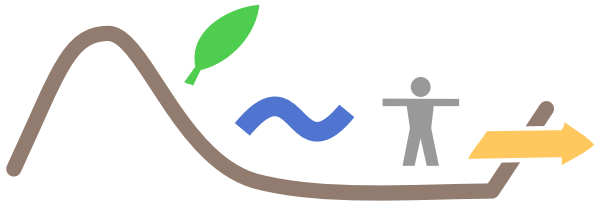


OpenFLUID

Software Environment
for Modelling Fluxes in Landscapes

Présentation générale

Utilisation d'OpenFLUID

JC. Fabre, M. Rabotin, D.Crevoisier,
A. Libres

LISAH - Laboratoire d'étude des Interactions Sol-Agrosystème-Hydrosystème

Plan

Plan

Modélisation intégrative du paysage

- Modélisation et simulation de processus spatialisés, interagissant fortement dans le temps et l'espace
  i.e. flux de matière (eau, terre, énergie, ...), activités humaines, ...
- Approche pluri-disciplinaire
- Approche équilibrée entre modélisation des processus et représentation de l'espace

⇒ Approche complexe

⇒ Nécessité d'outils en support à la modélisation intégrative du paysage

Historique

Développement du modèle MHYDAS au LISAH depuis 1995

- concepts de modélisation spatio-temporelle
- représentation des éléments du paysage
- hydrologie de surface et souterraine, polluants, érosion, agronomie, ...

Nouveaux besoins, nouveaux contextes (2000's)

- montée en puissance de la modélisation/simulation
- grande variété de processus spatiaux modélisés
- collaborations de recherche de plus en plus nombreuses (projets sur AO)

Objectifs de développement d'une plateforme "Paysage"

Un environnement logiciel pour la modélisation spatialisée

- Proposer un formalisme "aussi générique que possible" pour la représentation de l'espace
- Proposer une structure de couplage spatio-temporel
- Permettre la construction de modèles couplés adaptés aux objectifs scientifiques

Un outil pour le travail en collaboration

- Mettre à disposition un environnement logiciel ouvert, extensible, évolutif
- Proposer une approche standard pour le développement de modèles
- Capitaliser, partager, réutiliser des modèles, des connaissances, des compétences

Une démarche d'ingénierie logicielle

- Profiter des technologies récentes
- Améliorer la maintenance, l'évolutivité, la qualité
- Permettre l'interfaçage avec des outils/modèles existants

⇒ Développement de la plateforme OpenFLUID depuis 2005, première version en juillet 2007

Objectifs de développement d'une plateforme "Paysage"

Un environnement logiciel pour la modélisation spatialisée

- Proposer un formalisme "aussi générique que possible" pour la représentation de l'espace
- Proposer une structure de couplage spatio-temporel
- Permettre la construction de modèles couplés adaptés aux objectifs scientifiques

Un outil pour le travail en collaboration

- Mettre à disposition un environnement logiciel ouvert, extensible, évolutif
- Proposer une approche standard pour le développement de modèles
- Capitaliser, partager, réutiliser des modèles, des connaissances, des compétences

Une démarche d'ingénierie logicielle

- Profiter des technologies récentes
- Améliorer la maintenance, l'évolutivité, la qualité
- Permettre l'interfaçage avec des outils/modèles existants

⇒ Développement de la plateforme OpenFLUID depuis 2005, première version en juillet 2007

Objectifs de développement d'une plateforme "Paysage"

Un environnement logiciel pour la modélisation spatialisée

- Proposer un formalisme "aussi générique que possible" pour la représentation de l'espace
- Proposer une structure de couplage spatio-temporel
- Permettre la construction de modèles couplés adaptés aux objectifs scientifiques

Un outil pour le travail en collaboration

- Mettre à disposition un environnement logiciel ouvert, extensible, évolutif
- Proposer une approche standard pour le développement de modèles
- Capitaliser, partager, réutiliser des modèles, des connaissances, des compétences

Une démarche d'ingénierie logicielle

- Profiter des technologies récentes
- Améliorer la maintenance, l'évolutivité, la qualité
- Permettre l'interfaçage avec des outils/modèles existants

⇒ Développement de la plateforme OpenFLUID depuis 2005, première version en juillet 2007

Objectifs de développement d'une plateforme "Paysage"

Un environnement logiciel pour la modélisation spatialisée

- Proposer un formalisme "aussi générique que possible" pour la représentation de l'espace
- Proposer une structure de couplage spatio-temporel
- Permettre la construction de modèles couplés adaptés aux objectifs scientifiques

Un outil pour le travail en collaboration

- Mettre à disposition un environnement logiciel ouvert, extensible, évolutif
- Proposer une approche standard pour le développement de modèles
- Capitaliser, partager, réutiliser des modèles, des connaissances, des compétences

Une démarche d'ingénierie logicielle

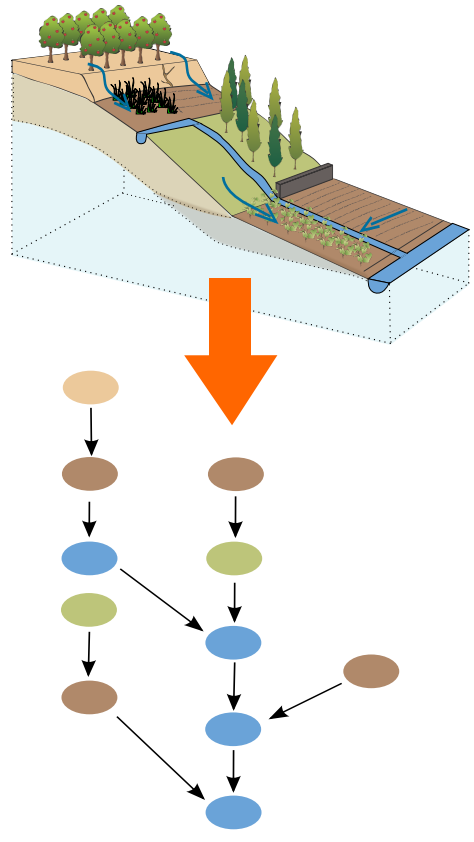
- Profiter des technologies récentes
- Améliorer la maintenance, l'évolutivité, la qualité
- Permettre l'interfaçage avec des outils/modèles existants

⇒ Développement de la plateforme OpenFLUID depuis 2005, première version en juillet 2007

Représentation Numérique du Paysage (RNP)

Représentation sous la forme d'un graphe d'unités spatiales connectées

- Les unités spatiales sont réparties en classes (surface, fossés, souterrain, . . .)
- Les unités spatiales sont qualifiées (propriétés geométriques, physiques, . . .)

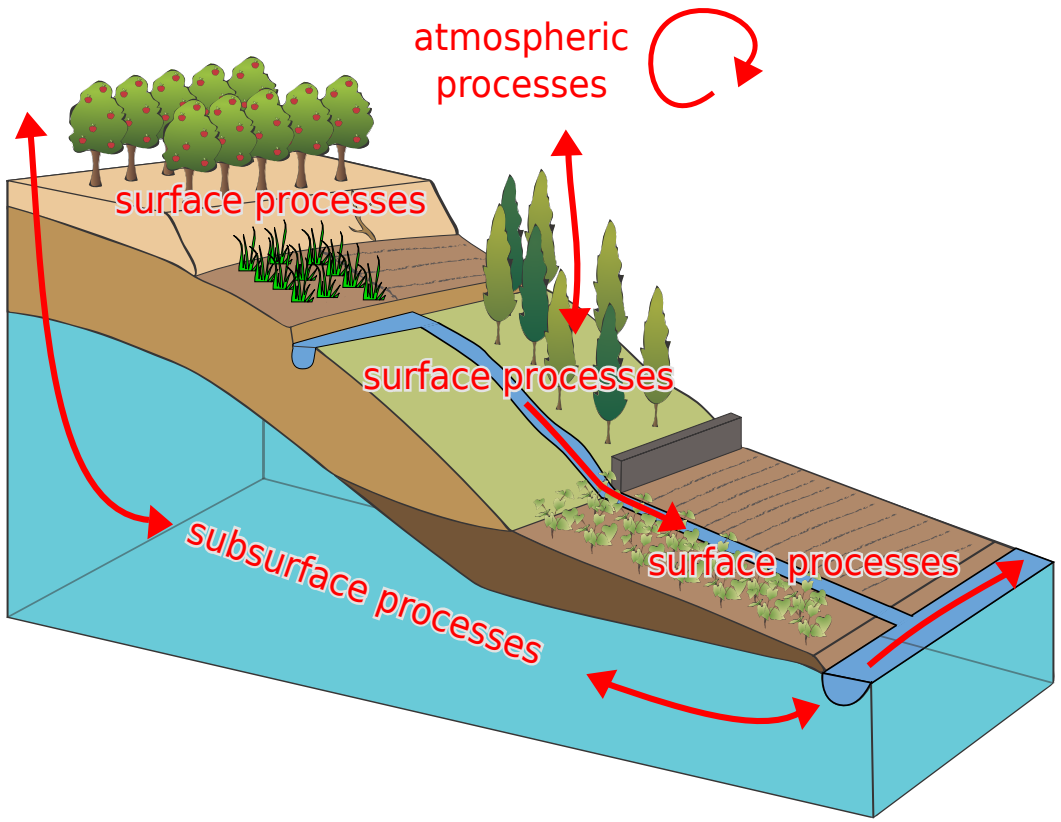


Ce graphe peut être construite en tenant compte

- du croisement de couches d'information spatiales (géométrie, relief, propriétés pertinentes, ...)
- des contraintes liées aux modèles qui vont être appliqués.

Modélisation de processus spatialisés

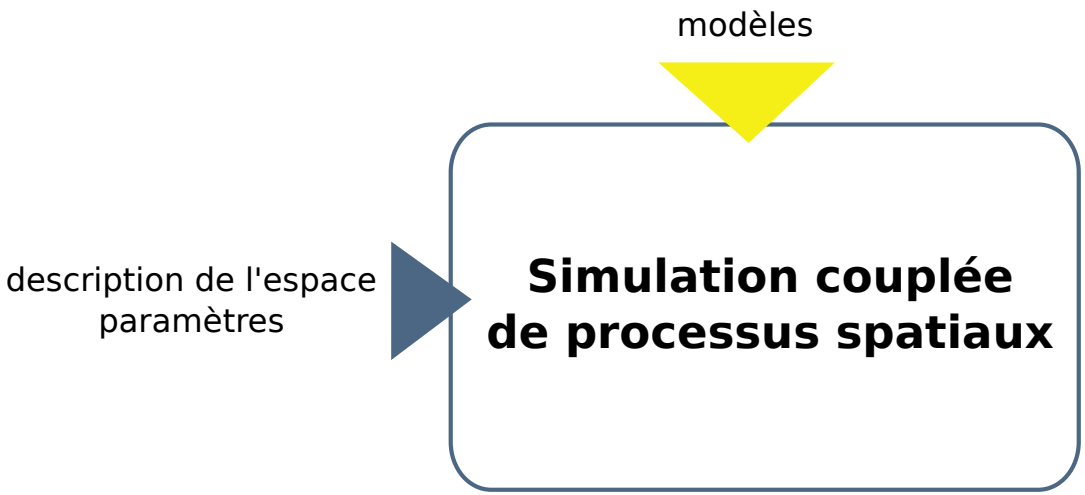
- Les processus sont associés à des unités spatiales
- Ces processus temporels donnent lieu à des modifications de l'état des unités spatiales ⇒ variables d'état des unités
- L'ensemble des processus couplés forme un modèle couplé



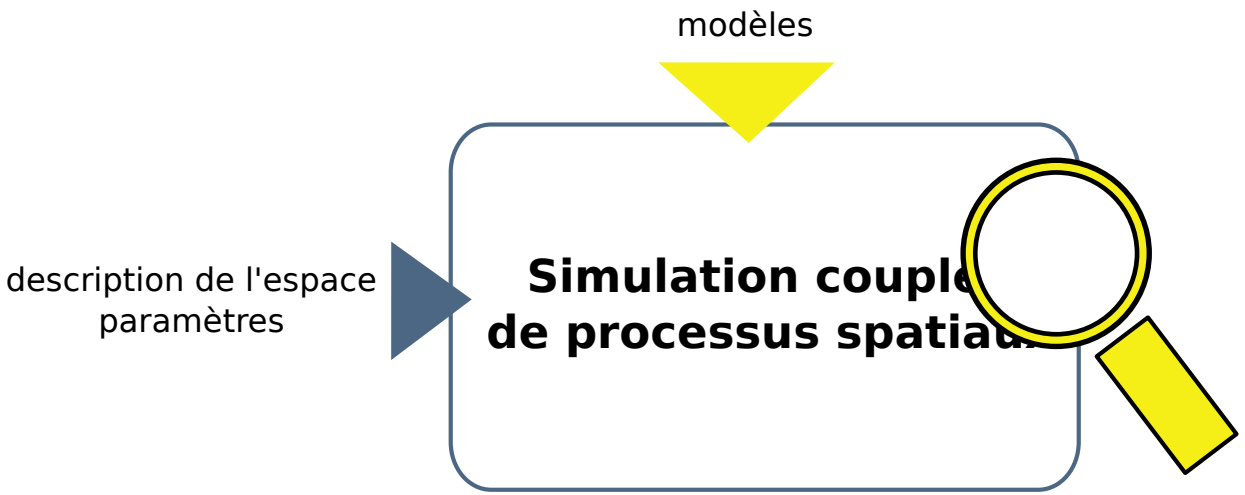
Démarche de simulation

Simulation couplée de processus spatiaux

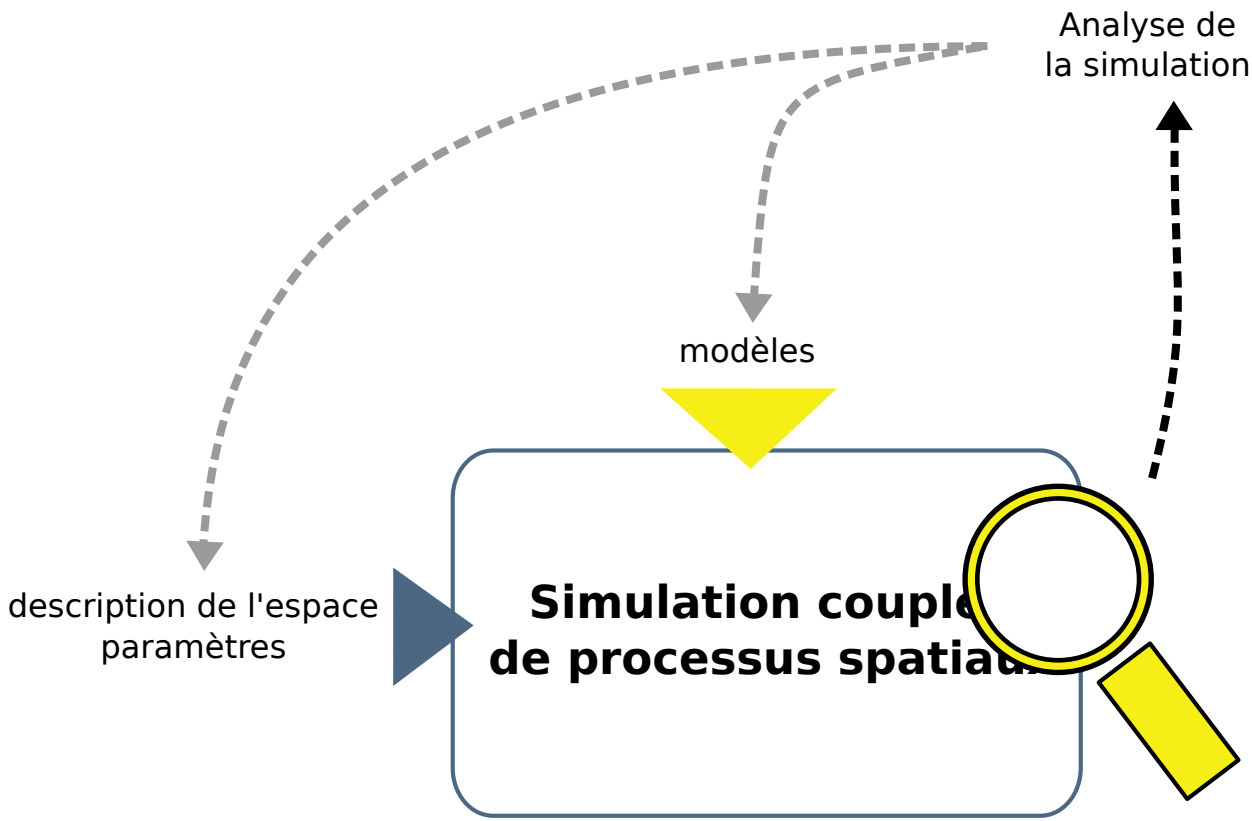
Démarche de simulation



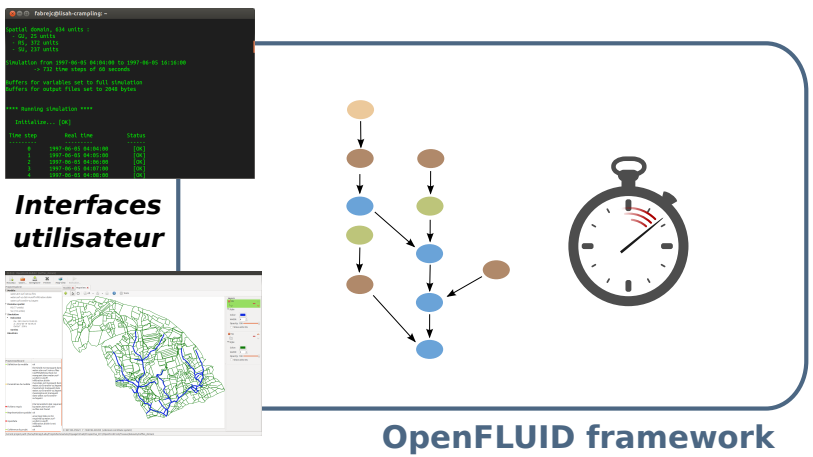
Démarche de simulation



Démarche de simulation



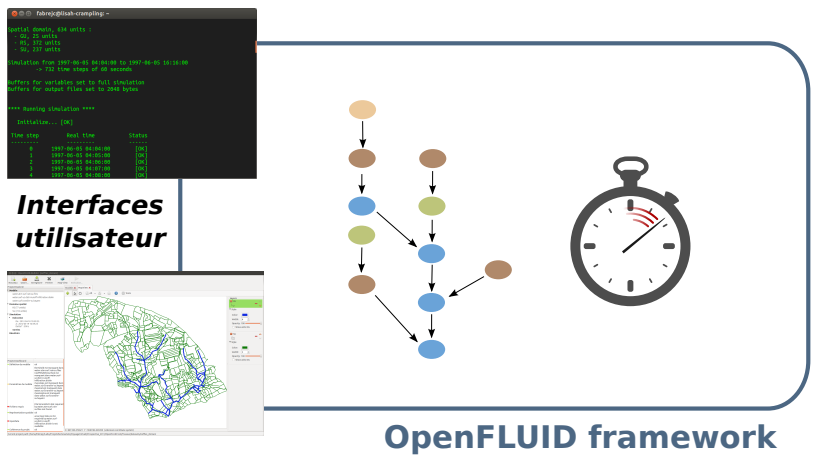
La plateforme OpenFLUID



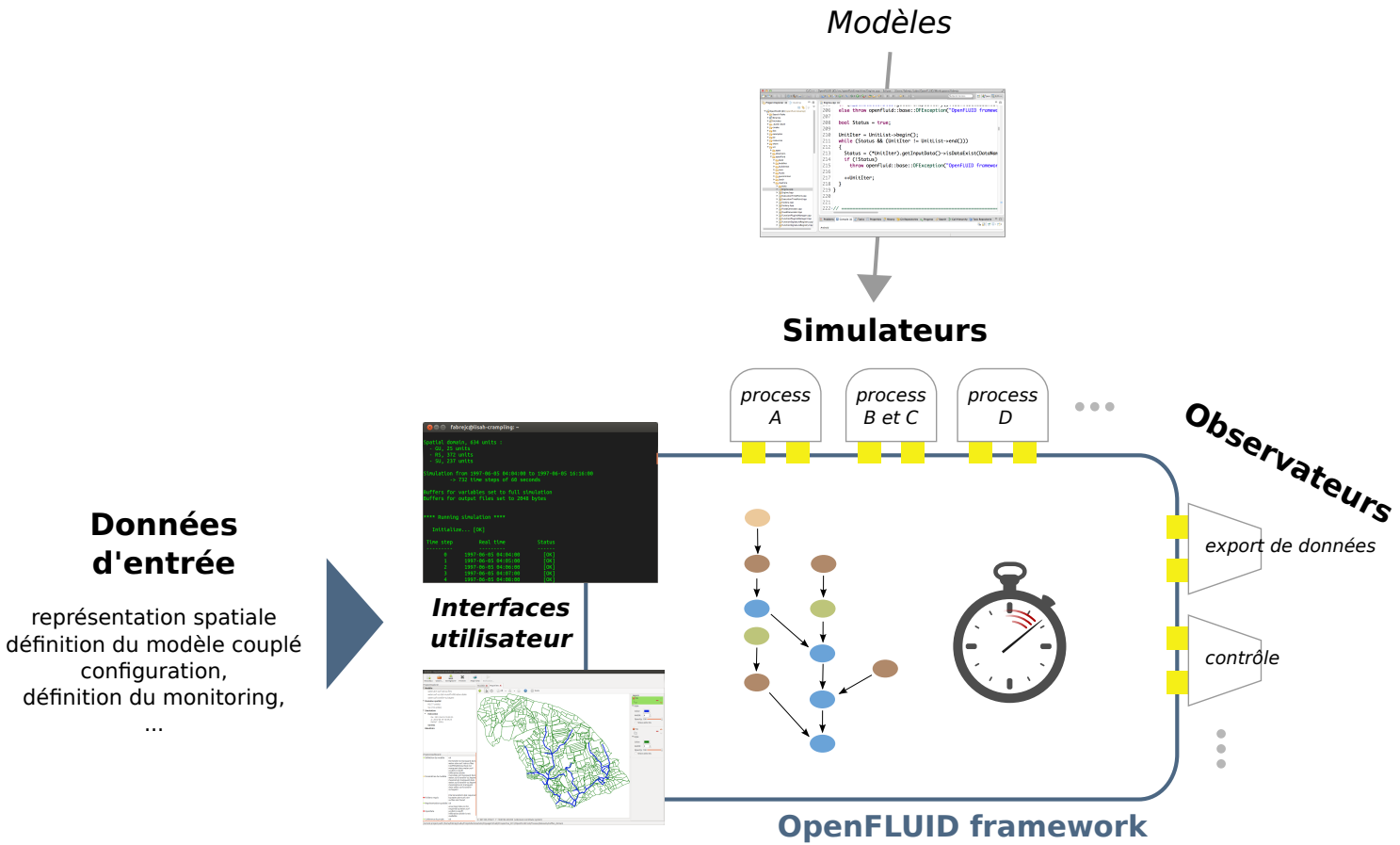
La plateforme OpenFLUID

Données d'entrée

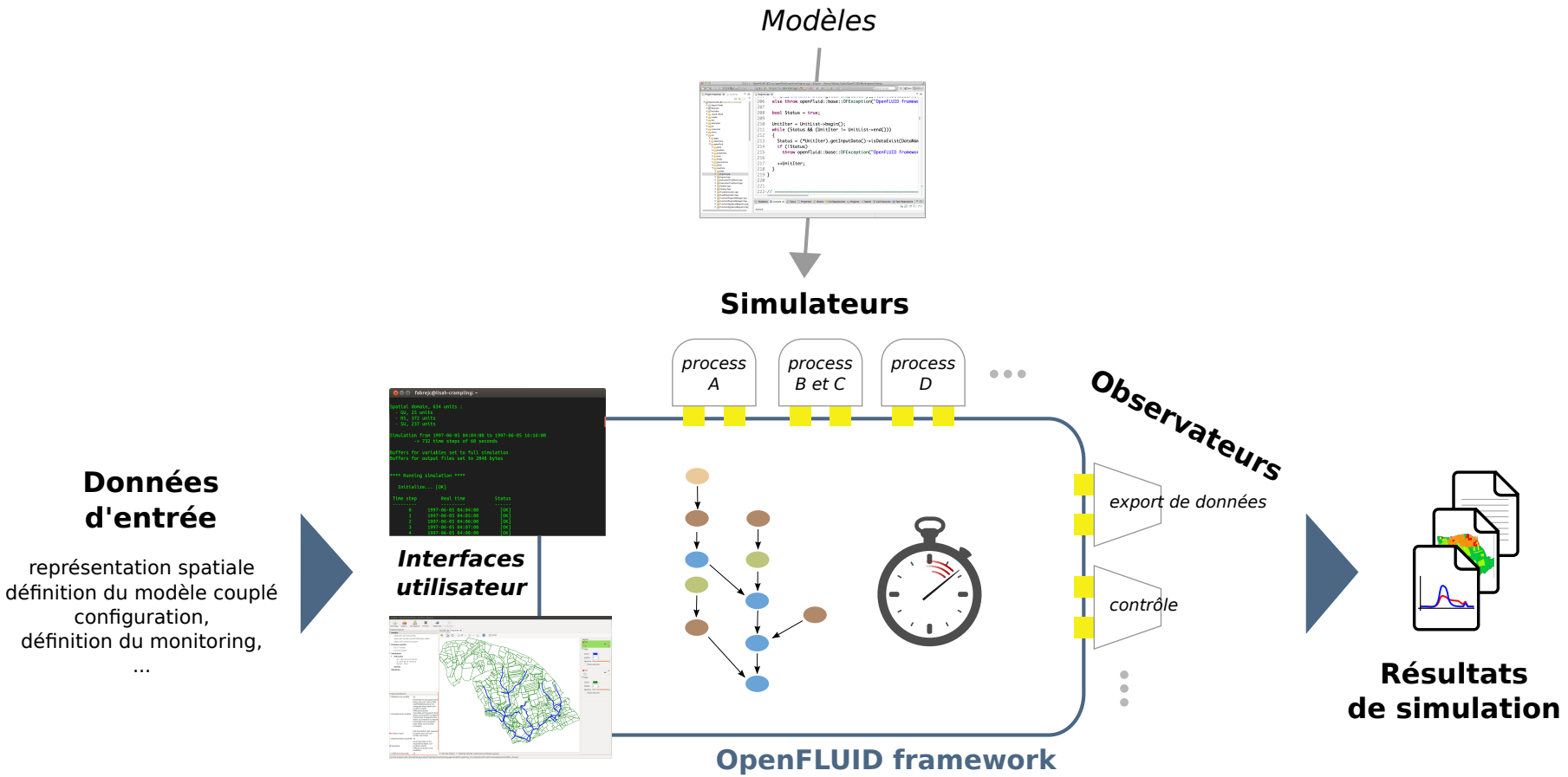
représentation spatiale
définition du modèle couplé
configuration,
définition du monitoring,
...



La plateforme OpenFLUID



La plateforme OpenFLUID



Plan

Framework OpenFLUID

Structure de construction et couplage de modèles, coeur de la plateforme

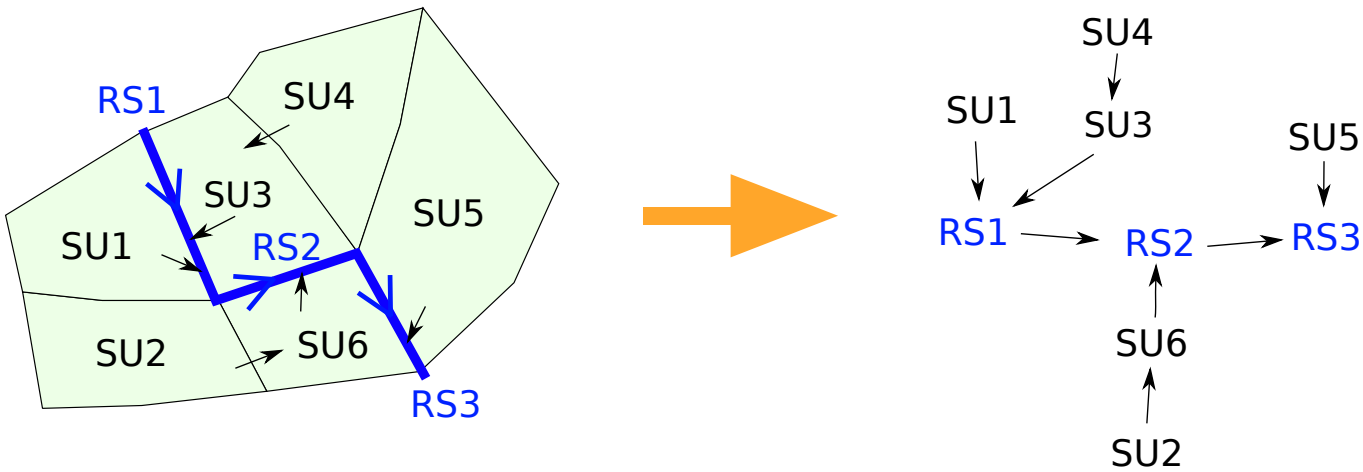
- Représentation de l'espace sous la forme de graphes connexes d'unités spatiales, avec multi-échelle possible
- Modélisation couplée basée sur des simulateurs branchés dynamiquement lors de la simulation
- Gestion du monitoring de la simulation
- Gestion du déroulement de la simulation et des échanges entre simulateurs (dans le temps et dans l'espace)

Le framework est un ensemble de bibliothèques logicielles intégrant les fonctionnalités de la plateforme (libopenfluid-xxx)

Représentation numérique du paysage (RNP)

Le paysage est représenté par OpenFLUID sous la forme d'un graphe orienté

- Les noeuds sont les unités spatiales représentant le paysage, rangés par classe d'unité
- Les arcs orientés sont les connexions entre les unités spatiales
- Chaque noeud porte des données propres à l'unité spatiale qu'il représente
- Un noeud peut contenir un graphe pour des approches multi-échelles



Simulateurs OpenFLUID

Un simulateur OpenFLUID est un code de calcul développé à partir d'un modèle mathématique

- Il simule un ou plusieurs processus spatiaux (transferts, évolutions)
- Il fait évoluer les états des unités spatiales
- Il déclare son comportement au travers de sa signature: unités spatiales utilisées, variables produites/requises, données d'entrée utilisées, ...

Un simulateur peut être

- développé "from scratch"
- l'encapsulation d'un code de calcul existant

Il est construit sous la forme d'un plugin pour le framework OpenFLUID.

Modèle couplé

Un modèle couplé est une liste de simulateurs qui seront branchés au framework lors de la simulation

Le couplage entre simulateurs se fait par l'échange spatio-temporel de variables

Chaque simulateur a son propre pas de temps.
Lors de chaque exécution, il précise la la planification pour sa réexecution, parmi les possibilités suivantes:

- une durée en secondes
- le deltat par défaut fixé pour la simulation
- une seule fois à la fin
- jamais

Observateurs OpenFLUID et Monitoring

Un observateur est un code source qui permet de ”regarder” les informations de simulation en continu. Il ne peut modifier l'état de la simulation.

Exemples d'observateurs

- export de résultats de simulation (format CSV, BD, . . .)
- export de représentations 2D/3D de simulations (vers SIG, GoogleEarth, . . .)
- contrôle en continu de simulations
- . . .

Un observateur est construit sous la forme d'un plugin pour le framework OpenFLUID.

Le monitoring est une liste d'observateurs qui seront branchés au framework lors de la simulation

Déroulement d'une simulation

Une simulation est définie par une période de simulation avec une date de début et une date de fin.

Un deltat par défaut est défini pour les simulateurs.

Une contrainte de planification peut être appliquée aux simulateurs

- Aucune contrainte
- La planification des simulateurs est vérifiée et doit être égale au deltat par défaut
- La planification des simulateurs est forcée pour être égale au deltat par défaut (risqué!)

Déroulement d'une simulation

Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.

t0

Mise en place de la période de simulation

Déroulement d'une simulation

Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.

Positionnement de la simulation à t0

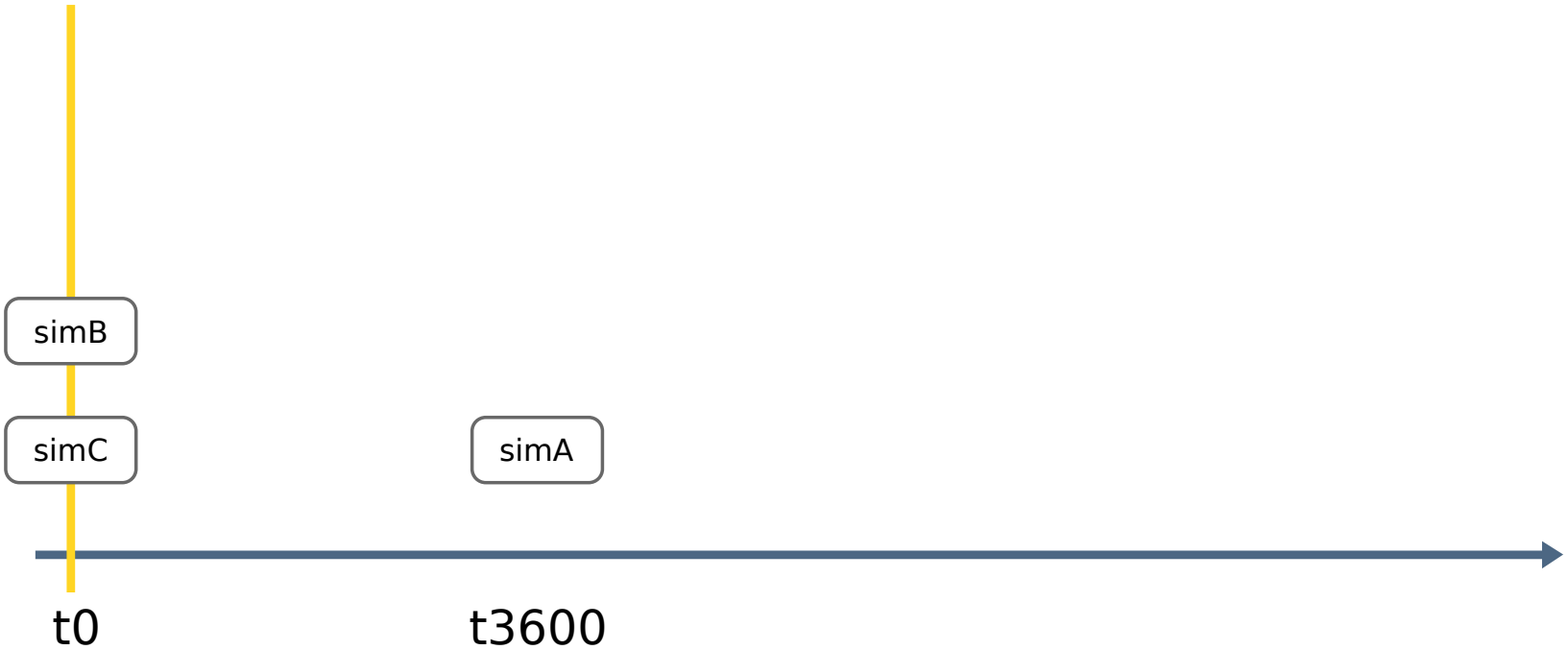
Déroulement d'une simulation

Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.

Exécution du simulateur simA à t0

Déroulement d'une simulation

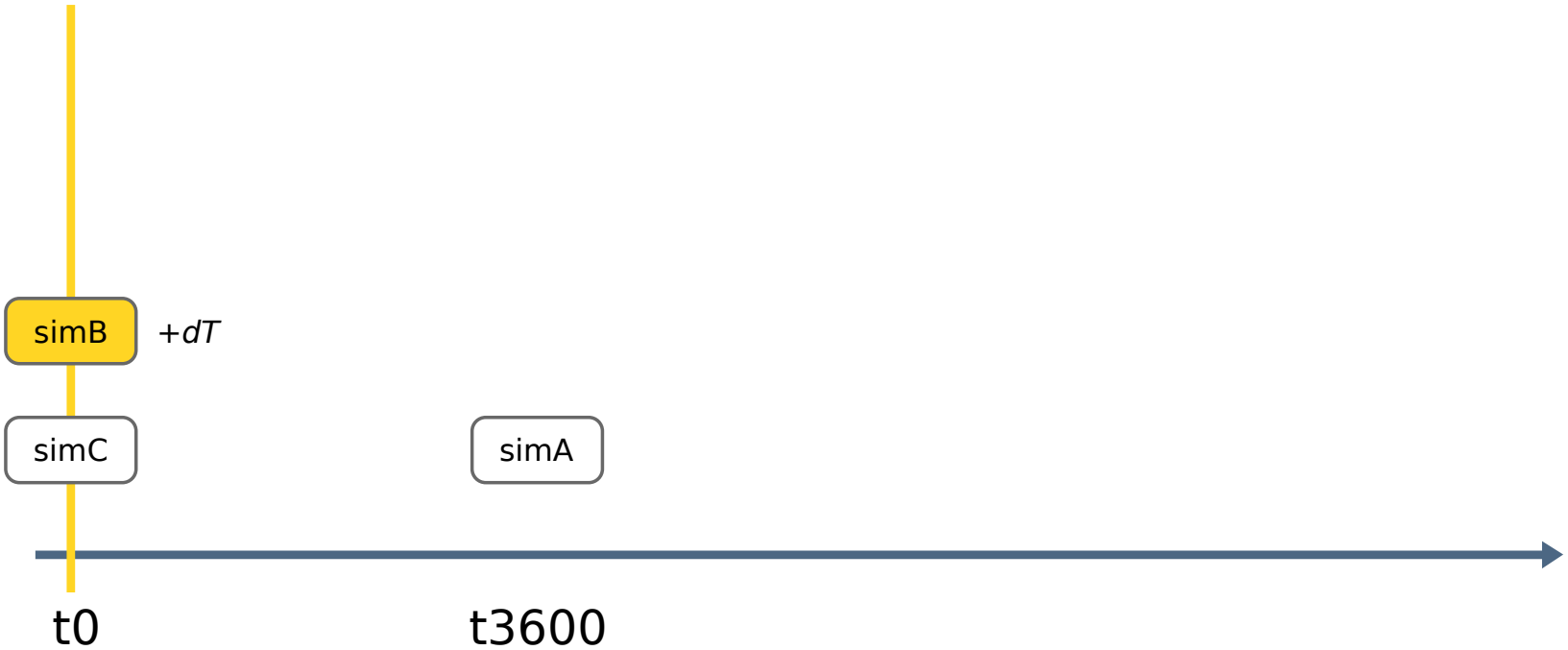
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de simA pour sa prochaine exécution

Déroulement d'une simulation

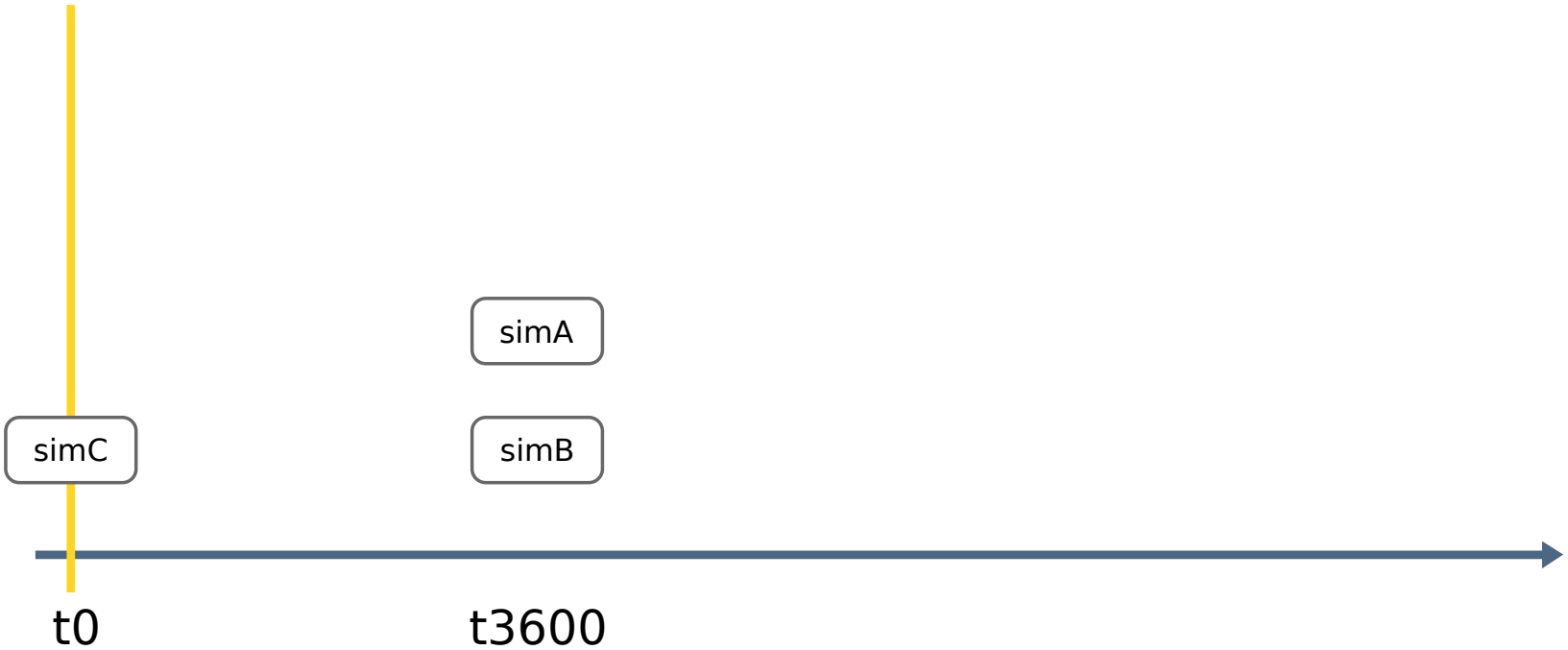
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du simulateur simB à t0

Déroulement d'une simulation

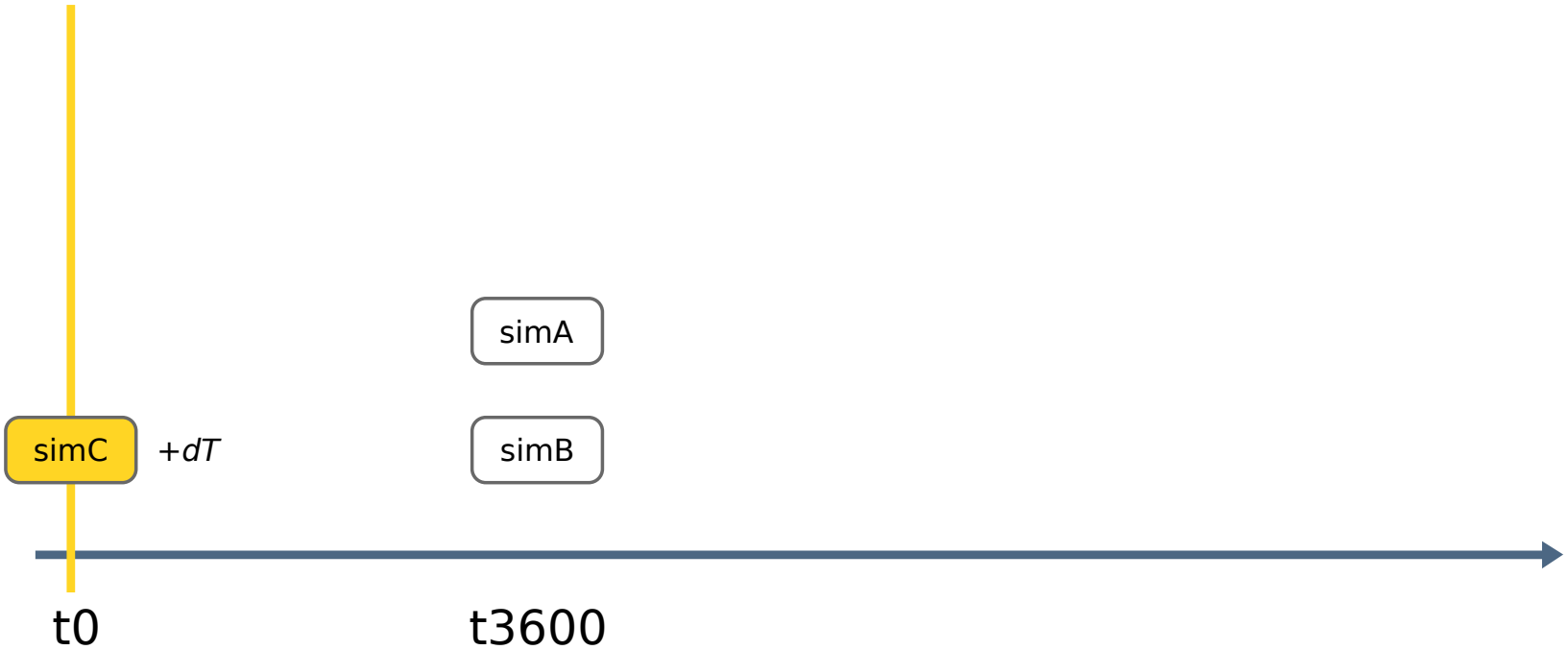
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de simB pour sa prochaine exécution (réagencement)

Déroulement d'une simulation

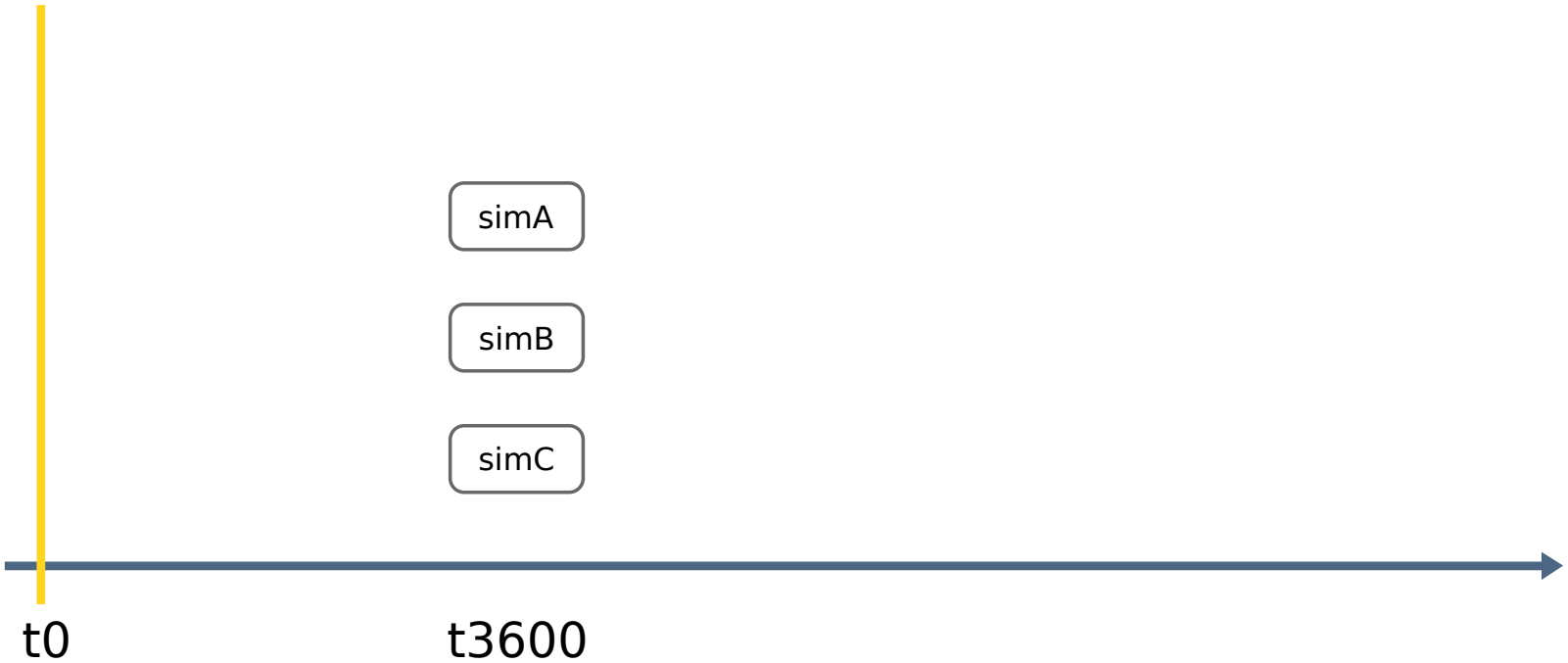
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du simulateur simC à t0

Déroulement d'une simulation

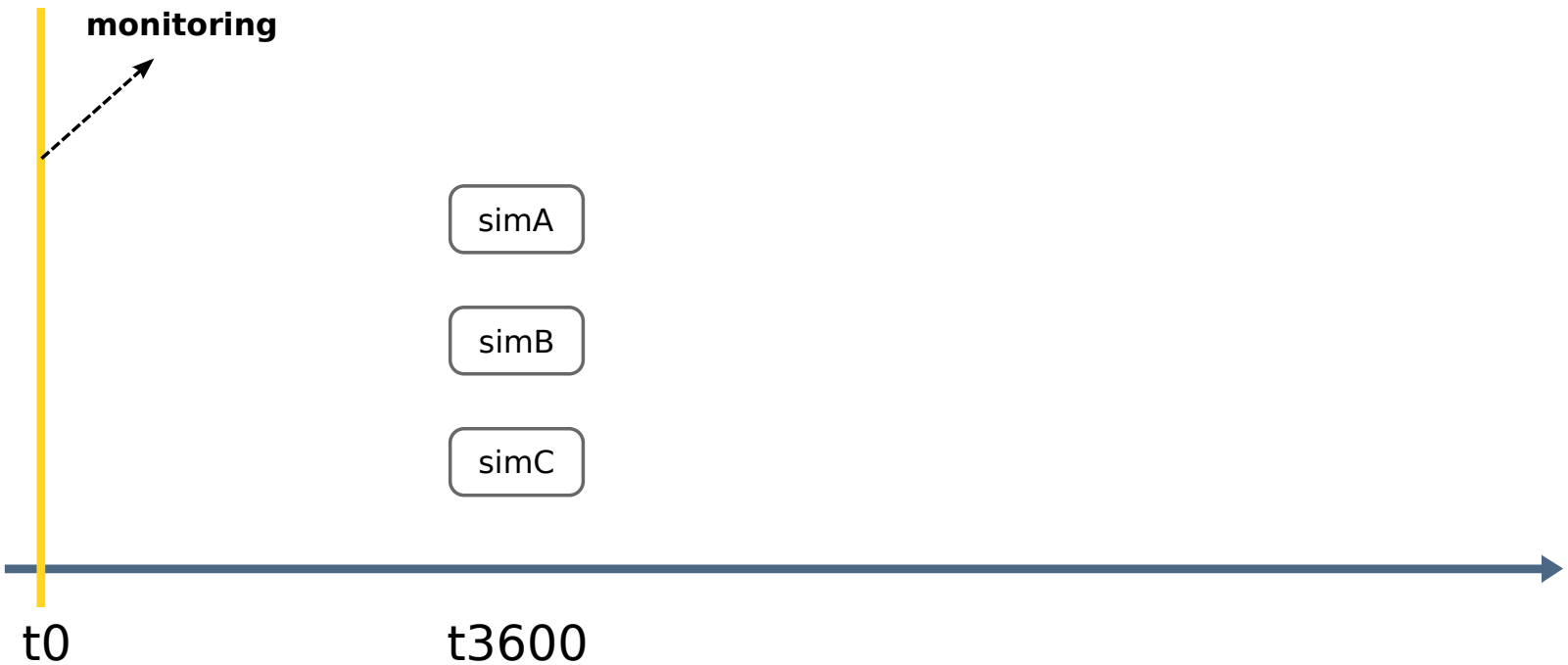
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de simC pour sa prochaine exécution

Déroulement d'une simulation

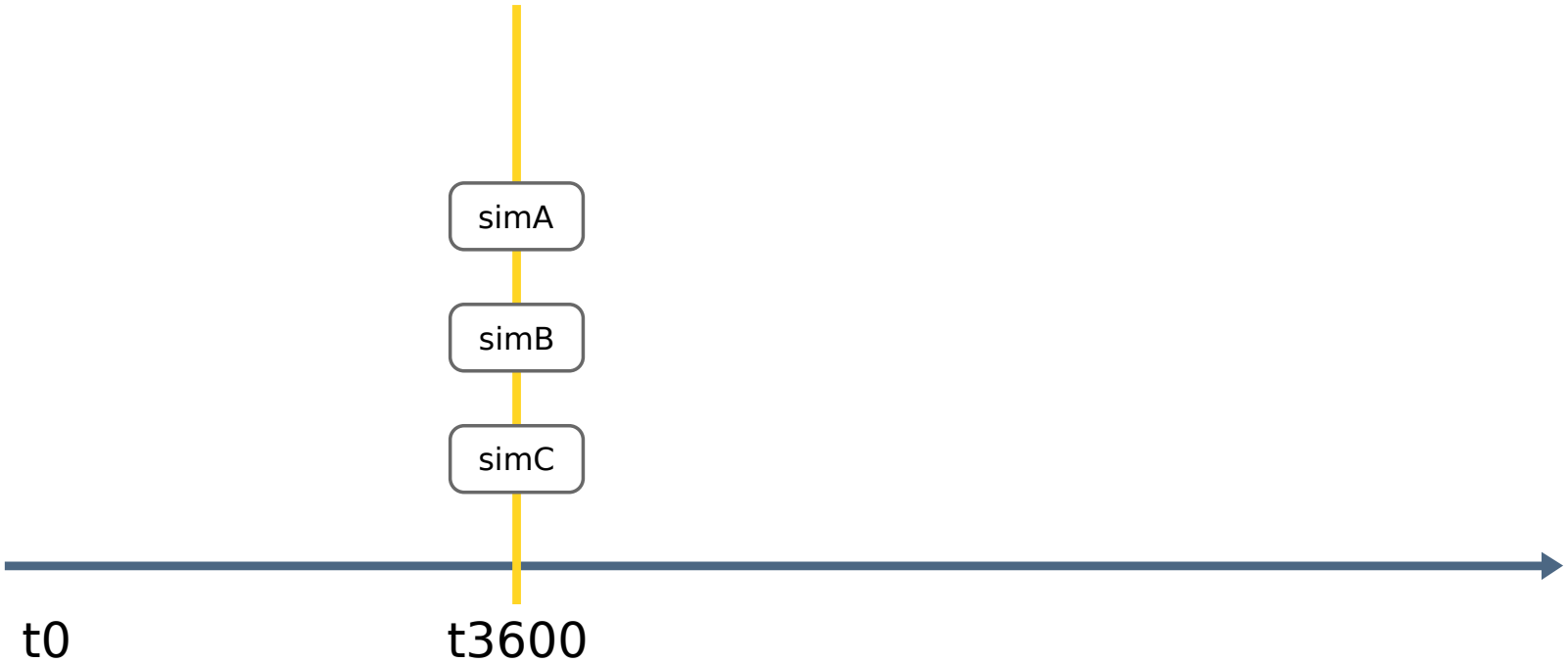
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du monitoring à t0

Déroulement d'une simulation

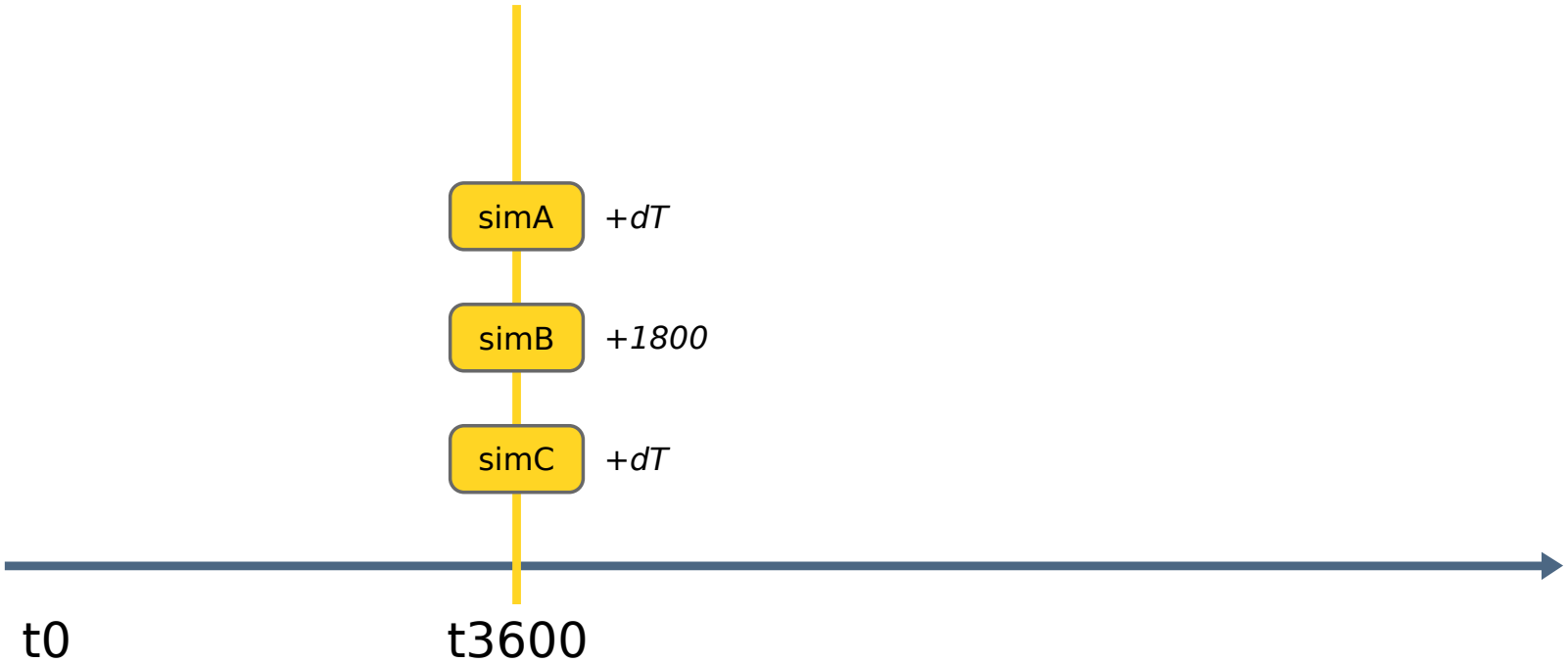
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de la simulation au prochain index de temps (t3600)

Déroulement d'une simulation

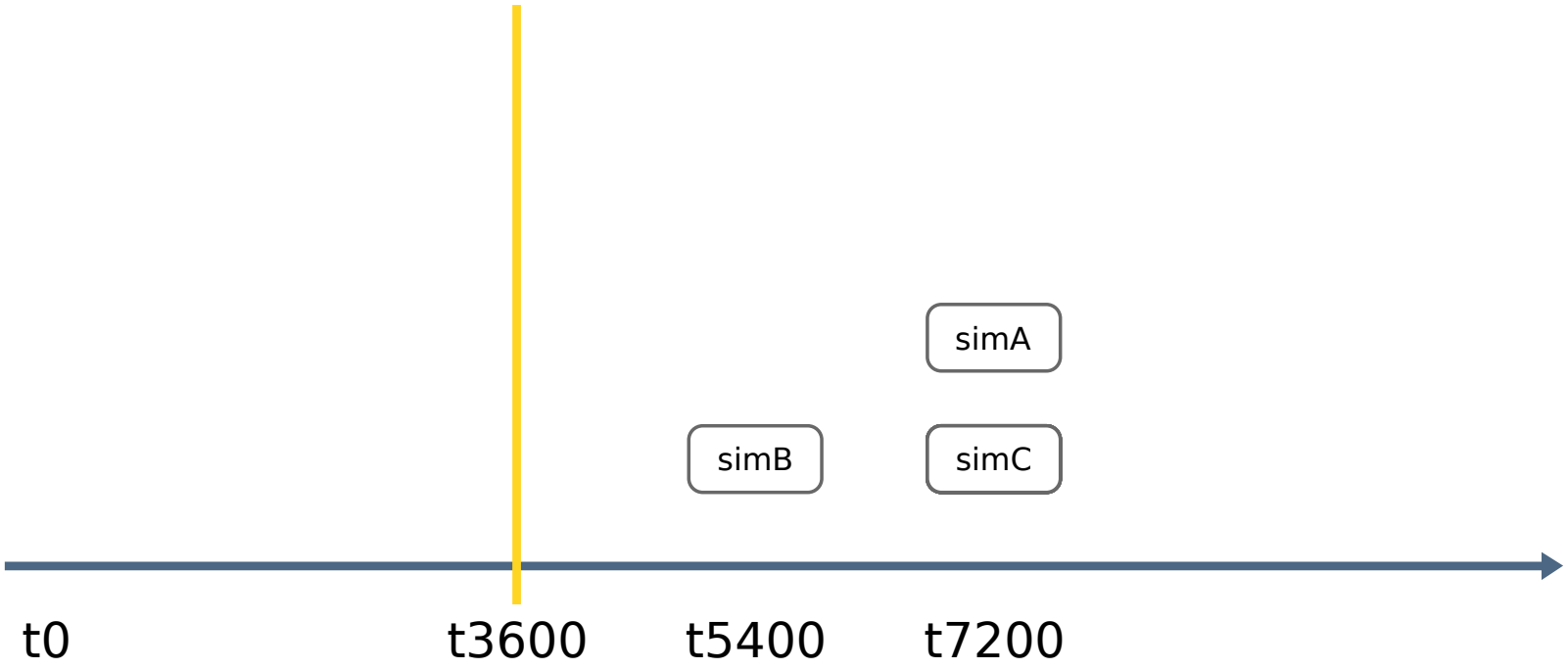
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution des simulateurs à t3600

Déroulement d'une simulation

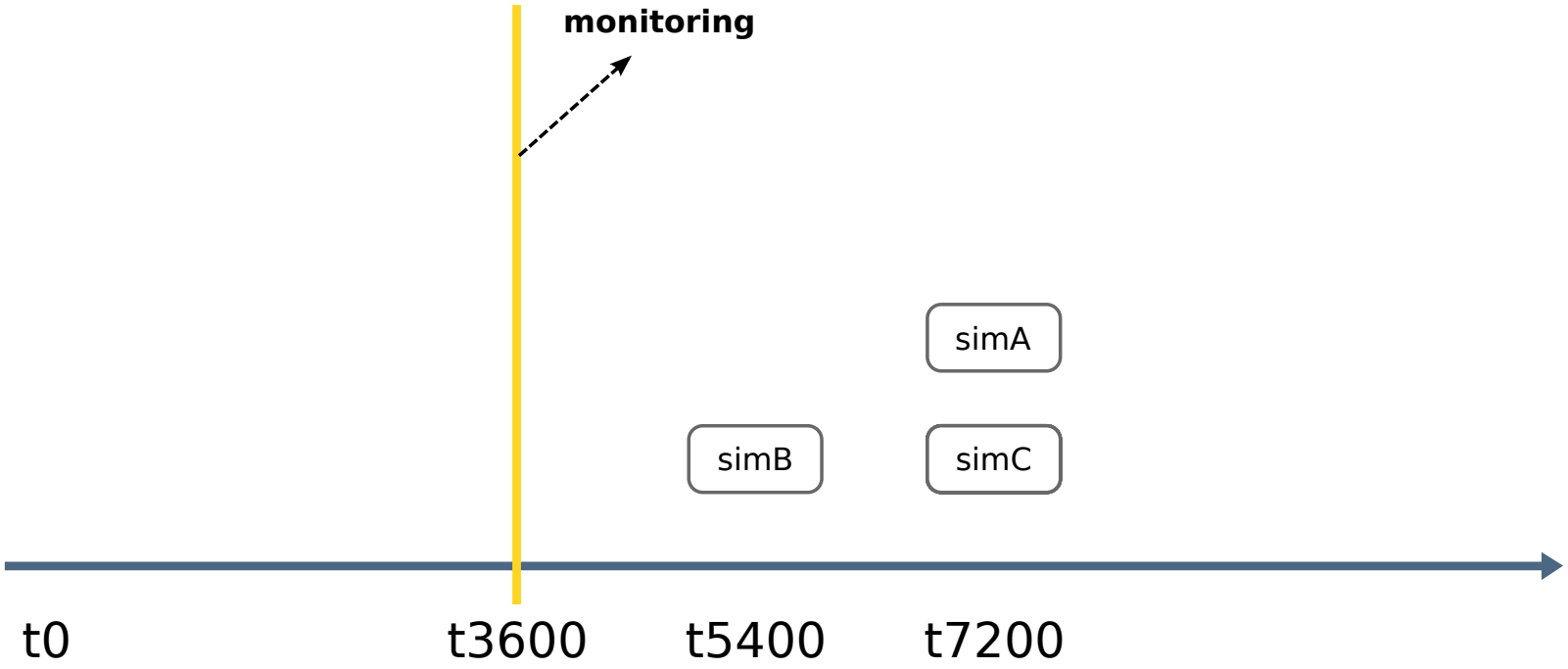
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement des simulateurs pour leurs prochaines exécutions

Déroulement d'une simulation

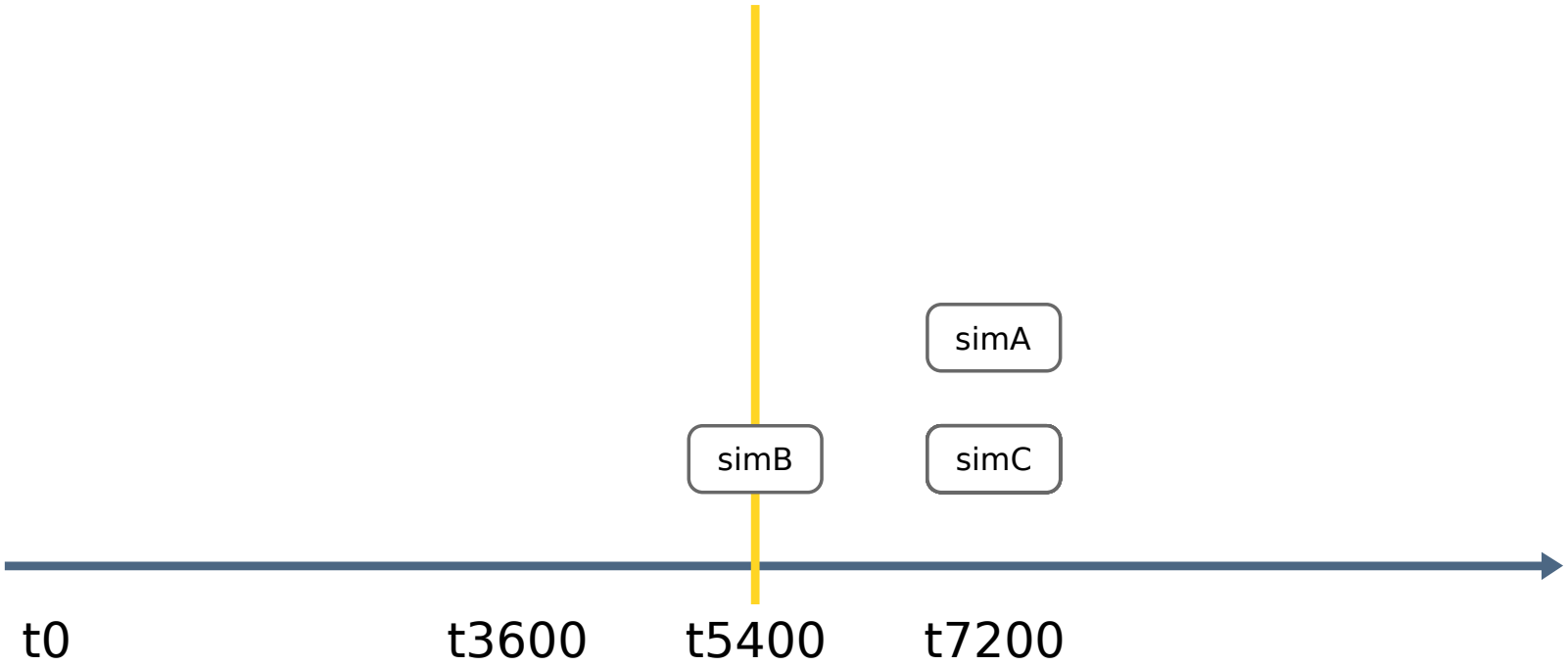
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du monitoring à t3600

Déroulement d'une simulation

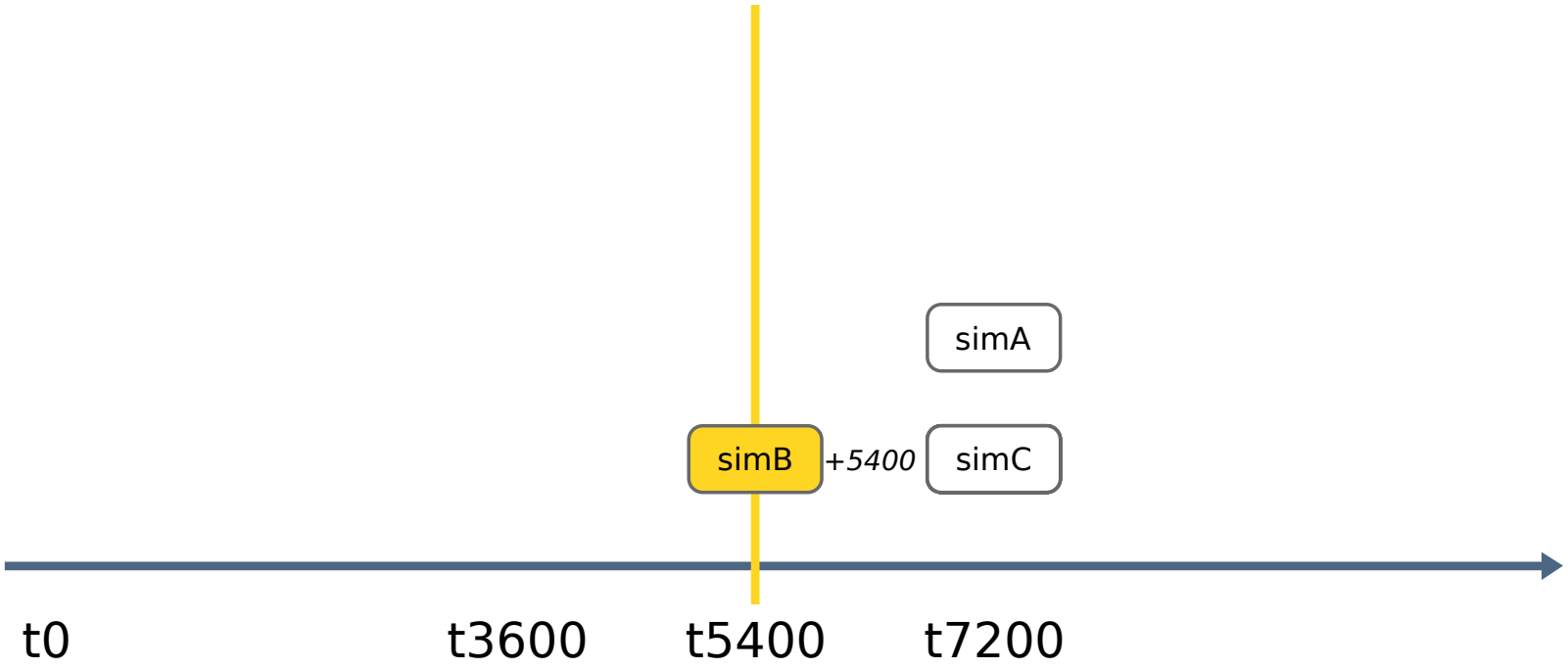
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de la simulation au prochain index de temps (t5400)

Déroulement d'une simulation

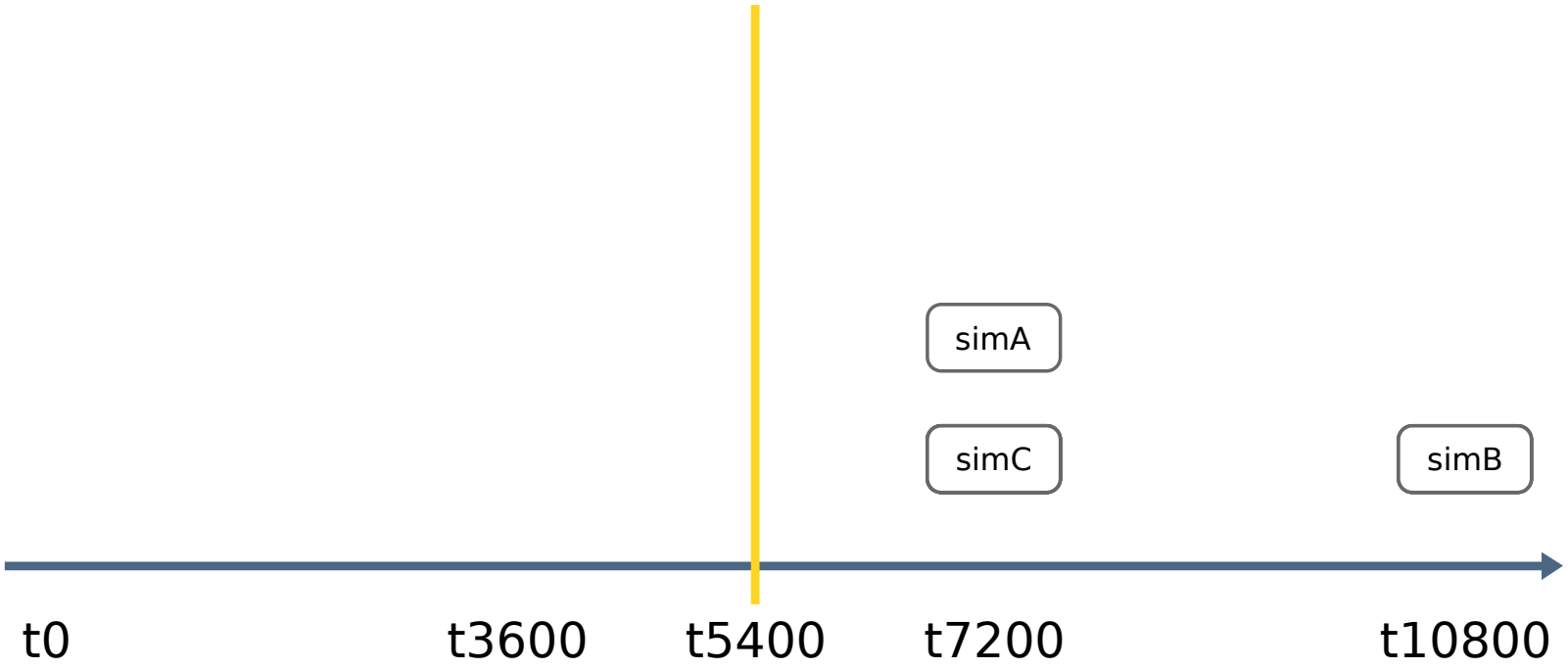
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du simulateur simB à t5400

Déroulement d'une simulation

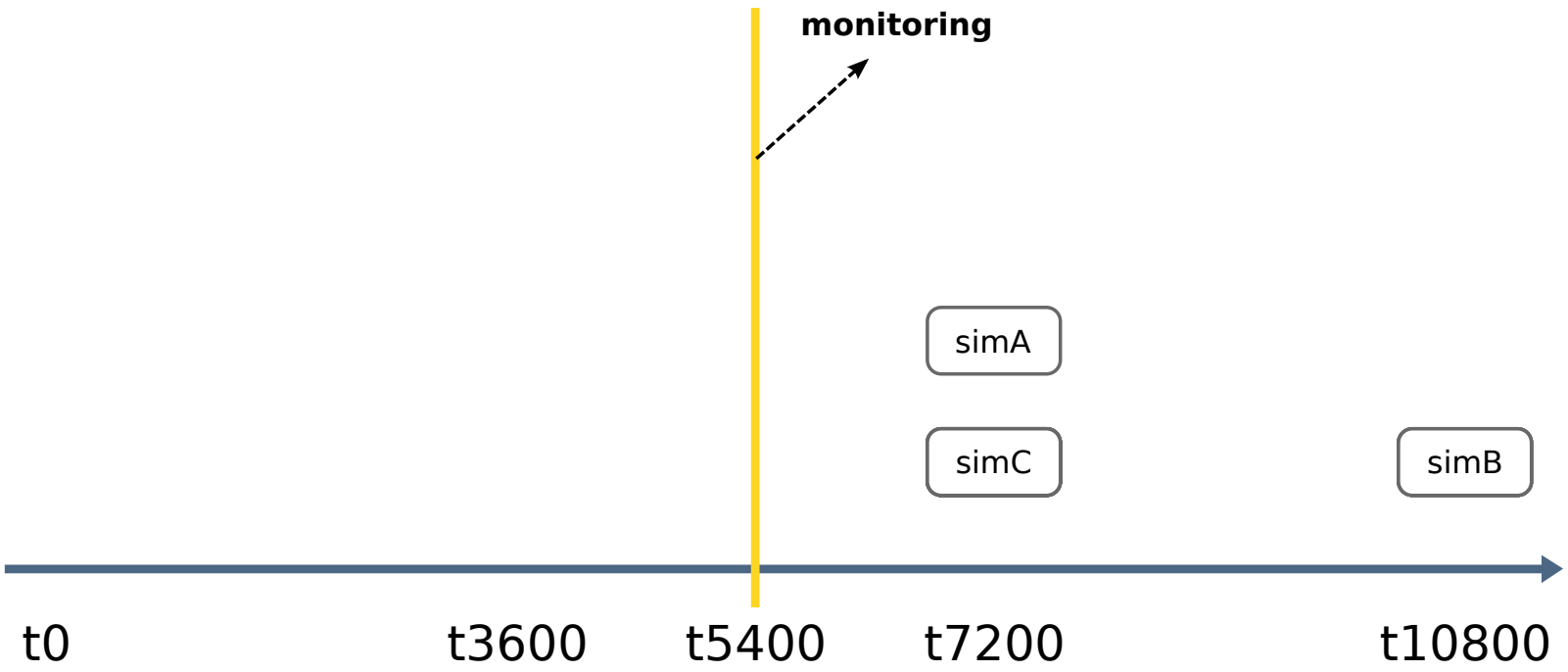
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de simB pour sa prochaine exécution

Déroulement d'une simulation

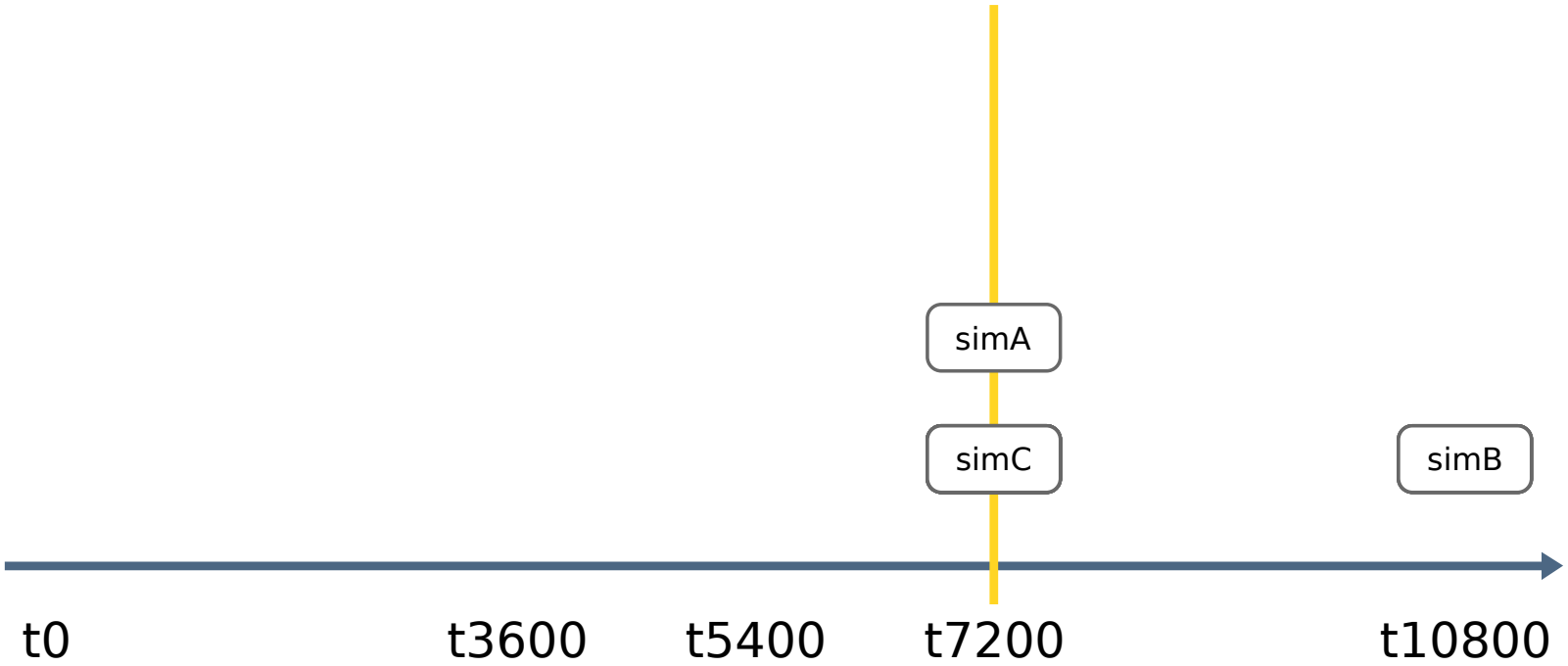
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du monitoring à t5400

Déroulement d'une simulation

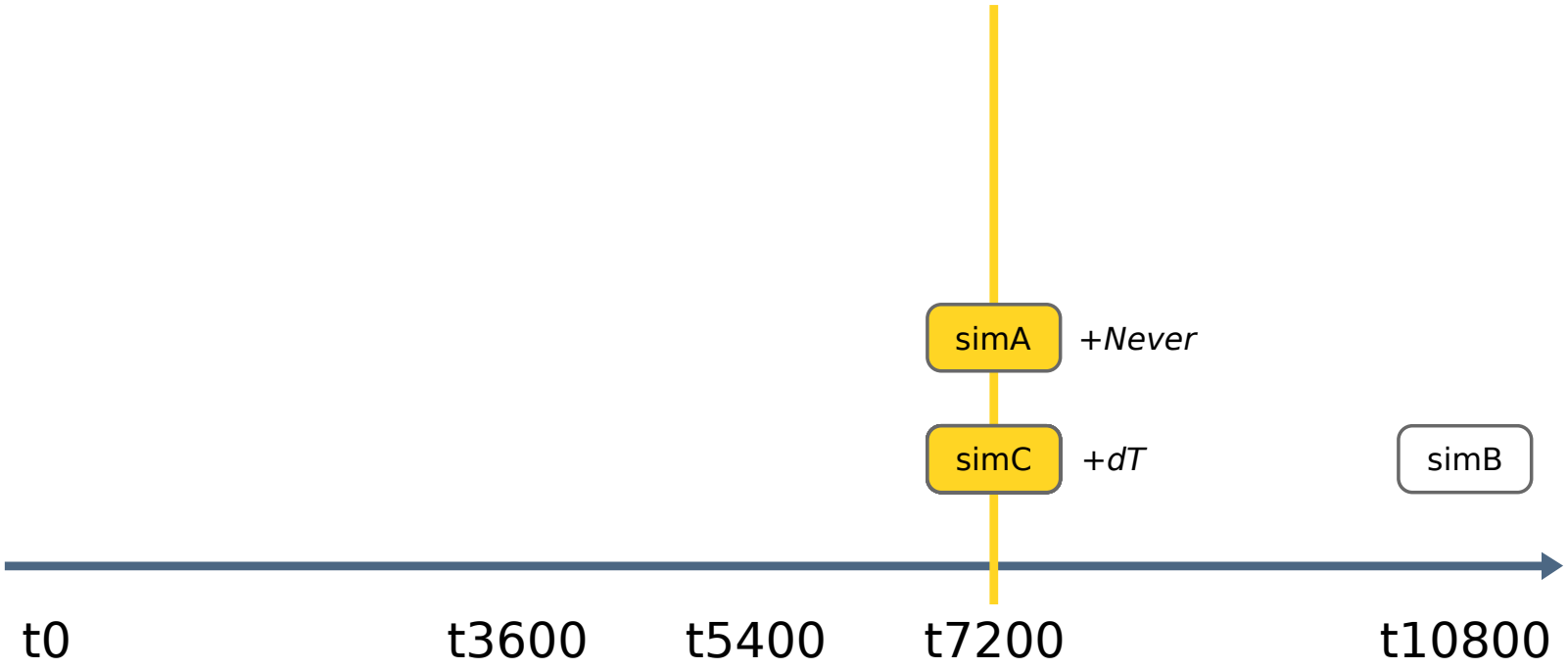
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de la simulation au prochain index de temps (t7200)

Déroulement d'une simulation

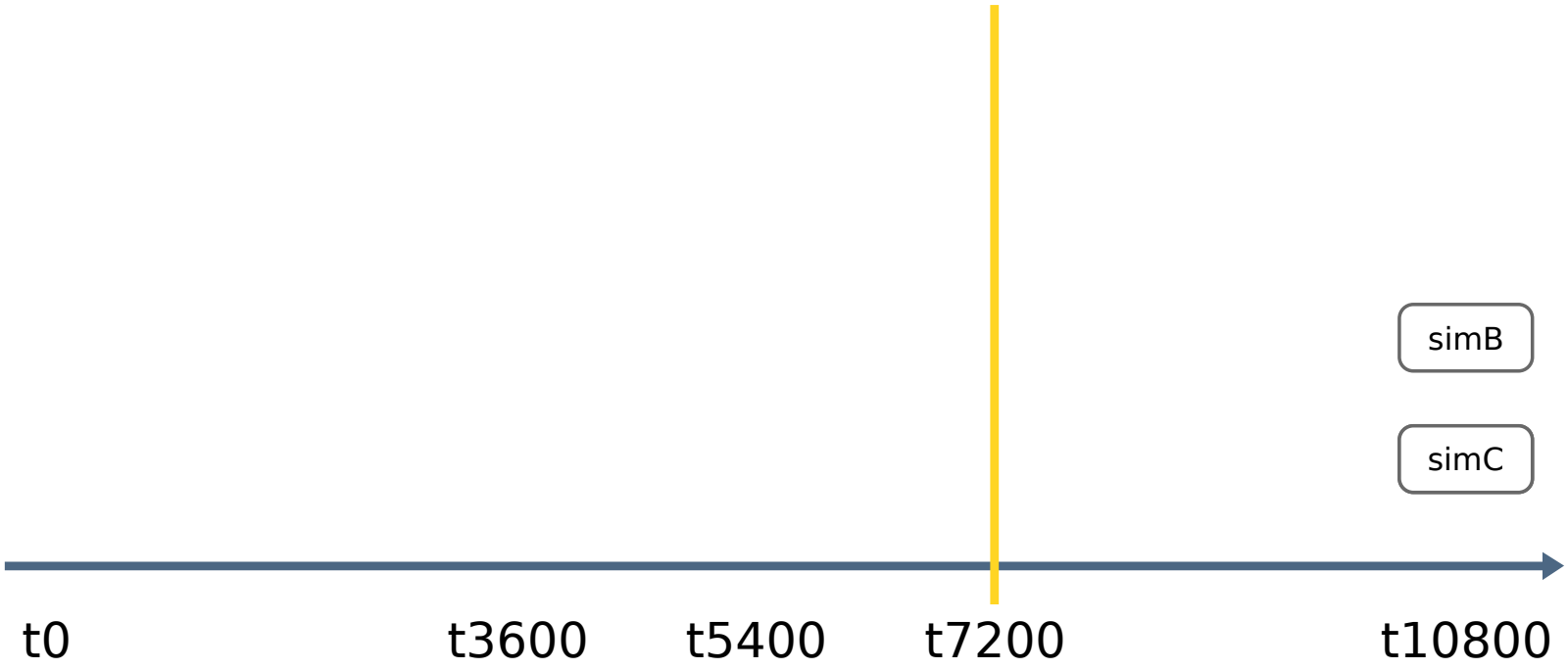
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution des simulateurs simA et simC à t7200

Déroulement d'une simulation

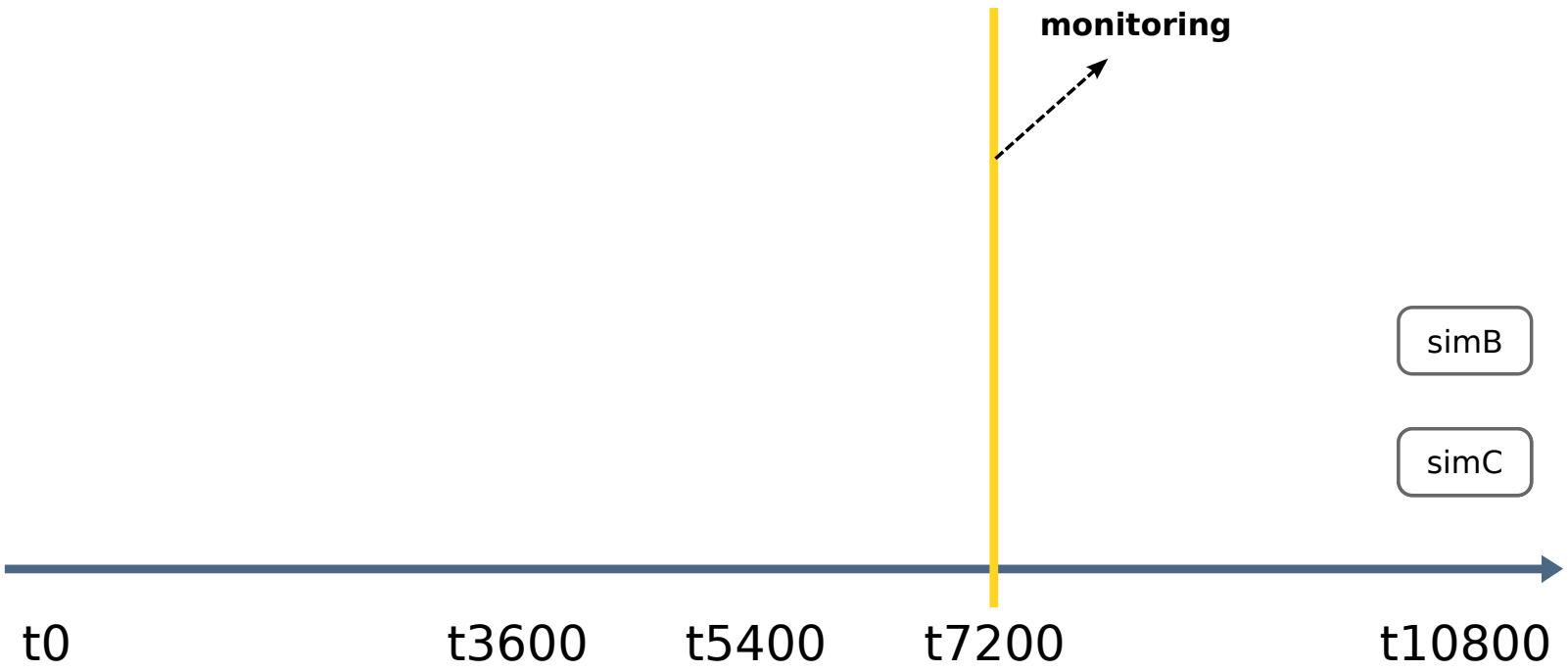
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de simC pour sa prochaines exécution (réagencement)

Déroulement d'une simulation

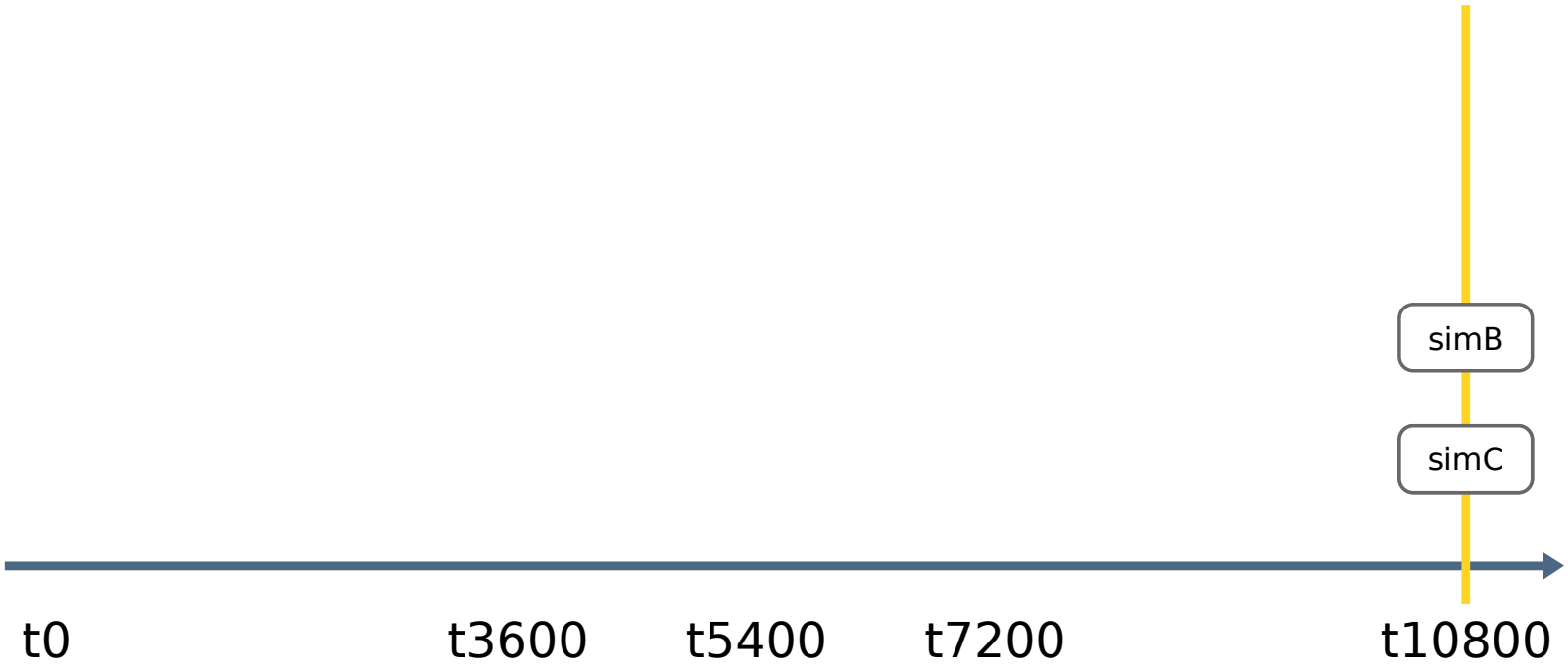
Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Exécution du monitoring à t7200

Déroulement d'une simulation

Exemple de simulation comprenant 3 simulateurs,
avec un deltat (dt) par défaut de 1h00 (3600s),
sans contrainte de planification.



Positionnement de la simulation au prochain index de temps (t10800)

Fonctionnalités complémentaires

- Vérification de cohérence des modèles construits
- Parallélisation automatisée des calculs des simulateurs (basée sur l'indépendance entre unités spatiales)
- Gestion des messages d'avertissements et d'erreurs
- Profilage de simulation
- Générateur de documentation scientifique à partir des simulateurs
- Librairie de traitement spatial (OpenFLUID-LandR)
- ...

Types de données OpenFLUID

Les données manipulées par OpenFLUID durant la simulation sont typées
Différents types de données sont définis

Types simples:

- BooleanValue : nombre booléen (0 ou 1)
- DoubleValue : nombre réel en double précision (1.52, 0.000025, ...)
- IntegerValue : nombre entier (1,18, 56325874, ...)
- StringValue : chaîne de caractères

Types composés:

- VectorValue : vecteur (1D) de nombres en double précision
- MatrixValue : matrice (2D) de nombres en double précision
- MapValue : liste clé-valeur de tout autre type de données

Variables

Les variables d'état sont centrales dans le couplage car échangées entre simulateurs tout au long de la simulation

- attachées aux unités spatiales via le graphe d'espace
- les valeurs sont associées à un index de temps dans la simulation (instant de production)
- les valeurs sont ajoutées/modifiées par les simulateurs

Un variable d'état est présente sur toutes les unités d'une même classe d'unité

Les variables d'état peuvent être de n'importe quel type OpenFLUID

Données d'entrée

Les données d'entrée sont des données initiales attachées aux unités spatiales.

- *morphologie : surface, longueur, largeur, profondeur, ...*
- *propriétés physiques : conductivité, teneur en eau,...*
- *coefficients : manning, ...*
- *paramètres descriptifs : occupation du sol, ...*

Elles sont stockées sous la forme de chaînes de caractères (StringValue), et peuvent être converties vers n'importe quel autre type OpenFLUID

Elles ne peuvent plus être modifiées une fois la simulation lancée.

Paramètres de simulateurs et d'observateurs

Les paramètres des simulateurs et des observateurs sont des paramètres qui s'appliquent à un simulateur ou un observateur en particulier

- attachés aux simulateurs ou observateurs

Ils sont stockés sous la forme de chaînes de caractères (StringValue), et peuvent être convertis vers n'importe quel autre type OpenFLUID

Evènements discrets

Les évènements discrets surviennent à un instant précis sur une unité spatiale donnée

- *opérations culturales : labours, épandage de produits, ...*
- *changement d'occupation du sol, rotation de culture, ...*
- *aménagements : curage d'un fossé, ...*

Ils portent des informations qui peuvent être traitées par les simulateurs.
Ces informations sont stockées sous la forme de chaînes de caractères (StringValue), et peuvent être converties vers n'importe quel autre type OpenFLUID

Ils peuvent être connus *à priori* sous la forme d'un calendrier, ou générés par les simulateurs au cours de la simulation.

Banque de données intégrée : Datastore

Le datastore permet d'intégrer des données non structurées, en complément des données standards

Actuellement, le datastore permet de gérér des données de type couche géographique raster ou vecteur.

A terme, intégration de nouvelle sources de données : liens BDs, chroniques, ...

Les données du datastore ne font pas partie des données de simulation, mais viennent en complément.

Générateurs de valeurs de variables

Les générateurs produisent des valeurs pour des variables
Ils sont intégrés au framework OpenFLUID, et ne nécessitent pas de simulateurs particuliers

- Générateur de valeurs constantes (fixed)
- Générateur de valeurs aléatoires (random)
- Générateur de valeurs à partir d'un fichier (inject)
- Générateur de valeurs à partir d'un fichier, avec interpolation temporelle du contenu (interp)

Cohérence du modèle couplé

Au travers de la signature de chaque simulateur, la cohérence du modèle couplé peut être vérifiée.

Cohérence spatio-temporelle des variables

- Une variable requise à un temps *t* sur une classe d'unité *u* doit être produite par un autre simulateur au même temps *t* et sur la même classe d'unité *u*
- Une même variable ne peut-être produite que par un seul simulateur

Cohérence spatiale des données d'entrée

- Une donnée d'entrée requise sur une classe d'unité *u* doit être présente sur la classe d'unité *u*

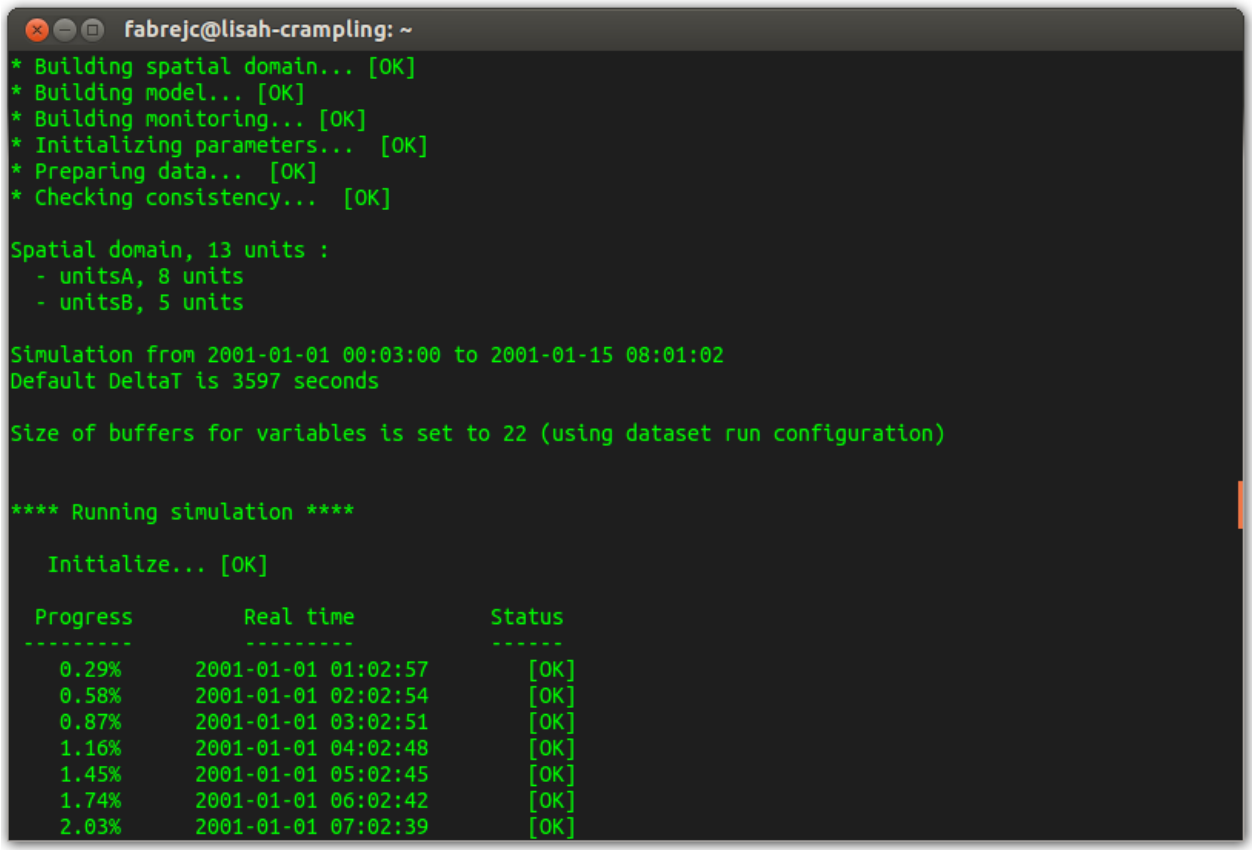
La cohérence du modèle couplé est automatiquement vérifiée par le framework OpenFLUID à la mise en place du modèle couplé

Plan

Application logicielle en ligne de commande

Exécution de simulations en ligne de commande à partir d'un jeu de données d'entrée

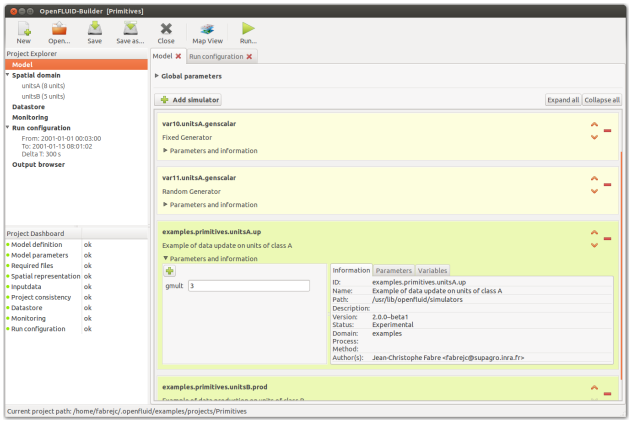
- utilisation simple ou en batch, sur cluster de calcul, . . .



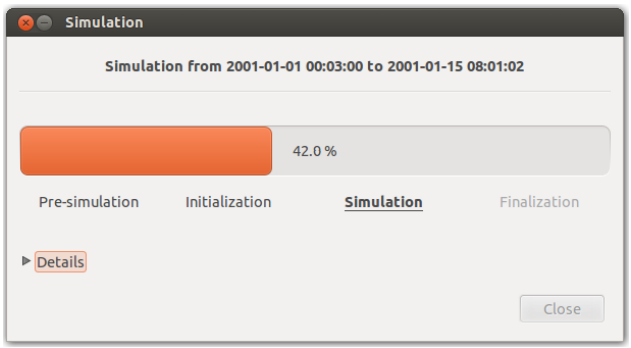
Application logicielle graphique (OpenFLUID-Builder)

Interface graphique pour la construction, la paramétrisation, l'exécution de simulation et la visualisation de résultats

- Construction simple de modèles et de représentations du paysage
- Paramétrisation
- Simulation
- Accès aux résultats



Extensible par ajout de Builder-extensions (plug-ins) : import/export de données (fichiers, BDs, ...), visualisations, utilisations d'outils externes, ...



Package ROpenFLUID

Utilisation d'OpenFLUID depuis l'environnement GNU R

- Chargement de jeux de données
- Paramétrage (données d'entrées, paramètres de simulateurs, ...)
- Exécution de simulations
- Exploitation des résultats

⇒ Profiter des fonctionnalités d'exploration de simulations sous R

- Analyse de sensibilité
- Propagation d'incertitude
- ...

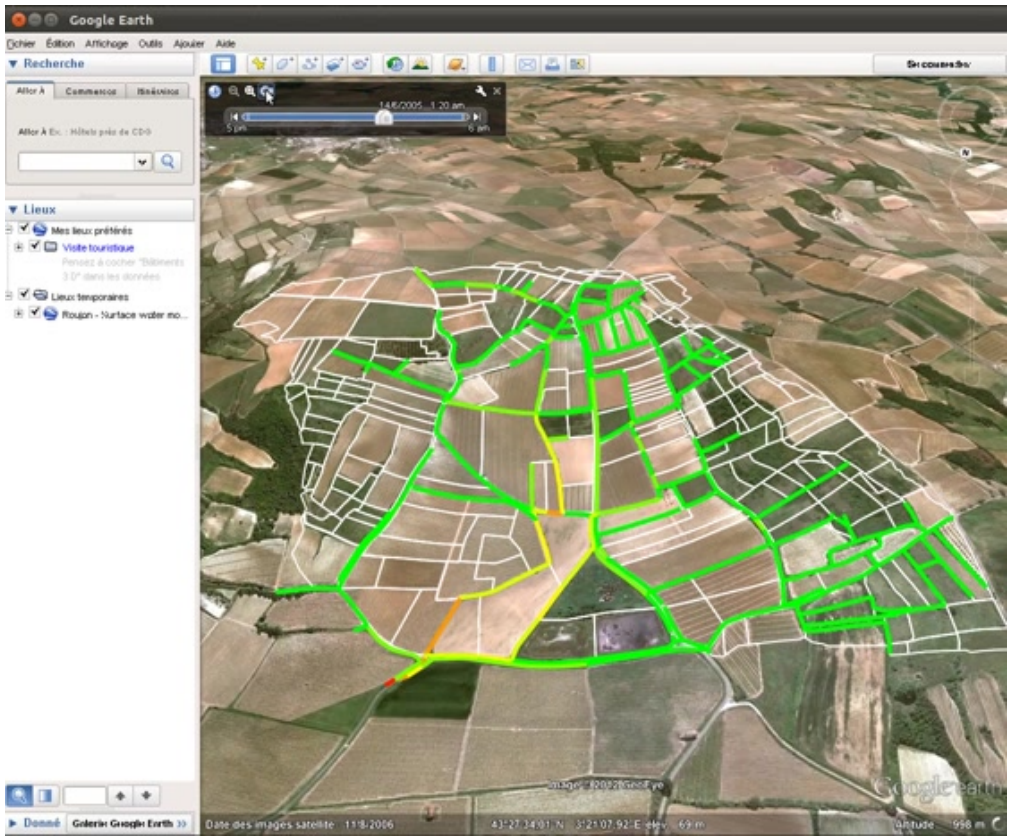
Lancement d'une simulation sous R

```
library('ROpenFLUID')
ofsim = OpenFLUID.loadDataset('/path/to/dataset')
OpenFLUID.runSimulation(ofsim)
data = OpenFLUID.loadResult(ofsim,'SU',15,'water.surf.Q.downstream')
```

Interfaçage avec des outils externes

Visualisation, SIG:

- VisIt
- Goole Earth
- Graphviz
- Qgis, GRASS
- ...



Analyse:

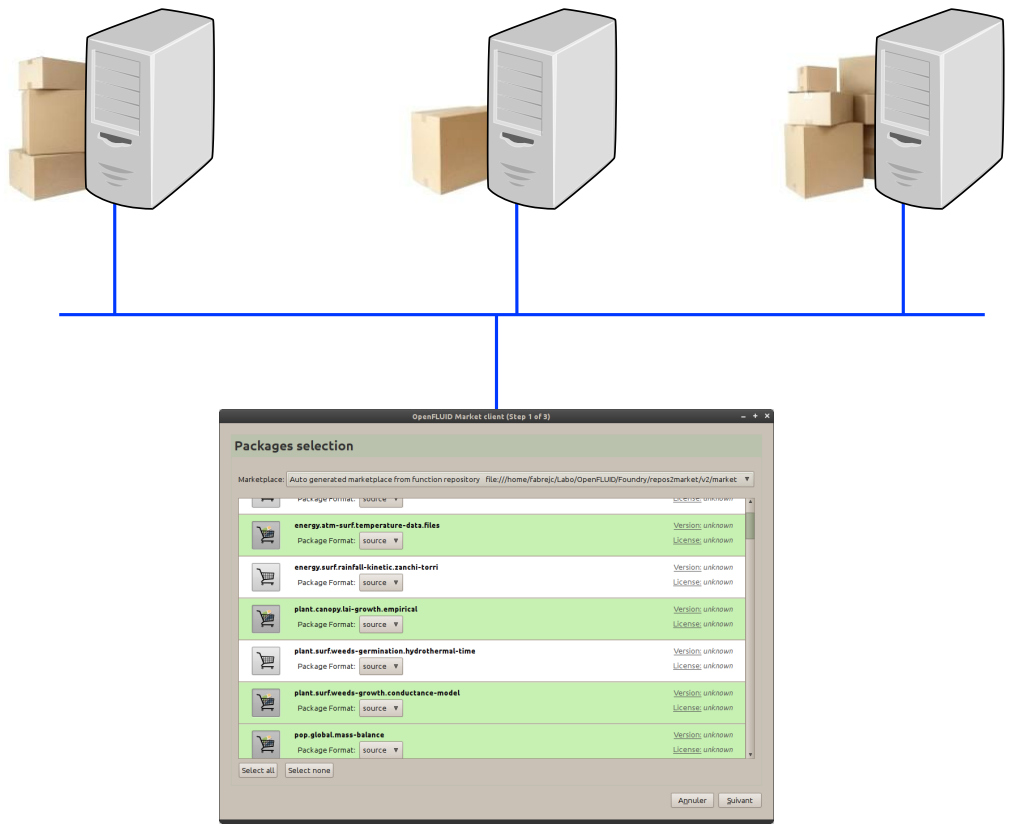
- GNU R
- PEST
- Octave
- ...

Via des observateurs

ou par post-traitement de données de simulation

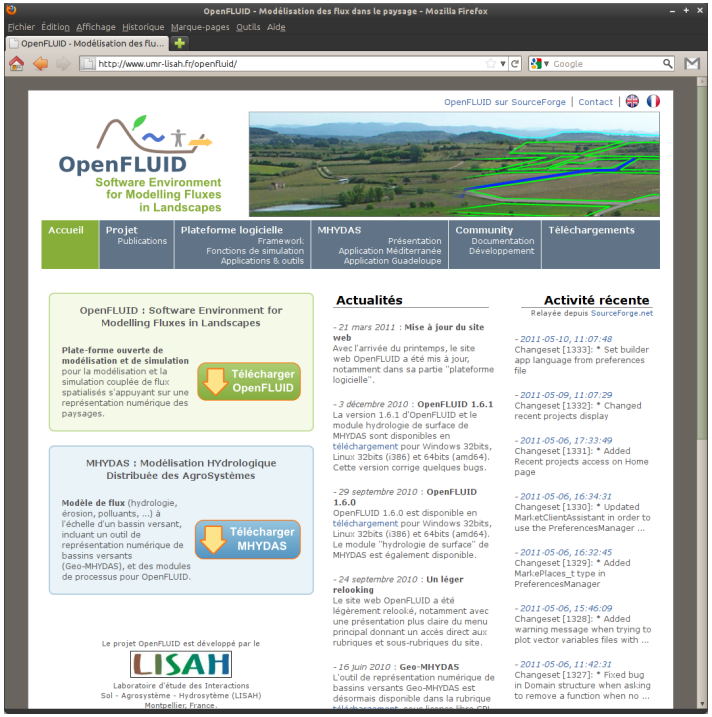
OpenFLUID Market

Installation automatisée de simulateurs, d'observateurs et de jeux de données, mis à disposition sur un dépôt (Marketplace OpenFLUID) local ou via internet (HTTP)



Ressources OpenFLUID en ligne

http://www.openfluid-project.org/



Une présentation générale

- Projet, applications, téléchargements

Un espace collaboratif (Community)

- Documentations utilisateurs et développeurs
- Guides, exemples de code, . . .

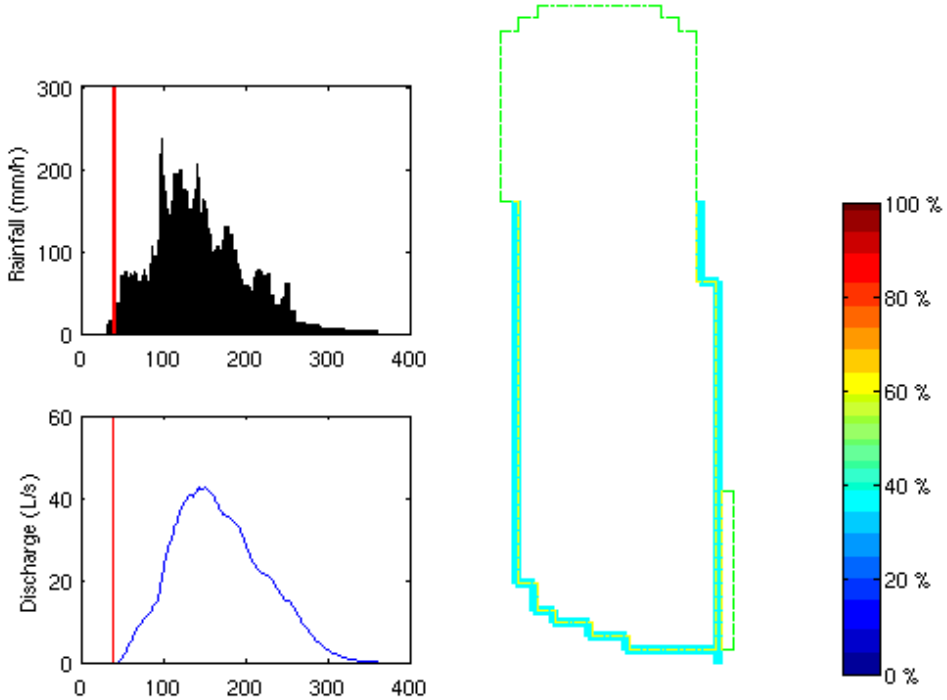
@OpenFLUID

Ruissellement et propagation

Parcelle AW6 - Roujan *(X. Louchart)*

Parcelle AW6 - Roujan :

- 1200 m^2
- 1070 unités spatiales



Simulation :

- sur 1 heure
- pas de temps : 10 s

Transfert en réseau hydrographique

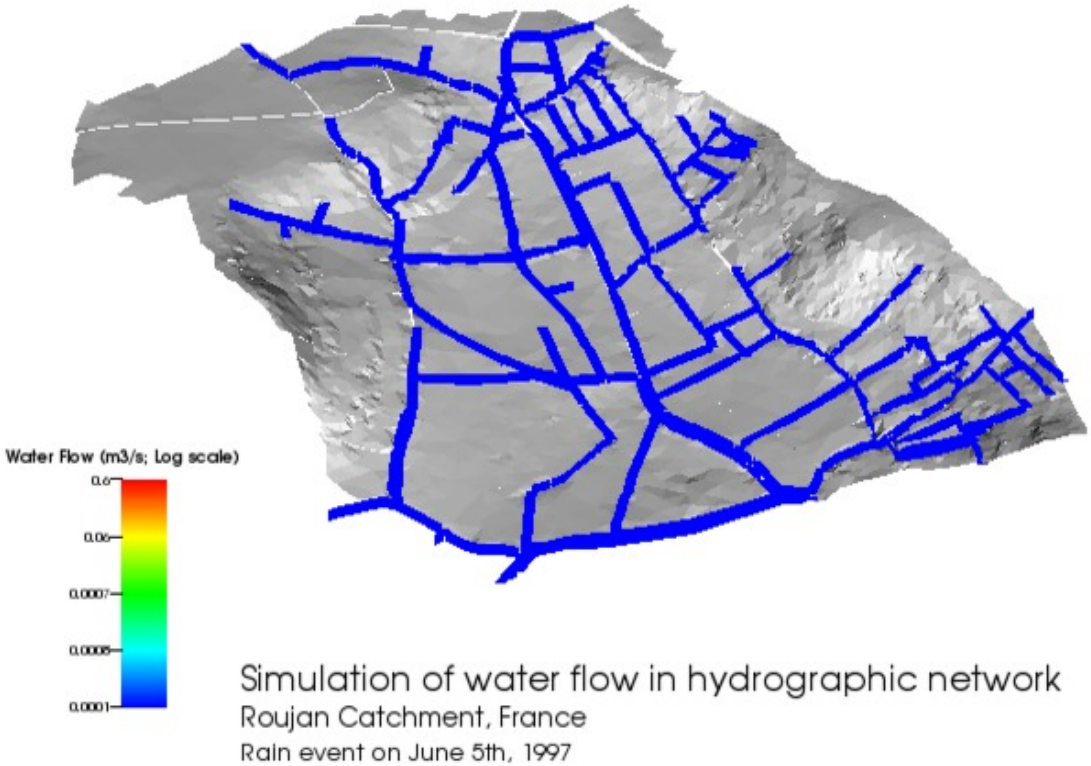
Bassin versant de Roujan *(R. Moussa)*

Bassin versant de Roujan :

- 0.91 km^2
- ~600 unités spatiales

Simulation :

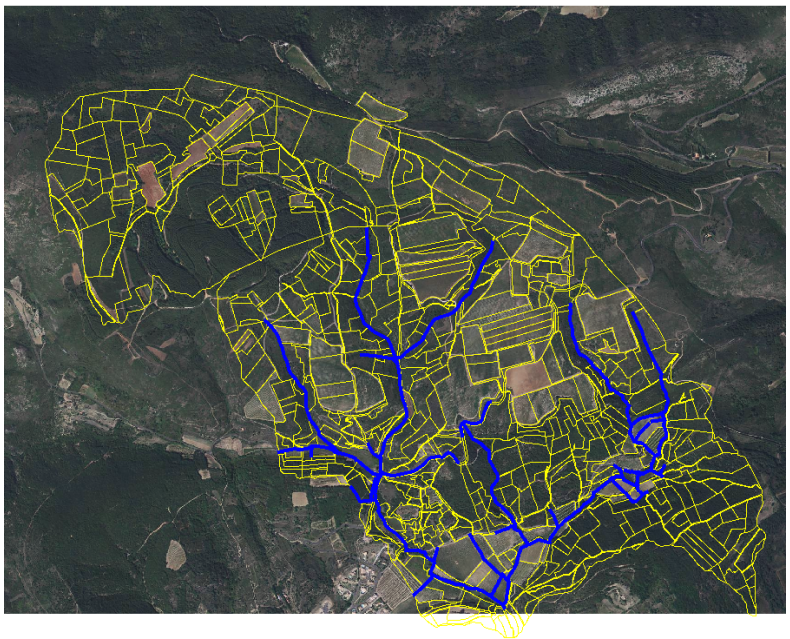
- sur 7 heures
- pas de temps : 60 s

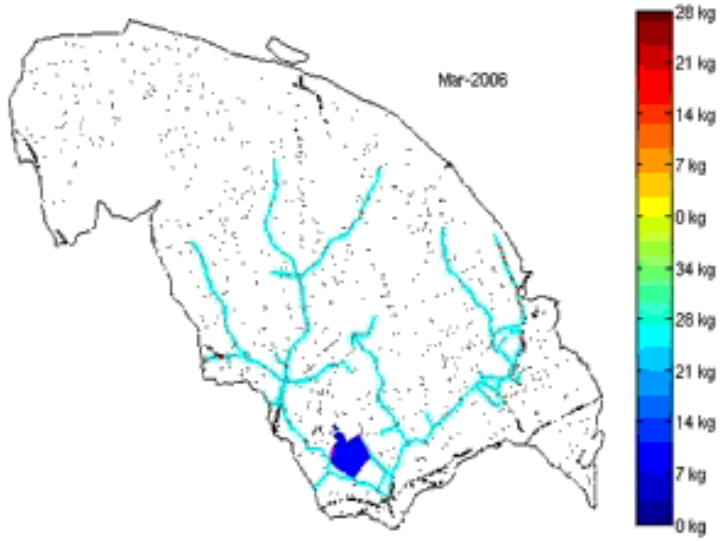


Simulation of water flow in hydrographic network
Roujan Catchment, France
Rain event on June 5th, 1997

Evolution des teneurs en pesticides dans le sol

Bassin versant de Neffiès *(X. Louchart)*





Bassin versant de Neffiès :

- 4 km^2
- ~800 unités spatiales

Simulation :

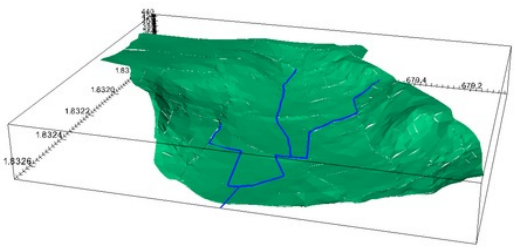
- sur 3 ans, selon pratiques de pulvérisation observées
- 10 insecticides et fongicides
- pas de temps : 24 H

Aménagement d'un réseau de fossés

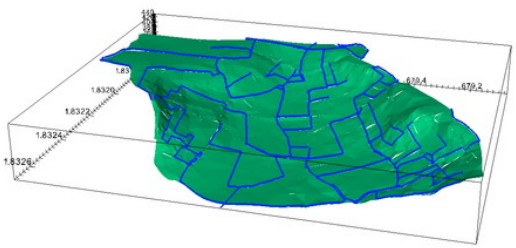
Bassin versant de Roujan *(F. Levavasseur)*

Scenarios d'aménagement du réseau de fossés sur le bassin versant de Roujan (0.91 km^2)

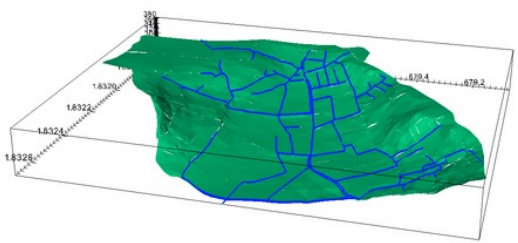
- évacuer l'eau des parcelles
- limiter les pertes en sol dues à l'érosion



Faible densité
38 fossés



Haute densité
322 fossés



Densité moyenne
223 fossés

Travaux présentés au Salon International de l'Agriculture (SIA, 25 février au 4 mars 2012, Paris)

Plan

Jeu de données d'entrée : organisation et formats

Un jeu de données d'entrée OpenFLUID est constitué d'un à plusieurs fichiers d'entrées standardisés au format XML, portant l'extension `.fluidx`.

Ce(s) fichier(s) doivent définir les sections suivantes

- `<domain>` : définition de la RNP, des paramètres et évènements distribués
- `<model>` : définition du modèle couplé
- `<monitoring>` : définition du monitoring
- `<run>` : configuration de la simulation

Il est possible pour les simulateurs et observateurs de gérer leurs propres fichiers d'entrée.

Jeu de données d'entrée : un seul ⇔ plusieurs fichiers

dataset.fluidx

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <domain>
    <!-- ici la definition et la parametrisation de l espace -->
  </domain>

  <model>
    <!-- ici la definition du modele -->
  </model>

  <monitoring>
    <!-- ici la édfinition du monitoring -->
  </monitoring>

  <run>
    <!-- ici la configuration de la simulation -->
  </run>
</openfluid>
```

Jeu de données d'entrée : un seul ⇔ plusieurs fichiers

model.fluidx

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <model>
    <!-- ici la definition
         du modele -->
  </model>
</openfluid>
```

domain.fluidx

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <domain>
    <!-- ici la definition
         et la parametrisation
         de l espace -->
  </domain>
</openfluid>
```

run_output.fluidx

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <run>
    <!-- ici la configuration de la simulation -->
  </run>
  <monitoring>
    <!-- ici la édfintion du monitoring -->
  </output>
</monitoring>
```

Section <domain>

La section domain est composée de 3 sous-sections:

- <definition> : définition des unités spatiales qui composent la représentation du paysage, avec leur topologie (connectivité)
- <inputdata> : données d'entrée distribuées, attachés à chaque unité spatiale
- <calendar> : évènements distribués, attachés à chaque unité spatiale

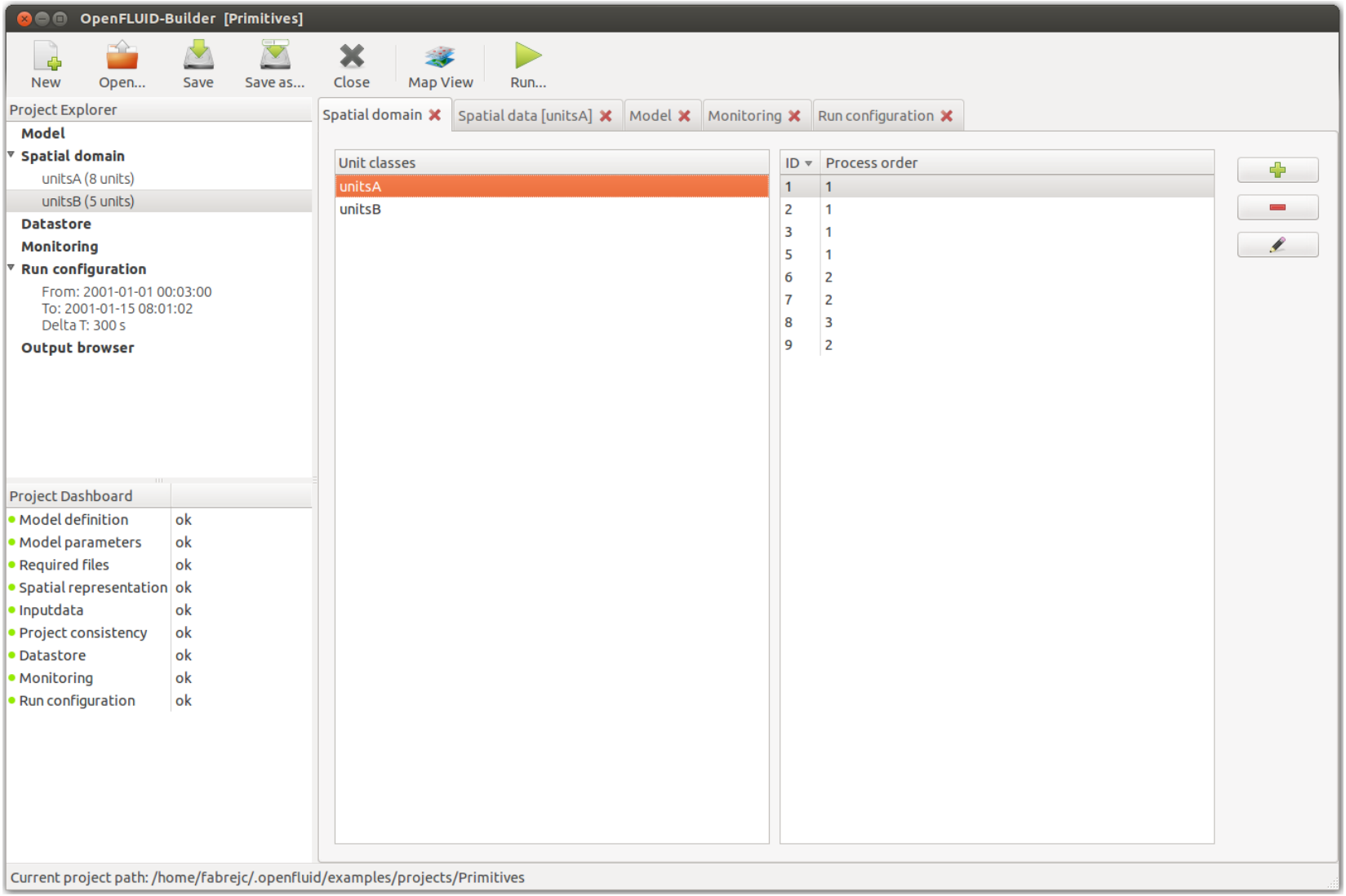
Section <domain>, sous-section <definition>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <domain>
    <definition>
      <unit class="SU" ID="1" pcsorder="1">
        <to class="SU" ID="5" />
      </unit>
      <unit class="SU" ID="5" pcsorder="2">
        <to class="RS" ID="9" />
      </unit>
      <unit class="RS" ID="9" pcsorder="1" />
    </definition>
  </domain>
</openfluid>
```

- déclaration d'une nouvelle unité : balise <unit> avec les attributs précisant la classe de l'unité, son ID et son ordre de traitement
- déclaration d'une connexion avec une autre unité : sous-balise <to> de la balise <unit>
- déclaration d'un lien de parenté avec une autre unité : sous-balise <childof> de la balise <unit>

Section <domain>, sous-section <definition>



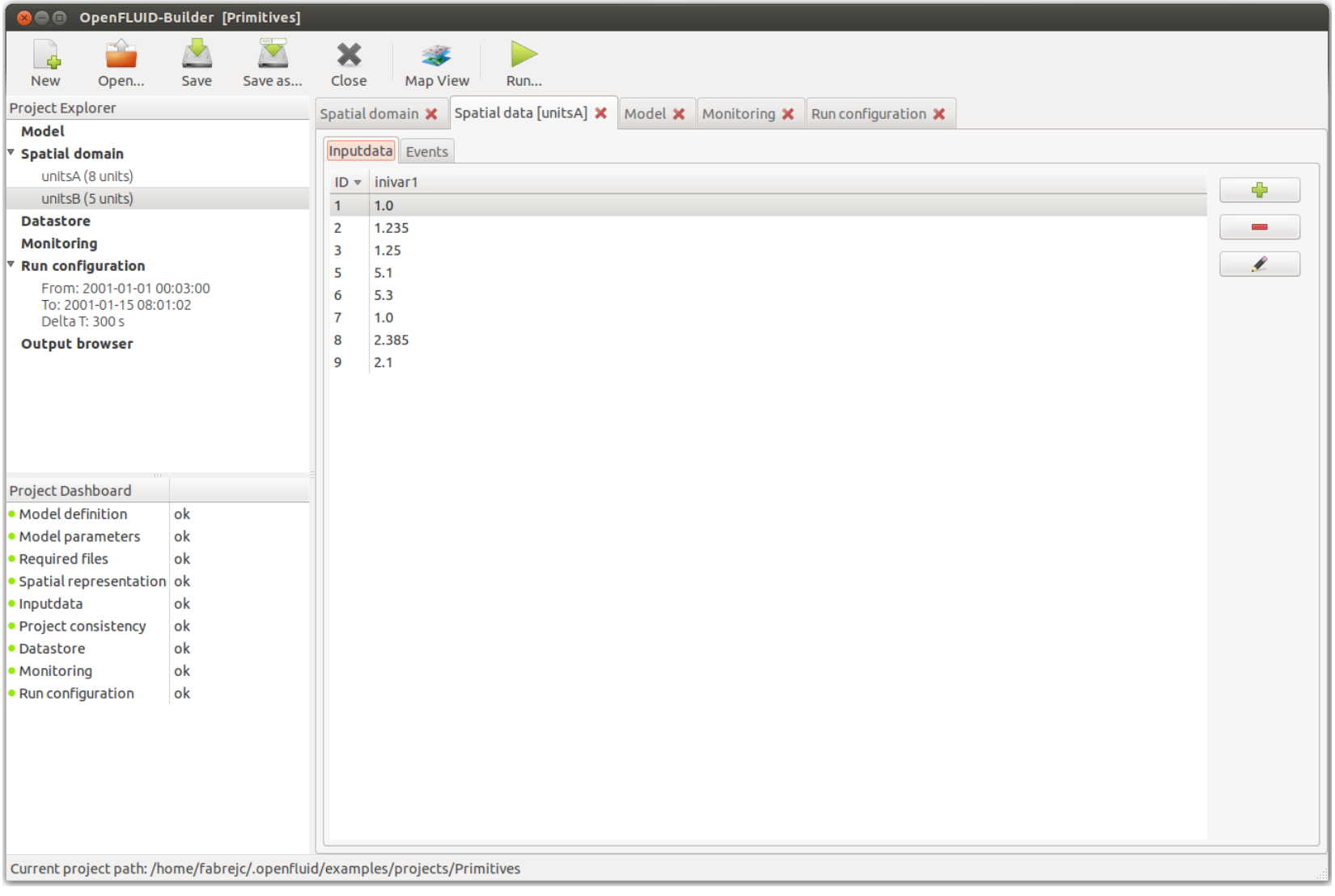
Section <domain>, sous-section <inputdata>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <domain>
    <inputdata unitclass="SU" colorder="area;slope;flowdist" />
1 4813.344   0.06265    14.366
2 293.982    0.27129    18.468
    </inputdata>
  </domain>
</openfluid>
```

- données en colonnes entre balises <inputdata> et </inputdata> (1ère colonne = ID de l'unité)
- ordre des colonnes (après colonne ID) précisé au travers de l'attribut colorder

Section <domain>, sous-section <inputdata>



Section <domain>, sous-section <calendar>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <calendar>
    <event unitclass="SU" unitID="1" date="1997-03-30␣12:00:00">
      <info key="molecule" value="diuron"/>
      <info key="percent_area" value="100"/>
      <info key="rate_ha" value="1"/>
    </event>
  </calendar>
</openfluid>
```

- un évènement est déclaré au travers d'une balise <event>, et est associé à une unité spatiale et à une date.
- il porte de 0 à n informations déclarées par une balise info, chaque valeur (attribut value) étant identifiée par une clé (attribut key)

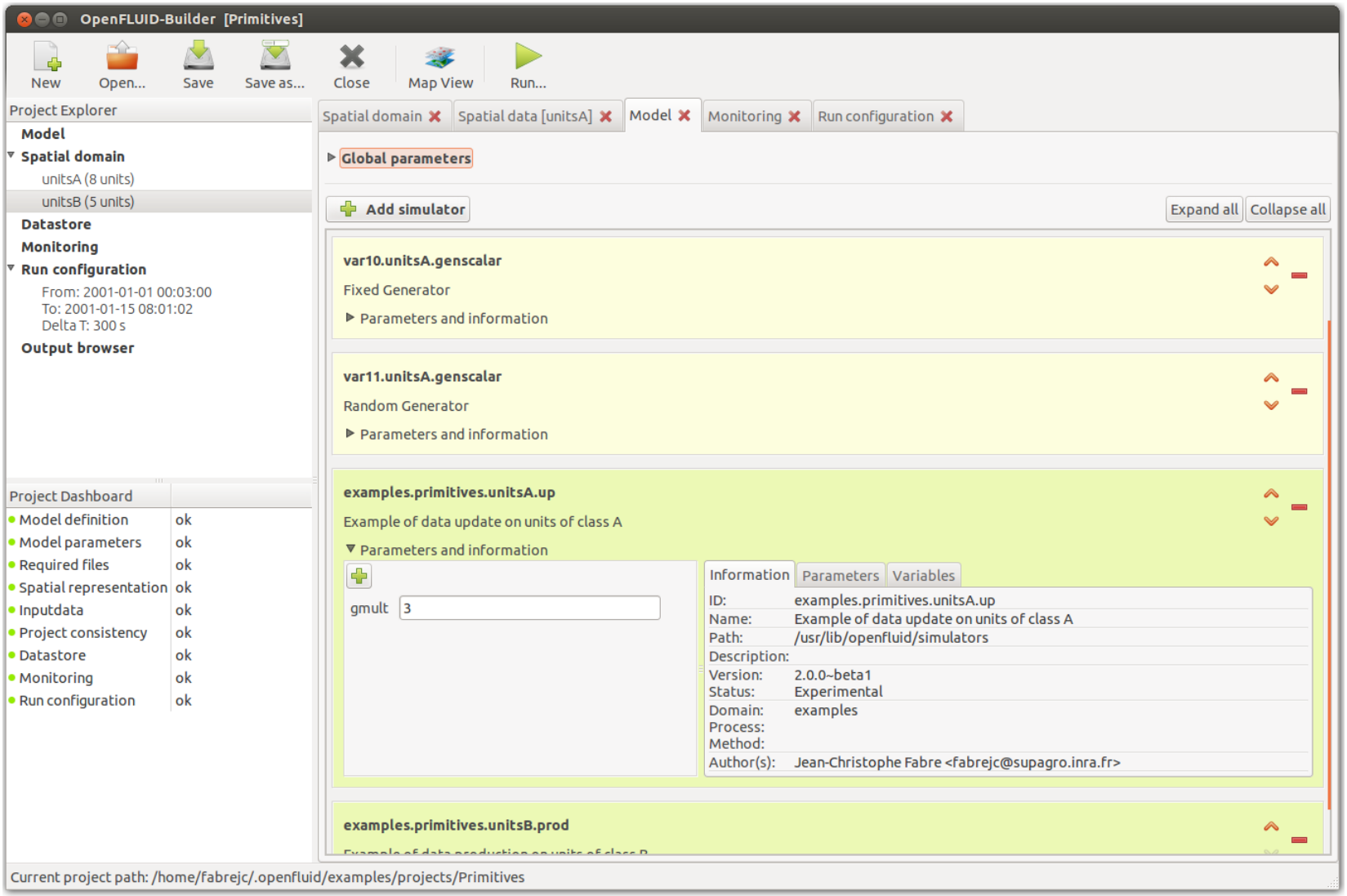
Section <model>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <model>
    <simulator ID="water.atm-surf.rain-su.files" />
    <simulator ID="water.surf-uz.runoff-infiltration.mseytoux" >
      <param name="resstep" value="0.000005" />
    </simulator>
  </model>
</openfluid>
```

- déclaration d'un simulateur : balise <simulator> avec attribut ID
- déclaration d'un paramètre de simulateur : balise <param> avec attributs name pour le nom du paramètre et value pour sa valeur

Section <model>



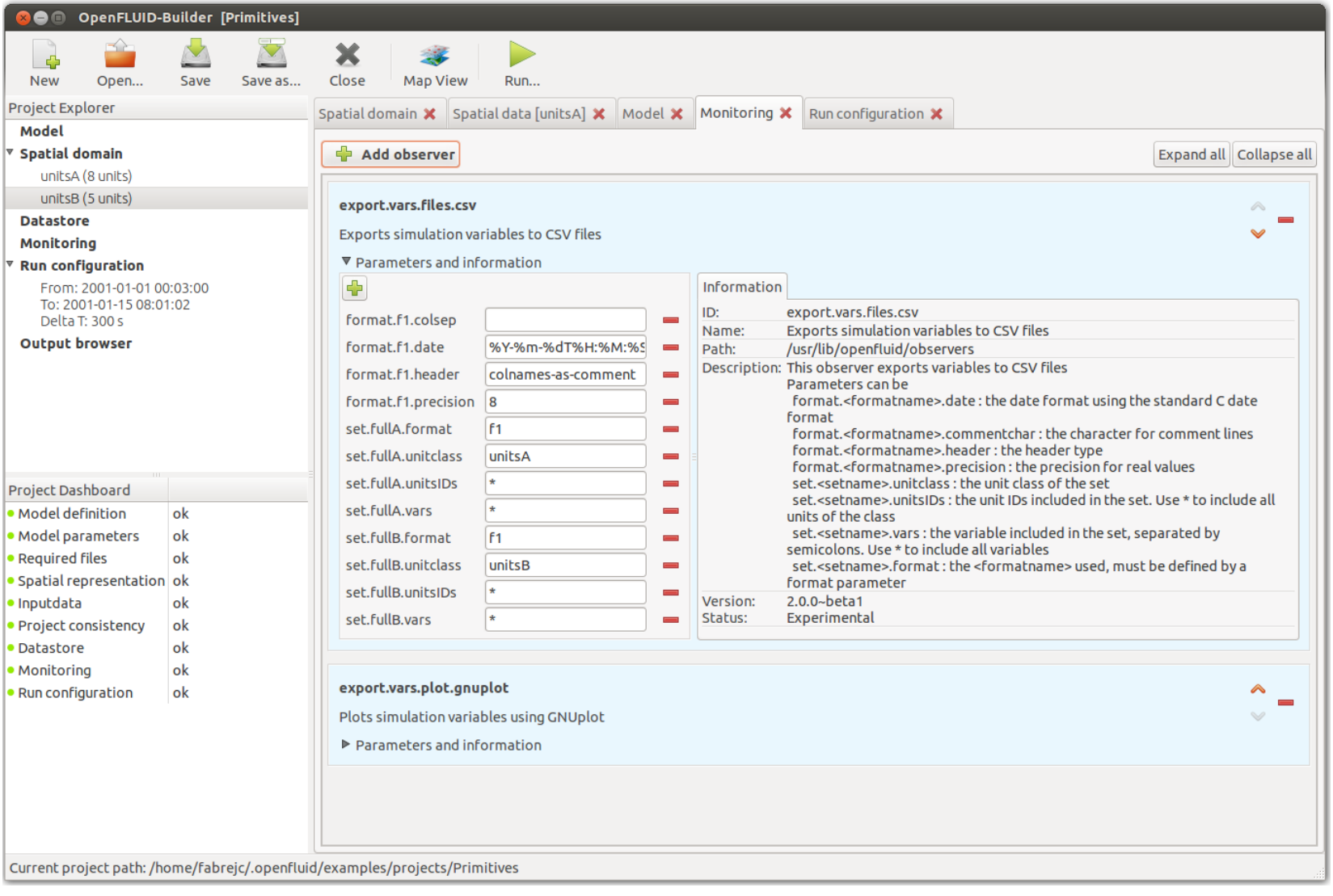
Section <monitoring>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <monitoring>
    <observer ID="export.vars.files.csv">
      <!-- params here -->
    </observer>
    <observer ID="export.vars.files.kml-anim">
      <!-- params here -->
    </observer>
  </output>
</openfluid>
```

- configuration du monitoring
- exports de données de simulations, contrôles, . . .

Section <monitoring>



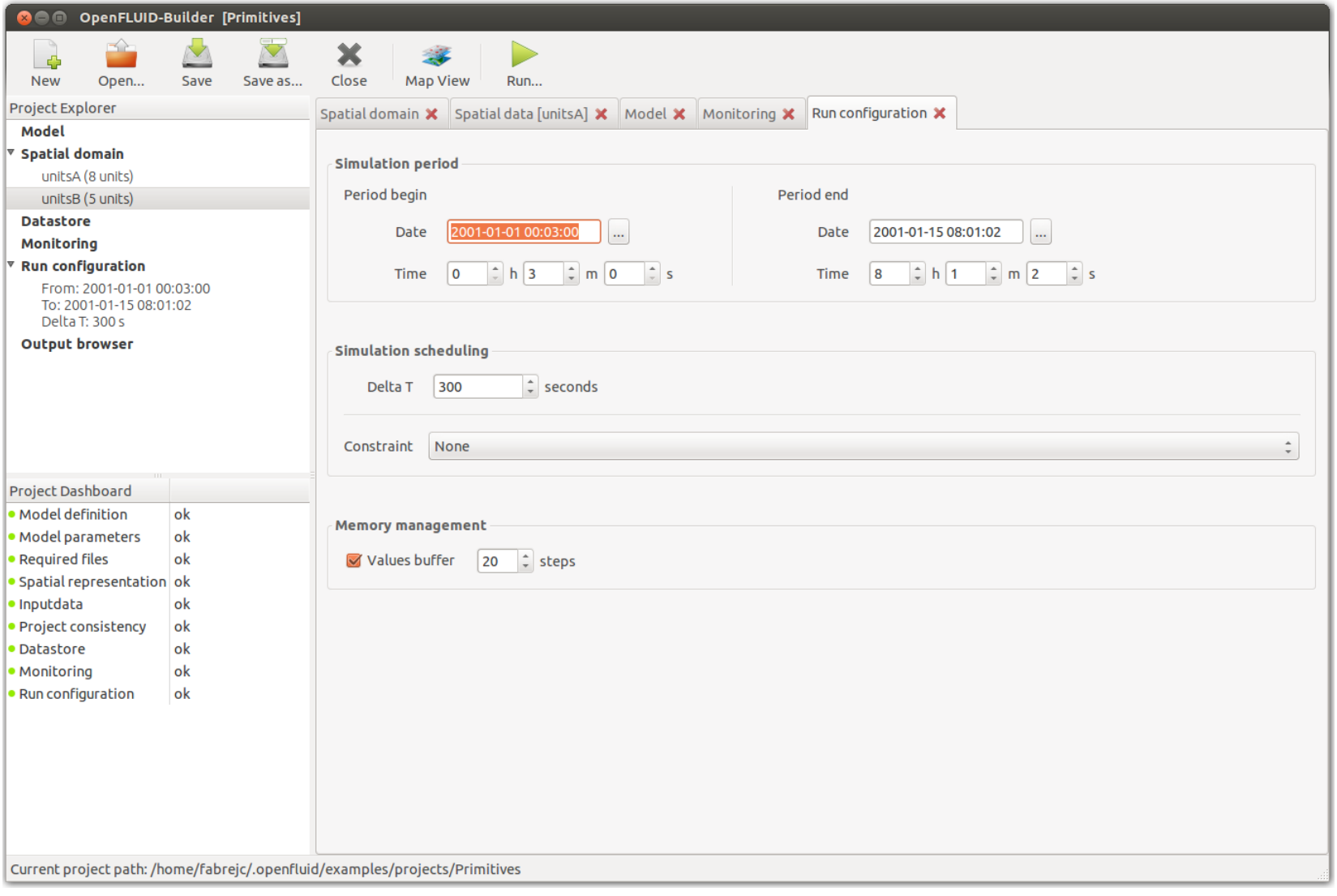
Section <run>

Exemple

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
  <run>
    <scheduling deltat="60" constraint="none" />
    <period begin="1997-03-29 03:00:18" end="1997-04-01 16:23:21" />
    <!-- <valuesbuffer size="10" /> -->
  </run>
</openfluid>
```

- la configuration de la simulation comprend obligatoirement une période de simulation (<period>) et un pas de temps d'échange (<deltat>)
- la gestion de la mémoire peut être affinée au travers de la balise <valuesbuffer> (optionnelle)

Section <run>


OpenFLUID-Builder [Primitives]
New
Open...
Save
Save as...
Close
Map View
Run...
Project Explorer
Model
Spatial domain
unitsA (8 units)
unitsB (5 units)
Datastore
Monitoring
Run configuration
From: 2001-01-01 00:03:00
To: 2001-01-15 08:01:02
Delta T: 300 s
Output browser
Project Dashboard
Model definition ok
Model parameters ok
Required files ok
Spatial representation ok
Inputdata ok
Project consistency ok
Datastore ok
Monitoring ok
Run configuration ok
Spatial domain
Spatial data [unitsA]
Model
Monitoring
Run configuration
Simulation period
Period begin
Date 2001-01-01 00:03:00
Time 0 h 3 m 0 s
Period end
Date 2001-01-15 08:01:02
Time 8 h 1 m 2 s
Simulation scheduling
Delta T 300 seconds
Constraint None
Memory management
Values buffer 20 steps
Current project path: /home/fabrejc/.openfluid/examples/projects/Primitives

Plan

Environnement

Répertoires par défaut (sous Unix/Linux) de l'utilisateur

- OpenFLUID personnel : `/home/username/.openfluid`
- Données en entrée : `/home/username/.openfluid/OPENFLUID.IN`
- Données résultats : `/home/username/.openfluid/OPENFLUID.OUT`
- Simulateurs : `/home/username/.openfluid/simulators`
- Observateurs : `/home/username/.openfluid/observers`

Variables d'environnement

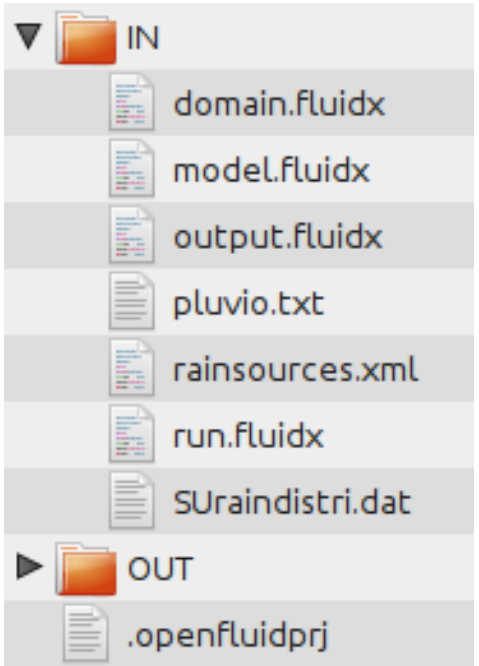
- `$OPENFLUID_SIMS_PATH` : chemins(s) de recherche supplémentaire(s) de simulateurs

Notion de projet OpenFLUID

Un projet OpenFLUID est un répertoire qui rassemble au minimum:

- un jeu de données d'entrée (sous-répertoire `IN`), contenant notamment les fichiers `.fluidx`
- un ou plusieurs jeu de résultats (sous-répertoire(s) `OUT`)
- des métadonnées du projet (fichier caché `.openfluidprj`)

  Nom du projet, auteurs, dates de création et de modification



OpenFLUID Builder recherche des projets existants dans un répertoire par défaut, qui peut être configuré depuis l'application

Principales options de la ligne de commande

Option	Description
`-i`	chemin du jeu de données d'entrée
`-o`	chemin de sauvegarde des résultats
`-w`	chemin de projet OpenFLUID
`-p`	chemins de recherche de simulateurs
`-n`	chemins de recherche d'observateurs
`-q`	affichage silencieux pendant la simulation
`-v`	affichage détaillé pendant la simulation
`-k`	activation du profiling de simulation
`-f`	liste des simulateurs disponibles
`-r`	rapport détaillé sur les simulateurs disponibles
`-e`	liste des observateurs disponibles
`-l`	rapport détaillé sur les observateurs disponibles
`--version`	retourne la version d'OpenFLUID

- Les options de la ligne de commande peuvent être combinées

Buddies

Les buddies ou "compagnons" sont des outils périphériques, accompagnant OpenFLUID dans sa mise en oeuvre
4 compagnons sont intégrés à OpenFLUID:

- sim2doc : génération automatique de documentation scientifique pour les simulateurs
- convert : convertisseur de formats de jeux de données d'entrée
- newsim : générateur de code source de simulateurs
- newdata : générateur de jeux de données vides formatés

Utilisation en ligne de commande `openfluid` avec les options `--buddy`, `--buddyopts`, `--buddyhelp`

Buddy `sim2doc`

Utilisation du code source des simulateurs pour en extraire les informations nécessaires

- Lecture de la signature du simulateur
- Insertion du contenu LaTeX placé en commentaires du code source entre les balises `<sim2doc>` et `</sim2doc>` (optionnel)

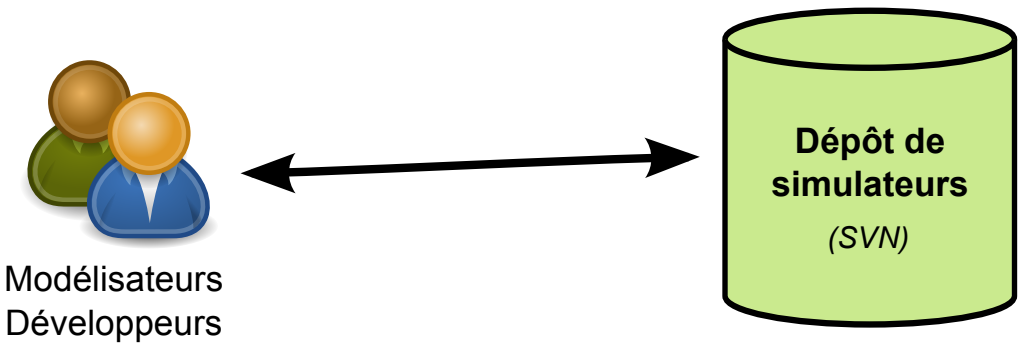
Génère un fichier LaTeX, pouvant être converti à la volée en PDF ou en HTML

Exemple

```
openfluid --buddy sim2doc --buddyopts \
  inputcpp=MySim.cpp,outputdir=./doc,PDF=1
```

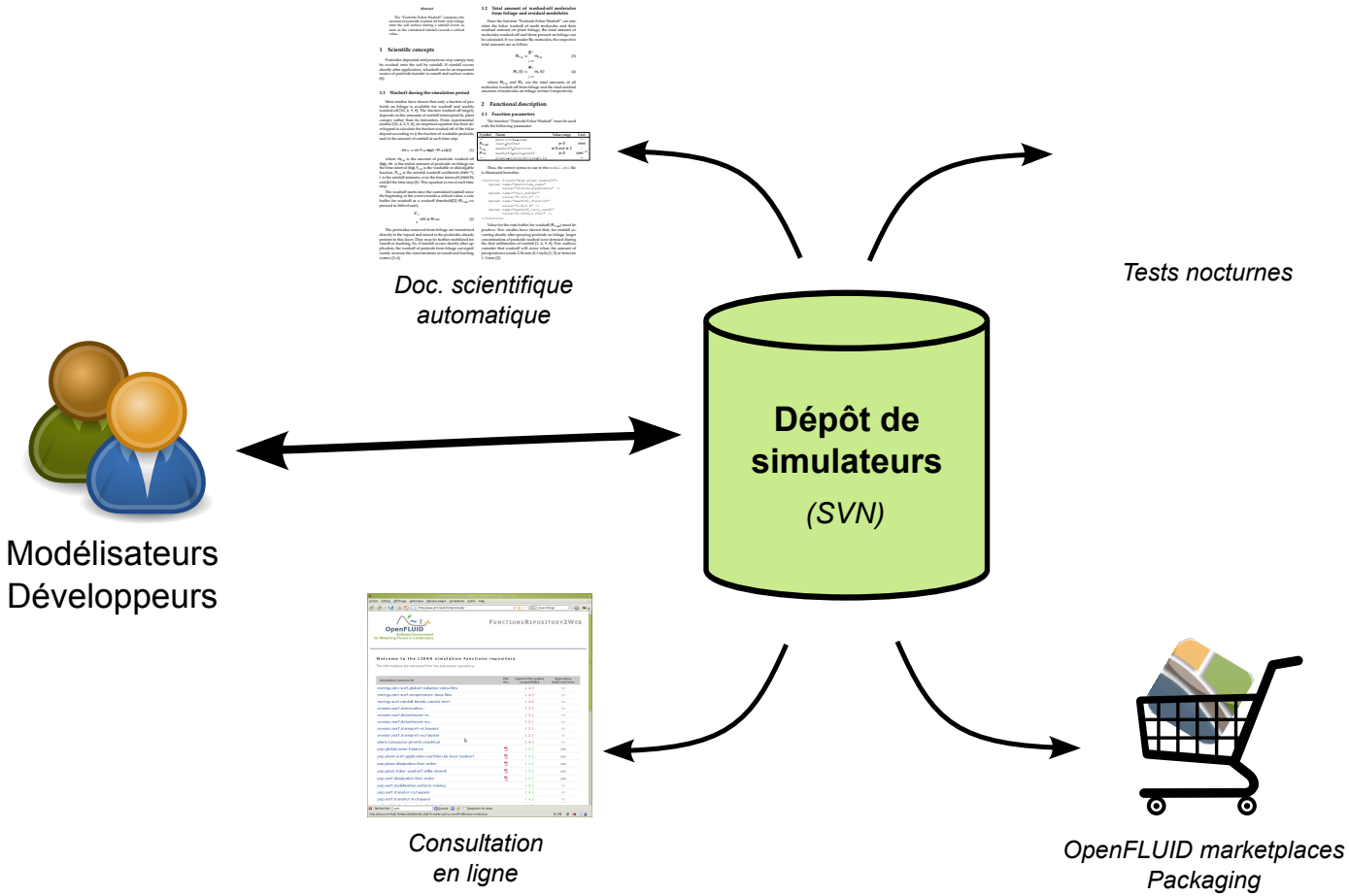
Dépôt de simulateurs

Structure pour la capitalisation et le partage de simulateurs développées au sein d'un groupe de travail



Dépôt de simulateurs

Structure pour la capitalisation et le partage de simulateurs développées au sein d'un groupe de travail



Dépôt de simulateurs du LISAH

Capitalisation des développements, packaging:

- Structure normalisée, avec outil de gestion du code source (subversion)
- https://www.umr-lisah.fr/svn/openfluid-functions/

Navigation dans le contenu:

- Accès à l'historique, aux documentations scientifiques, ...
- http://www.umr-lisah.fr/repos2web/

Tableau de suivi des tests

- Suite CMake/CTest/CDash
- http://www.umr-lisah.fr/cdash/

Quelques éléments de projet

Ingénierie logicielle

- Développement majoritairement en C++
- Multiplateformes Linux/Unix, Windows, (Mac OSX)
- Librairies support: Boost, LibXML2, Glib, GTK, libCURL, GDAL/OGR, GEOS
- Environnement de développement/tests/packaging/doc:
  Eclipse, CMake, GCC, CTest, CPack, Doxygen, LaTeX
- Code source géré sous Git,
  centralisé et disponible sur GitHub https://github.com/OpenFLUID

Cadre juridique

- Dépôt APP, dépôt de la marque OpenFLUID®
- Licence libre (GPLv3) ou sur mesure (commerciale, propriétaire, . . .)

Perspectives de développement

- Validation du module PyOpenFLUID permettant d'utiliser OpenFLUID depuis le langage Python
- Introduction de données attachées aux connections entres unités spatiales
- Consolidation de la librairie de traitement spatial OpenFLUID-LandR
- Accès au datastore depuis les simulateurs et observateurs
- Evolution du datastore : liens avec BDs, fichiers de données mesurées, ...
- Améliorer le développement d'approches orientées raster
- Passage à la librairie graphique Qt en lieu et place de GTK
- ...

Version 2.0 beta pour la formation

Version stable actuelle: OpenFLUID 1.7.2

Version pour la formation : OpenFLUID 2.0.0~beta1

- Remaniement en profondeur du code source (108000 lignes ajoutées, 85000 lignes supprimées)
- Doc en ligne incomplète
- Traduction builder non mise à jour
- Risques de plantages (merci les béta-testeurs!)
- sim2doc pas à jour
- générateur ”inject” pas à jour

Utiliser le bug board!

Références

J.C. Fabre, X. Louchart, R. Moussa, C. Dagès, F. Colin, M. Rabotin, D. Raclot, P. Lagacherie, and Voltz M.

OpenFLUID: a software environment for modelling fluxes in landscapes.

In *LANDMOD2010*, INRA, CIRAD, page 13pp, Montpellier, France, 2010. Quae.

J.-C. Fabre, M. Rabotin, D. Crevoisier, A. Libres, C. Dagès, R. Moussa, Ph. Lagacherie, D. Raclot, and M. Voltz.

OpenFLUID: an open-source software environment for modelling fluxes in landscapes.

In *Vol. 15, EGU2013-8821-1, EGU General Assembly 2013*, Vienna (Austria), 7-12 avril 2013.

M. Rabotin, J.-C. Fabre, A. Libres, P. Lagacherie, D. Crevoisier, and R. Moussa.

Using graph approach for managing connectivity in integrative landscape modelling.

In *Vol. 15, EGU2013-8851, EGU General Assembly 2013*, Vienne (Autriche), 7-12 avril 2013.