Ampère
Unité Mixte de Recherche CNRS
Génie Électrique, Électromagnétisme, Automatique, Microbiologie environnementale et Applications

Conception sûre de systèmes embarqués à base de COTS

Salam HAJJAR
Encadré par : Éric NIEL, Emil DUMITRESCU
Laboratoire AMPERE, INSA de Lyon

Modélisation des Systèmes Réactifs (MSR'13)
INRIA Rennes - Bretagne Atlantique, France
du 13 au 15 novembre 2013

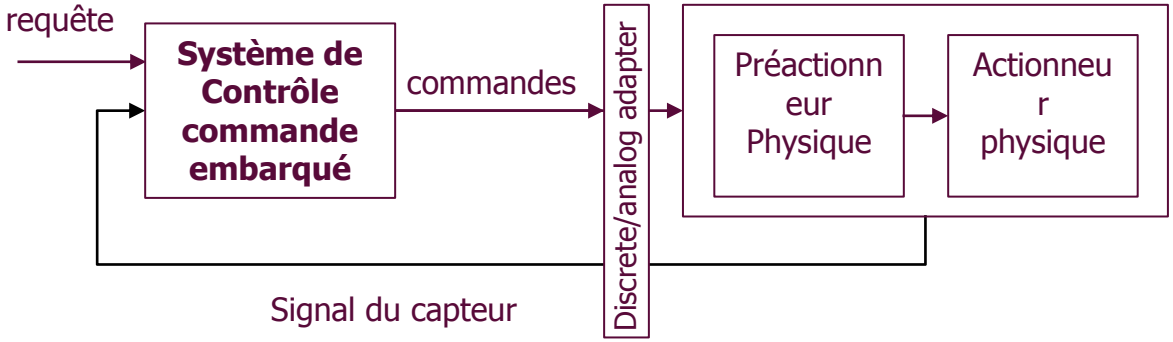
CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
Lyon 1
INSA LYON

Plan

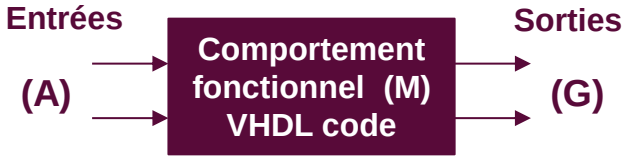
- **Contexte et problématique**
  - Contexte: Conception sûre de systèmes embarqués de contrôle-commande
  - Problématique : utilisation de COTS dans les systèmes embarqués matériels
- **Contribution**
  - Méthode de conception sûre:
    - Cible : systèmes basés sur des COTS
    - Renforcement automatique d'exigences fonctionnelles
  - Illustration : système de contrôle accès voyageurs d'un train
- **Conclusion et perspectives**
  - Résumé de la proposition
  - Perspectives

Définitions

- **Système de contrôle-commande (C-C) embarqué**:
  - Fonction de calcul dédiée à une mission spécifique dans un système
  - Contrôle: la partie physique d'un système
  - Commande: ordre (actionneurs, pré-actionneurs, etc.)

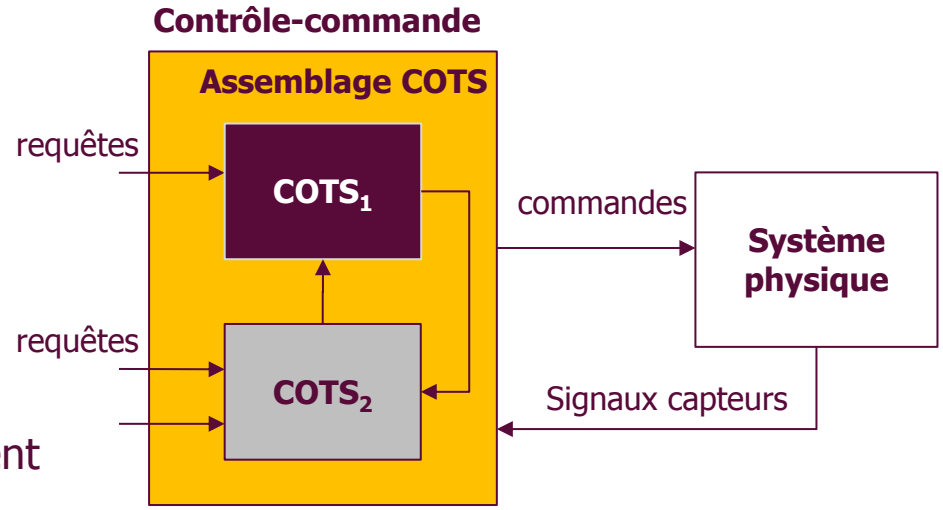


- **COTS: Composant sur étagère**
  - Composants générique, préfabriqué et réutilisable
  - Caractéristiques fonctionnels [S. Beydeda 2005]
    1. Interface (I)
    2. Préconditions (A)
    3. Post-conditions (G)
    4. Comportement fonctionnel (M) : le lien entrées/sorties/états internes



Définitions

- **Bénéfice**: économiser le temps et les coûts de conception
- Garantir la **Sûreté du système** de contrôle commande: satisfaction d'un ensemble d'exigences fonctionnelles prédéfinies par le concepteur.
- **Système basé sur COTS**
  - Assemblage de COTS
  - Réaliser une fonction de contrôle-commande complexe
  - COTS conçus séparément et pas forcement pour travailler ensemble
    - → Les interactions entre les COTS ne sont pas forcement sûres
    - → La fonction de contrôle-commande n'est pas sûre
      - → Le comportement de la partie physique n'est pas sûr



Objectif

Objectif des travaux

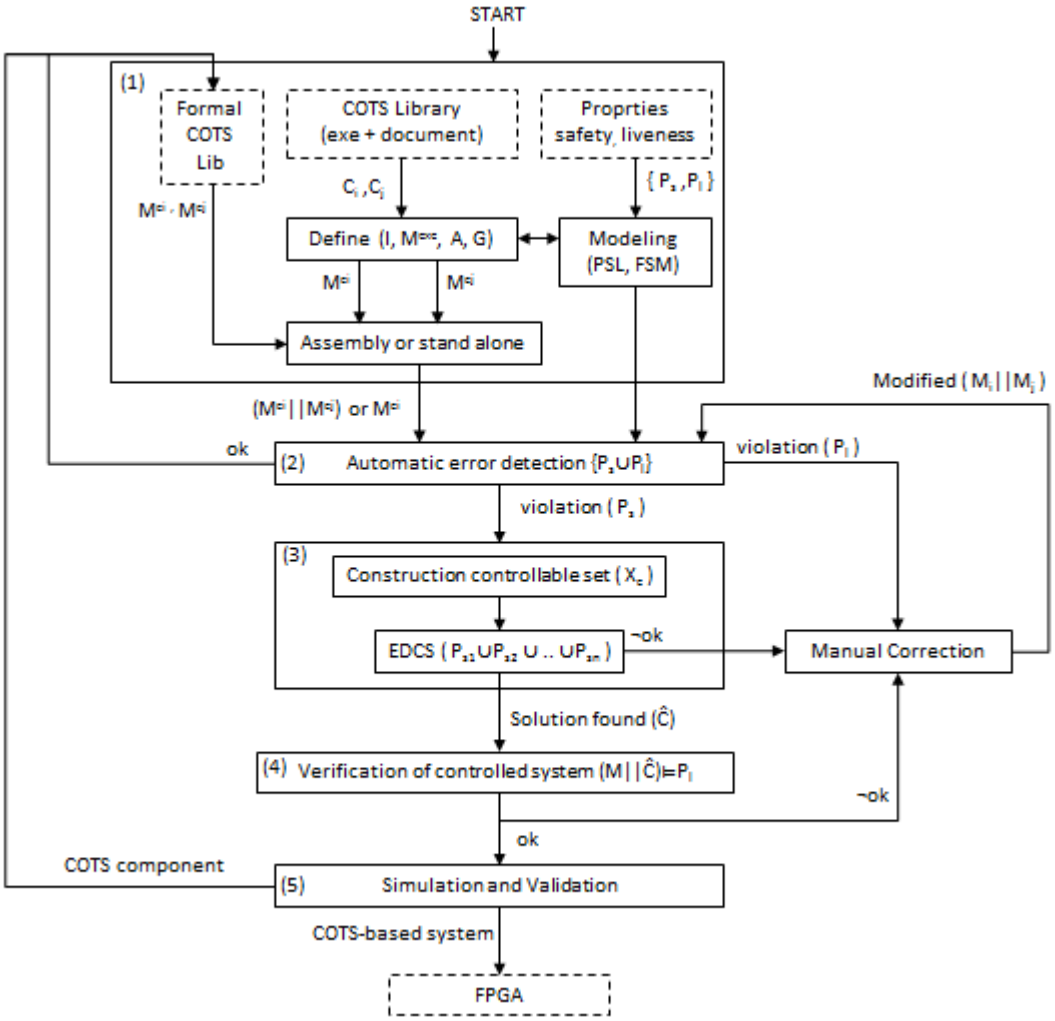
- Construction de fonctions de contrôle commande par l'assemblage des COTS

- Renforcement de la sûreté de l'assemblage en utilisant des techniques formelles et automatiques de vérification et de synthèse

Contribution

➢ **Méthode de conception sûre de systèmes embarqués**

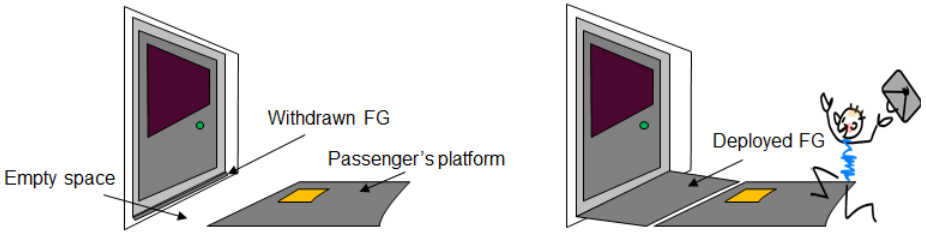
1. Modélisation
2. Détection automatique d'erreurs
3. Correction automatique d'erreurs
4. Vérification du système contrôlé
5. Simulation et validation

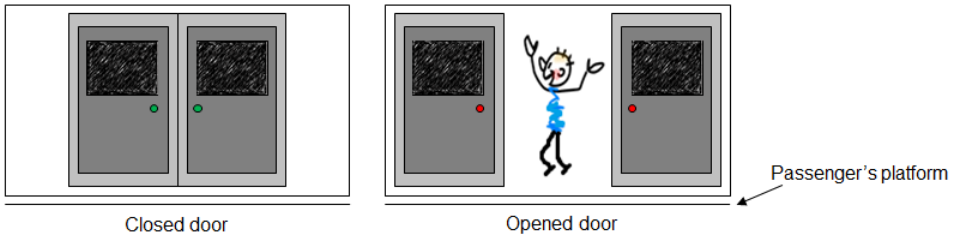


Exemple

➢ Système d'Accès voyageur

- Fourni par BOMBARDIER
- 2 COTS : C-C Porte et Emmarchement mobile



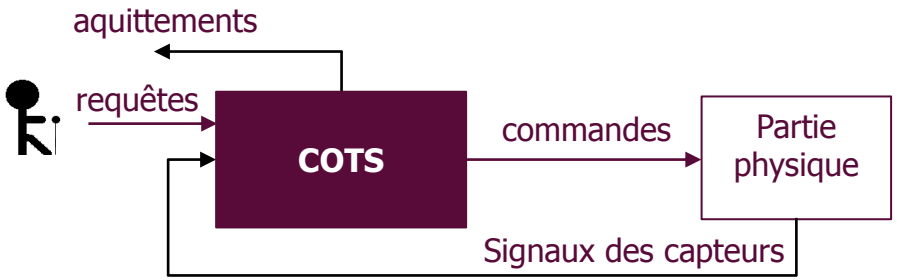


➢ Fonctionnement

- COTS porte : commande l' ouverture/fermeture de la partie physique « portes »
- COTS Emmarchement mobile : commande le déploiement/retrait de la partie physique « emmarchement mobile »

Exemple

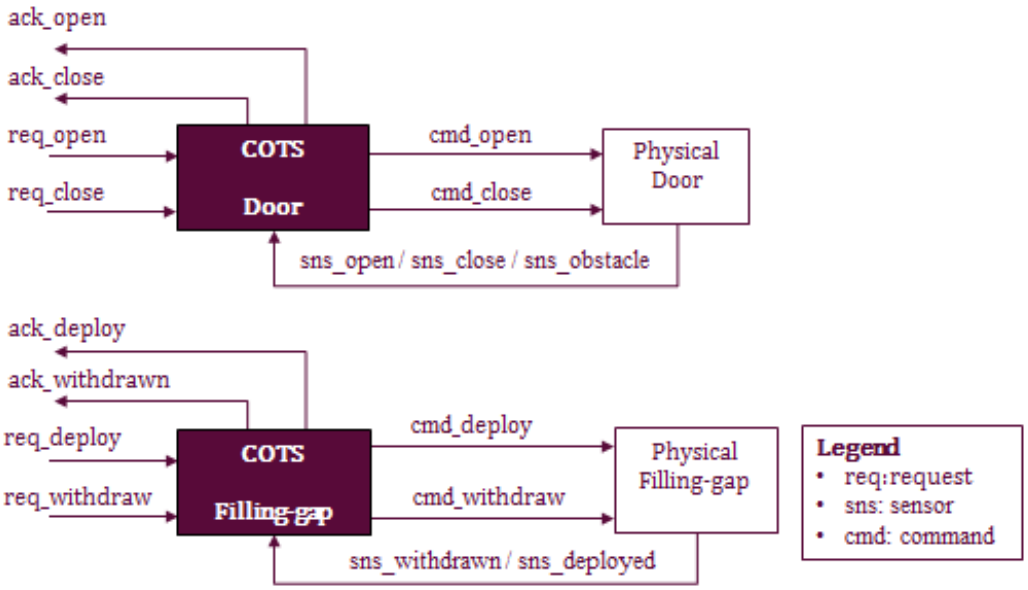
- **COTS environnent: la partie physique et le conducteur**



- **Le COTS est une boite noire**: le concepteur connait seulement ses entrées et sorties

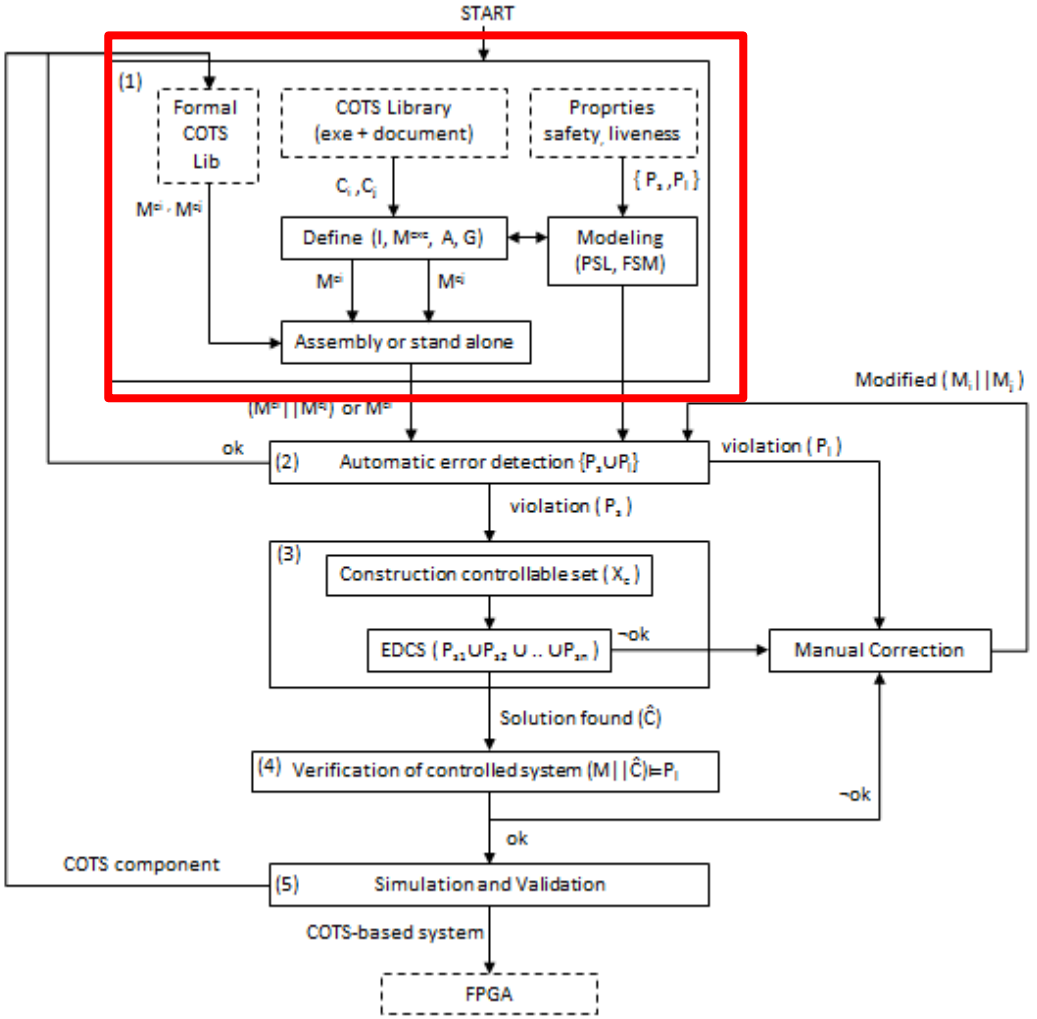
- **La porte**
  - Entrées
    - Requêtes (**req**) : open / close
    - Capteurs (**sns**) : open / closed / obstacle
  - Sorties
    - Commandes (**cmd**) : open / close
    - Acquittements (**ack**) : open / close

- **L'emmarchement mobile**
  - Entrées
    - Requêtes (**req**) : deploy / withdraw
    - Capteurs (**sns**) : withdrawn / deployed
  - Sorties
    - Commandes (**cmd**) : deploy / withdraw
    - Acquittements (**ack**) : deploy/withdraw



Étape (1) Modèlisation

1. Modélisation des COTS individuels
2. Modélisation de l'assemblage des COTS
3. Formalisation des exigences fonctionnelles



Étape (1) Modèlisation

1. Modélisation d'un COTS $C = (I^c, A^c, G^c, M^c)$

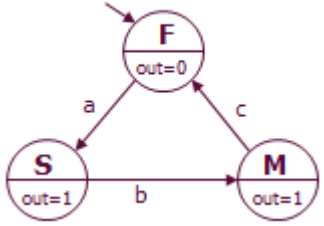
- **Interface**: un ensemble de variables Booléens qui représentent les entrées et les sorties du COTS

$$I^c = X^c \cup Y^c$$

- **Pré-conditions**: des hypothèses sur l'environnement $A^c = \{\Phi_a^C, M_a^C\}$
  - Logiques : Φ_a^C - assertions en logique temporelle
  - Opérationnelles : M_a^C - moniteurs exprimés par des machines à états finis

- **Post-conditions**: exigences garanties par le comportement du COTS si A^c sont respectées par l'environnement $G^c = \{\Phi_g^C, M_g^C\}$
  - Logiques : Φ_g^C
  - Opérationnelles : M_g^C

- **Modèle comportemental** :
  - Le code VHDL est donné par le fournisseur du COTS
  - Le modèle sous-jacent : FSM communicantes



Étape (1) Modèlisation

➢ Modèle du COTS porte:

Interface

$I^d = \{req\_open, req\_close, sns\_open, sns\_close, sns\_obst, cmd\_open, cmd\_close, ack\_open, ack\_close\}$

Requêtes | Capteurs | commandes | aquittements

Pré-condition

Les capteurs physiques de la porte s'activent infiniment souvent

$A^d = \{a_1^d, a_2^d\}$

a_1^d: $always\ eventually\ (sns\_open)$

a_2^d: $always\ eventually\ (sns\_close)$

Post-condition

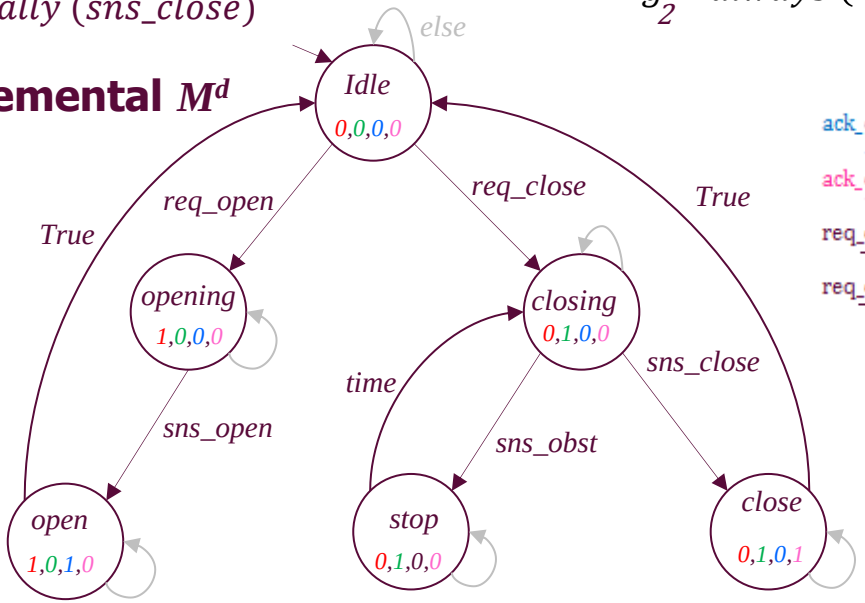
Toute requête d'ouverture/fermeture est traitée

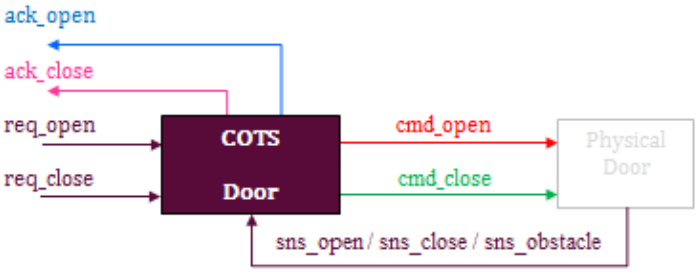
$G^d = \{g_1^d, g_2^d\}$

g_1^d: $always\ (req\_open) \rightarrow eventually\ (ack\_open)$

g_2^d: $always\ (req\_close) \rightarrow eventually\ (ack\_close)$

Modèle comportemental M^d





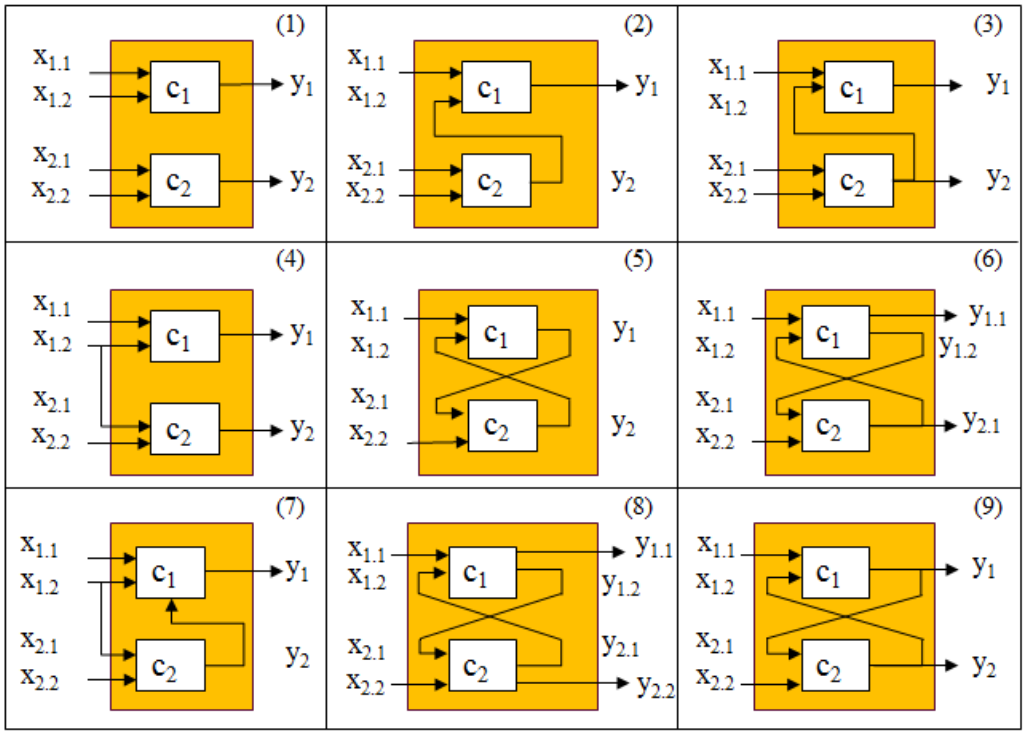
Étape (1) Modèlisation

- **Assemblage de COTS**: produit compositionnel

$$C_1 ||^C C_2 ||^C \dots ||^C C_n = (I^{asm}, A^{asm}, G^{asm}, M^{asm})$$

- Interface de l'assemblage
  $I^{asm} = I^{c1} \cup I^{c2} \cup \dots \cup I^{cn} \setminus I^{intern}$
- Pré-conditions de l'assemblage
  $A^{asm} \subseteq A^{c1} \cup A^{c2} \cup \dots A^{cn}$
- Post-conditions de l'assemblage
  $G^{asm} \subseteq G^{c1} \cup G^{c2} \cup \dots G^{cn}$
- Modèle du comportement de l'assemblage
  $M^{asm} = M^{c1} || M^{c2} || \dots M^{cn}$
  Produit synchrone des FSM

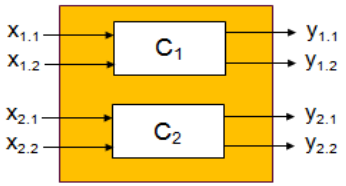
Patternes d'assemblage

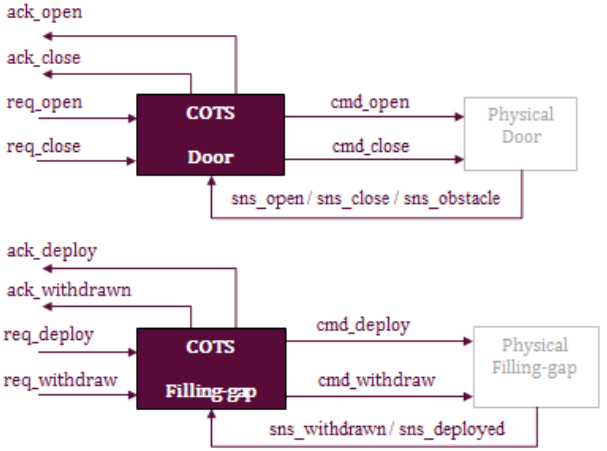


Étape (1) Modèlisation

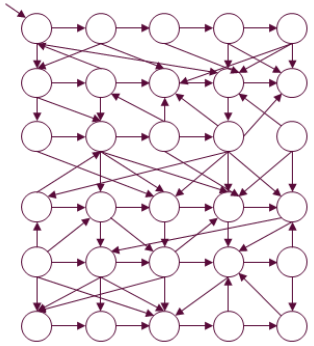
Example: Système d'accès voyageurs

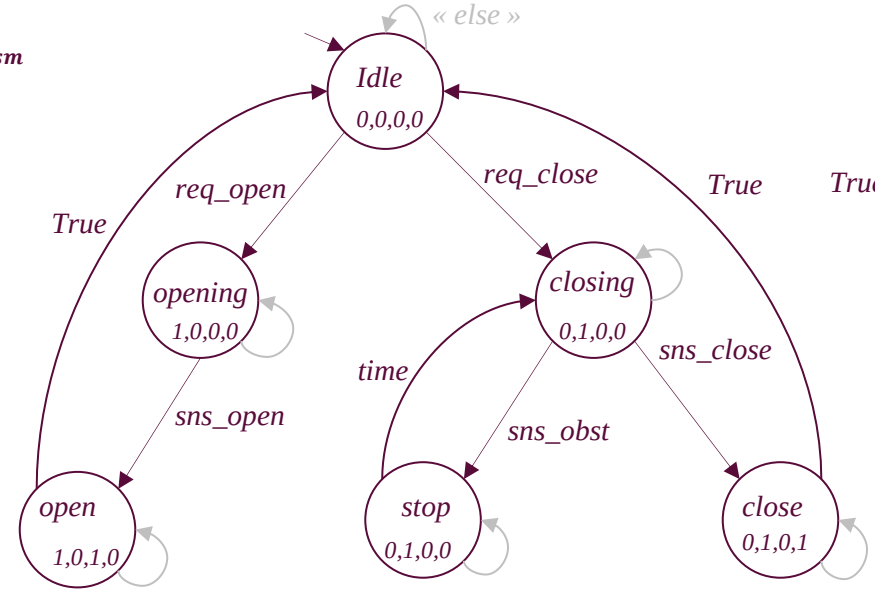
- $I^{asm} = I^{d} \cup I^{fg} \;:\; I^{intern} = \emptyset$
- $A^{asm} = A^{d} \cup A^{fg}$
- $G^{asm} = G^{d} \cup G^{fg}$
- $M^{asm} = M^{d} \,||\, M^{fg}$



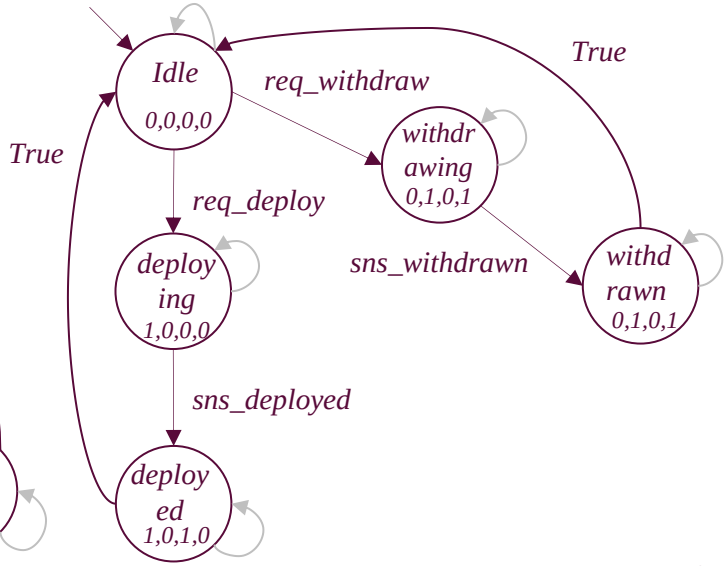


Aperçu du modèle M^{asm}





Porte M^{d}



Emmarchement mobile M^{fg}

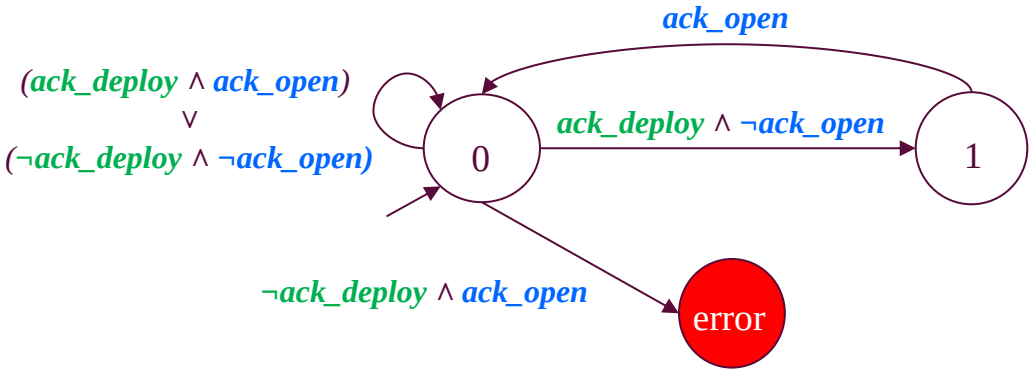
Étape (1) Modèlisation

➢ Propriétés

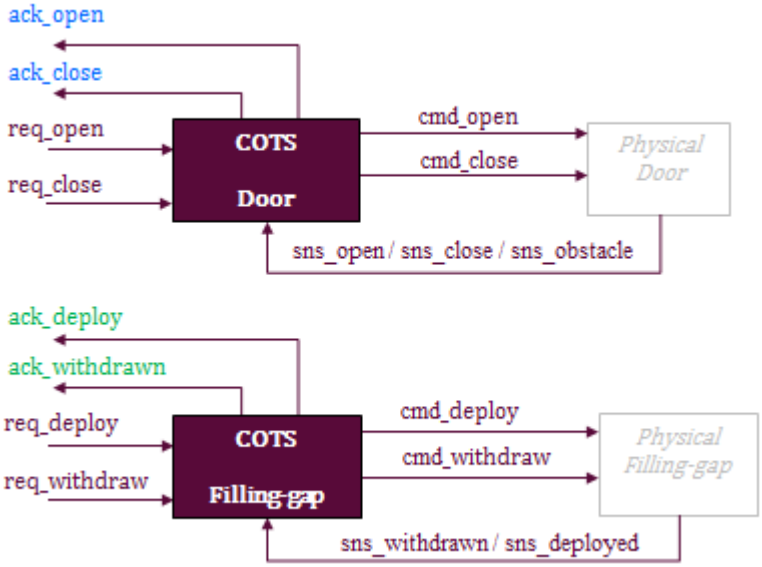
Exigences de sûreté suplémentaires, prédefinies par le concepteur pour assurer un comportement sûr de l'assemblage des COTS

- **Propriété globales** dénoté $P^{asm} = \{\Phi_p^{asm}, M_p^{asm}\}$
  - Concerne le comportement de l'assemblage
  - Modèle : logique temporelle ou moniteur

- p^{asm} : l'emmarchement mobile doit être déployé avant l'ouverture de la porte

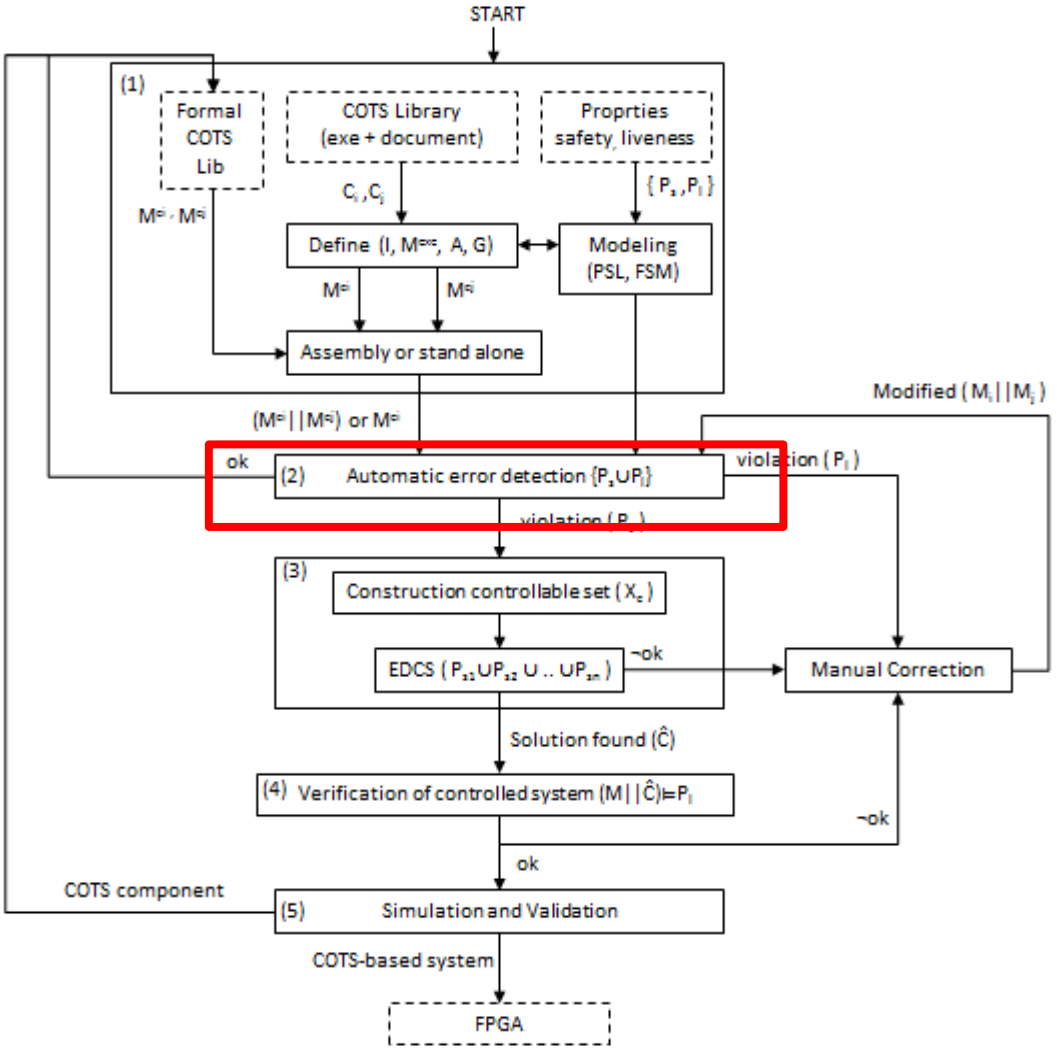


Modèle du moniteur exprimant M_p^{asm}



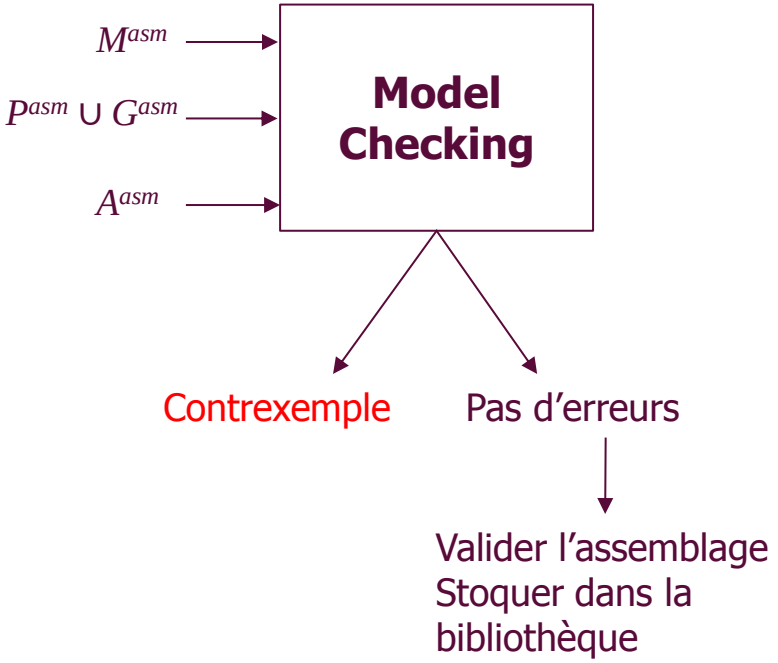
Étape (2) détection automatique des erreurs

- **Détéction automatique des erreurs : Model checking**
  - **Erreur locale** est la violation d'une post-condition causée par l'assemblage
  - **Erreur globale** est la violation d'une propriété globale



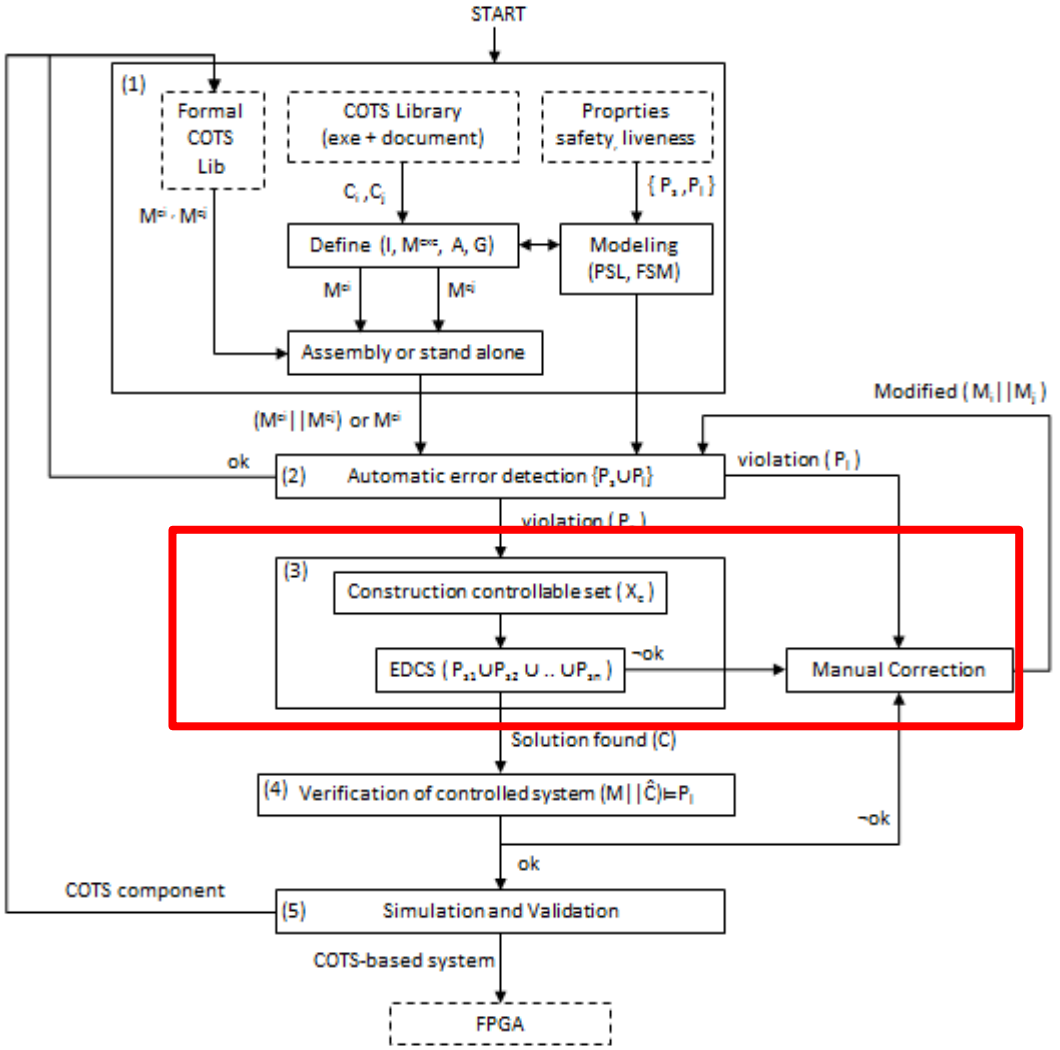
Étape (2) détection automatique des erreurs

- **Résultat négatif du model-checking:**
  - Erreur est détectée → Diagnostique
  - Un **contrexemple** est une séquence d'états qui mène à la violation d'une propriété
    De l'état initial jusqu'à l'état d'erreur



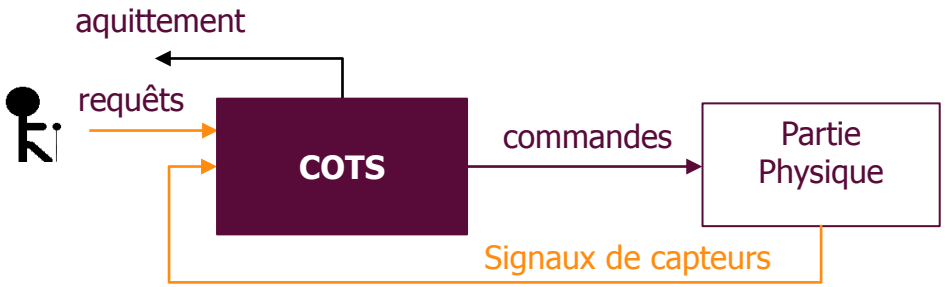
Étape (3) correction automatique des erreurs

- Correction selon le type de l'erreur
  - Erreur de vivacité → correction manuelle
  - Erreur de sûreté → correction automatique
    utiliser : Environment-aware SCD



Étape (3) correction automatique des erreurs

- **Environment-aware SCD basé sur SCD [Marchand et al. 2001]**



- **Nécessité de considérer l'environnement**

  Le bon fonctionnement du COTS dépend du bon fonctionnement de son environnement

- **Algorithme de la SCD**

  Calculer l'invariant sous contrôle qui satisfait une propriété (P)

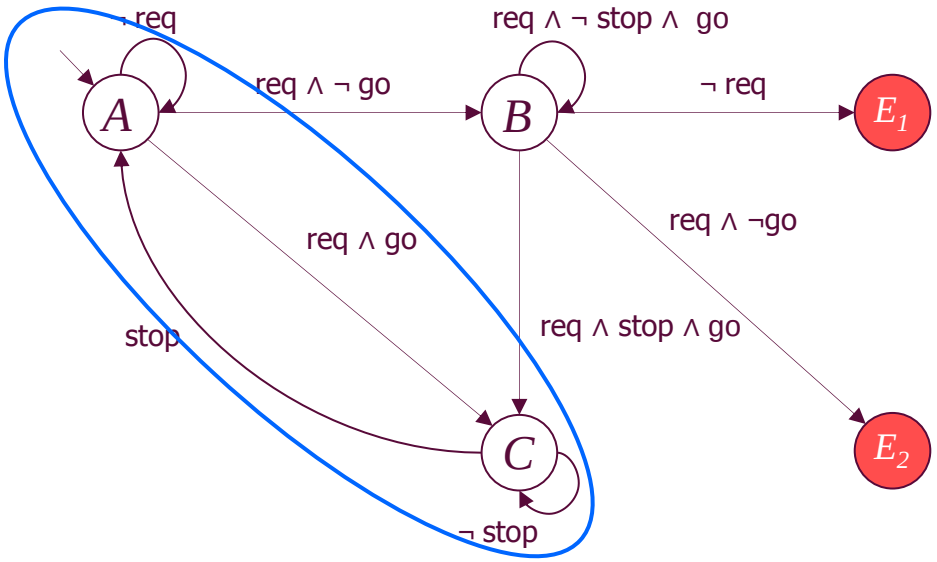
- **Algorithme de la ESCD**

  Calculer l'invariant sous contrôle qui satisfait une propriété (P) **pour un ensemble spécifique d'hypothèses d'environnement**

Étape (3) correction automatique des erreurs

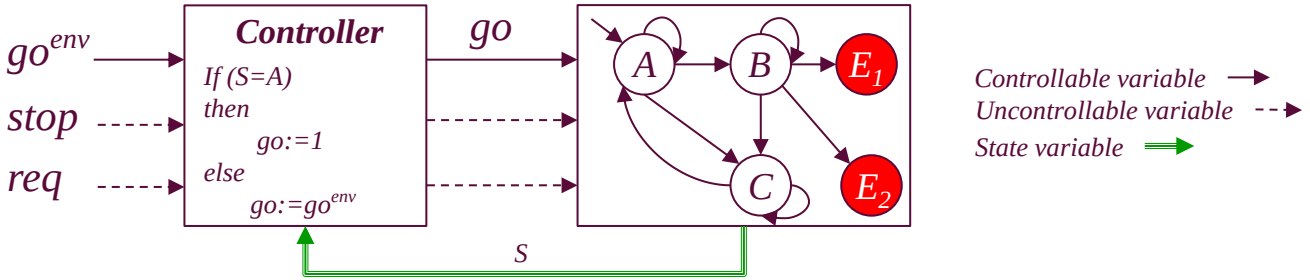
Example

- Système à 5 états
- Etats d'erreurs : E_1, E_2
- Variable Controllable : *go*
- Variables Uncontrollable : *req, stop*



- Invariant sous Contrôle = {A,C}

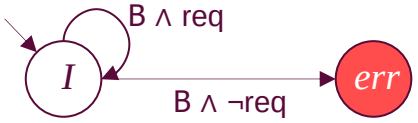
- **Contrôleur généré par SCD :**
  - ***If*** *(S=A)* ***then***
    go := 1
    Else
    go := go^{env}

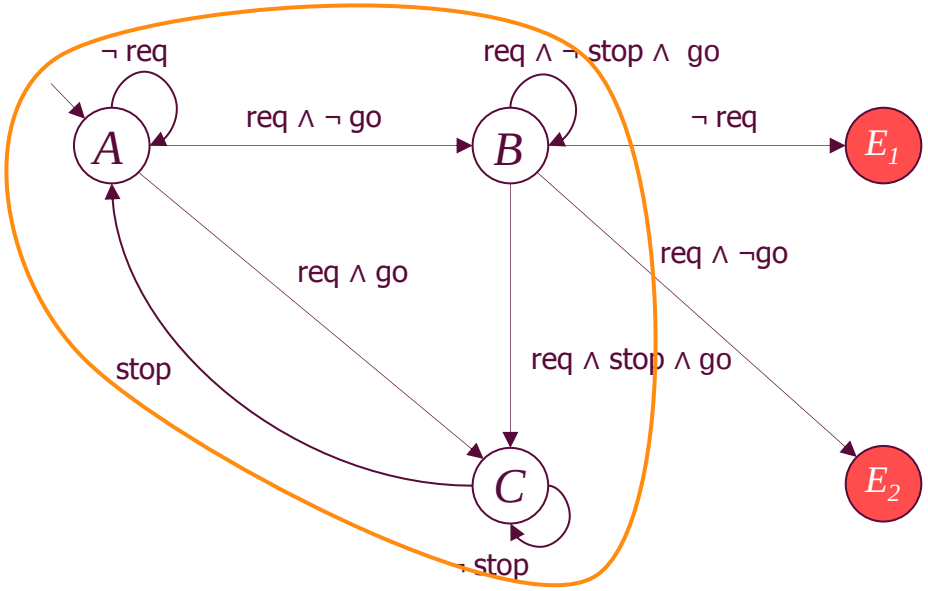


Étape (3) correction automatique des erreurs

➢ Example

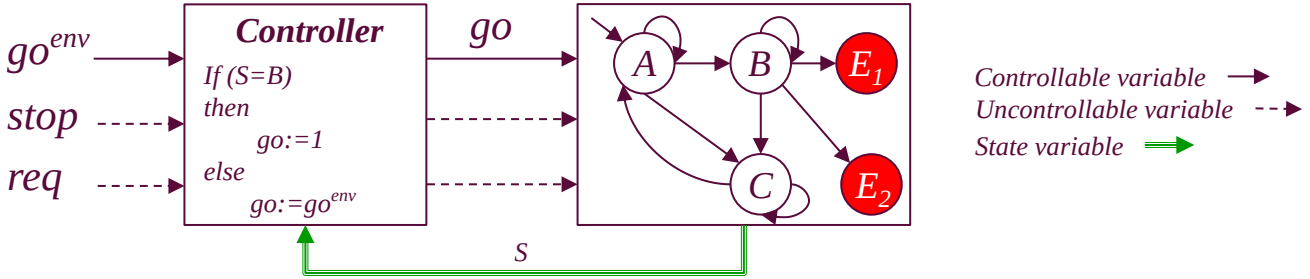
- Environment precondition:
  $always\ (B \rightarrow req)$





- Invariant sous Contrôle = {A,B,C}

- **Contrôleur généré par ESCD :**
  - ***If*** *(S=B)* ***then***
    go := 1
    Else
    go := go^{env}



Étape (3) correction automatique des erreurs

- **EDCS : COTS contrôlabilité des entrées**

  - Utiliser le contrexemple

  - Le set initial X_c^{init} des variables contrôlable est le set complet fourni par le contrexemple

    enlever les variables suivantes :

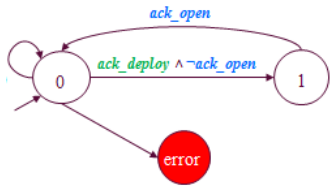
    1. Capteurs de la parties physique
    2. Données (message à transférer)
    3. Alerte et information d'alarme

  - Toutes les variables qui restent sont considérées contrôlables

Étape (3) correction automatique des erreurs

➢ Application à l'Accès voyageurs

- Propriété globale P^{asm} : l'emmarchement mobile doit être déployé avant l'ouverture de la porte

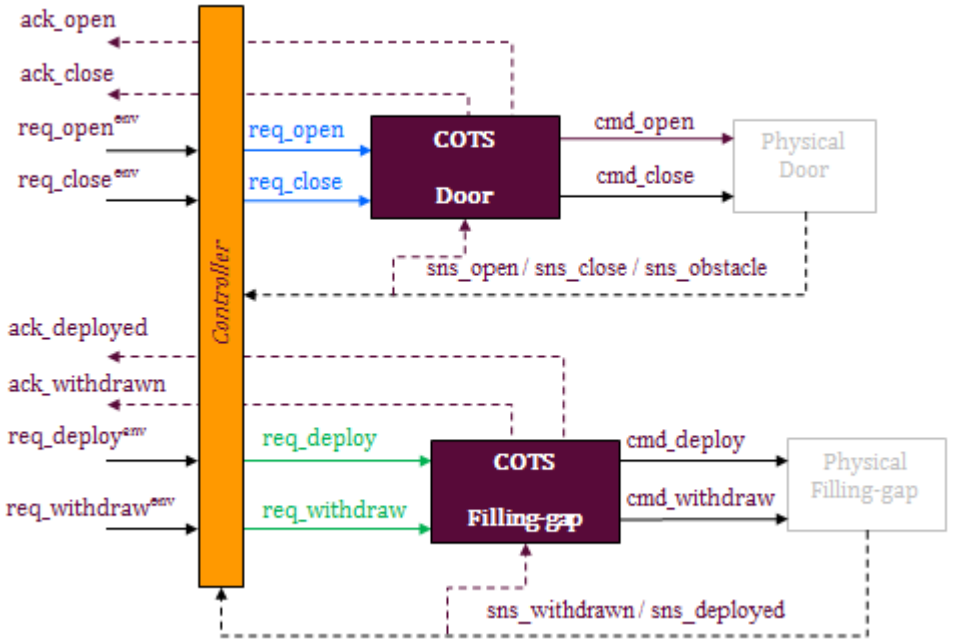


- Contrexemple $X_c^{init} = \{capteurs, requêtes, commandes\}$
  1. Eliminer les capteurs
  2. Eliminer les commandes

  Finalement $X_c = \{requêtes\}$: $req\_open, req\_close, req\_deploy, req\_withdraw$

- Générer automatiquement le contrôleur

- Assembler le contrôleur au modèle original de l'assemblage des COTS



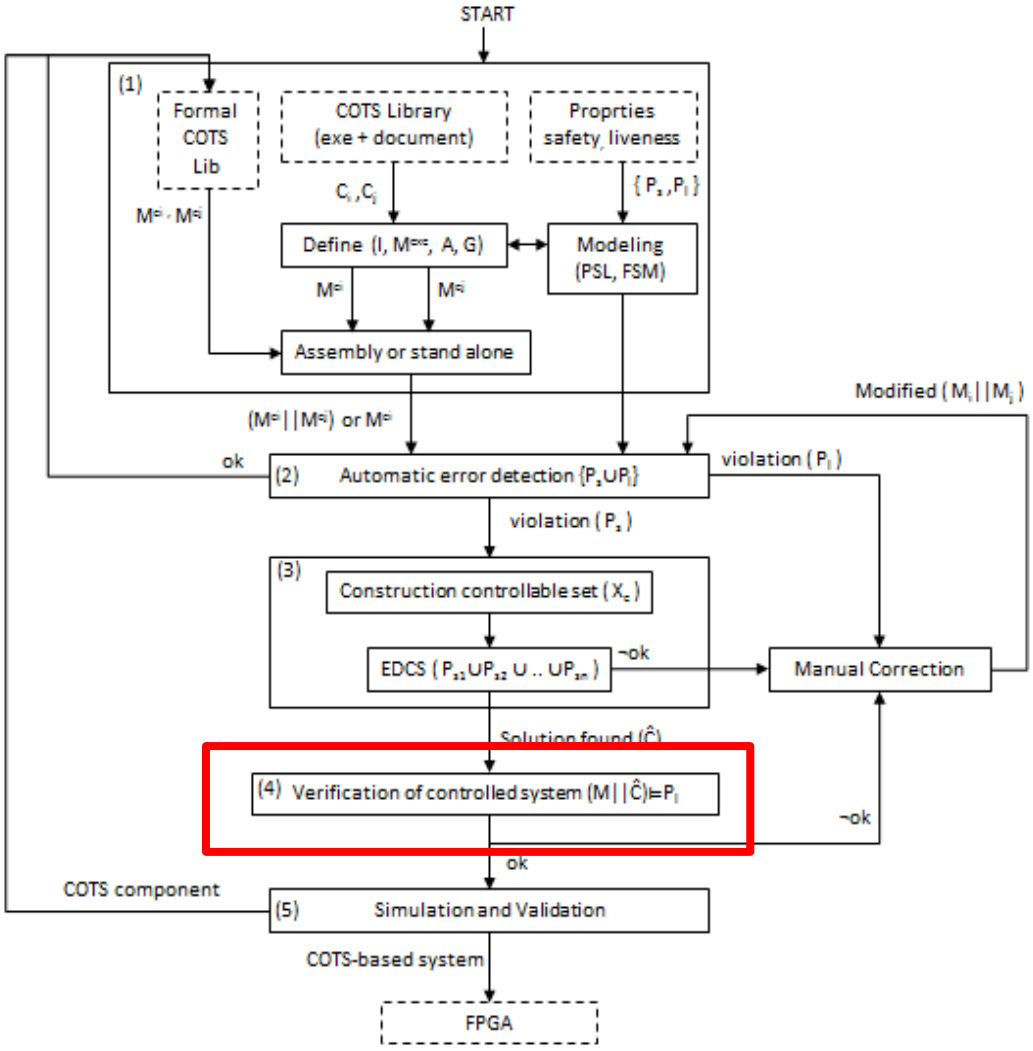
Étape (4) Vérification du système contrôlé

- Que fait le contrôleur?
  - Eliminer des comportements?
  - Inventer des événements?

- Vérification formelle du système contrôlé

  [Hajjar et al. 12, 13]

  - Vérifier le caractère **restrictif** du contrôleur
  - Vérifier la **passivité** du contrôleur



Étape (4) Vérification du système contrôlé

➢ Vérification formelle du système contrôlé

1. Le caractère **restrictif du contrôleur**

- **Contrôleur restreint:** enlève pas seulement les comportements menant vers les états d'erreur mais aussi il entraîne la violation de propriétés de vivacité

- Comment savoir si le contrôleur est restreint ou pas?
  1. Vérifier la satisfaction des post-conditions G^{asm}
  2. Vérifier formellement l'ensemble des propriétés de vivacité

  FINITE_TRANSACTION : always(request^{env} → eventually ackowledge)

- **Exemple : accès voyageurs**
  toute requête est acquittée

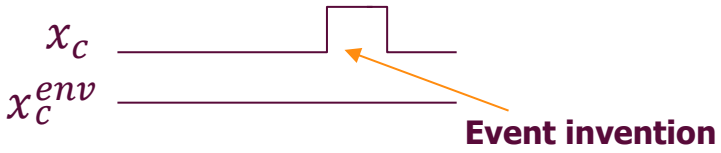
  $$finite\_open\_req : always(req\_open^{env} \rightarrow eventually\ ack\_open)$$

Étape (4) Vérification du système contrôlé

➢ Vérification formelle du système contrôlé

2. Passivité du contrôleur

- **Contrôleur passif** : se limite à empêcher l'occurrence d'événements
- **Invention d'un événement** : activation d'un signal par le contrôleur alors que celui-ci n'a pas été activé par l'environnement
- Comment se produit l'invention d'événement?
  Le contrôleur affecte la valeur (1) au contrôlable x_c alors que l'environnement souhaite lui affecter la valeur (0)



- NO_EVENT_INVENTION : never(request^{controlled} ∧ ¬request^{env})

- **Exemple accès voyageur**
  le contrôleur ne doit jamais commander la porte à s'ouvrir spontanément, sans être demandé à l'ouverture par le conducteur

$$passive_1 : never(open\_req \wedge \neg open\_req^{env})$$

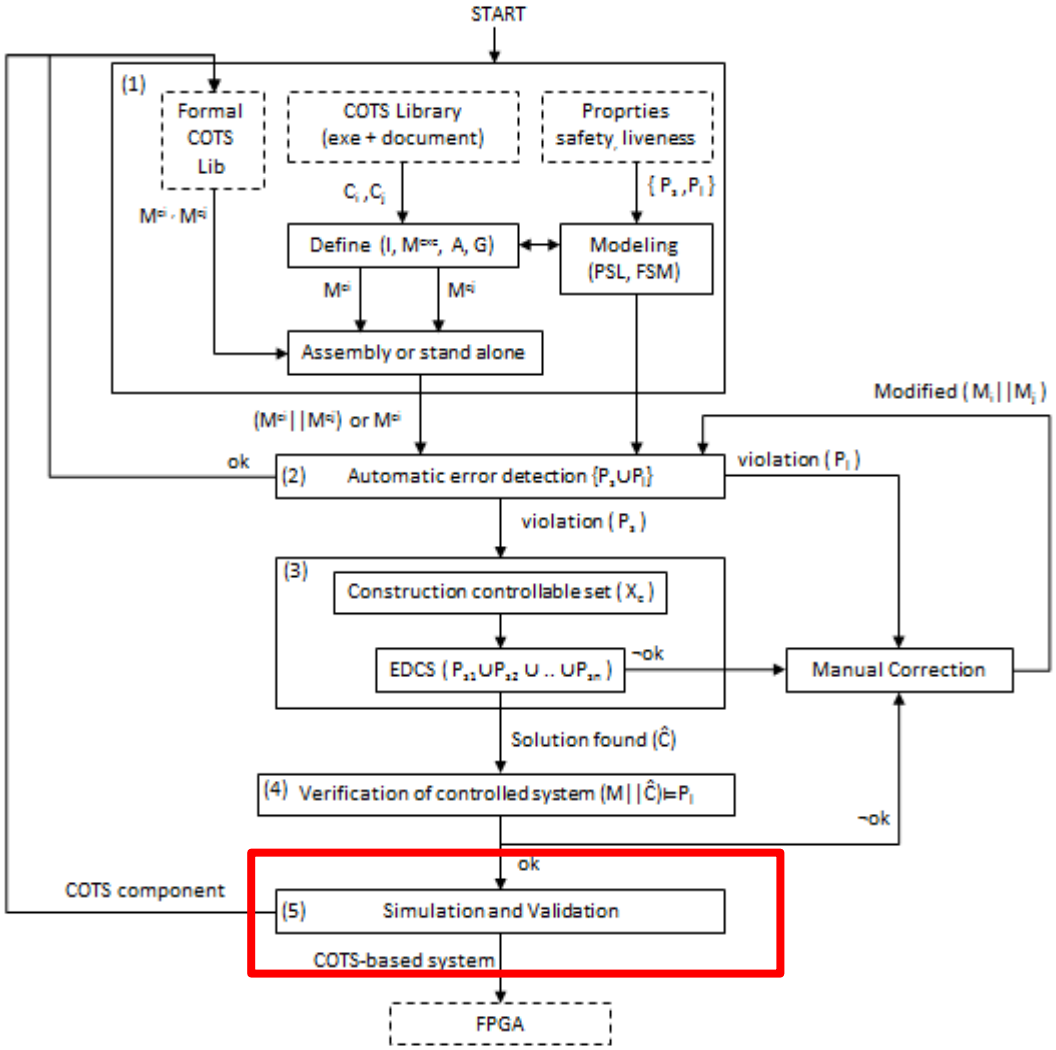
Étape (5) Simulation du système contrôlé

- **Simuler le système contrôlé**

- **But**
  - S'assurer que le système fonctionne correctement
  - Aider le concepteur à prendre la décision d'implémenter le système sur une puce FPGA et/ou le stoquer dans la bibliothèque des COTS

- **Réalisation**

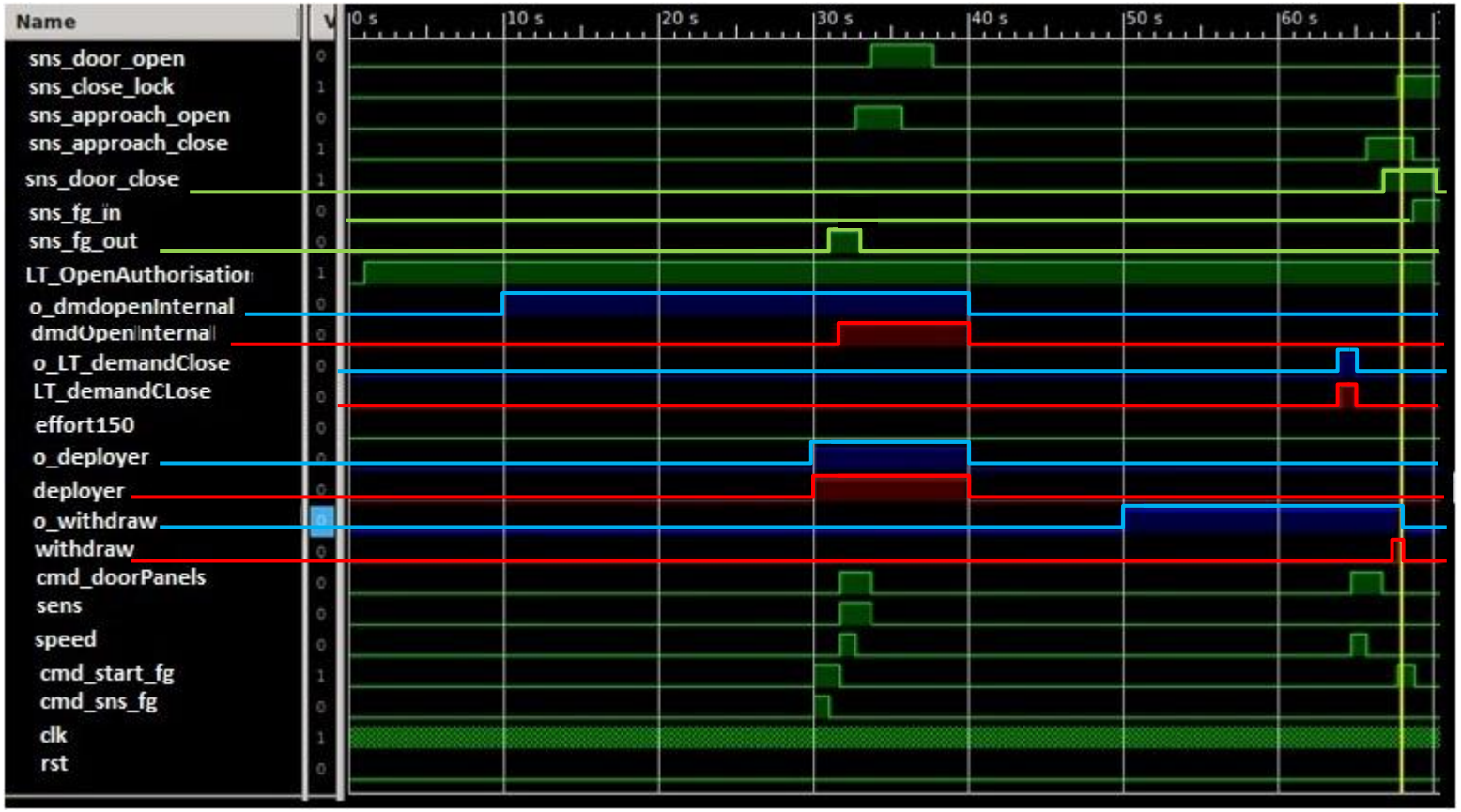
  Outils de simulation VHDL (Xilinx ISE)



Étape (5) Simulation du système contrôlé

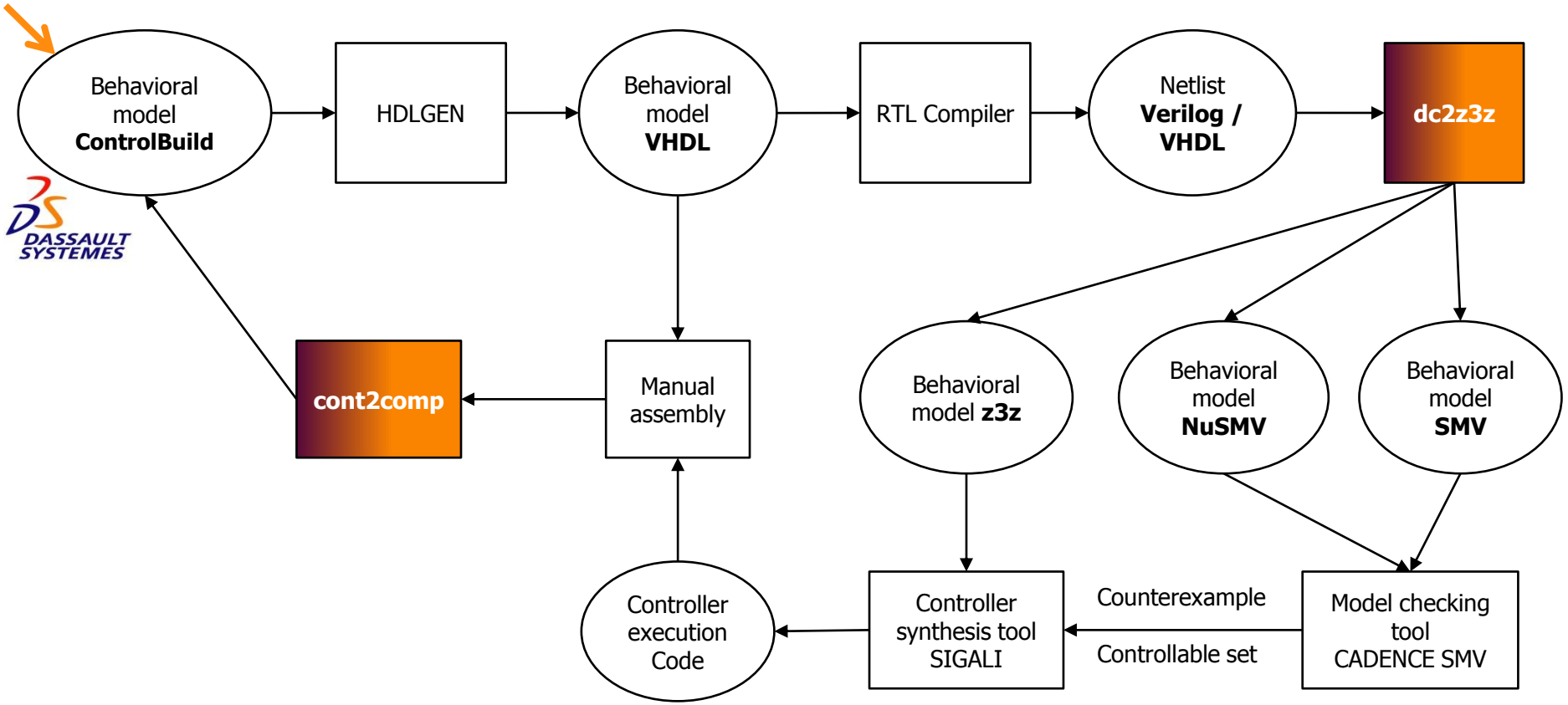
- Simuler le système contrôlé pour un scénario prédéfini

controllables environment correspondence of controllable uncontrollables



- à la seconde 10 le conducteur demande l'ouverture de la porte (l'emmarchement mobile n'est pas encore déployé), le contrôleur empêche l'ouverture jusqu'à la seconde 30 où l'emmarchement mobile est bien déployé.

Chaîne d'outils de l'implémentation



Conclusion

- **Conceptuellement : Aider le concepteur**
  - Proposer une méthode de conception sûre
    - Une démarche en 5 étapes : de la modélisation jusqu'à la simulation
    - **Synergie entre la vérification formelle et la synthèse de contrôleurs discrets**
  - Corriger automatiquement des erreurs de conception
    Employer DCS

- **Techniquement :**
  - Proposition de la ESCD
  - Définition de la contrôlabilité des entrées en utilisant le Contrexemple
  - Evaluation du comportement du contrôleur en vérifiant
    sa passivité
    son caractères restrictifs

- **Application**
  - Utilisation de l'ESCD dans un cadre industriel
    validation de la méthode sur le système d'accès voyageurs

Perspectives

Perspectives

- Renforcer des propriétés de vivacité
  - EDCS pour la vivacité bornée
  - Utiliser la technique de synthèse optimale pour approximer une garantie de vivacité
- Développer un mécanisme automatique pour le choix des variables contrôlables
- Vérification automatique de la correspondance des pré/post-conditions lors de l'assemblage de COTS

Conception sûre de systèmes embarqués à base de COTS

Merci