



DÉVELOPPEMENT DE SYSTÈMES CRITIQUES AVEC LA MÉTHODE EVENT-B

INTRODUCTION À LA MODÉLISATION AVEC EVENT-B

🎓 3A cursus ingénieurs - Mention Sciences du Logiciel





🏛️ CentraleSupélec - Université Paris-Saclay - 2024/2025



Idir AIT SADOUNE

idir.aitsadoune@centralesupelec.fr

PLAN

-  Les méthodes formelles
-  L'analyse des besoins
-  L'activité de modélisation
-  En résumé ...

[Retour au plan](#) - [Retour à l'accueil](#)

PLAN

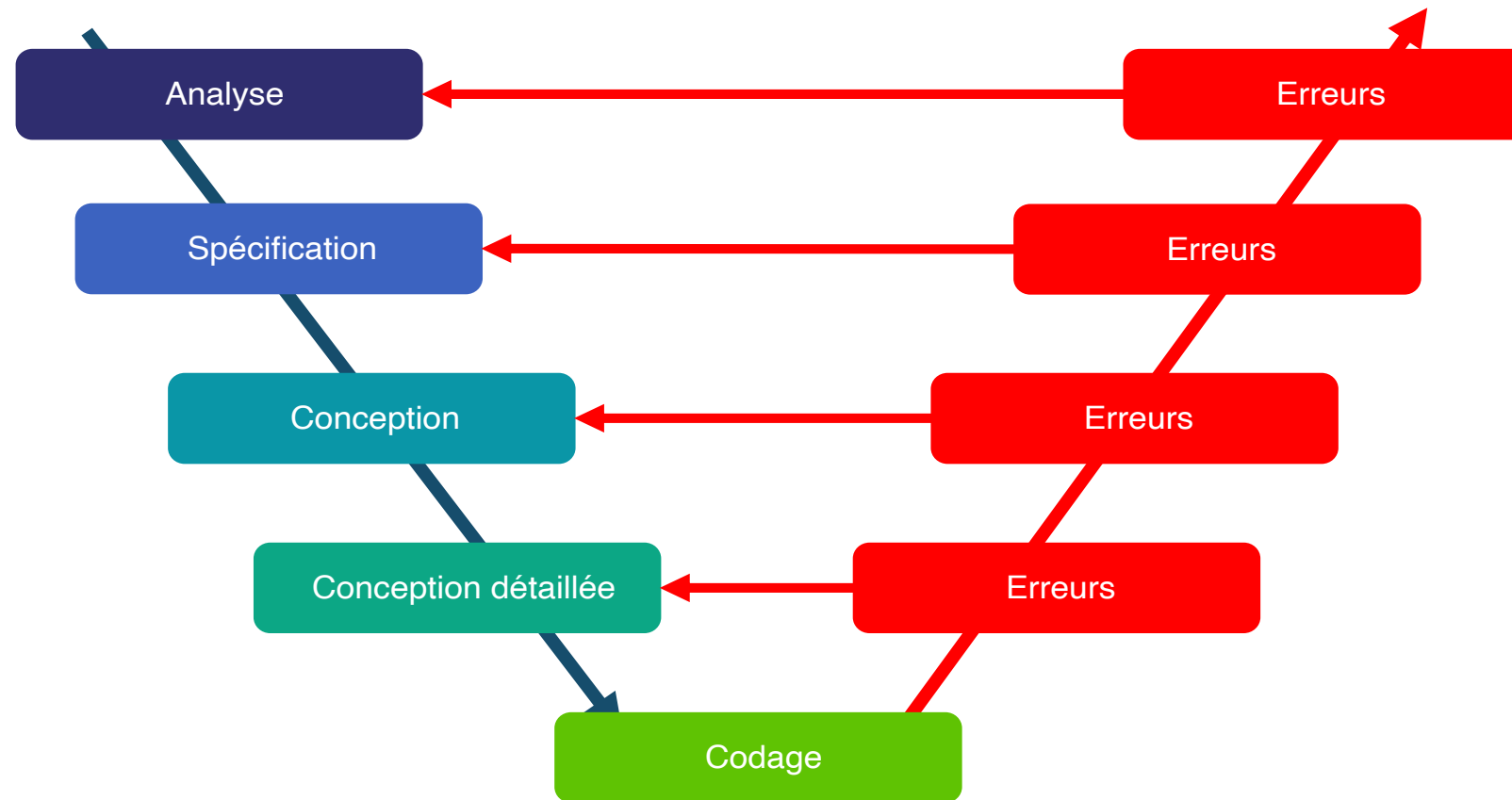
- > Les méthodes formelles
- > L'analyse des besoins
- > L'activité de modélisation
- > En résumé ...

[Retour au plan](#) - [Retour à l'accueil](#)

LE LOGICIEL INFORMATIQUE



CYCLE DE DÉVELOPPEMENT



Des **erreurs** possibles à toutes les étapes du développement.

LOGICIELS CRITIQUES

- **Une défaillance** dans un logiciel peut avoir des **conséquences catastrophiques** (humaines, financières, ...).
- Exemple du calculateur de bord d'**Ariane 5**
 - ➡ Vol 241/5101 du 25 janvier 2018



SITUATIONS À ÉVITER !!!

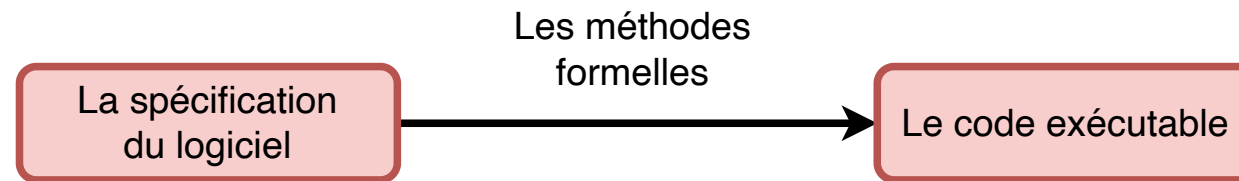


SOLUTIONS

- Les **règles** et les **techniques** de programmation.
- Le **support** des langages de programmation.
- Les **méthodologies de conception** et de développement.
- Le **test**.
- **Les méthodes formelles.**

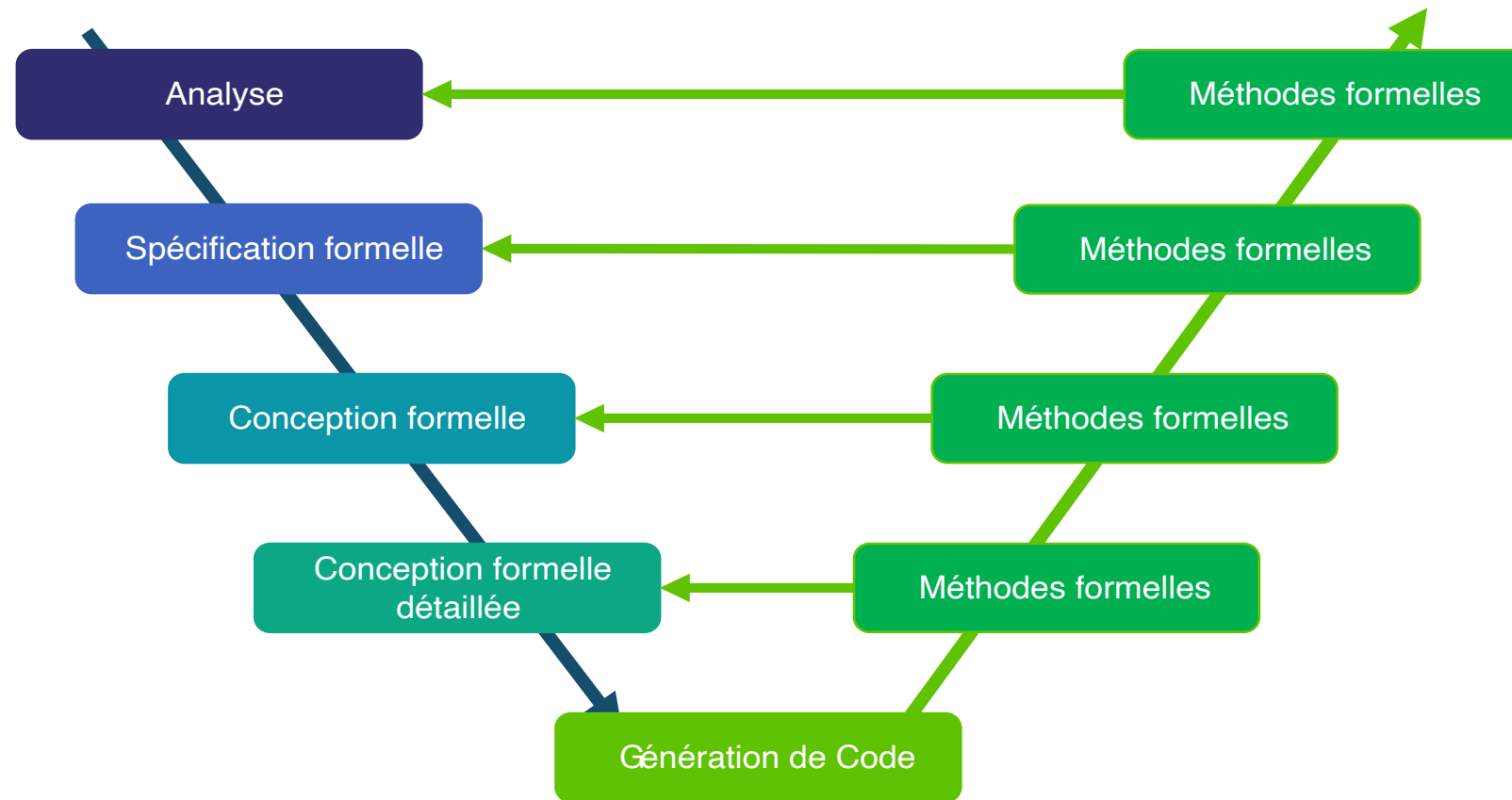
OBJECTIF DES MÉTHODES FORMELLES

➡ Aider les **ingénieurs** à effectuer la **transformation** suivante :



ne semble pas différent de **la programmation** ordinaire

LA PLACE DES MÉTHODES FORMELLES



Utiliser les **méthodes formelles** dans **toutes les étapes**.

QUI RECOMMANDE LES MÉTHODES FORMELLES ?

- **Normes européennes**

L'utilisation de spécifications formelles seule rend **les exigences non ambiguës**.

- **Normes de l'aéronautique**

L'utilisation de méthodes formelles a pour but d'**éliminer les erreurs de spécification**, de **conception** et de **codage** lors du développement.

- **Normes du ferroviaire**

Pour les spécifications, des méthodes formelles sont recommandées car le modèle formel fournit **précision**, **non ambiguïté** et **cohérence**.

EXEMPLES DE NORMES

- Les **normes européennes** EN 50126, EN 50128, EN 50129
 - ➡ des **standards** utilisés dans le **domaine ferroviaire**.
 - ➡ requises pour les fournisseurs d'**équipements de contrôle-commande**.



LES MÉTHODES FORMELLES RECOMMANDÉES

- Quelques **méthodes formelles** recommandées par les **normes** :
 - ➡ "CSP, HOL, LOTOS, Temporal Logic, **B Method**, Model Checking ..."
 - ➡ **page 103** de la norme **EN 50128**

LES TYPES DE MÉTHODES FORMELLES

- Une méthode formelle est une approche systématique utilisée pour déterminer si un programme respecte des propriétés spécifiques.
- Différents types de méthodes formelles (selon cette définition) :
 - Type checking ou la vérification de type
 - Static analysis ou l'interprétation abstraite
 - Model checking ou la vérification de modèles
 - Theorem proving ou la preuve de théorèmes

TYPE CHECKING

- Contrôler les **propriétés de bas niveau** des variables dans un programme.
- Un **type** définit :
 - un **ensemble de valeurs** à affecter à une **variable**
 - les **opérations** que l'on peut effectuer sur une variable
 - la façon dont une variable sera **stockée dans la mémoire**
- **Type checking** contrôle si :
 - **les affectations** de valeur à une variable sont **correctes**
 - la variable n'est utilisée que dans **les opérations autorisées**
- Cela se fait **automatiquement** dans les **compilateurs**.

STATIC ANALYSIS

- C'est une **technique automatique** utilisée pour vérifier qu'un programme n'aura pas certaines **erreurs d'exécution**.
- **Erreurs d'exécution** typiques détectées :
 - division par zéro
 - débordement lié au tableau
 - débordement arithmétique (virgule flottante)
 - ...
- L'analyse est effectuée en **abstrayant les variables du programme** et en exécutant l'**abstraction résultante**
 - ➡ l'**interprétation abstraite** peut conduire à une **fausse alerte**

MODEL CHECKING

- Les propriétés à vérifier **ne sont pas des propriétés du programmes**
 ▹ ce sont des propriétés **du modèles du programme**
- Habituellement, ces modèles désignent des **machines à états finis**
 (**états** et **transitions**)
- Les **propriétés à vérifier** sont souvent des **propriétés temporelles**
 (**accessibilité** d'un état)
- Les **Model checkers** fonctionnent **automatiquement**

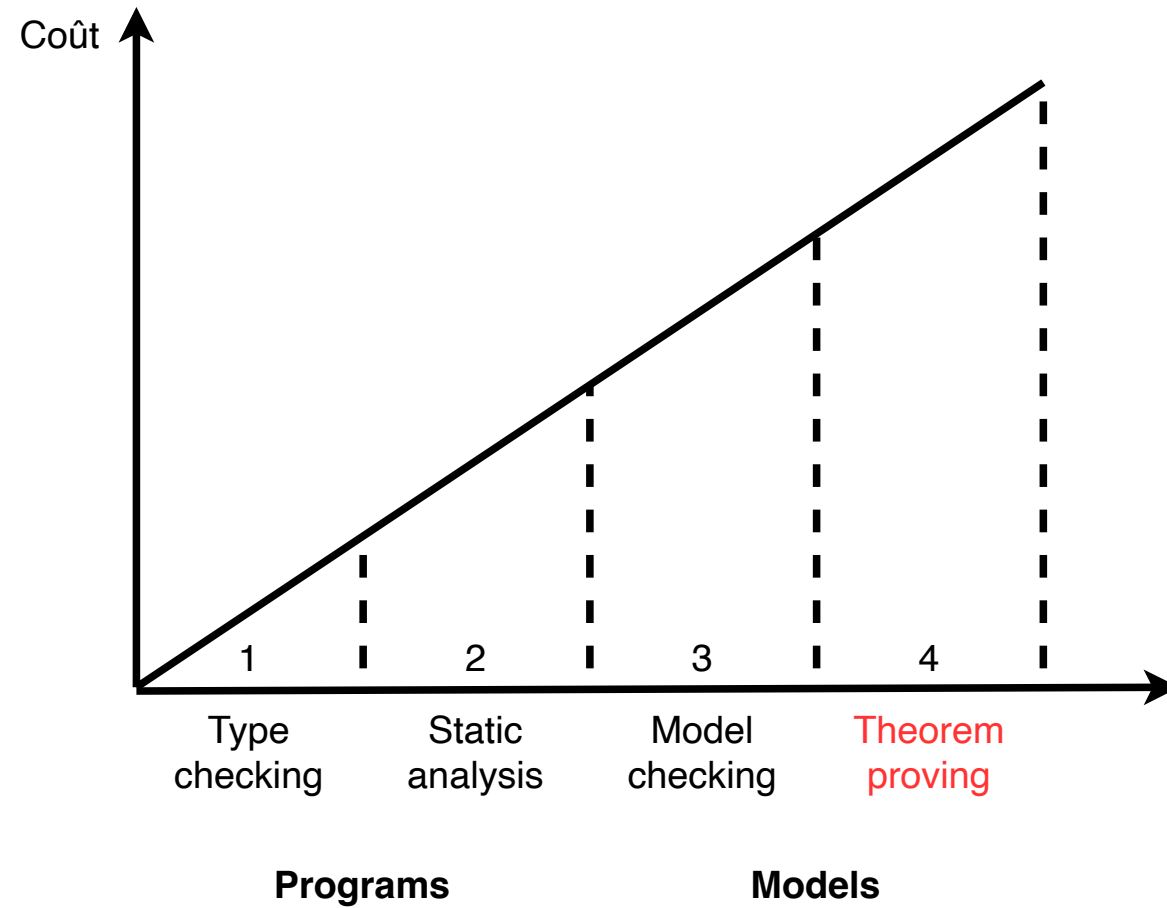
THEOREM PROVING

- C'est l'approche que je vais **développer dans ce cours**.
 - la méthode **Event-B**
- Les propriétés à prouver (par **démonstration** semi-automatique) font partie des modèles :
 - **théorèmes** et **invariants**.
- La méthode **Event-B** se base sur la construction de modèles par **raffinements successifs**.
- A la fin du processus, le modèle le plus concrêt (le **dernier niveau de raffinement**) est traduit **automatiquement en programme**.

COMPARAISONS

- En **Type checking** et en **Static Analysis**, on travaille sur des **programmes**
- En **Model Checking** et en **Theorem Proving**, on travaille sur des **modèles**
- En **Type checking** et en **Theorem Proving**, on prouve une propriété qui **fait partie de l'objet analysé**
- En **Static Analysis** et en **Model Checking**, on prouve une propriété qui est **proposé en externe**

COÛTS



POURQUOI UTILISER DES MÉTHODES FORMELLES BASÉES SUR LA PREUVE ?

- Lorsque le **risque** est trop élevé (ex. les systèmes critiques)
- Quand il n'y a **rien de mieux disponible**.
- Quand les gens remettent en question leur **processus de développement**.
- La décision d'utiliser des méthodes formelles est **toujours stratégique**.

QUESTIONS À SE POSER SUR LA MÉTHODE FORMELLE À UTILISER

- Y a-t-il une **théorie** derrière la méthode formelle ?
- Quel type de **langage** utilise la méthode formelle choisie ?
- Existe-t-il un mécanisme de **raffinement** ?
- Y a-t-il un **prouveur automatique** efficace ?

PLAN

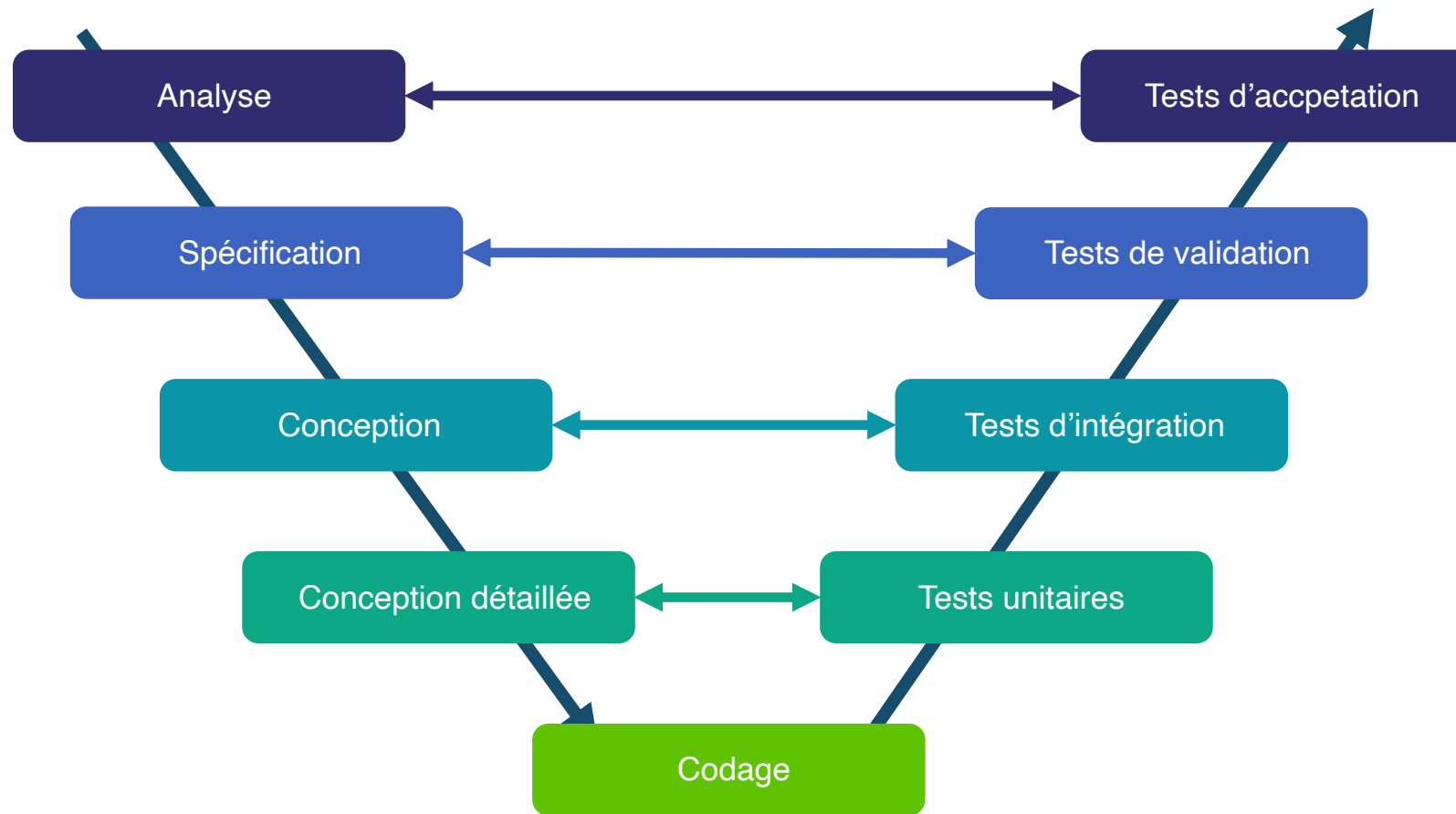
- > Les méthodes formelles
- > L'analyse des besoins
- > L'activité de modélisation
- > En résumé ...

[Retour au plan](#) - [Retour à l'accueil](#)

AVANT LA MODÉLISATION

- Définir les principaux **objectifs** du futur système
- Définir les **exigences**
- Étude de **faisabilité**

LE CYCLE DE DÉVELOPPEMENT



Partie la plus faible : le cahier des charges (**Analyse**)

LE CAHIER DES CHARGES

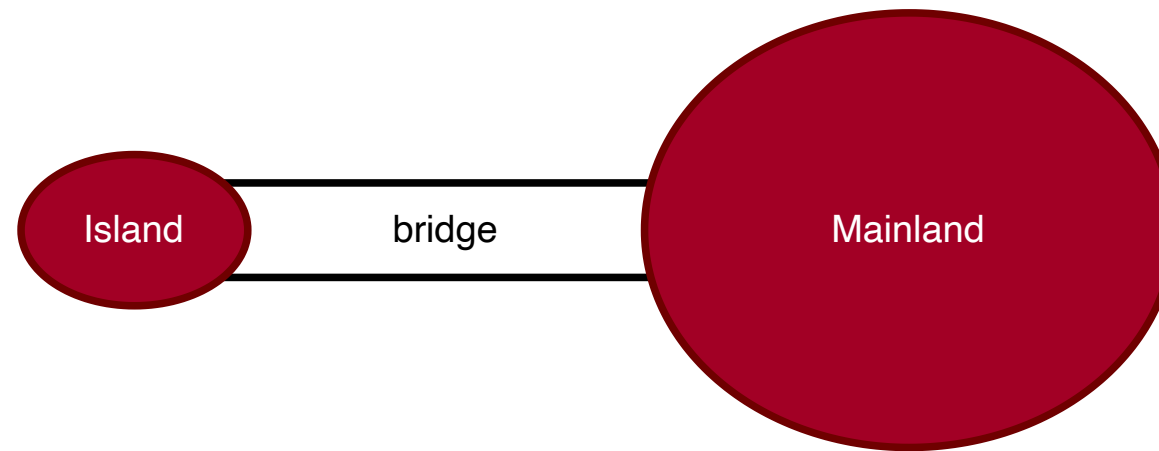
- **Importance** de ce document (sa position dans le cycle de vie)
- Les cahiers des charges sont généralement **difficiles à exploiter**
- D'où très souvent la nécessité de **le réécrire**

EXEMPLE D'UN CAHIER DES CHARGES

- Le système que nous allons construire est **un logiciel connecté à certains équipements**.
- Il existe deux types d'exigences :
 1. les exigences concernées par **le matériel**, labellisées **EQP**,
 2. les exigences concernés par **la fonction du système**, étiquetés **FUN**.
- La fonction de ce système est de **contrôler les voitures sur un pont étroit**.
Ce pont est censé relier **le continent** à **une petite île**.

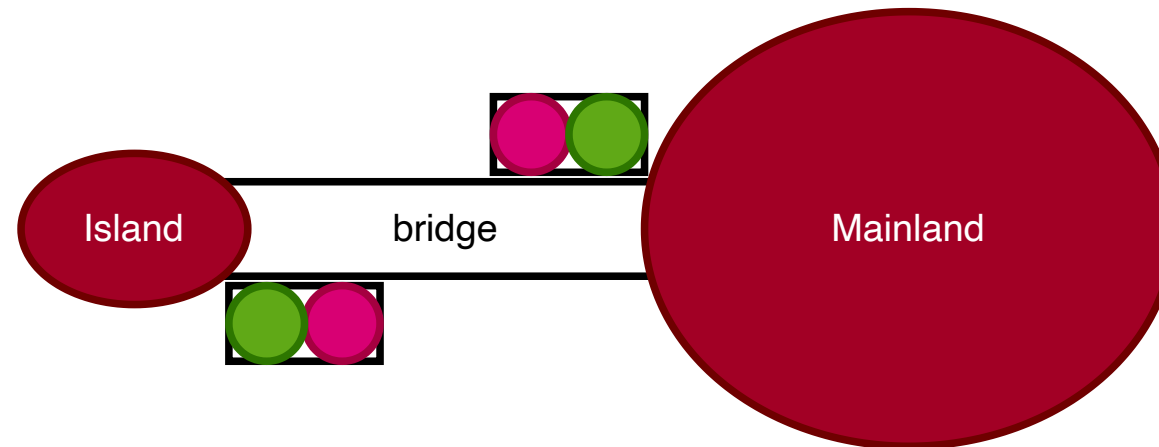
EXEMPLE D'UN CAHIER DES CHARGES

- **FUN-1** → the system is controlling cars on a bridge between the mainland and an island.



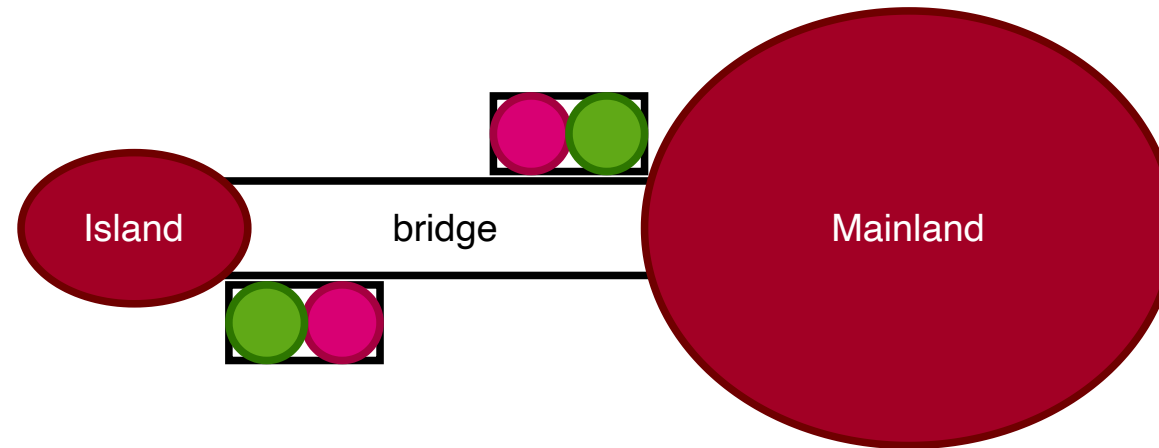
EXEMPLE D'UN CAHIER DES CHARGES

- **EQP-1** → the system has two traffic lights with two colors: green and red, one of the traffic lights is situated on the mainland and the other one on the island Both are close to the bridge.



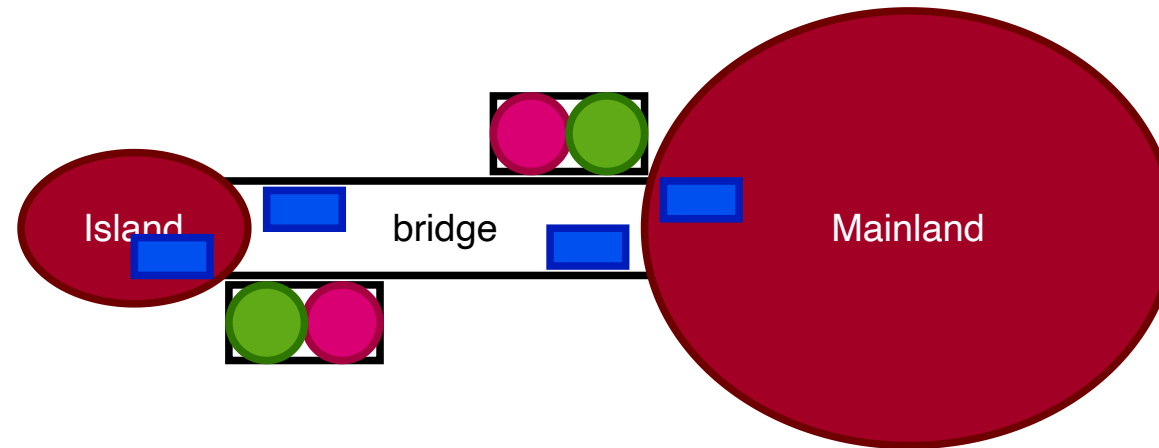
EXEMPLE D'UN CAHIER DES CHARGES

- **EQP-2** → the traffic lights control the entrance to the bridge at both ends of it.
- **EQP-3** → cars are not supposed to pass on a red traffic light, only on a green one.



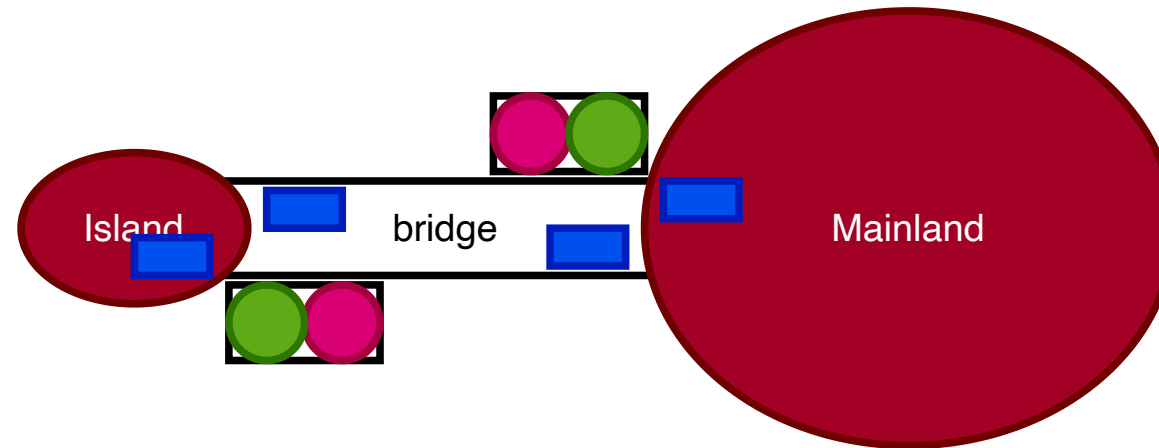
EXEMPLE D'UN CAHIER DES CHARGES

- **EQP-4** → the system is equipped with four car sensors each with two states: on or off.
- **EQP-5** → the sensors are used to detect the presence of cars entering or leaving the bridge.



EXEMPLE D'UN CAHIER DES CHARGES

- **FUN-2** → the number of cars on the bridge and the island is limited.
- **FUN-3** → the bridge is one way or the other, not both at the same time.



PLAN

- > Les méthodes formelles
- > L'analyse des besoins
- > L'activité de modélisation
- > En résumé ...

[Retour au plan](#) - [Retour à l'accueil](#)

LES MÉTHODES B ET EVENT-B

- **Méthodes formelles** permettant le **développement de logiciels sûrs**.
- Conçue par le mathématicien français **J.R Abrial** en **1996**.
- Repose sur les travaux menés à l'université d'Oxford par **C.A.R. Hoare**.
- **The B-Book** et **Modeling in Event-B** de J.R. Abrial sont les ouvrages fondamentaux des méthodes **B** et **Event-B**.
- La méthode **Event-B** est basée sur les **mathématiques discrètes** :
 - **Théorie des ensembles** (ensembles, relations, fonctions, ...)
 - **Logique du premier ordre** (calcul des prédicats)
- En utilisant des **conventions**, cela devrait nous faciliter le raisonnement
- Les modèles permettent de **raisonner** sur le futur système

EXEMPLES DE DÉVELOPPEMENT

- Systèmes de trains entièrement **automatiques**
- Ligne 14 du métro parisien (octobre 1998)
- Navette de l'aéroport de Roissy CDG (mars 2007)
- Dans chaque cas, seule la **partie critique** pour la sécurité est réalisée avec une méthode formelle utilisant la preuve (**la méthode B**)

COMPARAISON DES DÉVELOPPEMENTS

	Paris	Roissy
Nombre de lignes ADA (à partir de B)	86 000	158 000
Nombre de preuves	27 800	43 610
Pourcentage de la preuve interactive	8.3 %	3.3 %

Dans les deux cas :

pas de tests unitaires et **pas de tests d'intégration**

MÉTHODE B & SECTEURS D'ACTIVITÉS

- Premier métro autonome ([Ligne 14 - Projet Meteor - 1998](#))
- Ferroviaire : [exemples de projets](#)
- Automobile : [exemples de projets](#)
- Energie : [exemples de projets](#)
- Micro-électronique : [exemples de projets](#)
- Défense : [exemples de projets](#)

L'ÉTAT D'UN MODÈLE

- Un **modèle discret** est d'abord constitué d'un **état**
- L'état est représenté par des **constantes** et des **variables**
- Les **constantes** sont liées par certaines **propriétés**
- Les **variables** sont liées par des **invariants**
- Les **propriétés** et les **invariants** sont écrits à l'aide d'expressions de **la théorie des ensembles**

UNE VUE SCHÉMATIQUE DU MODÈLE

CONTEXT ctx_1
EXTENDS ctx_2

SETS s
CONSTANTS c
AXIOMS

$A(s, c)$
THEOREMS
 $T(s, c)$
END

MACHINE mch_1
REFINES mch_2
SEES ctx_i

VARIABLES v
INVARIANTS
 $I(s, c, v)$
THEOREMS
 $T(s, c, v)$

...
END

LES ÉVÉNEMENTS D'UN MODÈLE (TRANSITIONS)

- Un **modèle discret** est également constitué d'un certain nombre d'événements
- Un **événement** est composé d'une **garde** et d'une **action**
- La **garde** indique la **condition d'activation** de l'événement
- L'**action** indique la façon dont **l'état est modifié** par l'événement
- Les **gardes** et les **actions** sont écrites à l'aide d'expressions de **la théorie des ensembles**

UNE VUE SCHÉMATIQUE DU MODÈLE

CONTEXT ctx_1
EXTENDS ctx_2

SETS s
CONSTANTS c
AXIOMS

$A(s, c)$
THEOREMS
 $T(s, c)$
END

MACHINE mch_1
REFINES mch_2
SEES ctx_i

VARIABLES v
INVARIANTS

$I(s, c, v)$
THEOREMS
 $T(s, c, v)$
EVENTS
 $[events_list]$
END

$event \hat{=}$
any x
where
 $G(s, c, v, x)$
then
 $BA(s, c, v, x, v')$
end

INTERPRÉTATION OPÉRATIONNELLE

- L'exécution d'un événement est censée **ne pas prendre de temps**
- Deux événements **ne peuvent pas se produire simultanément**
- Si tous les événements ont des gardes fausses, **le système s'arrête**
- Si plusieurs événements ont des gardes vraies, **l'un d'eux est choisi** de manière non déterministe, et **son action modifie l'état**
- La phase précédente est **répétée** (si possible)

INTERPRÉTATION OPÉRATIONNELLE

```
1 Initialize;  
2 while (some events have true guards) {  
3   Choose one such event;  
4   Modify the state accordingly;  
5 }
```

- L'arrêt n'est pas nécessaire : **un système discret peut s'exécuter indéfiniment**
- Cette interprétation est juste donnée ici pour une **compréhension informelle**
- L'**interprétation** d'un système discret est **donnée par les preuves** qui peuvent être effectuées sur lui

UNE VUE SCHÉMATIQUE DU MODÈLE

```

CONTEXT  $ctx_1$ 
EXTENDS  $ctx_2$ 

SETS  $s$ 
CONSTANTS  $c$ 
AXIOMS
   $A(s, c)$ 
THEOREMS
   $T(s, c)$ 
END
  
```

```

MACHINE  $mch_1$ 
REFINES  $mch_2$ 
SEES  $ctx_i$ 

VARIABLES  $v$ 
INVARIANTS
   $I(s, c, v)$ 
THEOREMS
   $T(s, c, v)$ 
EVENTS
   $[events\_list]$ 
END
  
```

```

event  $\hat{=}$ 
  any  $x$ 
  where
     $G(s, c, v, x)$ 
  then
     $BA(s, c, v, x, v')$ 
  end
  
```

$$\begin{aligned}
 &A(s, c) \vdash T(s, c) \\
 &A(s, c) \wedge I(s, c, v) \vdash T(s, c, v) \\
 &A(s, c) \wedge I(s, c, v) \wedge G(s, c, v, x) \vdash \exists v'. BA(s, c, v, x, v') \\
 &A(s, c) \wedge I(s, c, v) \wedge G(s, c, v, x) \wedge BA(s, c, v, x, v') \vdash I(s, c, v') \\
 &\dots
 \end{aligned}$$

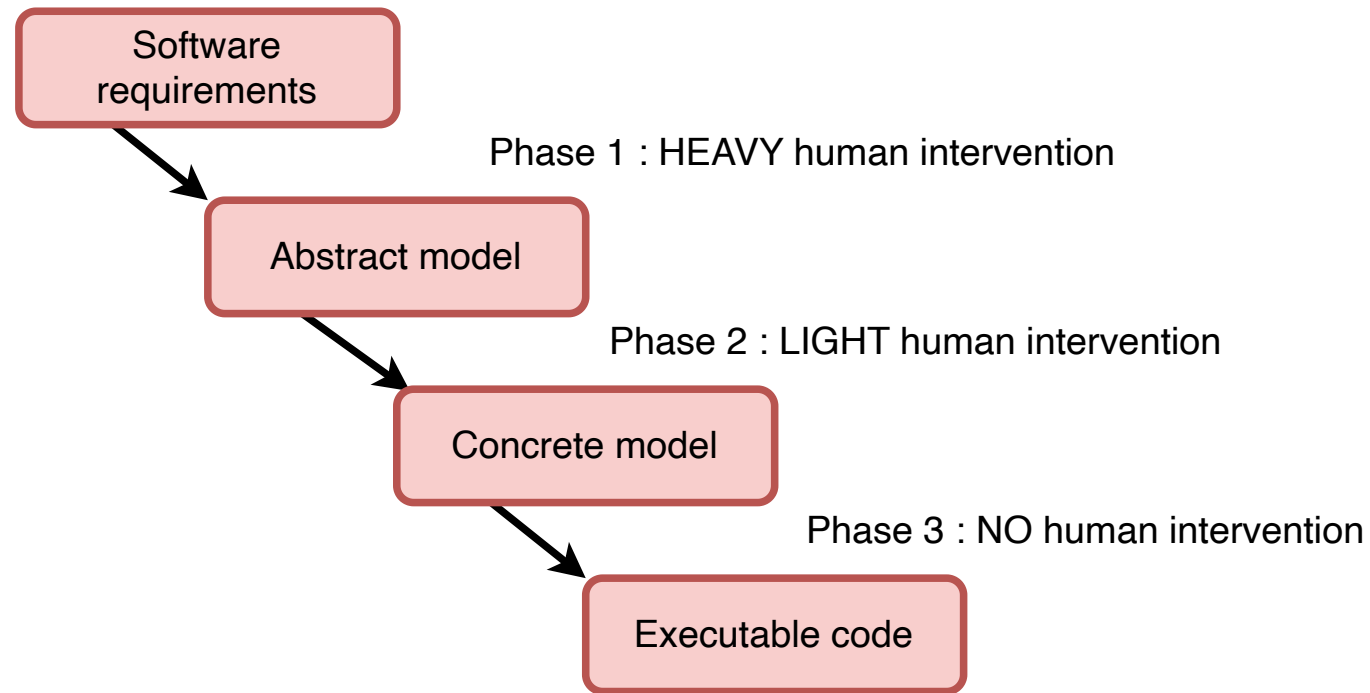
DÉVELOPPER DES SYSTÈMES COMPLEXES

- La formalisation contient des modèles :
 - des **futurs composants logiciels**
 - des **futurs équipements** autour de ces composants
- La construction globale du modèle peut être **très complexe**
- Le **raffinement** peut être utilisé pour maîtriser cette complexité

LE RAFFINEMENT

- Le **raffinement** nous permet de construire un modèle **progressivement**
- Nous construirons **une séquence ordonnée** de modèles plus précis
- Chaque modèle est **un raffinement** de celui qui le précède
- Une analogie : regarder à travers **un microscope**
- Extensions **spatiales** et **temporelles**
- **Raffinement des données**

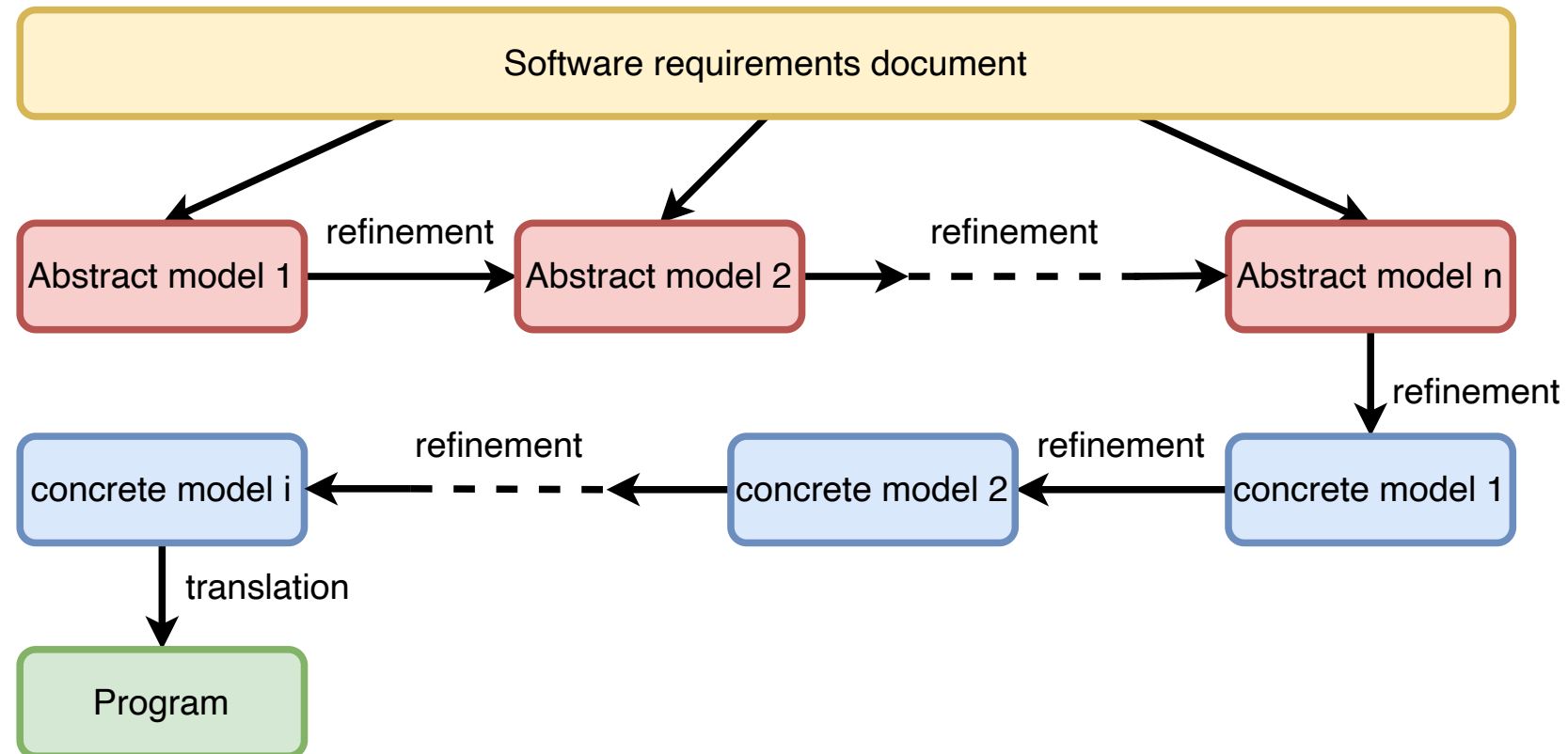
LES PHASES DU DÉVELOPPEMENT FORMEL



LES PHASES DU DÉVELOPPEMENT FORMEL

- Construction du **modèle abstrait**
 - **le document d'exigences** logicielles est donné
 - les détails de ce document sont progressivement **extraits**
 - le modèle abstrait est construit par **raffinements successifs**
- Construction du **modèle concret**
 - la construction du modèle concret se fait par **raffinements successifs**
 - les **constructions mathématiques** se transforment en **objets informatiques**
 - les **événements** se transforment en **structures de programmation**
- Obtention du **code exécutable**
 - le modèle concret est vérifié par un **outil de traduction**
 - le modèle concret est traduit en **un programme classique**

LES PHASES DU DÉVELOPPEMENT FORMEL



PLAN

- > Les méthodes formelles
- > L'analyse des besoins
- > L'activité de modélisation
- > En résumé ...

[Retour au plan](#) - [Retour à l'accueil](#)

OBJECTIFS DU COURS

- Donner un aperçu sur les activités de **modélisation** et du **raisonnement formel**
- Montrer que les programmes peuvent être **corrects par construction**
- Montrer que la modélisation peut être rendue **pratique**
- Illustrer cette approche par de nombreux **exemples**

CE QUE VOUS APPRENDREZ

À la fin du cours, vous devriez être à l'aise avec :

- ▢ La **modélisation** (versus programmation)
- ▢ L'**abstraction** et le **raffinement**
- ▢ Quelques **techniques mathématiques** utilisées pour le raisonnement
- ▢ La pratique de **la preuve** comme moyen de **construire des programmes**
- ▢ L'utilisation de certains **outils de preuve**

L'ÉVALUATION ...

Le contrôleur [Adaptive Exterior Light and Speed Control System](#)

- l'étude de cas de la conférence internationale ABZ 2020
- projet sur deux jours : $2 \times 3h$

MERCI

[Version PDF des slides](#)

[Retour à l'accueil](#) - [Retour au plan](#)