VLE : *Virtual Laboratory Environmnent*

Journée Devlog sur l'apport de l'IDM
Pour la mise en œuvre des modèles scientifiques

Gauthier Quesnel

INRA - MIAT

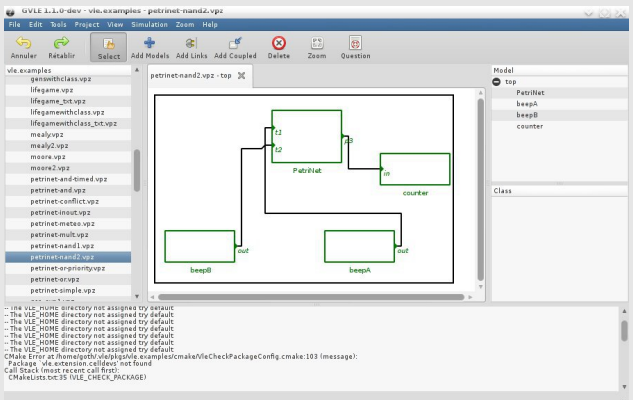
Introduction

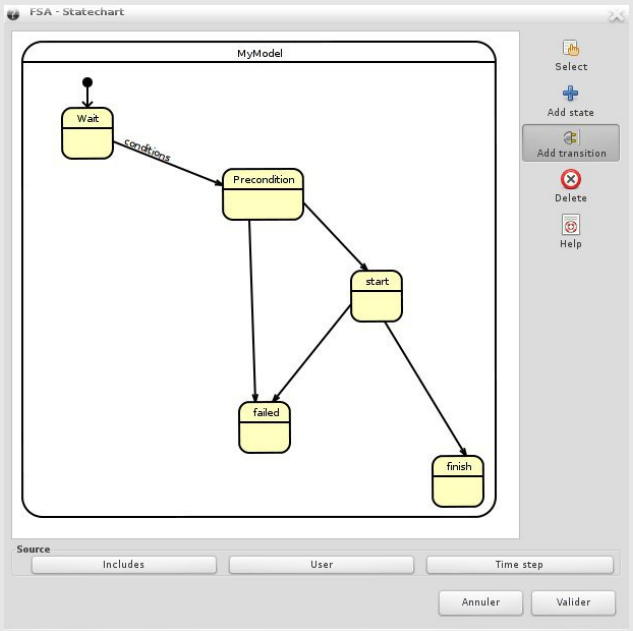
VLE : Virtual Laboratory Environment

- VLE [1] est un environnement de multimodélisation et de simulation de systèmes complexes dynamiques.
- Il est basé sur le formalisme à événements discrets DEVS [2]

VIRTUAL LABORATORY ENVIRONMENT

Complex systems modelling and simulation





DEVS

DEVS, un formalisme de M&S de systèmes dynamiques de bas niveau

- Initié en 1976 par B. P. Zeigler et est issu des mathématiques discrètes
- Un formalisme à événements discrets

DEVS

Simulation de la trajectoire d'une balle

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

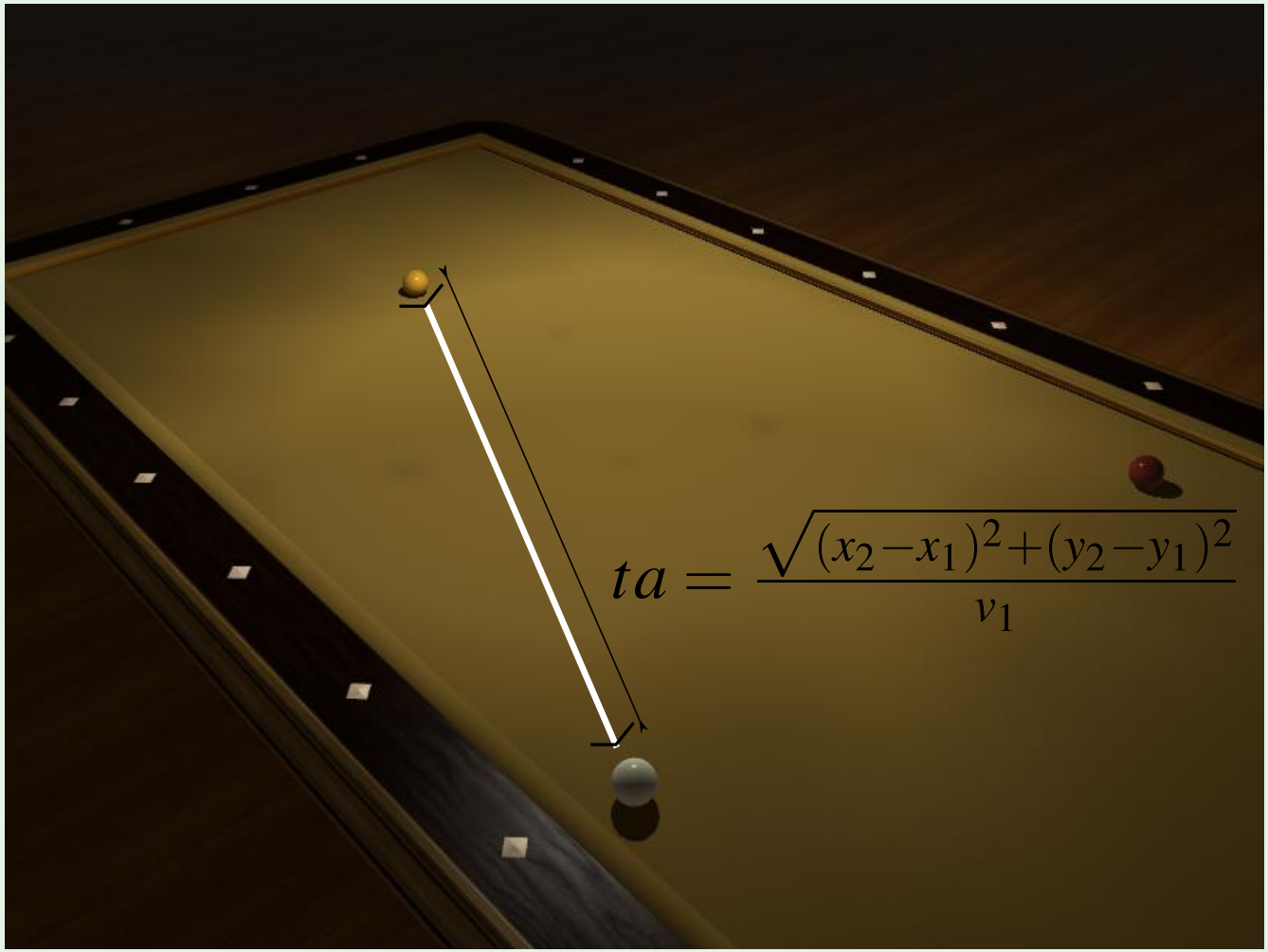
DEVS

Simulation de la trajectoire d'une balle

Temps discrets : Quel est mon état a t ?

DEVS

Simulation de la trajectoire d'une balle



Événement discret : Quand le prochain changement d'état aura lieu ?

DEVS

DEVS, un formalisme de M&S de systèmes dynamiques de bas niveau

- Initié en 1976 par B. P. Zeigler et est issu des mathématiques discrètes
- Un formalisme à événements discrets
- Propose deux types de modèles atomiques et couplés :
  - les modèles atomiques sont composés d'états et de fonctions de transitions d'états
  - les modèles couplés proposent une approche modulaire et hiérarchique de la M&S
  - possède une propriété importante : un modèle couplé possède les mêmes propriétés qu'un modèle atomique
- propose un ensemble d'algorithmes : les simulateurs abstraits

DEVS

DEVS, un formalisme de M&S de systèmes dynamiques de bas niveau

- Initié en 1976 par B. P. Zeigler et est issu des mathématiques discrètes
- Un formalisme à événements discrets
- Propose deux types de modèles atomiques et couplés :
  - les modèles atomiques sont composés d'états et de fonctions de transitions d'états
  - les modèles couplés proposent une approche modulaire et hiérarchique de la M&S
  - possède une propriété importante : un modèle couplé possède les mêmes propriétés qu'un modèle atomique
- propose un ensemble d'algorithmes : les simulateurs abstraits

Propriété: les formalismes de systèmes dynamiques peuvent être traduits en DEVS

DEVS

DEVS, un formalisme de M&S de systèmes dynamiques de bas niveau

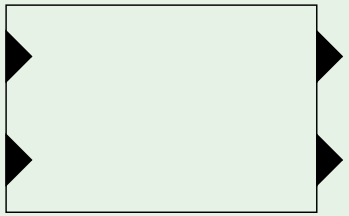
- Initié en 1976 par B. P. Zeigler et est issu des mathématiques discrètes
- Un formalisme à événements discrets
- Propose deux types de modèles atomiques et couplés :
  - les modèles atomiques sont composés d'états et de fonctions de transitions d'états
  - les modèles couplés proposent une approche modulaire et hiérarchique de la M&S
  - possède une propriété importante : un modèle couplé possède les mêmes propriétés qu'un modèle atomique
- propose un ensemble d'algorithmes : les simulateurs abstraits

Propriété: les formalismes de systèmes dynamiques peuvent être traduits en DEVS

VLE : Une implémentation des simulateurs abstraits

DEVS, Description du modèle atomique

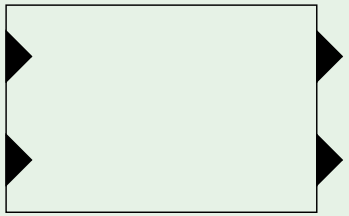
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées

DEVS, Description du modèle atomique

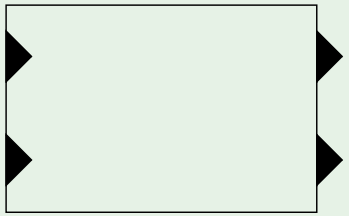
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées

DEVS, Description du modèle atomique

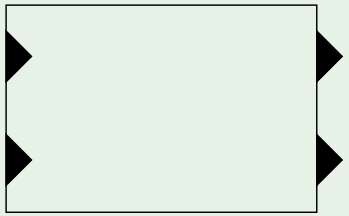
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système

DEVS, Description du modèle atomique

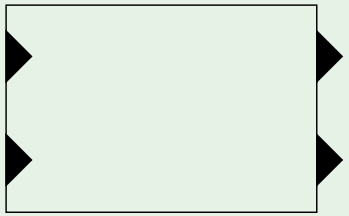
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système
- δ_{ext} : fonction de transition externe : $\delta_{ext} : S \times X \rightarrow S$
  représente les réponses du système aux événements externes

DEVS, Description du modèle atomique

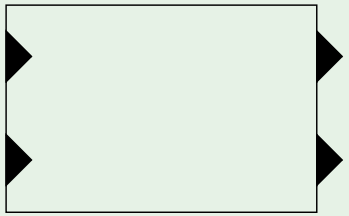
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système
- δ_{ext} : fonction de transition externe : $\delta_{ext} : S \times X \rightarrow S$
  représente les réponses du système aux événements externes
- δ_{int} : fonction de transition interne : $\delta_{int} : S \rightarrow S$
  représente les évolutions autonomes

DEVS, Description du modèle atomique

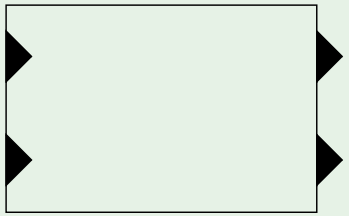
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système
- δ_{ext} : fonction de transition externe : $\delta_{ext} : S \times X \rightarrow S$
  représente les réponses du système aux événements externes
- δ_{int} : fonction de transition interne : $\delta_{int} : S \rightarrow S$
  représente les évolutions autonomes
- δ_{con} : fonction de conflit si δ_{int} et δ_{ext} sont programmées à la même date

DEVS, Description du modèle atomique

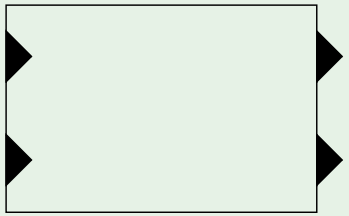
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système
- δ_{ext} : fonction de transition externe : $\delta_{ext} : S \times X \rightarrow S$
  représente les réponses du système aux événements externes
- δ_{int} : fonction de transition interne : $\delta_{int} : S \rightarrow S$
  représente les évolutions autonomes
- δ_{con} : fonction de conflit si δ_{int} et δ_{ext} sont programmées à la même date
- $ta(s)$: le temps pendant lequel le modèle reste dans l'état S. $ta \rightarrow R_0^+$

DEVS, Description du modèle atomique

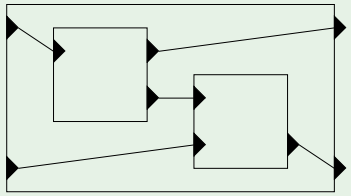
$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$



- X : l'ensemble des ports d'entrée et des valeurs attachées
- Y : l'ensemble des ports de sortie et des valeurs attachées
- S : l'ensemble des états du système
- δ_{ext} : fonction de transition externe : $\delta_{ext} : S \times X \rightarrow S$
  représente les réponses du système aux événements externes
- δ_{int} : fonction de transition interne : $\delta_{int} : S \rightarrow S$
  représente les évolutions autonomes
- δ_{con} : fonction de conflit si δ_{int} et δ_{ext} sont programmées à la même date
- $ta(s)$: le temps pendant lequel le modèle reste dans l'état S. $ta \rightarrow R_0^+$
- λ : la fonction de sortie : $\lambda : S \rightarrow Y$ représente les influences externes

DEVS, Description du modèle couplé

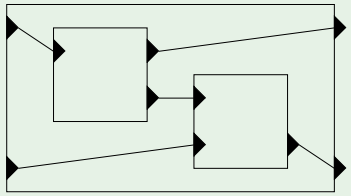
$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$



- X l'ensemble des ports d'entrée et des valeurs associées.

DEVS, Description du modèle couplé

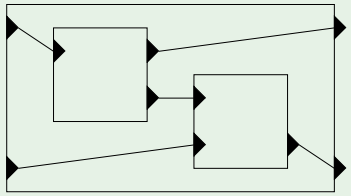
$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$



- X l'ensemble des ports d'entrée et des valeurs associées.
- Y l'ensemble des ports de sortie et des valeurs associées.

DEVS, Description du modèle couplé

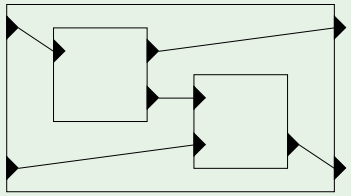
$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$



- X l'ensemble des ports d'entrée et des valeurs associées.
- Y l'ensemble des ports de sortie et des valeurs associées.
- D l'ensemble des identifiants des sous modèles avec : $\{M_d | d \in D\}$.

DEVS, Description du modèle couplé

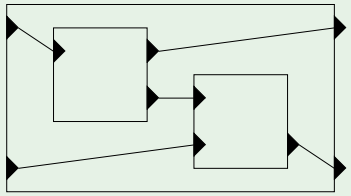
$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$



- X l'ensemble des ports d'entrée et des valeurs associées.
- Y l'ensemble des ports de sortie et des valeurs associées.
- D l'ensemble des identifiants des sous modèles avec : $\{M_d | d \in D\}$.
- $\forall d \in D \cup \{N\}$, I_d l'ensemble des perturbateurs de : d
  $I_d \subseteq D \cup \{N\}, d \notin I_d$

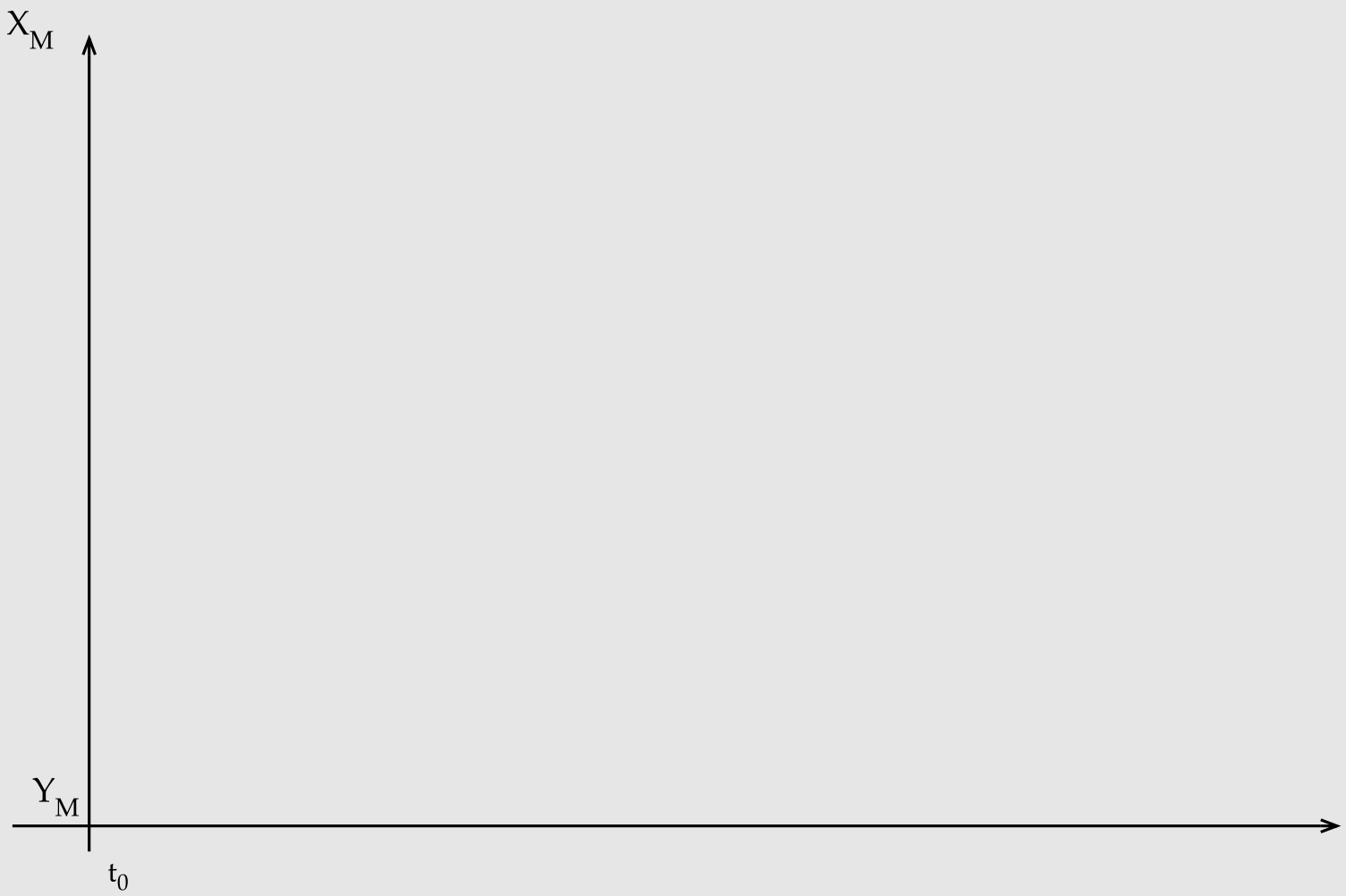
DEVS, Description du modèle couplé

$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$

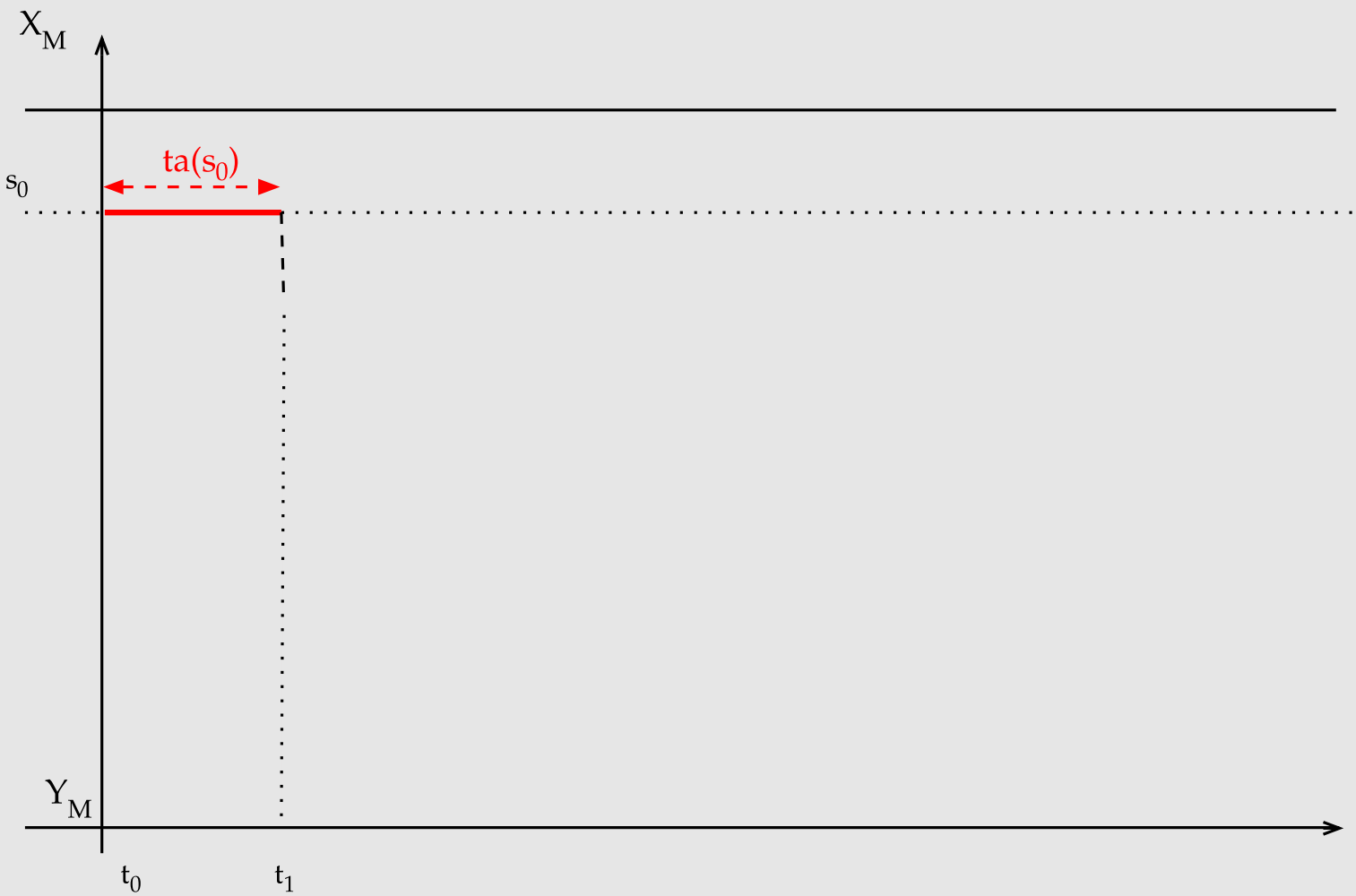


- X l'ensemble des ports d'entrée et des valeurs associées.
- Y l'ensemble des ports de sortie et des valeurs associées.
- D l'ensemble des identifiants des sous modèles avec : $\{M_d | d \in D\}$.
- $\forall d \in D \cup \{N\}$, I_d l'ensemble des perturbateurs de : d
  $I_d \subseteq D \cup \{N\}, d \notin I_d$
- $\forall d \in D \cup \{N\}$
  $\forall i \in I_d$, $Z_{i,d}$ est une fonction i-to-d :
  $Z_{i,d} : X \to X_d$, if $i = N$ (connexions d'entrée)
  $Z_{i,d} : Y_i \to Y$, if $d = N$ (connexions de sorties)
  $Z_{i,d} : Y_i \to X_d$ if $i \neq N$ et $d \neq N$ (connexions internes)

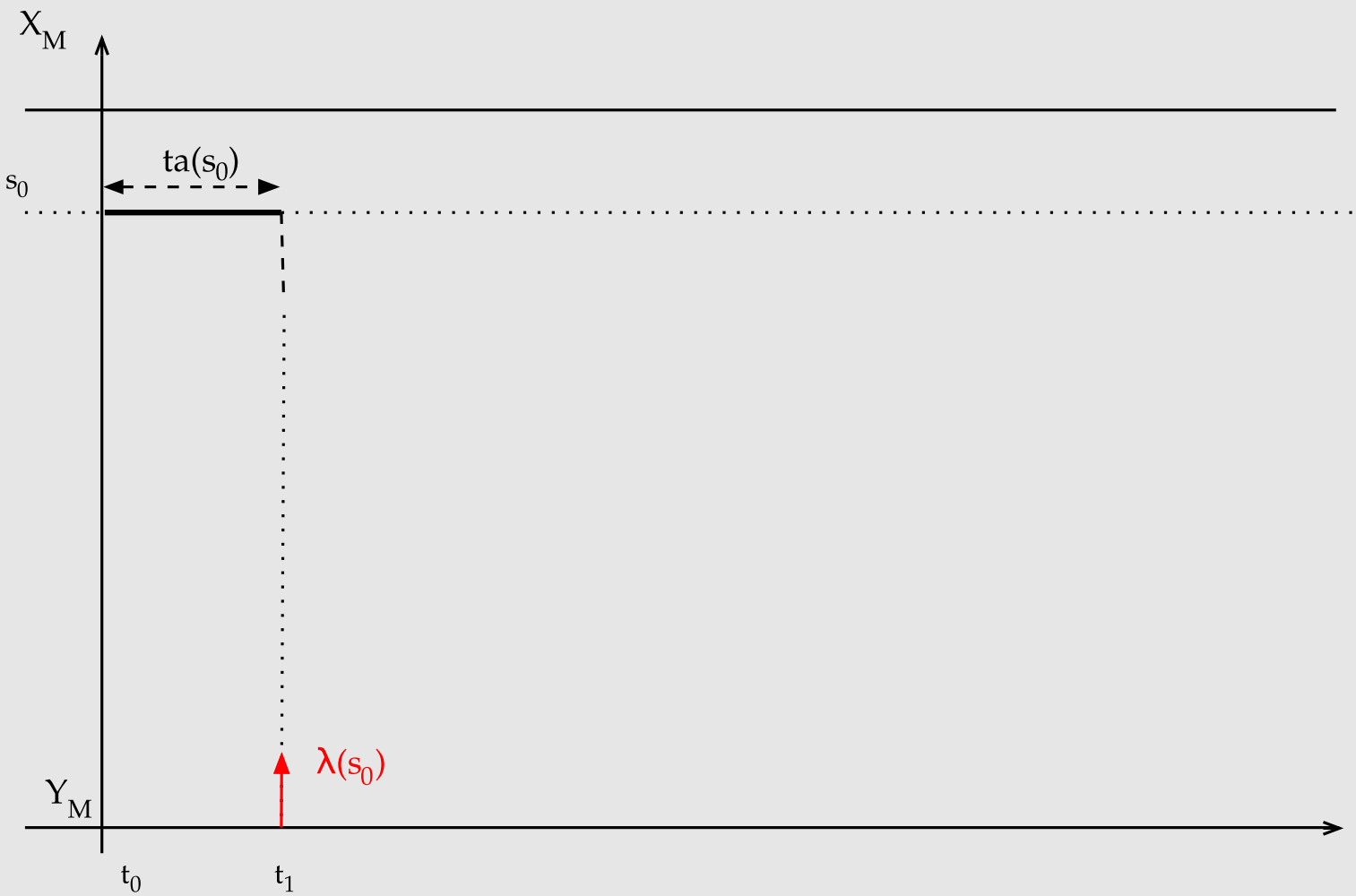
DEVS, Dynamique des états



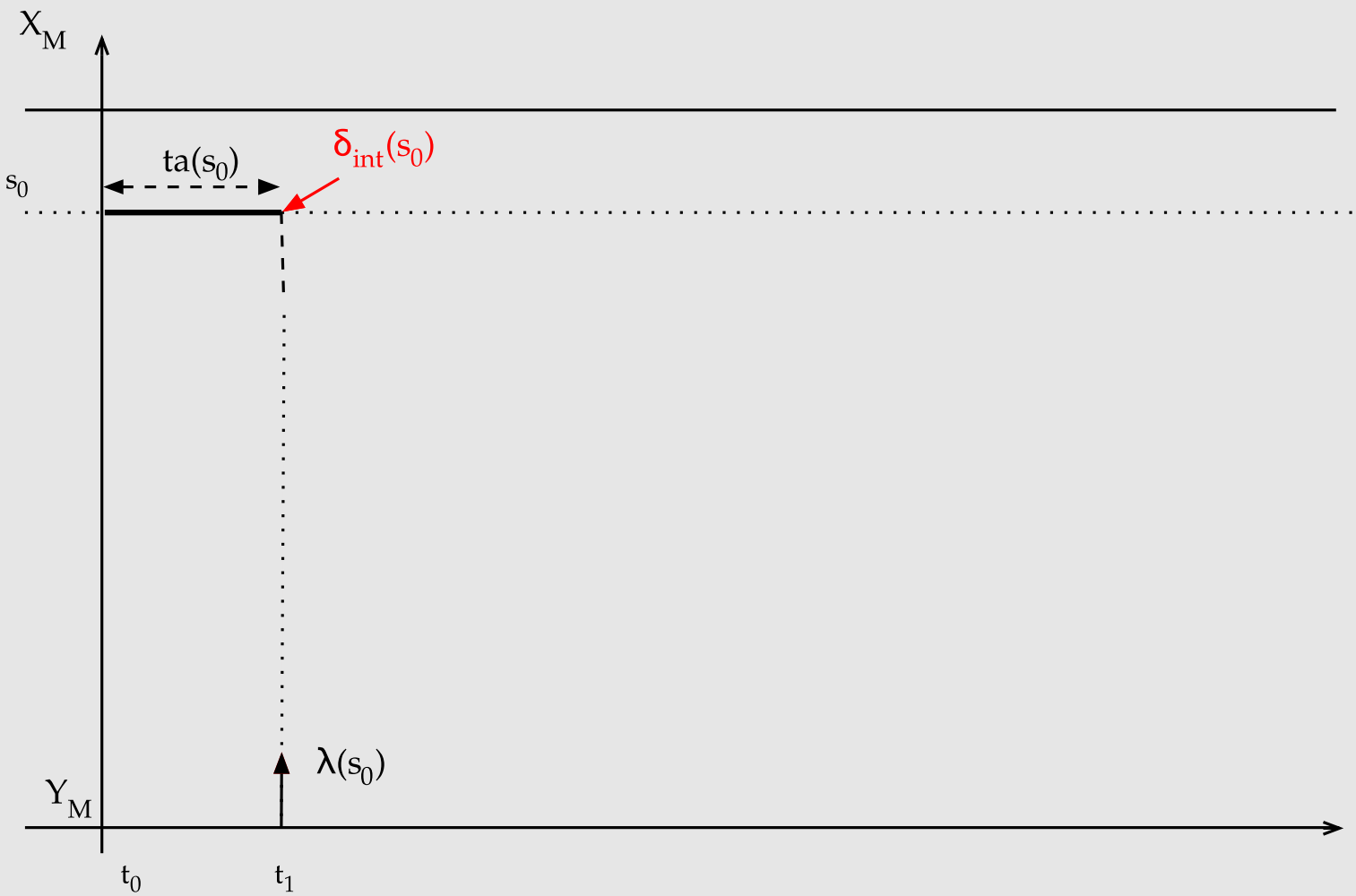
DEVS, Dynamique des états



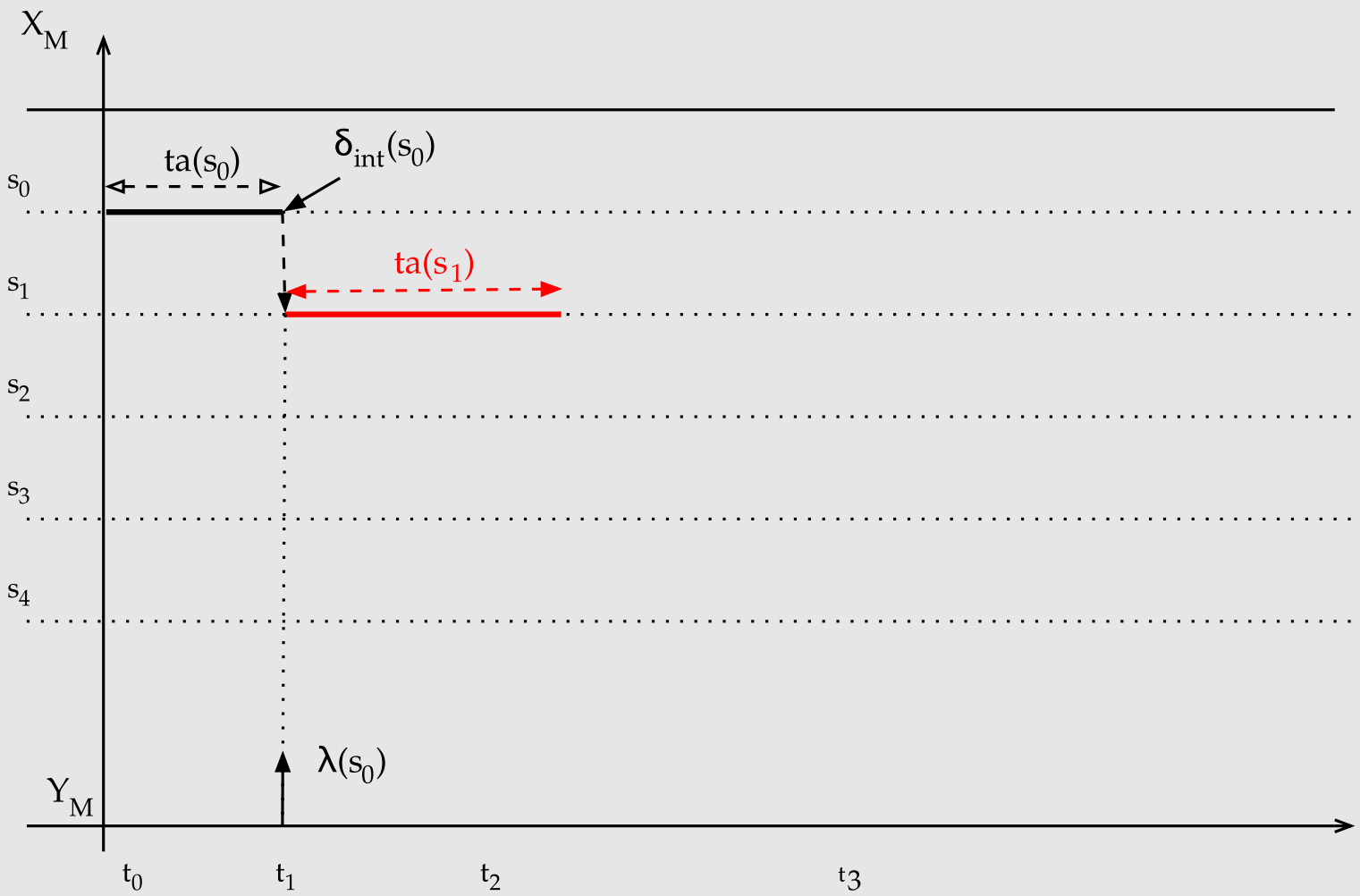
DEVS, Dynamique des états



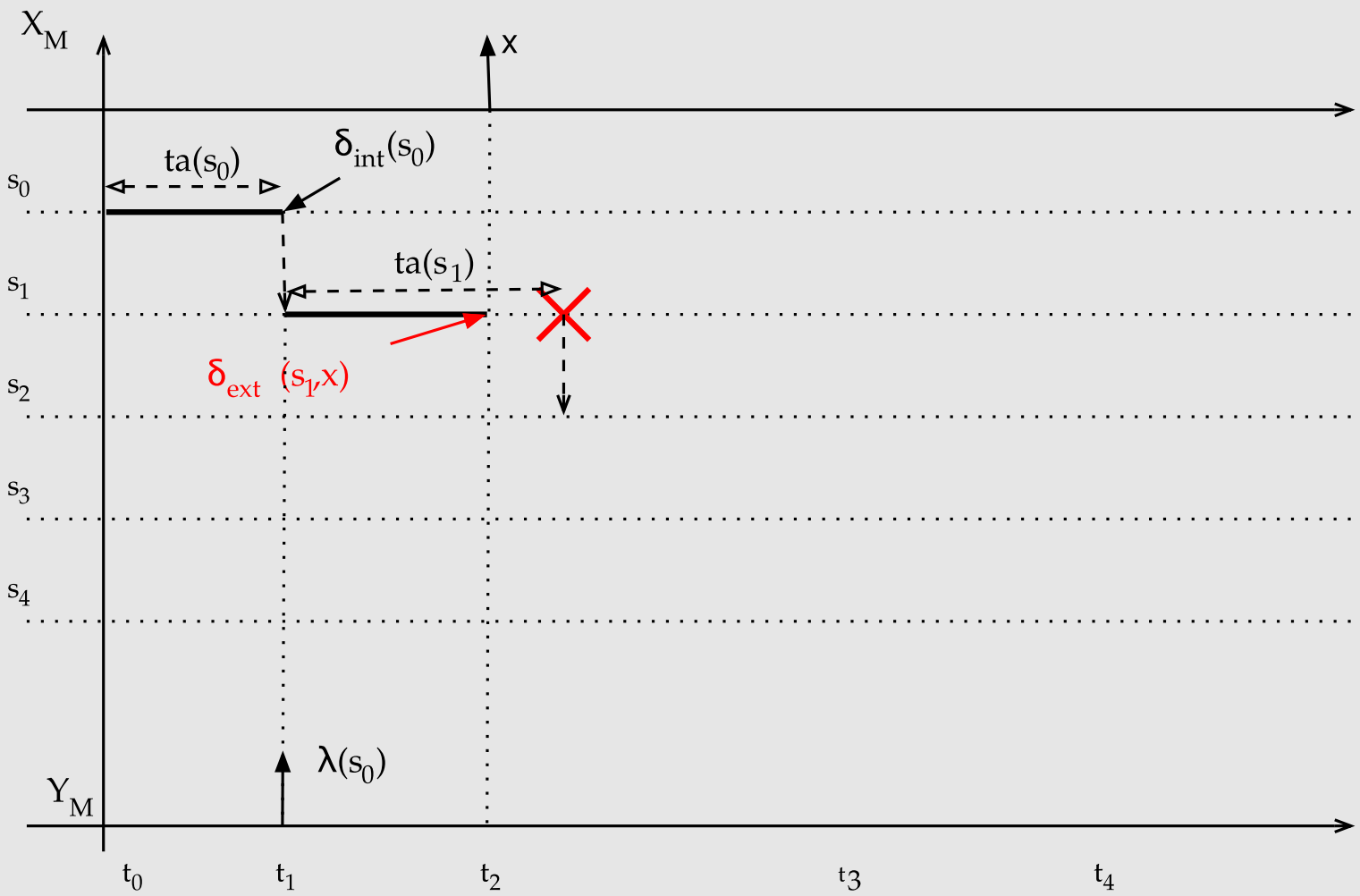
DEVS, Dynamique des états



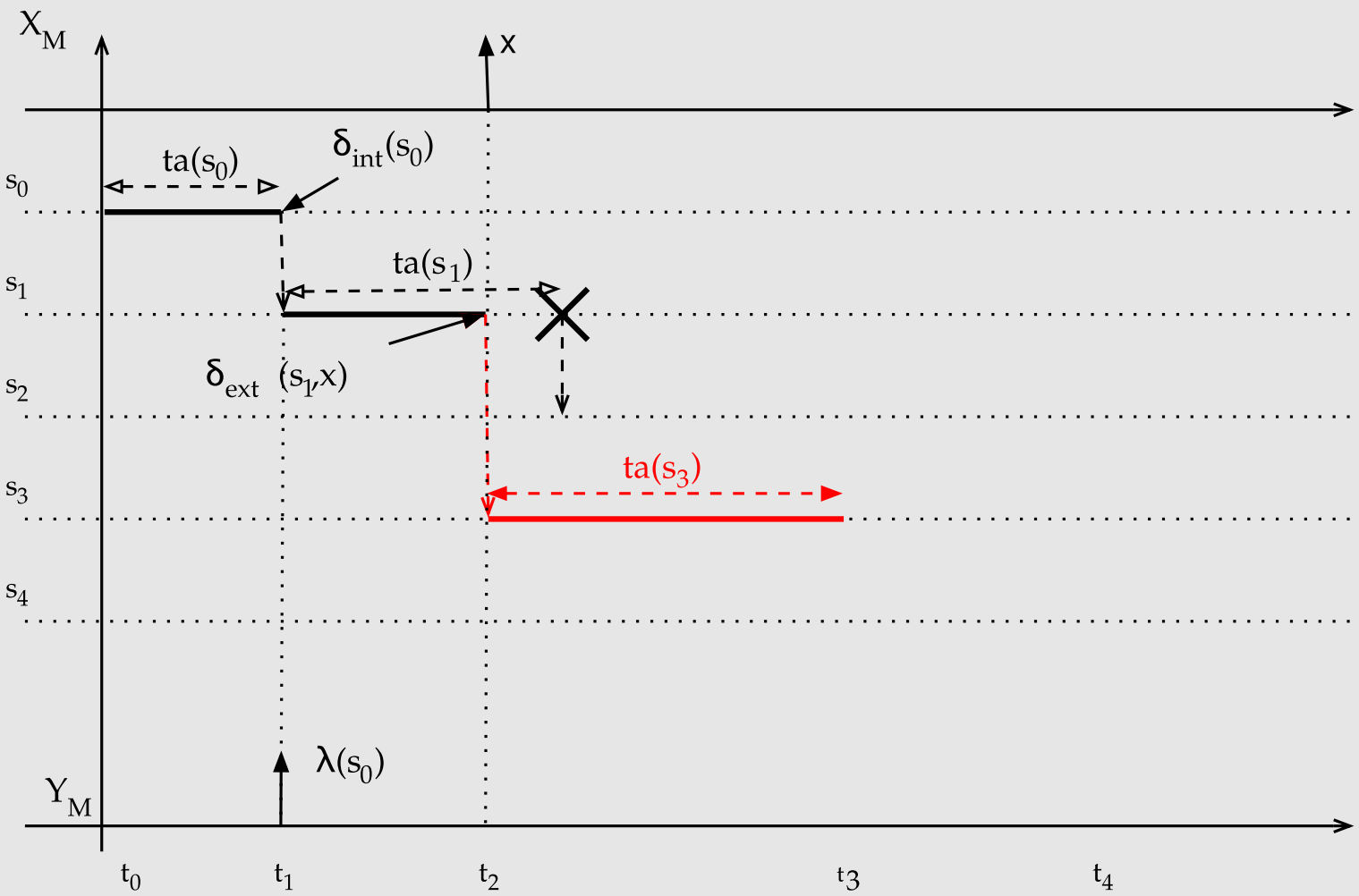
DEVS, Dynamique des états



DEVS, Dynamique des états



DEVS, Dynamique des états



DEVS, Structure dynamique

Réseau de modèles : déportent les connexions dans un modèle dit *exécutif* :

$$DSDEVN_N = \langle X_N, Y_N, \chi, M_\chi \rangle$$

où :

- X_N et Y_N sont les ports d'entrée et de sortie du réseau
- χ le nom du modèle *exécutif*
- M_χ le modèle *exécutif*

$$M_\chi = \langle X_\chi, Y_\chi, S_\chi, \delta^\chi_{int}, \delta^\chi_{ext}, ta_\chi, \lambda_\chi, \Sigma^*, \gamma \rangle$$

Avec :

$$\begin{cases} \gamma : S_\chi \rightarrow \Sigma^* \text{ la fonction de structure} \\ \Sigma* : \text{ l'ensemble des structures} \\ \Sigma_\alpha \in \Sigma^* : \\ \qquad \Sigma_\alpha = \langle D, EIC, EOC, IC \rangle \end{cases}$$

Projet VLE

VLE

- Virtual Environment Laboratory, projet démarré en 2003,
- Environnement de Multimodélisation et de simulation de systèmes complexes basé sur DEVS
- VLE fournit une bibliothèque C++ appelée VFL (VLE Foundation Library)
- VLE fournit un simulateur en ligne de commande, une interface graphique de modélisation, un paquet R pour l'analyse et la visualisation de résultats, un programme Python pour le développement de service web et un système de paquets pour étendre VLE

Projet VLE

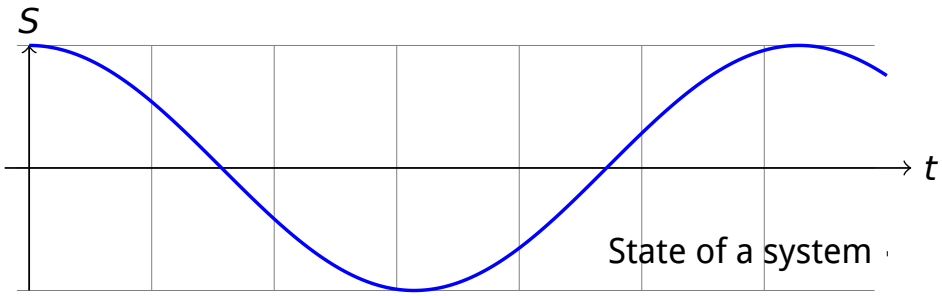
VLE

- Virtual Environment Laboratory, projet démarré en 2003,
- Environnement de Multimodélisation et de simulation de systèmes complexes basé sur DEVS
- VLE fournit une bibliothèque C++ appelée VFL (VLE Foundation Library)
- VLE fournit un simulateur en ligne de commande, une interface graphique de modélisation, un paquet R pour l'analyse et la visualisation de résultats, un programme Python pour le développement de service web et un système de paquets pour étendre VLE

Noyau de simulation

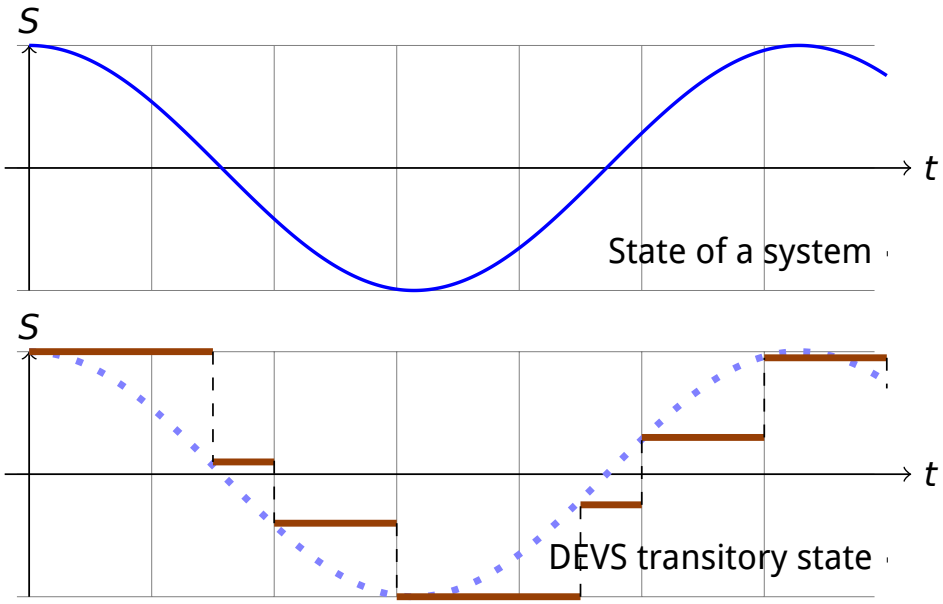
- Un noyau DSDE (F. Barros) : combine Parallel DEVS et Dynamic-Structure DEVS
- Un sous-système d'observation

VLE, Extension d'observation



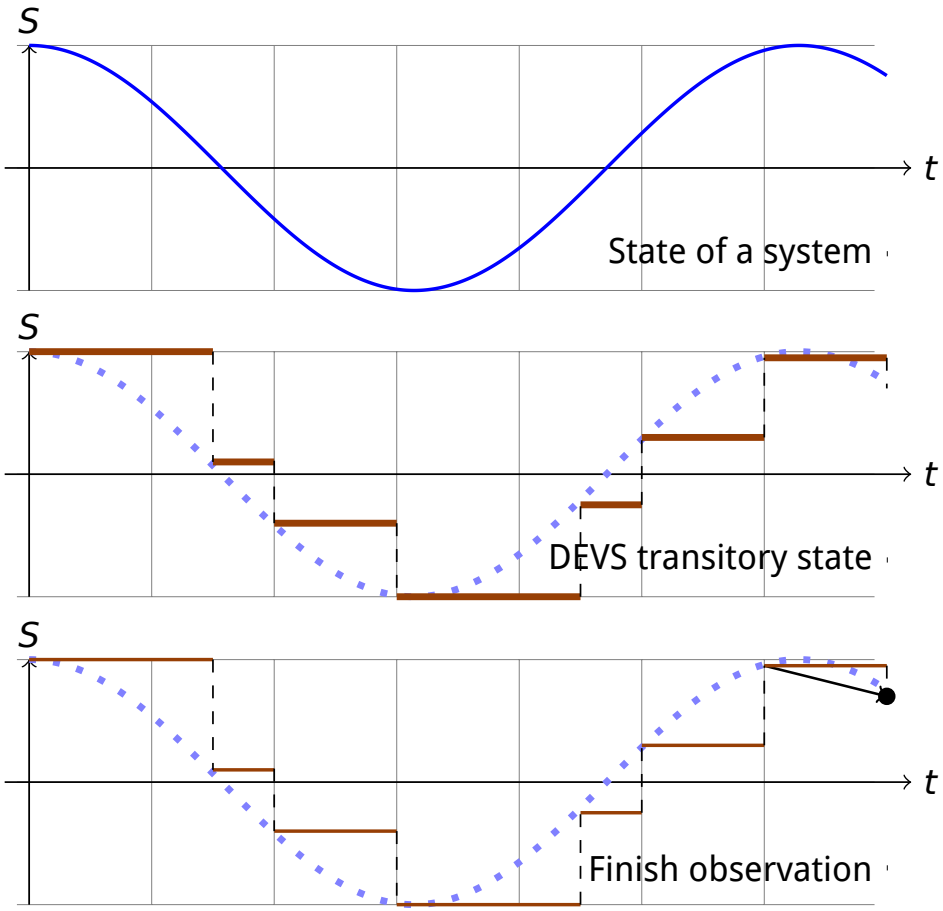
1. Montre l'évolution de l'état S d'un système dynamique sur t

VLE, Extension d'observation



1. Montre l'évolution de l'état S d'un système dynamique sur t
2. L'évolution de l'état avec un modèle DEVS (exemple)

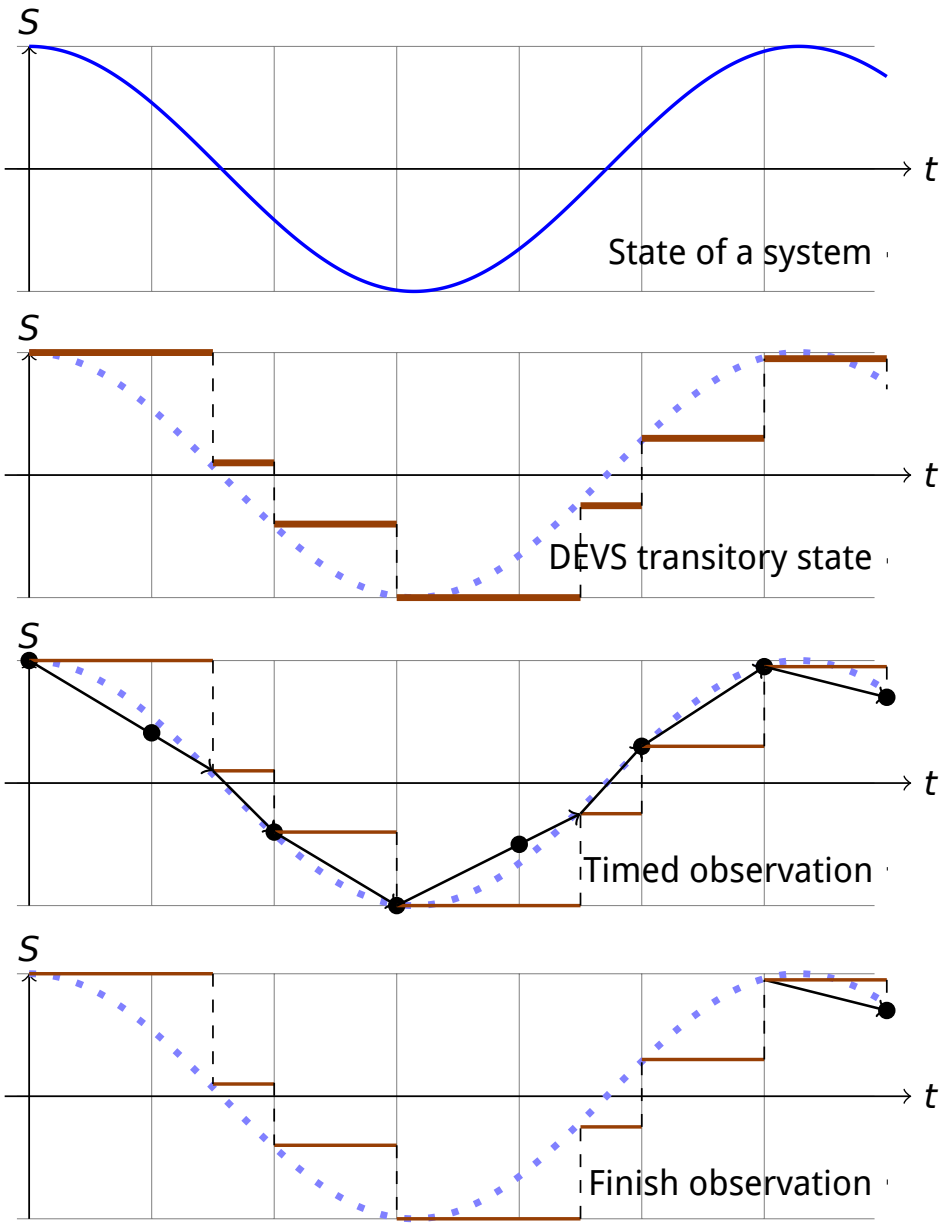
VLE, Extension d'observation



1. Montre l'évolution de l'état S d'un système dynamique sur t
2. L'évolution de l'état avec un modèle DEVS (exemple)
3. L'observation par pas de temps

Les points sont les valeurs d'observation retournées par la fonction d'observation utilisée par δ_{obs}

VLE, Extension d'observation



1. Montre l'évolution de l'état S d'un système dynamique sur t
2. L'évolution de l'état avec un modèle DEVS (exemple)
3. L'observation par pas de temps
4. L'observation à la date finale

Les points sont les valeurs d'observation retournées par la fonction d'observation utilisée par δ_{obs}

Multimodélisation : Définitions

Multimodélisation

- Un multimodèle est un modèle qui rassemble plusieurs paradigmes ou formalismes dans sa réalisation.
  - → Augmentation de la puissance descriptive du modèle
  - → Introduit la notion de couplage
- La multimodélisation est l'ensemble des concepts, outils et techniques de construction de multimodèles.

Comment coupler des modèles hétérogènes ?

- Coupler des représentations de la dynamique des sous-systèmes
- Intégration des notions de temps, d'espace, d'états et de transition

Multimodélisation : Définitions

Multimodélisation

- Un multimodèle est un modèle qui rassemble plusieurs paradigmes ou formalismes dans sa réalisation.
  - → Augmentation de la puissance descriptive du modèle
  - → Introduit la notion de couplage
- La multimodélisation est l'ensemble des concepts, outils et techniques de construction de multimodèles.

Comment coupler des modèles hétérogènes ?

- Coupler des représentations de la dynamique des sous-systèmes
- Intégration des notions de temps, d'espace, d'états et de transition

Directions possibles

- Co-simulation : chaque sous modèle a son propre simulateur.
- La spécification des sous-systèmes dans un formalisme unique : Réécriture de tous les sous-systèmes dans le même formalisme

Multimodélisation : Nos propositions

Sous-formalismes

Nous fournissons un ensemble de sous-formalismes développé comme des sous-classes du modèle atomic DEVS):

- Solvers pour la simulation d'équations différentielles ordinaires d'ordre 1
  - Euler, Runge Kutta order 4, QSS 1 et QSS 2
- Difference equation (recurrence relation)
- Petri net (with timed transition, inhibitor arc)
- Finite state automaton:
  - Harel statechart (proche de la spécification des statechart UML), Mealy, Moore and FDDEVS
- Decision making (agent, CSP, planificateur HTN, simulation plan, etc).

Multimodélisation : Équations aux différences

L'extension DifferenceEquation permet de développer des modèles à temps discret qui calculent la valeur d'une variable réelle en t en fonction d'elle-même à $t - \Delta t$, $t - 2\Delta t$, ... et en fonction d'autres variables réelles en t, $t - \Delta t$, $t - 2\Delta t$, ...

Formellement

$$
\begin{aligned}
\forall i \in \{1, \ldots, n\}, & \\
X_i(t) &= f_i(\underline{Z}_i(t), \underline{W}_i(t)) \\
\underline{Z}_i(t) &= \left[X_i(t - \Delta t), \ldots, X_i(0)\right] \\
\underline{W}_i(t) &= \left[\ldots, \underline{W}_j(t), \ldots\right] \text{ pour } j \in \{1, \ldots, n\} \text{ et } j \neq i \\
\underline{W}_j(t) &= \left[X_j(t), X_j(t - \Delta t), \ldots, X_j(0)\right]
\end{aligned}
$$

Multimodélisation : simulation d'équation différentielles du premier ordre

QSS : Quantized State System

Méthode proposée par E. Kofman en 2001 pour résoudre des équations différentielles du premier ordre basée sur :

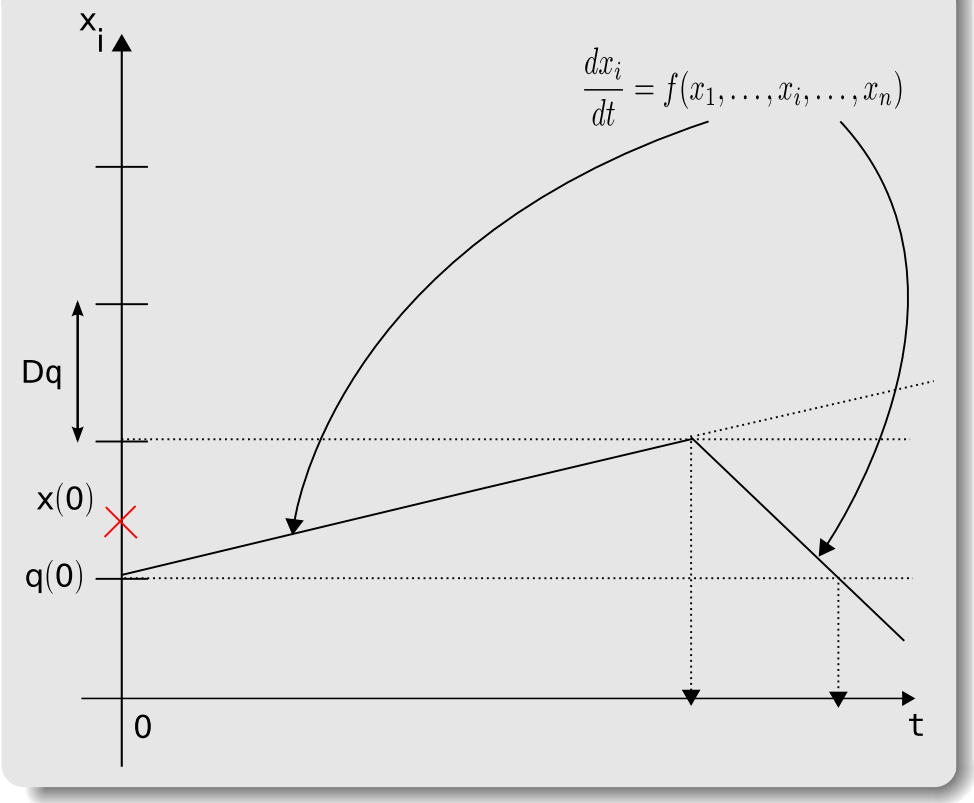
- la discrétisation des valeurs au lieu du temps : quantification
- le calcul du pas de temps en cohérence avec la pente calculée grâce à l'équation
- plusieurs algorithmes existent en fonction du degré d'approximation (aujourd'hui, il existe QSS1, QSS2 et QSS3)

DESS : Differential Equation System Specification

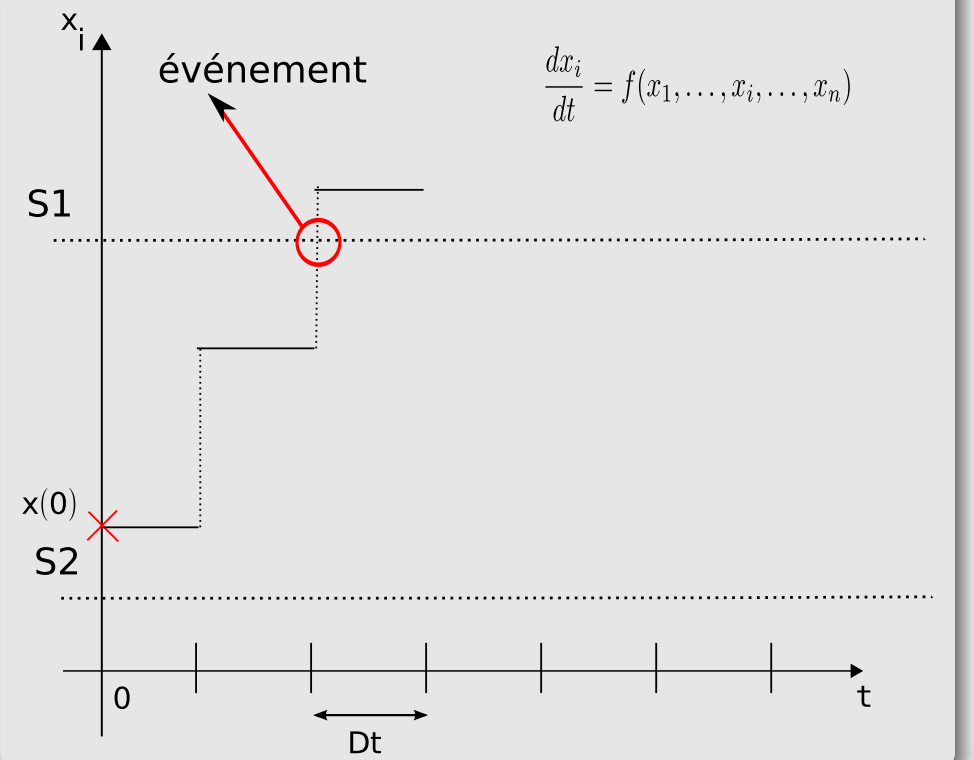
- À la différence de QSS, DESS repose sur des schémas d'intégration classiques (Euler, Runge Kunta, ...) à pas de temps d'intégration constant
- Les événements de sortie sont produits sur des seuils :
  - la variable d'intégration vient de franchir ou est égale à un seuil

Multimodélisation : Simulation d'équation différentielles du premier ordre

QSS

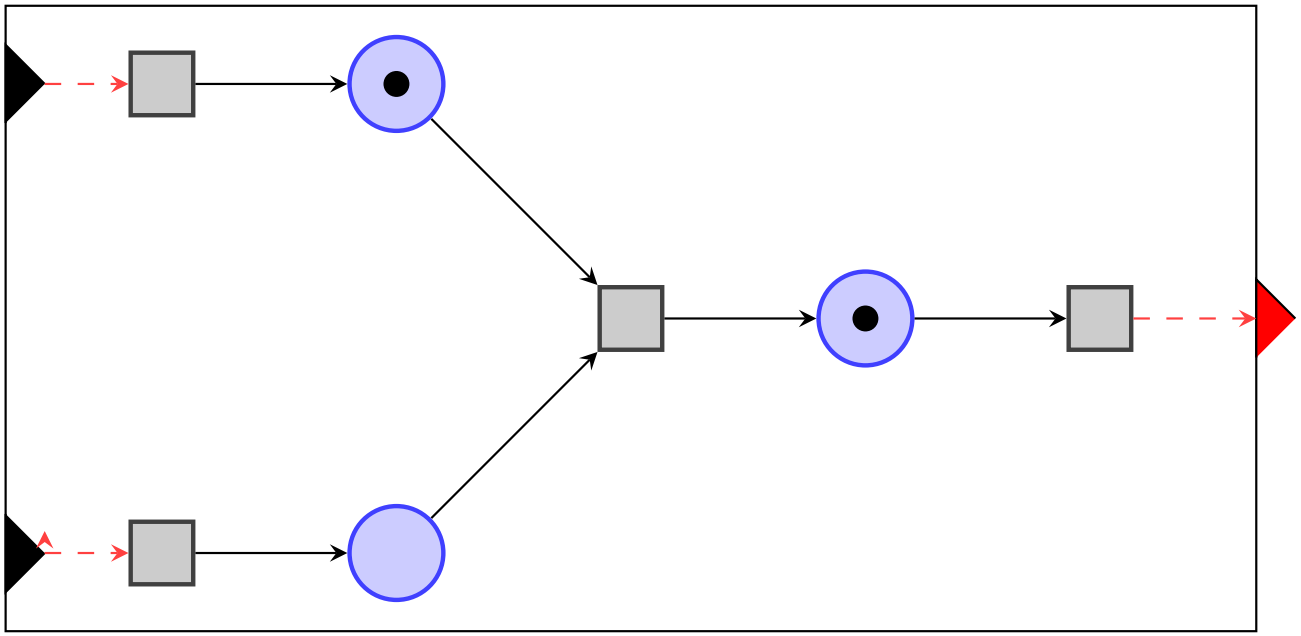


DESS



Multimodélisation : Réseau de Petri

Encapsulation d'un formalisme intemporel comme les réseaux de Petri.



À noter

Le réseau de Petri possède des extensions temporelle, à priorité, stochastique, etc. : High Level PetriNet

Développement de modèles

VLE

- VLE fourni une bibliothèque partagée C++ qui embarque le noyau de simulation. Ainsi :
  - les utilisateurs doivent développer des classes C++ de modèles atomiques, modèles exécutives ou de sous-formalismes

```
class Dynamics
{
public:
  virtual Time timeAdvance() const;

  virtual void internalTransition(const Time& time);

  virtual void externalTransition(const Time& time,
                                  const ExternalEventList& lst);
  virtual void output(const Time& time,
                                  ExternalEventList& out) const;
  virtual void confluentTransition(const Time& time,
                                  const ExternalEventList& lst);
  virtual Value* observation(const ObservationEvent& event) const;
}
```

Développement de modèles

Sous-formalismes

DEVS permet de développer des sous-formalismes de DEVS. Avec VLE nous fournissons des extensions :

dynamiques qui encapsulent un formalisme mathématique comme les équations différentielles, réseau de Petri, etc.

structurelles qui manipulent la structure du modèle : création d'automates cellulaires, des graphes communication, etc.

Développement de modèles

Sous formalismes

- Nous proposons une API simplifiée
- Nous proposons des générateurs de code C++

Par exemple, avec le formalisme des équation aux différences, nous fournissons une seule fonction `compute`. L'api des modèles atomiques est cachées.

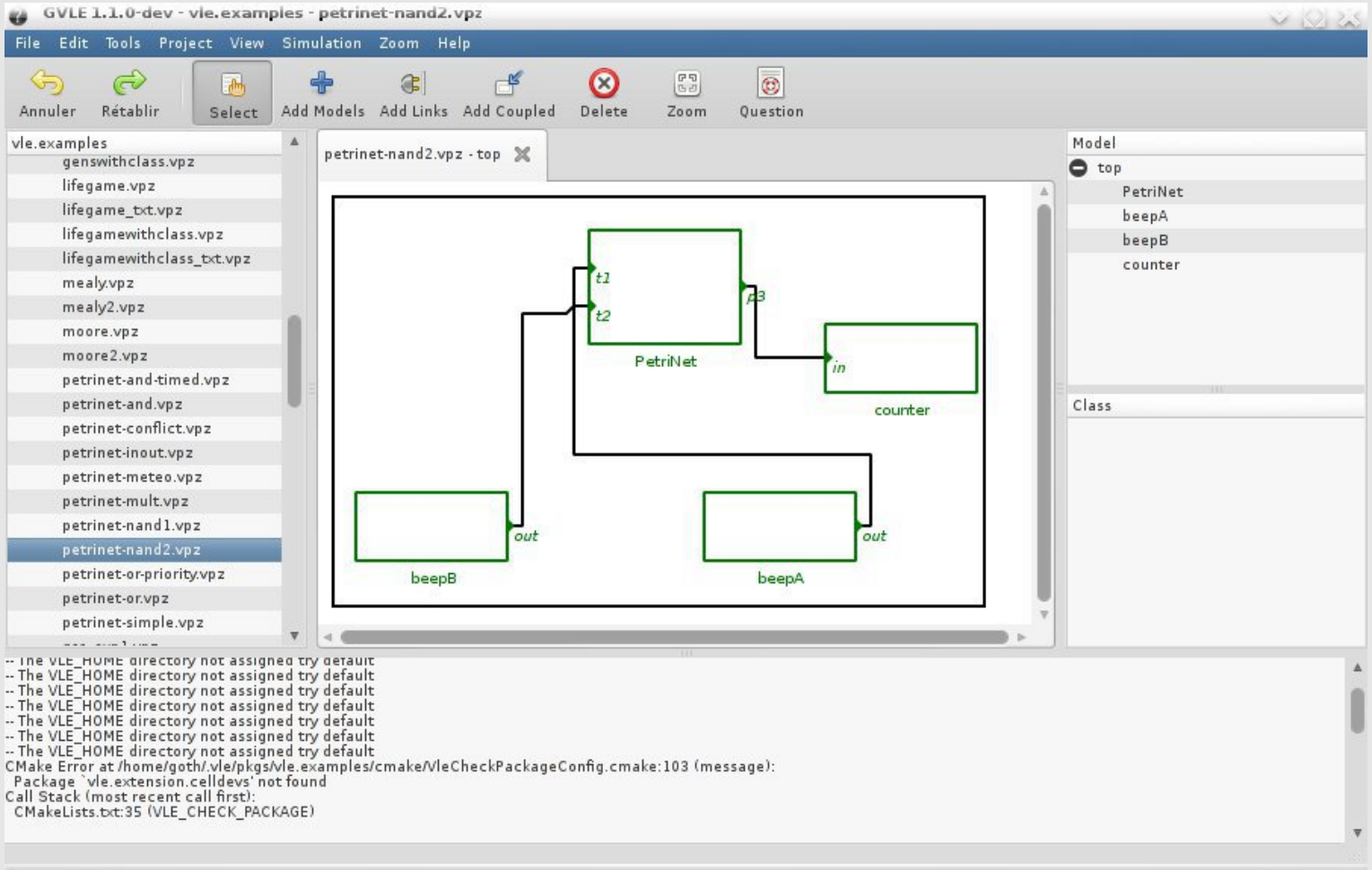
```
class MyModel: public vle::extension::DifferenceEquation
{
  [...]

  virtual double compute(const Time& time)
  {
    x = y(-1) + z(-1);
    y = z(-1);
  }
};
```

Développement de modèles

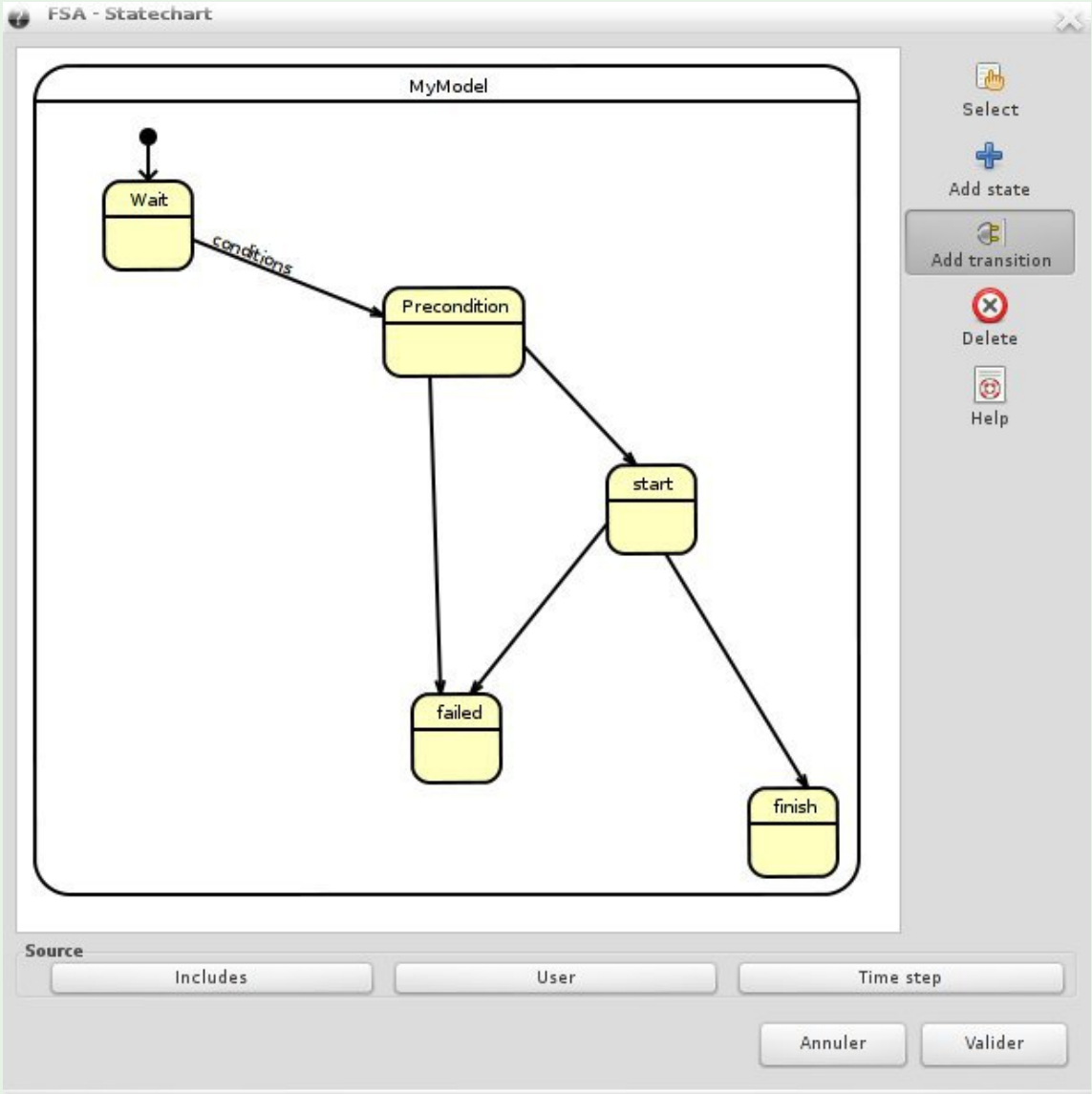
GVLE : GUI of VLE, un IDE

Pour développer des codes sources, la structure du modèle, les conditions initiales, les observations, les cadres expérimentaux :



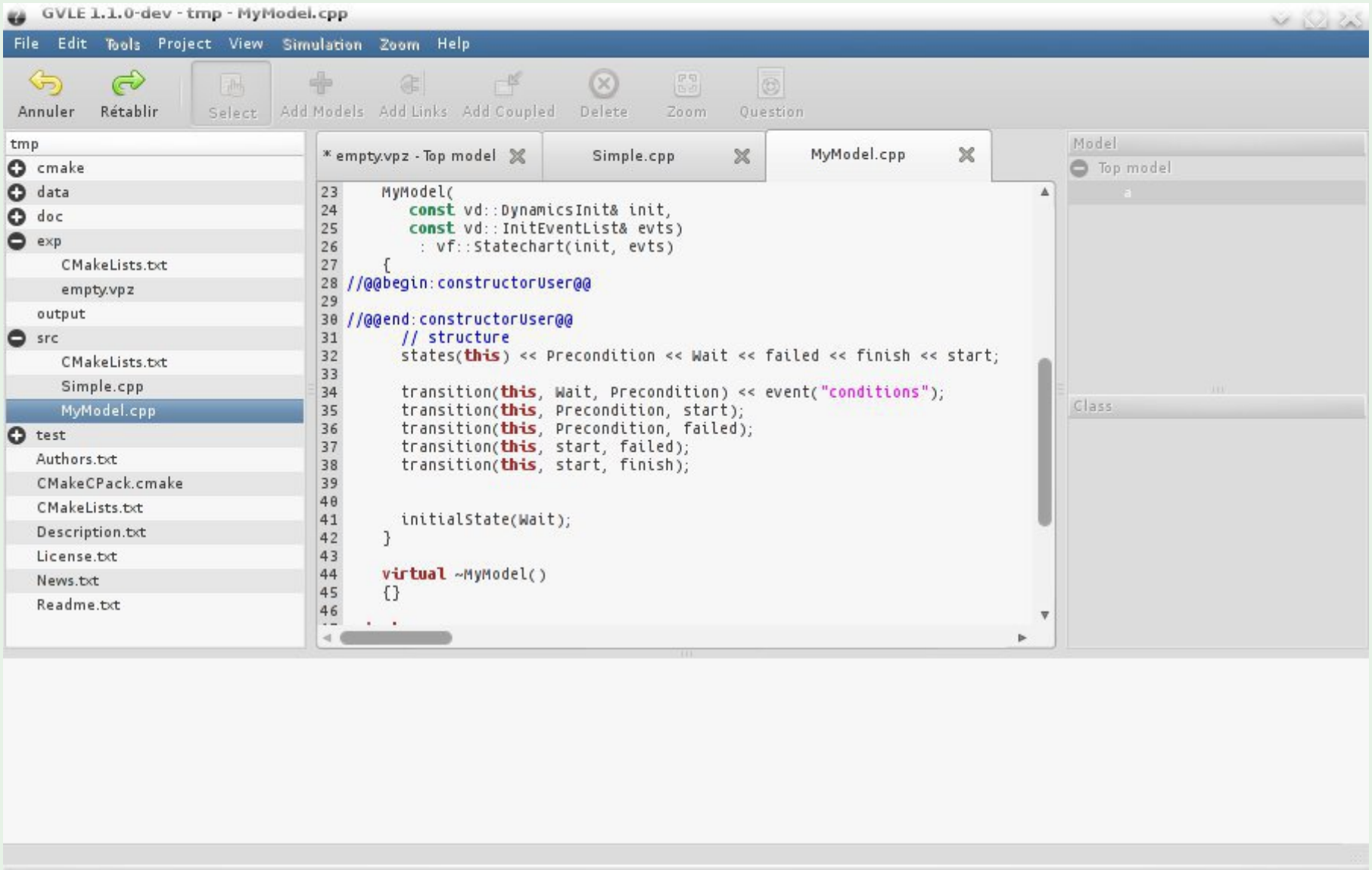
Développement de modèles : exemple

GUI pour développer un automate



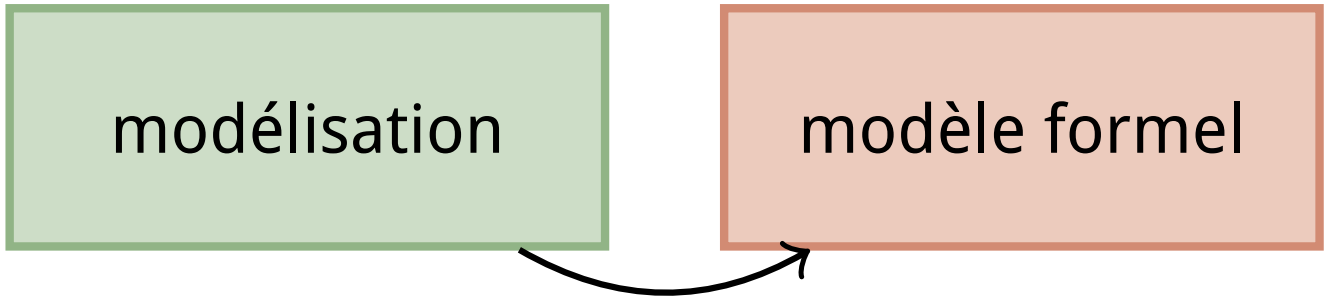
Développement de modèles : exemple

Génération de code source



VLE : Environnement complet de M&S

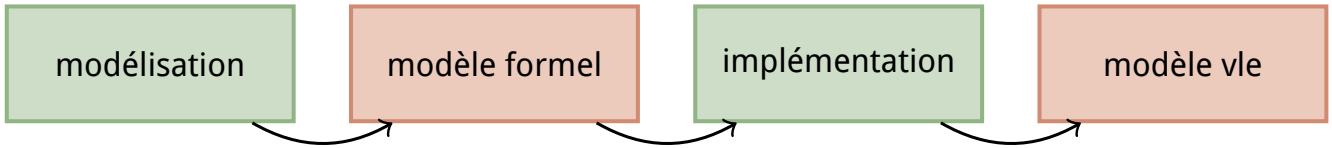
Le cycle de modélisation



Modélisation à l'aide des outils classiques de modélisation, équations différentielles, équations aux différences, automates, etc. tout en restant dans une modélisation de systèmes dynamiques.

VLE : Environnement complet de M&S

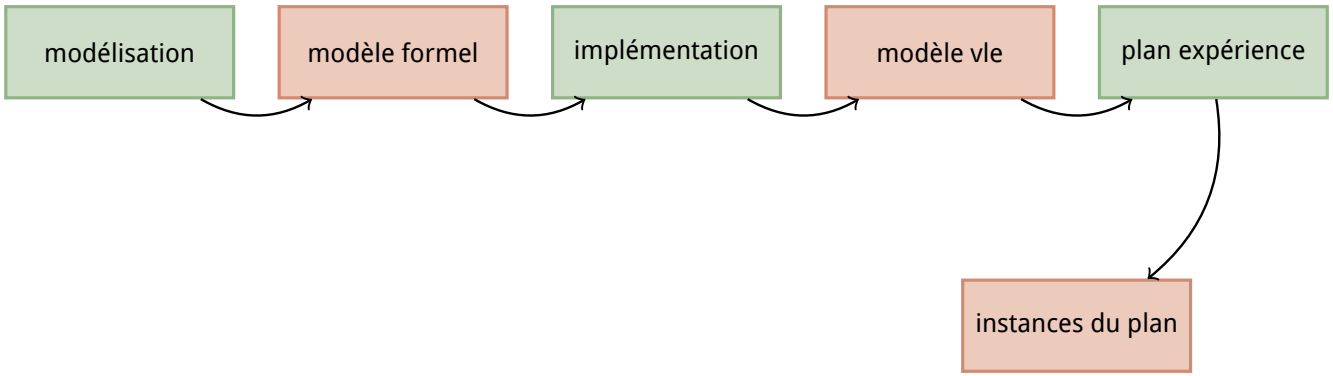
Le cycle de modélisation



Implémentation en code informatique de votre modèle, utilisation des bibliothèques de vle et des extensions proposées et de l'interface graphique gvle pour composer vos modèles.

VLE : Environnement complet de M&S

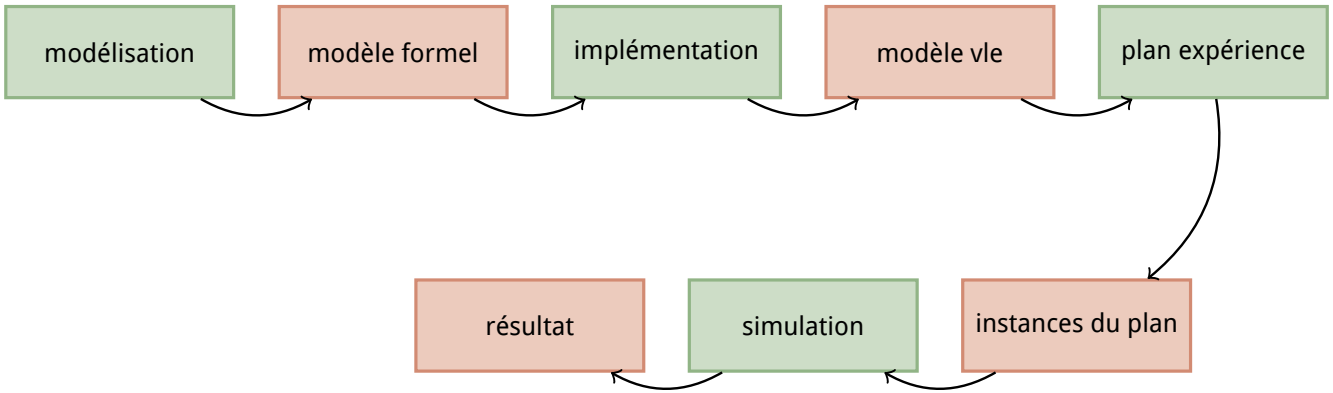
Le cycle de modélisation



Préparer la plan d'expérience, initialisation des paramètres et des variables, le nombre de répliquas, les modèles à observer, comment, et où diriger les données : gvle.

VLE : Environnement complet de M&S

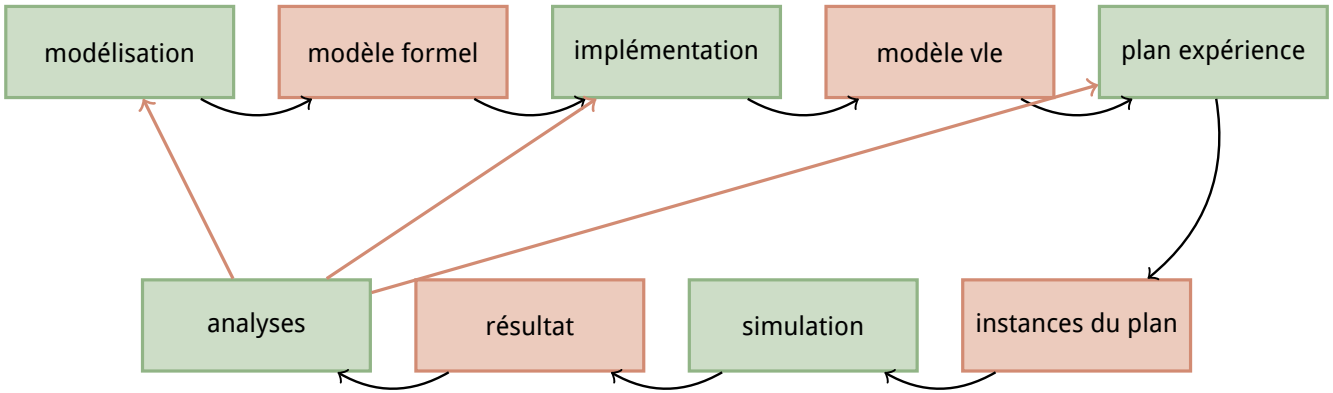
Le cycle de modélisation



Exécute les simulations depuis vle, gvle, rvle, pyvle, sur une machine locale ou une grille de calculs.

VLE : Environnement complet de M&S

Le cycle de modélisation



Offrir aux modélisateurs l'accès à DEVS, pour la modélisation (de modèles hétérogènes), la simulation (sur grille de calculs), et l'analyse des sorties, et si possible en proposant une intégration dans leurs outils.

Conclusion

Vers où ?

- Plus de formalismes (Multi-agents, équations différentielles spatialisées etc.)
- Plusieurs noyaux de simulation DEVS spécialisés (parallélisation, distribution, hybride temps réel etc.)
- Intégrer des travaux de validation
- Plus d'IHM
- Plus de port (Matlab, Scilab, etc.)

Conclusion

Vers où ?

- Plus de formalismes (Multi-agents, équations différentielles spatialisées etc.)
- Plusieurs noyaux de simulation DEVS spécialisés (parallélisation, distribution, hybride temps réel etc.)
- Intégrer des travaux de validation
- Plus d'IHM
- Plus de port (Matlab, Scilab, etc.)

Autre projet : PROTEUS

- « Plate-forme pour la Robotique Organisant les Transferts Entre Utilisateurs et Scientifiques »
- Groupe de Recherche en Robotique et partenaires industriels (Dassault Aviation, CEA, Thales, LASMEA, INRIA, Onera, etc.
- Utilisation de VLE comme simulateur de robot

Références

G. Quesnel, R. Duboz, and É. Ramat.

The Virtual Laboratory Environment -- An operational framework for multi-modelling, simulation and analysis of complex dynamical systems.

Simulation Modelling Practice and Theory, 17:641--653, April 2009.

B. P. Zeigler, D. Kim, and H. Praehofer.

Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.

Academic Press, 2000.