CSiMoon : Calcul scientifique, méthodologie orientée objets et environnement : de l'analyse mathématique à la programmation

CSiMoon

Stéphane Labbé (1), Jacques Laminie (1) et Violaine Louvet (2)

(1) Université Paris 11, Laboratoire de Mathématique et (2) Université Lyon 1, Maply

Introduction

Introduction - Motivation - Philosophi

- Analyse des activités de calcul de l'équipe.
  - Les domaines d'applications : essentiellement E.D.P.

Introduction - Motivation - Philosophi

- Analyse des activités de calcul de l'équipe.
  - Les domaines d'applications : essentiellement E.D.P.
  - Les domaines de recherche : math. , méthode de résolution, discrétisation, algèbre linéaire, parallélisme, graphique, ...

Introduction - Motivation - Philosophi

- Besoins de l'équipe du point de vue recherche mais aussi enseignement.
  - recherche : voir + haut

Introduction - Motivation - Philosophi

- Besoins de l'équipe du point de vue recherche mais aussi enseignement.
  - recherche : voir + haut
  - enseignement : de la programmation procédurale vers l'objet, nécessité d'un modèle de programmation

Introduction - Motivation - Philosophi

- Besoins de l'équipe du point de vue recherche mais aussi enseignement.
  - recherche : voir + haut
  - enseignement : de la programmation procédurale vers l'objet, nécessité d'un modèle de programmation
  - Capitalisation des acquis

Donc, un modèle de programmation

- souple, réutilisable, facile à modifier, ...

Donc, un modèle de programmation

- souple, réutilisable, facile à modifier, ...
- permettant une séparation des domaines de recherche (équations, algorithmiques, discrétisations, systèmes linéaires, ...)

Donc, un modèle de programmation

- souple, réutilisable, facile à modifier, ...
- permettant une séparation des domaines de recherche (équations, algorithmiques, discrétisations, systèmes linéaires, ...)
- portable, auto-documenté, instrumenté, multi-langages

Donc, un modèle de programmation

- souple, réutilisable, facile à modifier, ...
- permettant une séparation des domaines de recherche (équations, algorithmiques, discrétisations, systèmes linéaires, ...)
- portable, auto-documenté, instrumenté, multi-langages

PROGRAMMATION OBJET

Programmation Orientée Objet pour l

- Une méthodologie de programmation orientée objet pour
  - ✓ la mise sur pied de bibliothèques propres aux besoins du laboratoire, modulaires et cohérentes,

Programmation Orientée Objet pour

- Une méthodologie de programmation orientée objet pour
  - ✓ la mise sur pied de bibliothèques propres aux besoins du laboratoire, modulaires et cohérentes,
  - ✓ donner aux étudiants un cadre de programmation pour améliorer leur apprentissage de l'informatique scientifique,

Programmation Orientée Objet pour

- Une méthodologie de programmation orientée objet pour
  - ✓ la mise sur pied de bibliothèques propres aux besoins du laboratoire, modulaires et cohérentes,
  - ✓ donner aux étudiants un cadre de programmation pour améliorer leur apprentissage de l'informatique scientifique,
  - ✓ une meilleure exploitation des compétences de chacun et une capitalisation du savoir-faire.

Programmation Orientée Objet pour

- Un environnement de programmation pour
  - ✓ disposer d'outils d'instrumentation, de couplage, de génération de documentation...

Programmation Orientée Objet pour

- Un environnement de programmation pour
  - ✓ disposer d'outils d'instrumentation, de couplage, de génération de documentation...
  - ✓ une meilleure exploitation et une meilleure rationalisation des moyens informatiques.

Environnement de programmation

Les fonctionnalités :

- ✓ Instrumentation des codes pour une exploitation graphique décentralisée,

Environnement de programmation

Les fonctionnalités :

- ✓ Instrumentation des codes pour une exploitation graphique décentralisée,
- ✓ Couplages de codes,

Environnement de programmation

Les fonctionnalités :

- ✓ Instrumentation des codes pour une exploitation graphique décentralisée,
- ✓ Couplages de codes,
- ✓ Contrôle interactif de l'exécution des codes,

Environnement de programmation

Les fonctionnalités :

- ✓ Instrumentation des codes pour une exploitation graphique décentralisée,
- ✓ Couplages de codes,
- ✓ Contrôle interactif de l'exécution des codes,
- ✓ Génération automatique de documentation.

Des mathématiques à la programmation

Problématique

- Problématique : comment donner une description des discrétisations compatible avec la programmation.

Discrétisation

Soit à résoudre : $\mathcal{P}u = f$, avec u dans W_1 et f dans W_2

$$W_1 \xrightarrow{\mathcal{P}} W_2$$

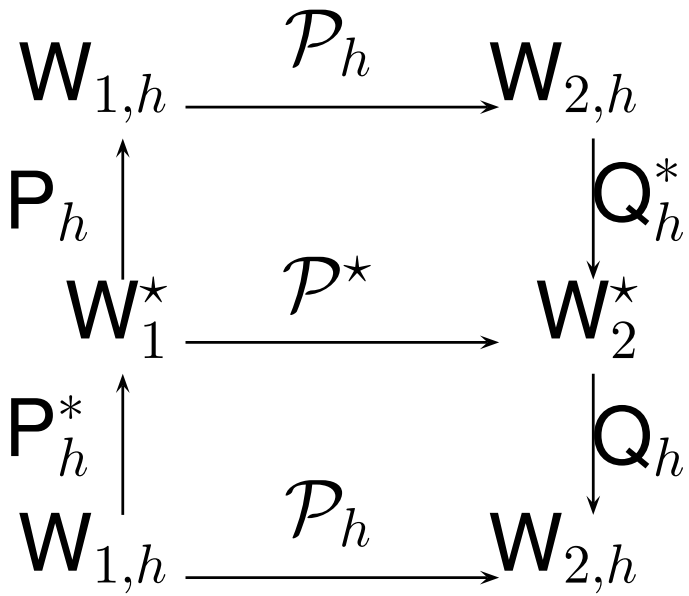
Discrétisation

Soit à résoudre : $\mathcal{P}^\star u = f$, avec u dans $\mathsf{W}_1^\star \supset \mathsf{W}_1$ et f dans $\mathsf{W}_2^\star \supset \mathsf{W}_2$, $\mathcal{P}^\star_{|\mathsf{W}_1} = \mathcal{P}$

$$\mathsf{W}_1^\star \xrightarrow{\mathcal{P}^\star} \mathsf{W}_2^\star$$

Discrétisation

Soit à résoudre : $\mathcal{P}_h u = \mathsf{Q}_h^* f$, avec u dans $\mathsf{W}_{1,h}$ et f dans W_2



Niveaux de discrétisation

- **Le niveau continu** : correspond à la représentation abstraite du problème indépendemment de la discrétisation.

Niveaux de discrétisation

- **Le niveau continu** : correspond à la représentation abstraite du problème indépendemment de la discrétisation.
- **Le niveau discret** : représentation de la discrétisation, en particulier des projecteurs.

Niveaux de discrétisation

- **Le niveau continu** : correspond à la représentation abstraite du problème indépendemment de la discrétisation.
- **Le niveau discret** : représentation de la discrétisation, en particulier des projecteurs.
- **Le niveau algébrique** : représentation des systèmes algébriques finaux en terme de manipulation de matrices et de vecteurs.

Découpage des niveaux

Le découpage s'effectue suivant les quatre types principaux d'objets

- Le problème : $\mathcal{P}u = f$ et ses héritiers,

Découpage des niveaux

Le découpage s'effectue suivant les quatre types principaux d'objets

- Le problème : $\mathcal{P}u = f$ et ses héritiers,
- Les opérateurs : $\mathcal{P}$ et ses héritiers,

Découpage des niveaux

Le découpage s'effectue suivant les quatre types principaux d'objets

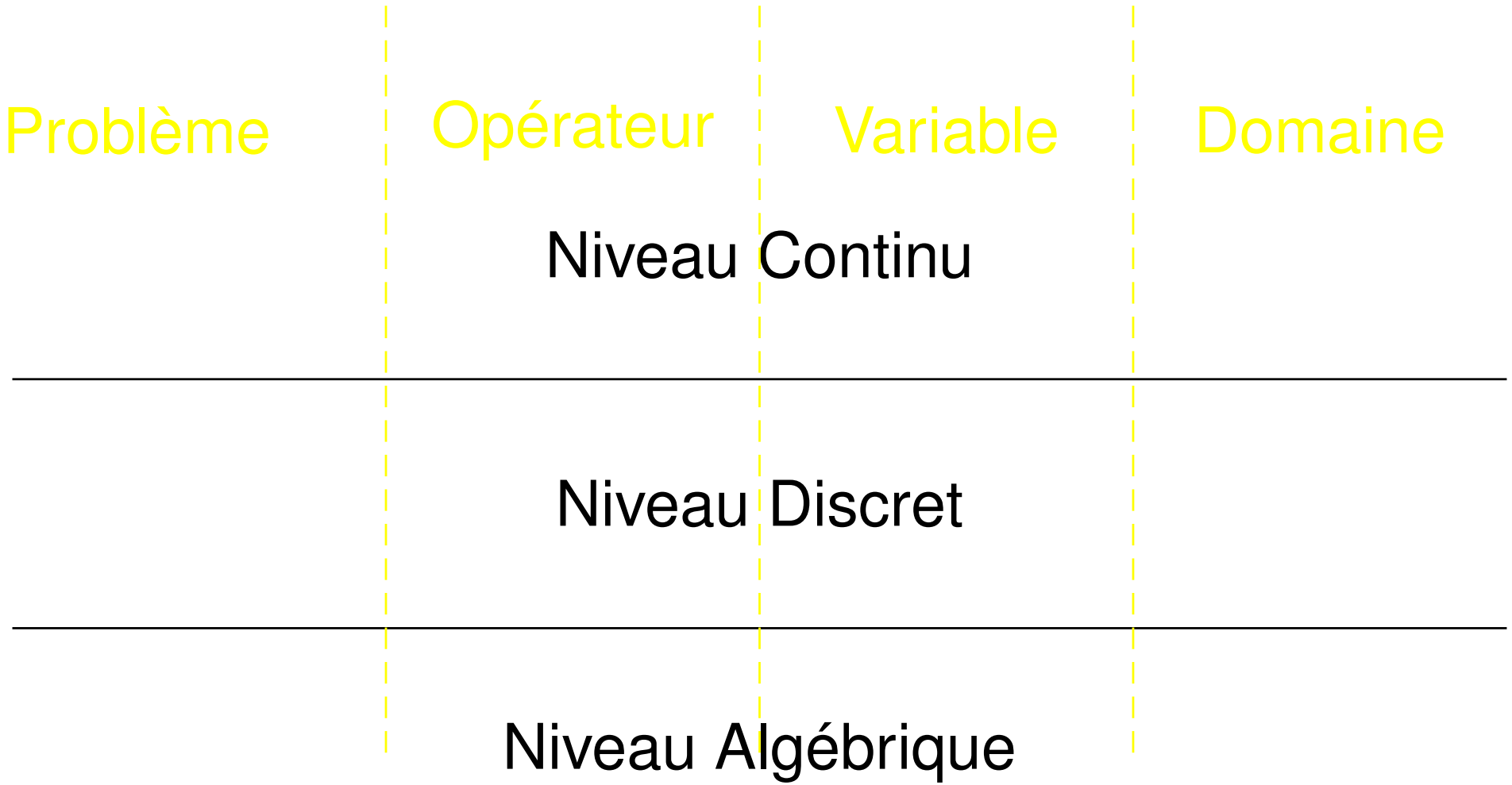
- Le problème : $\mathcal{P}u = f$ et ses héritiers,
- Les opérateurs : $\mathcal{P}$ et ses héritiers,
- Les variables : les W_1, W_2 et leurs héritiers,

Découpage des niveaux

Le découpage s'effectue suivant les quatre types principaux d'objets

- Le problème : $\mathcal{P}u = f$ et ses héritiers,
- Les opérateurs : $\mathcal{P}$ et ses héritiers,
- Les variables : les W_1, W_2 et leurs héritiers,
- Les domaines : les domaines sur lesquels sont contruits les espaces et leurs héritiers.

Découpage des niveaux



Le cahier des charges

Les contraintes

- Indépendance des niveaux de programmation pour chaque métier,

Le cahier des charges

Les contraintes

- Indépendance des niveaux de programmation pour chaque métier,
- Contrôle descendant des objets,

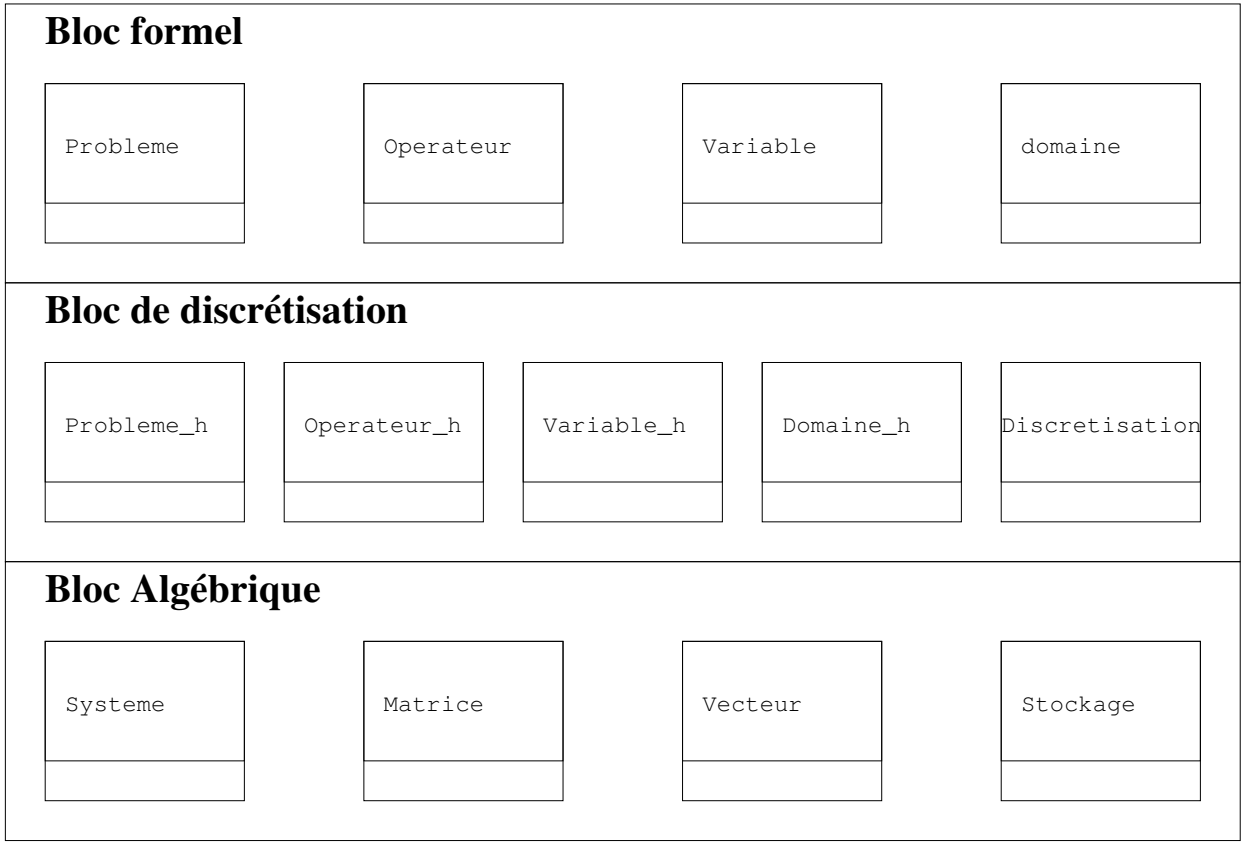
Le cahier des charges

Les contraintes

- Indépendance des niveaux de programmation pour chaque métier,
- Contrôle descendant des objets,
- Modularité des objets.

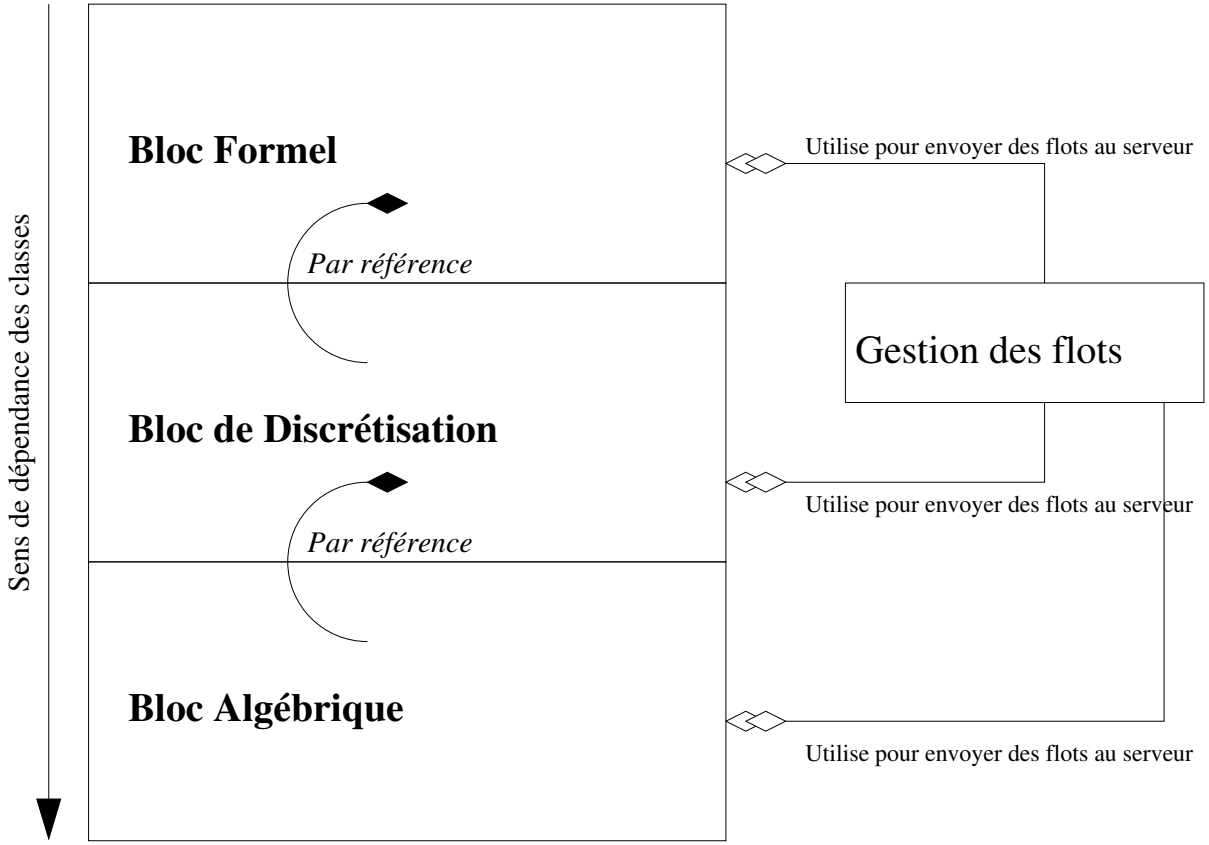
Les objets

Positionnement des objets dans les niveaux



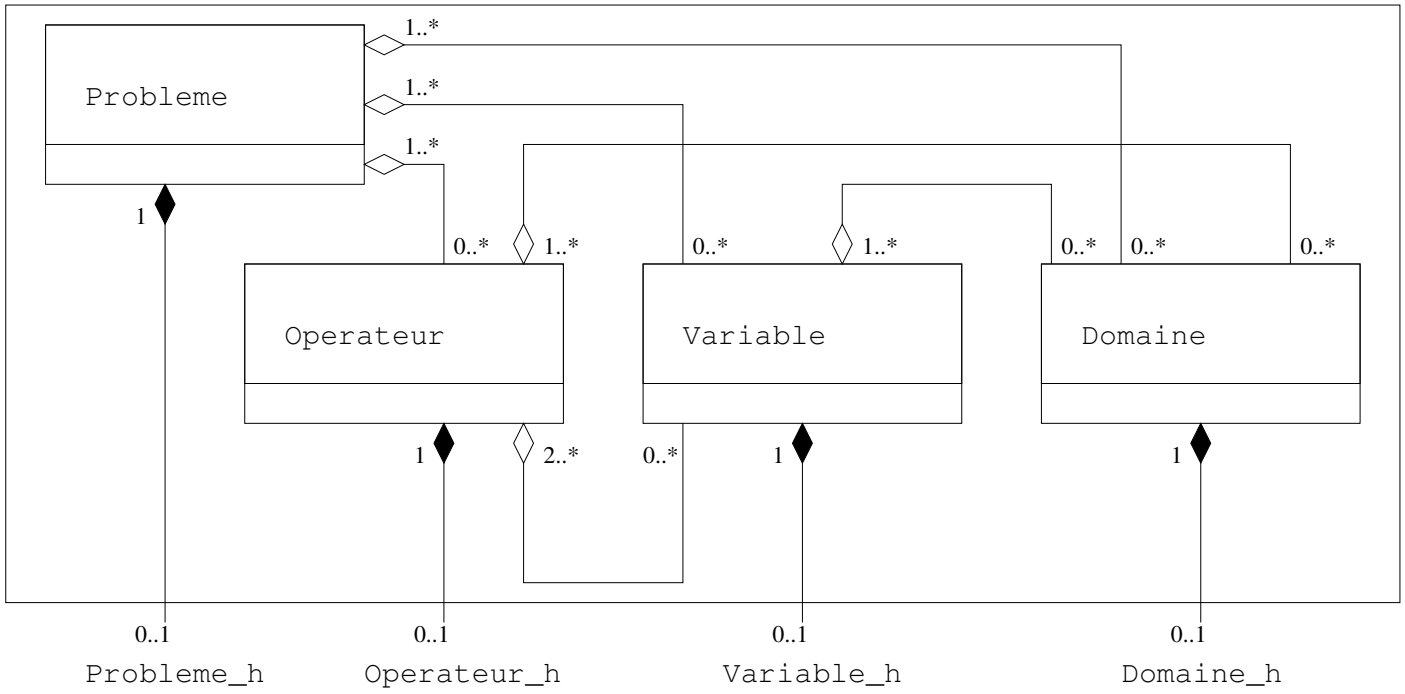
Les objets

Liens inter-niveaux



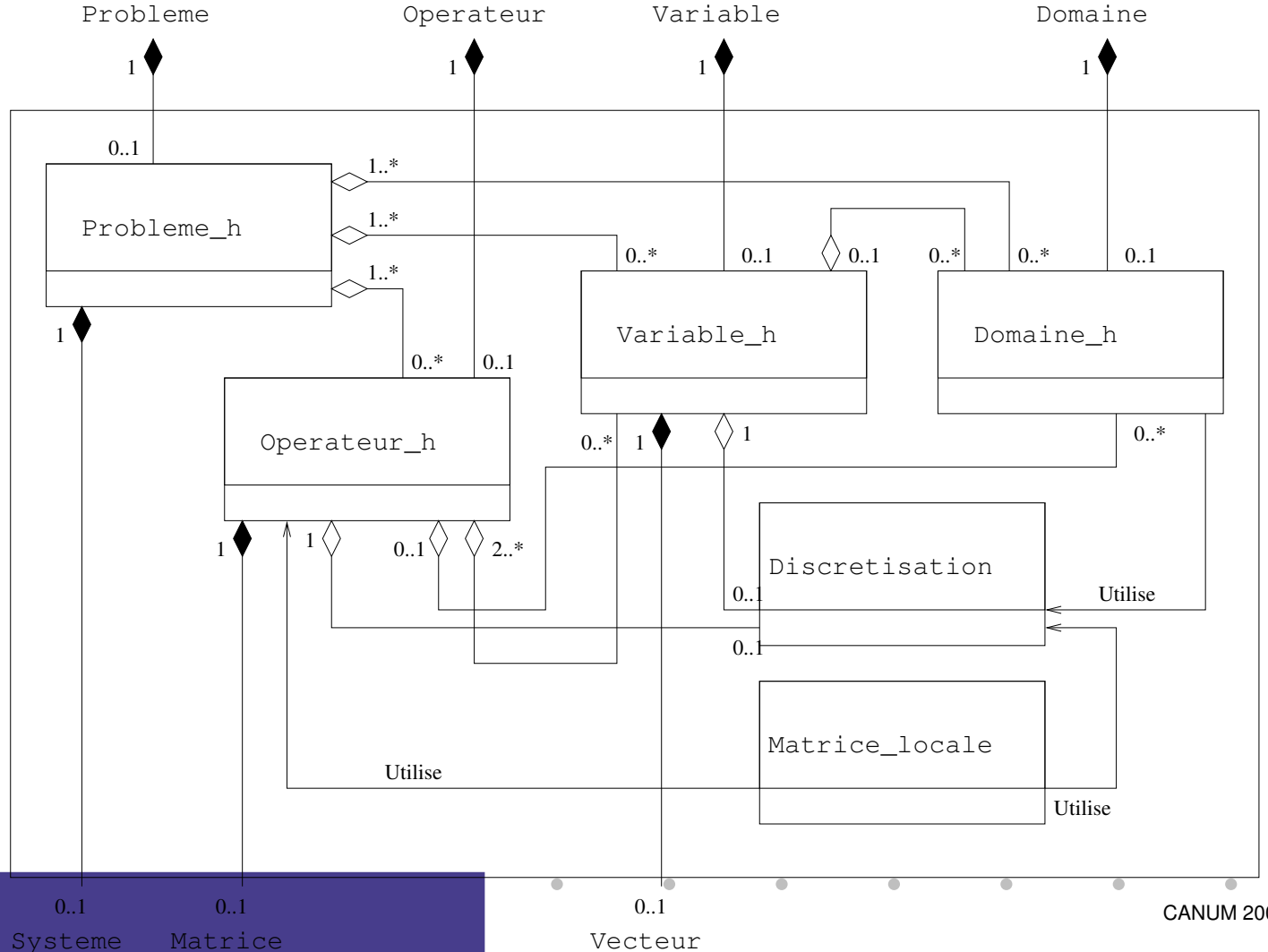
Les objets

Niveau continu



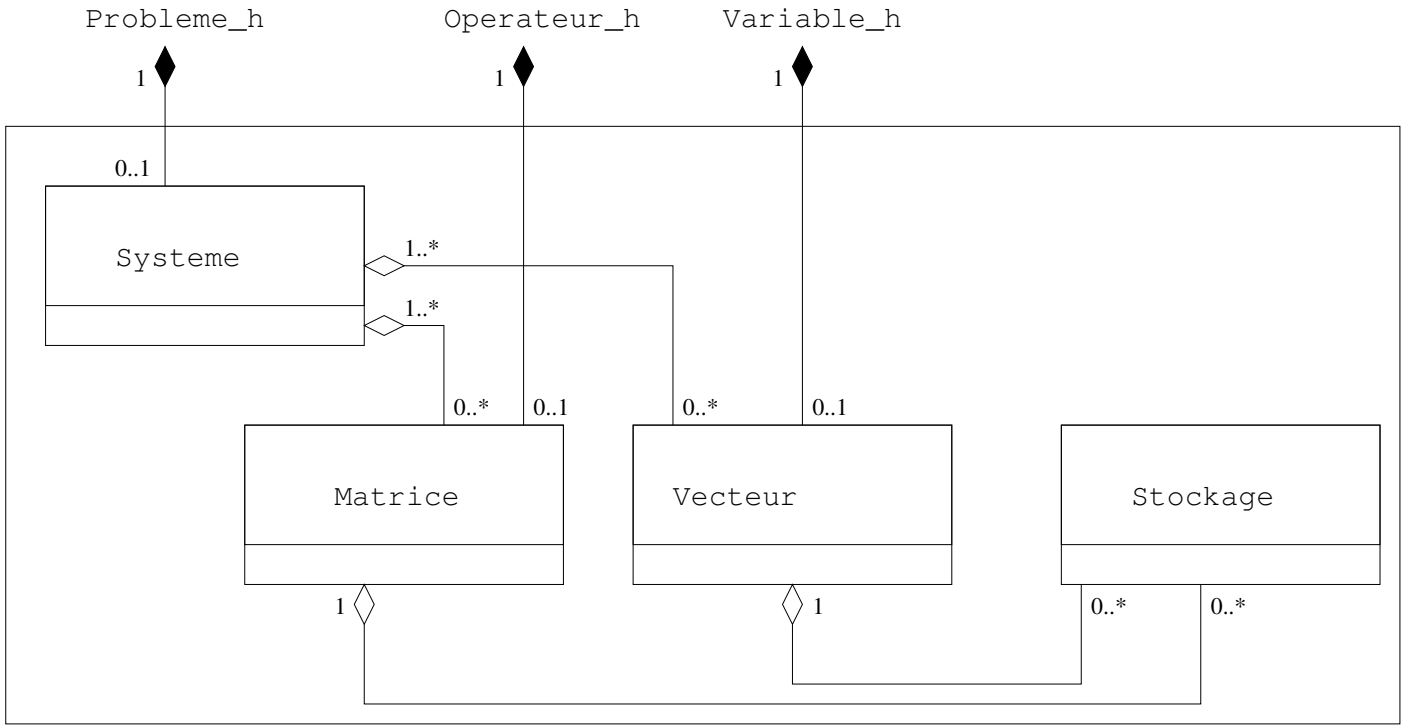
Les objets

Niveau discret



Les objets

Niveau algébrique



Composants et instrumentation

Couplage de composants

- Introduction

Couplage de composants

- Introduction
- Méthodologie de développement

Couplage de composants

- Introduction
- Méthodologie de développement
- Architecture et Infrastructure de communication

Couplage de composants

- Introduction
- Méthodologie de développement
- Architecture et Infrastructure de communication
- API de couplage

Couplage de composants

- Introduction
- Méthodologie de développement
- Architecture et Infrastructure de communication
- API de couplage
- Visualisation

Couplage de composants

- Introduction
- Méthodologie de développement
- Architecture et Infrastructure de communication
- API de couplage
- Visualisation
- Perspectives

Introduction

- Fonctionnalités :

Introduction

- Fonctionnalités :
  - couplage

Introduction

- Fonctionnalités :
  - couplage
  - exploitation, contrôle

Introduction

- Fonctionnalités :
  - couplage
  - exploitation, contrôle
  - outil de développement (documentation).

Introduction

- Fonctionnalités :
  - couplage
  - exploitation, contrôle
  - outil de développement (documentation).
- Contraintes :

Introduction

- Fonctionnalités :
  - couplage
  - exploitation, contrôle
  - outil de développement (documentation).
- Contraintes :
  - Plateforme distribuée, dynamique (temps réel) et collaborative,

Introduction

- Fonctionnalités :
  - couplage
  - exploitation, contrôle
  - outil de développement (documentation).
- Contraintes :
  - Plateforme distribuée, dynamique (temps réel) et collaborative,
  - multi-client et multi-composants.

Méthodologie de développement

Méthodologie = Formalisme + Processus

- UML (Unified Modeling Language) = Notations standardisées avec une sémantique précise

Méthodologie de développement

Méthodologie = Formalisme + Processus

- UML (Unified Modeling Language) = Notations standardisées avec une sémantique précise
- UP (Unified Process) = démarche méthodologique ; itératif et incrémental

Conceptualisation → Analyse

↑ ↓

Test, validation ← Implémentation ← Conception

Architecture et Infrastructure de com

Prise en compte des contraintes techniques = 3 grands sous-systèmes :

- Le composant $\Rightarrow$ Super-calculateur,

Architecture et Infrastructure de com

Prise en compte des contraintes techniques = 3 grands sous-systèmes :

- Le composant $\Rightarrow$ Super-calculateur,
- La plate-forme de pilotage $\Rightarrow$ Poste client

Architecture et Infrastructure de com

Prise en compte des contraintes techniques = 3 grands sous-systèmes :

- Le composant $\Rightarrow$ Super-calculateur,
- La plate-forme de pilotage $\Rightarrow$ Poste client
- La gestion des données des composants $\Rightarrow$ Serveur

Architecture et Infrastructure de com

Prise en compte des contraintes techniques = 3 grands sous-systèmes :

- Le composant $\Rightarrow$ Super-calculateur,
- La plate-forme de pilotage $\Rightarrow$ Poste client
- La gestion des données des composants $\Rightarrow$ Serveur

$\Rightarrow$ Architecture de type 3 tiers.

Protocole de communication

Contraintes :

- Sur protocole TCP/IP

Protocole de communication

Contraintes :

- Sur protocole TCP/IP
- Mapping C++ (composants) et JAVA (interface graphique)

Protocole de communication

Contraintes :

- Sur protocole TCP/IP
- Mapping C++ (composants) et JAVA (interface graphique)
- Objet

Protocole de communication

Contraintes :

- Sur protocole TCP/IP
- Mapping C++ (composants) et JAVA (interface graphique)
- Objet
- Normalisé et reconnu (pérennité)

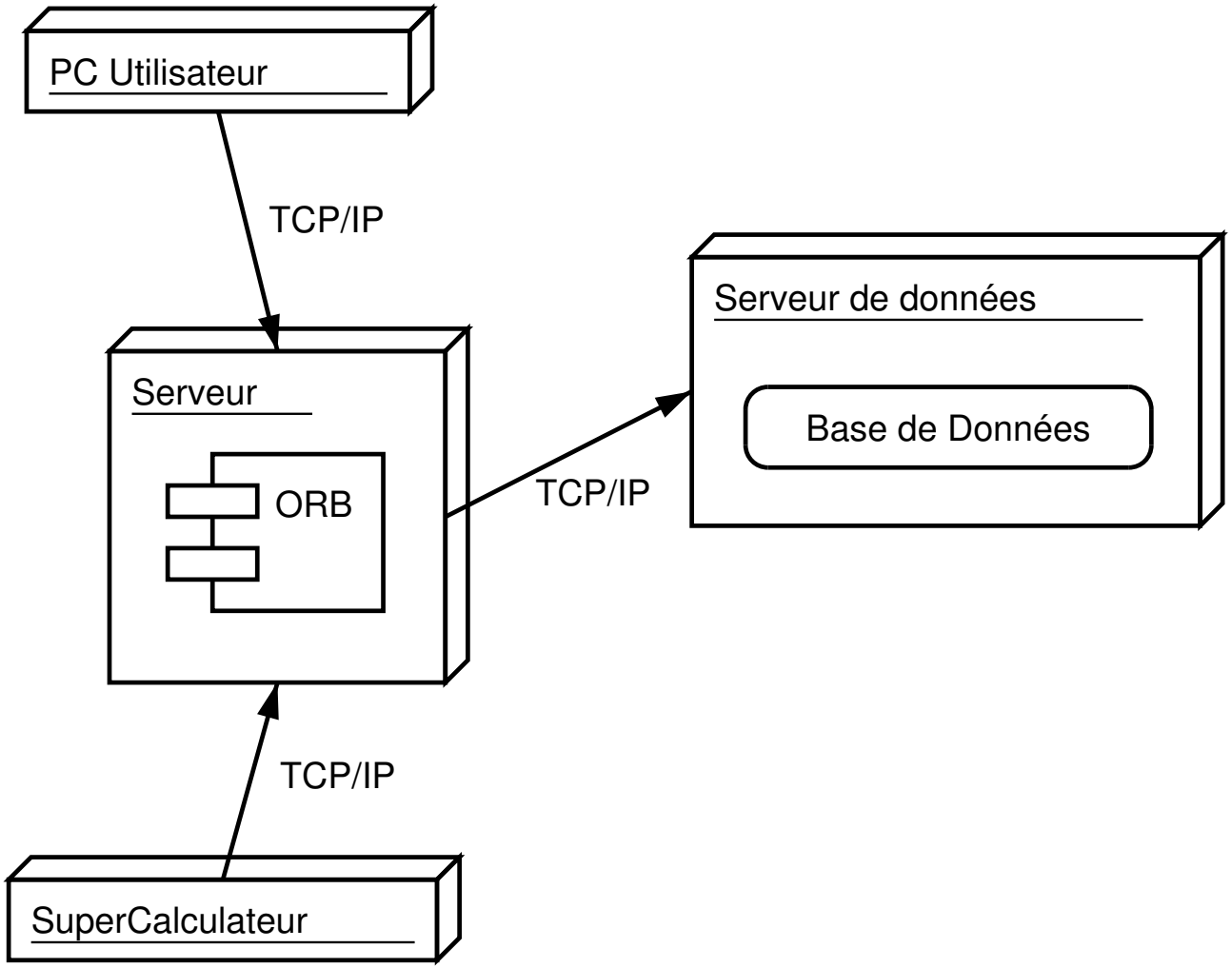
Protocole de communication

Contraintes :

- Sur protocole TCP/IP
- Mapping C++ (composants) et JAVA (interface graphique)
- Objet
- Normalisé et reconnu (pérennité)

$\Rightarrow$ CORBA

Architecture de la plate-forme



API de couplage

- Composition :

API de couplage

- Composition :
  - Flux de données distribués

API de couplage

- Composition :
  - Flux de données distribués
  - Contrôle du composant

API de couplage

- Composition :
  - Flux de données distribués
  - Contrôle du composant
  - Profiling, mémoire

API de couplage

- Composition :
  - Flux de données distribués
  - Contrôle du composant
  - Profiling, mémoire
- Langages :

API de couplage

- Composition :
  - Flux de données distribués
  - Contrôle du composant
  - Profiling, mémoire
- Langages :
  - Natif C++, Java

API de couplage

- Composition :
  - Flux de données distribués
  - Contrôle du composant
  - Profiling, mémoire
- Langages :
  - Natif C++, Java
  - Interfaçage Fortran 90

Communications API-serveur

- Classe de connexion au serveur : initialisation des services CORBA

Communications API-serveur

- Classe de connexion au serveur : initialisation des services CORBA
- Classes de factory des objets distribués

Communications API-serveur

- Classe de connexion au serveur : initialisation des services CORBA
- Classes de factory des objets distribués
- Enregistrement des objets distribués sur le service de nommage

Communications API-serveur

- Classe de connexion au serveur : initialisation des services CORBA
- Classes de factory des objets distribués
- Enregistrement des objets distribués sur le service de nommage
- Contrôle du composant par l'intermédiaire du service d'événement

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions
- Création des flux : nom, format, permissions

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions
- Création des flux : nom, format, permissions
- Envoi des données par surcharge de l'opérateur ≪

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions
- Création des flux : nom, format, permissions
- Envoi des données par surcharge de l'opérateur $\ll$
- Surcharge du *cout*

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions
- Création des flux : nom, format, permissions
- Envoi des données par surcharge de l'opérateur $\ll$
- Surcharge du *cout*
- Méthodes *push* et *pop* pour le profiling

Utilisation de l'API

- Méthode globale d'initialisation des communications avec le serveur : propriétaire, temps de vie, permissions
- Création des flux : nom, format, permissions
- Envoi des données par surcharge de l'opérateur $\ll$
- Surcharge du *cout*
- Méthodes *push* et *pop* pour le profiling
- Méthode globale de fin d'exécution : conservation ou non des flux sur le serveur

Visualisation

Principes :

- JAVA $\Rightarrow$ portable

Visualisation

Principes :

- JAVA $\Rightarrow$ portable
- Temps réel

Visualisation

Principes :

- JAVA $\Rightarrow$ portable
- Temps réel
- Sécurisé par login/mot de passe

Visualisation

Fonctionnalités :

- Contrôle des exécutions

Visualisation

Fonctionnalités :

- Contrôle des exécutions
- Modification des permissions

Visualisation

Fonctionnalités :

- Contrôle des exécutions
- Modification des permissions
- Sauvegarde des données sur fichier

Visualisation

Fonctionnalités :

- Contrôle des exécutions
- Modification des permissions
- Sauvegarde des données sur fichier
- Visualisation textuelle, 1D, 2D des flux de données, visualisation couplée

Visualisation

Fonctionnalités :

- Contrôle des exécutions
- Modification des permissions
- Sauvegarde des données sur fichier
- Visualisation textuelle, 1D, 2D des flux de données, visualisation couplée
- Accès à la documentation du composant (scripts perl)

Perspectives court terme

- Actuellement : prototype avec toutes les fonctionnalités présentées

Perspectives court terme

- Actuellement : prototype avec toutes les fonctionnalités présentées
- 2^{ieme} itération :

Perspectives court terme

- Actuellement : prototype avec toutes les fonctionnalités présentées
- 2^{ieme} itération :
  - Abstraire l'environnement de communication (différents ORB)

Perspectives court terme

- Actuellement : prototype avec toutes les fonctionnalités présentées
- 2^{ieme} itération :
  - Abstraire l'environnement de communication (différents ORB)
  - Utilisation de Doxygen pour la génération de documentation extraite des sources

Perspectives court terme

- Actuellement : prototype avec toutes les fonctionnalités présentées
- 2^{ieme} itération :
  - Abstraire l'environnement de communication (différents ORB)
  - Utilisation de Doxygen pour la génération de documentation extraite des sources
  - flux entrants et couplage de composant

Perspectives moyen terme

- n^{ieme} itération :

Perspectives moyen terme

- n^{ieme} itération :
  - Abstraire l'environnement de communication (middleware de type Grid)

Perspectives moyen terme

- n^{ieme} itération :
  - Abstraire l'environnement de communication (middleware de type Grid)
  - Utilisation d'XML pour décrire les flux distribués

Perspectives moyen terme

- n^{ieme} itération :
  - Abstraire l'environnement de communication (middleware de type Grid)
  - Utilisation d'XML pour décrire les flux distribués
  - plug-ins de visualisation (AVS, ...)

Perspectives moyen terme

- n^{ieme} itération :
  - Abstraire l'environnement de communication (middleware de type Grid)
  - Utilisation d'XML pour décrire les flux distribués
  - plug-ins de visualisation (AVS, ...)
  - Composant parallèle

Conclusion et perspectives

Conclusion et perspectives

- Ce qui existe
  - problème test elliptique, Schrödinger stochastique

Conclusion et perspectives

- Ce qui existe
  - problème test elliptique, Schrödinger stochastique
  - mailleur régulier et lecture de maillage, calcul des compléments

Conclusion et perspectives

- Ce qui existe
  - problème test elliptique, Schrödinger stochastique
  - mailleur régulier et lecture de maillage, calcul des compléments
  - Discrétisation EF DF

Conclusion et perspectives

- Ce qui existe
  - problème test elliptique, Schrödinger stochastique
  - mailleur régulier et lecture de maillage, calcul des compléments
  - Discrétisation EF DF
  - Méthode de stokage Full, ELL CRS, tableau de tableaux

Conclusion et perspectives

- les développements en cours
  - EF mixtes-hybrides Petrof-Galerkin

Conclusion et perspectives

- les développements en cours
  - EF mixtes-hybrides Petrof-Galerkin
  - VF

Conclusion et perspectives

- les développements en cours
  - EF mixtes-hybrides Petrof-Galerkin
  - VF
  - DF sur un nuage de points

Conclusion et perspectives

- les développements en cours
  - EF mixtes-hybrides Petrof-Galerkin
  - VF
  - DF sur un nuage de points
  - Interface avec Paraview, AVS/Express

Conclusion et perspectives

- les développements en cours
  - EF mixtes-hybrides Petrof-Galerkin
  - VF
  - DF sur un nuage de points
  - Interface avec Paraview, AVS/Express
  - Classe Tableau (avec A. Lichnewsky) Pete Pooma