration
  - Echanges entre les équipes IFPEN et CEA
  - Difficultés et axes d'amélioration

- **Evolution d'Arcane au cours de l'avenant n°4 (2017-2020)**
  - Gestion du maillage
  - Invasion d'Alien
  - Nouvelle structure Arccon, Arccore, Axlstar

- **Directions de travail pour le futur d'Arcane, préparation du prochain avenant (2021-2024)**
  - Adaptation aux nouvelles architectures matérielles
  - Outils d'aide au développement
  - Aspects Génie Logiciel
  - Mise en Open Source

Le futur d'Arcane, prochain avenant (2021-2024)

Adaptation aux nouvelles architectures matérielles

- **Architectures**
  1. Vectorisation
  2. Augmentation du nombre de cœurs de calcul par nœud
  3. Processeurs hétérogènes (GPGPU)

- **Pour les deux premiers points, Arcane dispose déjà de mécanismes permettant d'utiliser ces fonctionnalités**
  - ces existantes pour le HPC (vectorisation, multi-threading) pas forcément utilisées

- **L'expérience montre que cela reste compliqué à utiliser**

- **En général, les priorités dans le développement logiciel sont mises sur l'aspect fonctionnel et l'aspect HPC est souvent secondaire**

Le futur d'Arcane, prochain avenant (2021-2024)

Adaptation aux nouvelles architectures matérielles

- **Pour exploiter les nouvelles architectures, nos homologues américains ont développé de nouvelles plateformes logicielles**
  - Kokkos (Sandia National Labs)
  - RAJA (Lawrence Livermore National Labs)

- **Points communs de ces 2 plateformes**
  - Elève le niveau d'abstraction et rend opaque l'implémentation (pas d'appel OpenMP visible dans le code)
  - Exclusivement C++
  - Ne prend en charge que l'aspect mono nœud (pas de MPI)
  - Utilisation massive des fonctionnalités récentes du C++ (lambda fonction template)
  - => **accroit la complexité de codage, augmente les temps de compilation et réduit la lisibilité du code**
  - Sur GPGPU, nécessité de gérer plusieurs zones mémoires
  - Tentative d'homogénéiser les mécanismes de gestion de la concurrence (multi-threading) et GPGPU via un mécanisme de boucle

Exemple: boucles concurrentes

Formes de boucles similaires entre Arcane/Kokkos/RAJA

ARCANE

```
Parallel::Foreach(allCells(),[=](CellVectorView cells){
  ENUMERATE_CELL(icell,cells){
    sound_speed[icell] = sqrt(adiabatic_cst[icell]*pressure[icell]/density[icell]);
  } });
```

```
Parallel::ForeachItem(allCells(),[=](CellIterator cells){
    sound_speed[icell] = sqrt(adiabatic_cst[icell]*pressure[icell]/density[icell]);
});
```

KOKKOS

```
Kokkos::parallel_for (nb_cell, KOKKOS_LAMBDA (const int icell) {
   sound_speed[icell] = sqrt(adiabatic_cst[icell]*pressure[icell]/density[icell]);
  });
```

RAJA

```
RAJA::forall<exec_policy>(RAJA::RangeSegment(0, N), [=] (int icell) {
 sound_speed[icell] = sqrt(adiabatic_cst[icell]*pressure[icell]/density[icell]);
});
```

Le futur d'Arcane, prochain avenant (2021-2024)

Adaptation aux nouvelles architectures matérielles

- Il sera de plus en plus difficile en C++ de cacher l'architecture matérielle sous jacente
- L'utilisation de DSL (Nablab) permet cette séparation entre l'aspect métier (numérique) et informatique.

Le futur d'Arcane, prochain avenant (2021-2024)

Adaptation aux nouvelles architectures matérielles : STRATÉGIE RETENUE

- **Développer des mini-applications open-source utilisant Arcane**
  - Hydrodynamique explicite (similaire à Pennant)
  - Trajectographie Monte Carlo (similaire à Quicksilver)
  - Code utilisant un solveur linéaire (pour tester aussi Alien)
- **Utiliser ces applications comme démonstrateurs pour le support du multi-threading et des GPGPU**
- **Collaboration possible avec les constructeurs (Atos) et/ou le milieu académique (ECR).**

Le futur d'Arcane, prochain avenant (2021-2024)

Outils d'aide au développement : pistes de travail

- **Ajout de mécanismes de profiling**
  - Trace des appels MPI (déjà existant) : OTF2, JSON
  - Support pour des évènements utilisateurs
- **Outils de traçage de l'évolution des performances dans les codes au cours des versions**
- **Outil de debug avancé (à la HyODA)**
- **Aide au développement pour le multi-threading**
  - Détection des accès non thread-safe dans les variables
  - Faciliter l'écriture de tests notamment pour le multi-threading

Outils d'aide au développement : Utilisation Directe

- **Objectif**
  - Pouvoir utiliser Arcane plus facilement qu'avec la configuration actuelle qui nécessite une boucle en temps, un fichier de configuration, des AXL, …

```
int main(int argc,char* argv[])
{
  CommandLineArguments cmd_line_args(&argc,&argv);
  ArcaneSimpleExecutor simple_exec;
  simple_exec.initialize(cmd_line_args);
  ISubDomain* sd = simple_exec.createSubDomain();
  MeshReaderMng mrm(sd);
  IMesh* mesh = mrm.readMesh("Mesh1","sod.vtk");
  ENUMERATE_CELL(icell,mesh->allCells()){
    …
  }
}
```

Le futur d'Arcane, prochain avenant (2021-2024)

Outils d'aide au développement : Utilisation du C#

- **Deux modes prévus**
  - Utilisation simple pour des plugin.
  - Réalisation directe de modèles numériques en C#

- **Voir exposé correspondant**

Le futur d'Arcane, prochain avenant (2021-2024)

Aspects Génie Logiciel : Objectifs de la modularité

- **Pouvoir réutiliser certaines parties de Arcane en dehors de Arcane**
  - déjà fait avec Arccore pour Alien
- **Pouvoir plus facilement faire évoluer certains composants**
  - Maillages
  - Variables
  - Services
  - Mécanismes d'échange de message (MPI, mémoire distribuée)
- **Simplifier les tests**
- **Cela nécessitera des modifications dans les codes**
  - Par exemple certaines classes n'auront plus forcément accès au ISubDomain
  - Modifications dans les fichiers d'en-tête.

Le futur d'Arcane, prochain avenant (2021-2024)

Mise en Open Source

- **Pourquoi ?**
  - Besoin de simplification des développements
  - Difficultés pour tester les nouvelles versions de Arcane en environnement CEA/IFPEN

- **La mise en Open Source permet de**
  - Simplifier les développements via l'accès à un ensemble de ressources uniques accessibles depuis Internet
  - Faciliter les éventuelles collaborations et le travail des stagiaires
  - Montrer aux constructeurs les mécanismes utilisés par Arcane afin de les évaluer/améliorer
  - Augmenter/Améliorer les tests via l'utilisation de ressources machines gratuites

- **Le CEA et l'IFPEN sont d'accord pour mettre Arcane sous licence Open Source**
  - Dans un premier temps, les sources ne seront pas accessibles de manière publique

- **Objectifs**
  - Acter la mise en open source lors de la rédaction du prochain avenant en **septembre 2020**

Conclusion

DÉVELOPPEMENTS PRÉVUS

- **2020**
  - Finalisation avenant 2017-2020
  - Préparation avenant 2021-2024
  - Préparation mise en open source de Arcane
- **2021**
  - Structures de maillages spécifiques (cartésien, polyédrique) et nouveaux mécanismes d'accès aux connectivités
  - Exploration des fonctionnalités du C++20 (concepts, modules)
- **2022**
  - Séparation de Arcane en plusieurs composantes à la manière de ce qui a été fait pour Arccore/Axlstar.
  - Première version de démonstration de faisabilité GPGPU fin 2022

Commissariat à l’énergie atomique et aux énergies alternatives - www.cea.fr

DE LA RECHERCHE À L'INDUSTRIE