MODÉLISATION ET VALIDATION FORMELLE DE SYSTÈMES GLOBALEMENT ASYNCHRONES ET LOCALEMENT SYNCHRONES

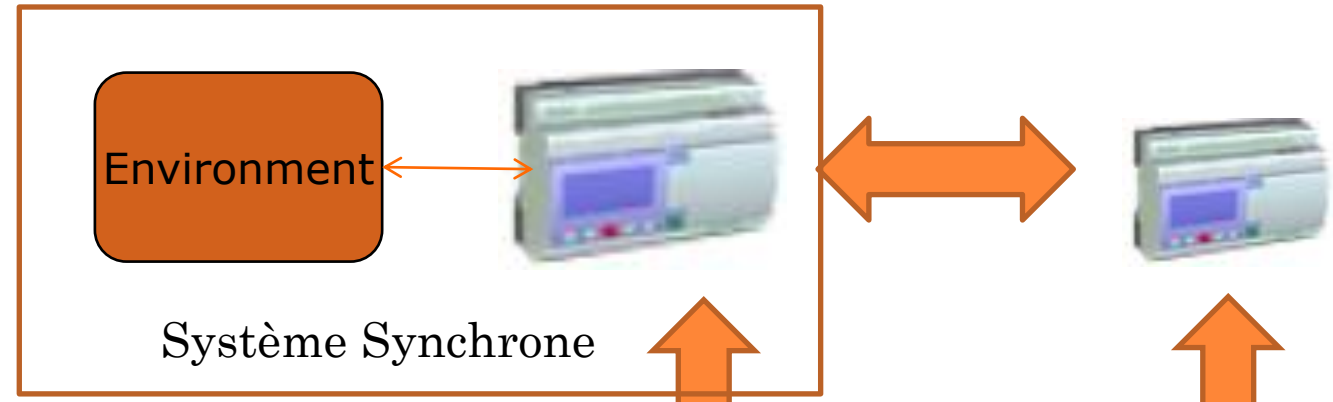
Fatma Jebali, Mouna Tka, Christophe Deleuze, Frédéric Lang, Radu Mateescu, Ioannis Parissis

Plan

1. Contexte général
2. GALS: validation synchrone
3. GALS: validation asynchrone
4. Conclusion et travaux futurs

Contexte General

GALS: Globally Asynchronous Locally Synchronous



Système Synchrone

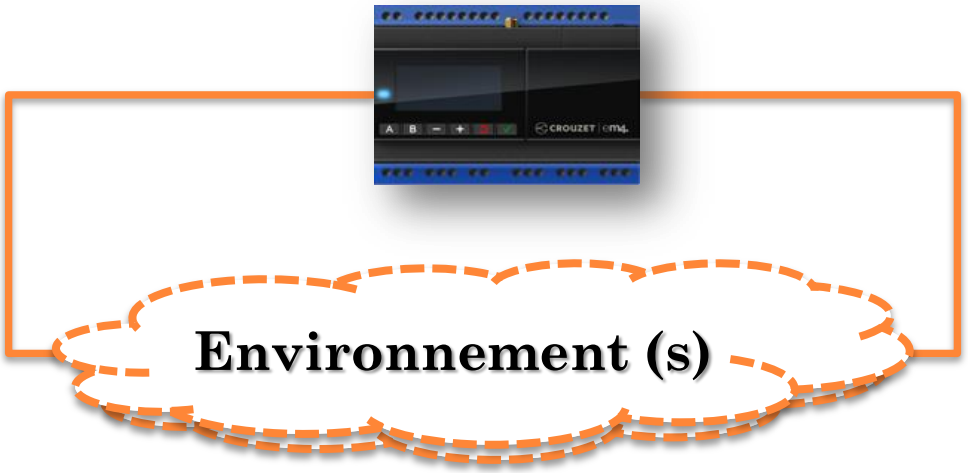
✓ Validation Synchrone

Interaction asynchrone

✓ Validation Asynchrone

CLP
Contrôleur logique programmable

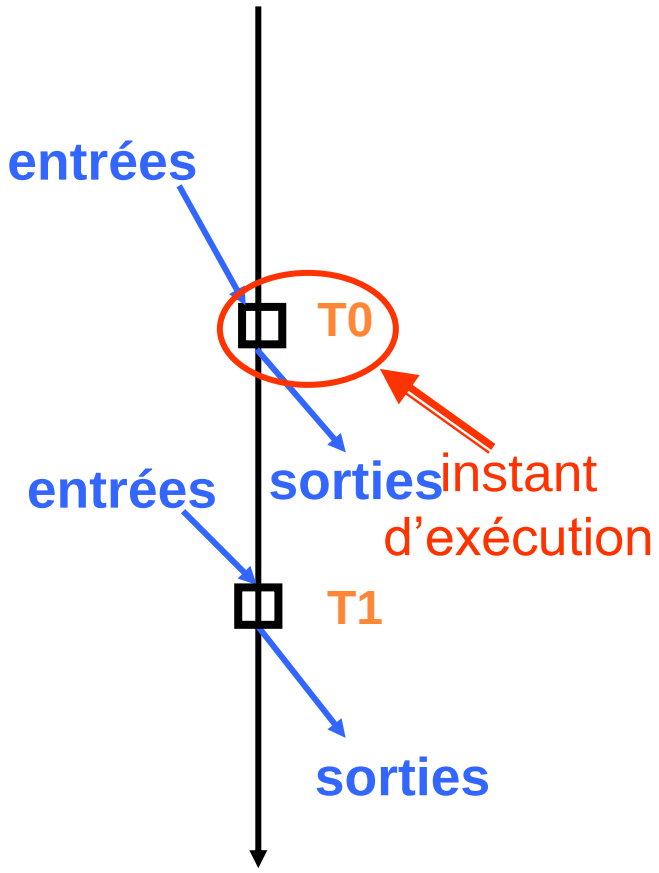
Validation Synchrone



Système Synchrone

Fonctionnement cyclique :

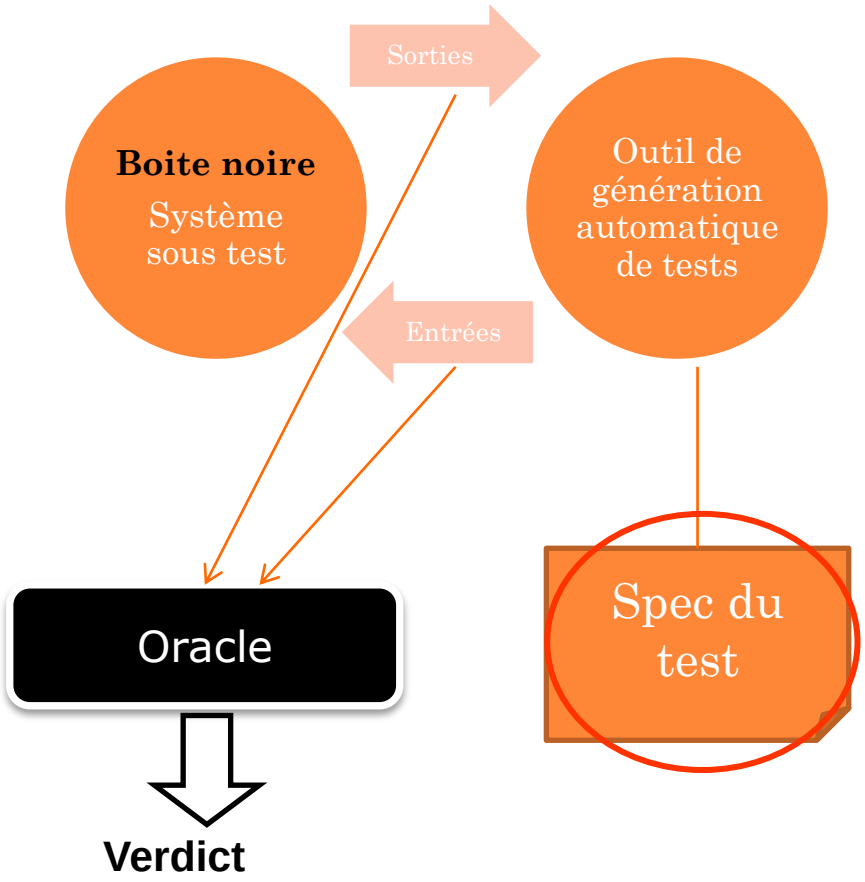
1. Lecture des entrées
2. Exécution/Calcul des sorties
3. Ecriture des sortie



Automatisation du test

un générateur de séquences de test :

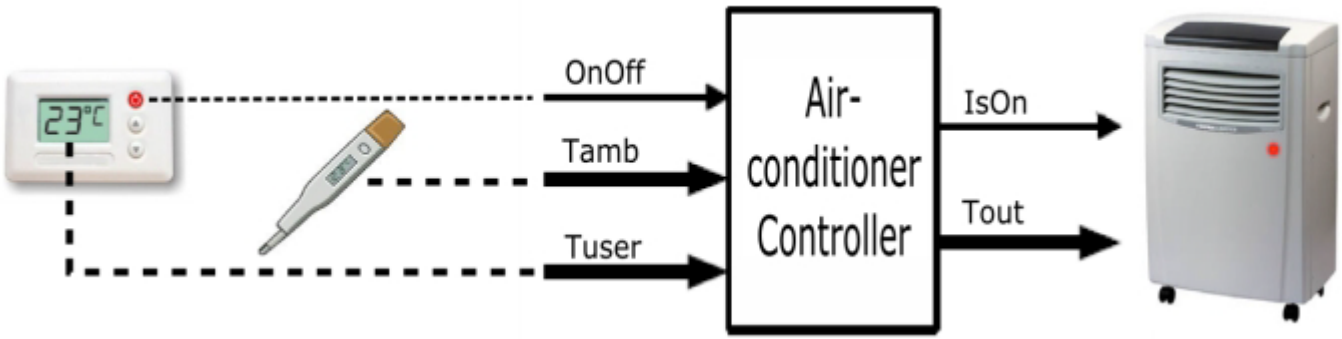
1. lit les sorties
2. choisit des valeurs pour les entées…
3. les injecte



Synchronous Programs Testing Language

- Variables:
  - sorties du système
  - entrées du système
  - internes à l'environnement
- Profil:
  - ensemble de contraintes reliant les variables déclarées et limitant l'espace des états possibles.
  - Chaque profil est spécifique à un environnement bien déterminé d'utilisation du système.
- Scénario:
  - Une histoire qui décrit une évolution de l'environnement dans le temps

EXEMPLE:



•OnOff(Booléen) : vraie lorsque l'utilisateur appuie sur le bouton Marche-Arrêt du climatiseur;

•Tamb(entier) : valeur en degrés Celsius de la température ambiante;

•Tuser(entier) : valeur en degrés Celsius de la température sélectionnée par l'utilisateur.

•IsOn(Booléen) : Etat du climatiseur;

•Tout (entier) : définit la température de l'air que le climatiseur fait sortir.

Catégories

```
categ Season
    { group Summer
        { 20 < Tamb ; Tamb < 44 ;
          Tuser = ComputeTemperature(Tamb) }
    group Winter
        { 5 <= Tamb ; Tamb <= 21 }
    }
categ TypePlace
    {
    group Office { OnOff=prob(true, 0.001) }
    group Home { OnOff=prob(true, 0.1) }
    }
```

Profil: Catégories

- Un même système peut être utilisé dans des environnements déférents.
- On découpe l'environnement en plusieurs aspects, les catégories.
- Par exemple
  - catégorie Saison
    - été
    - hiver
    - …
  - catégorie Lieu
    - bureau
    - appartement
    - …

- Profil = combinaison d'instances de catégories

1. (été, bureau)
2. (été, appartement)
3. etc

- guide la création du profil
- possibilité de réaliser des catégories pré-définies pour aider le testeur.

Scénario

- Histoire qui décrit l'évolution de l'environnement dans le temps.

- séquence de deux types d'éléments:

  - des étapes ponctuelles (points de passage):
    - à un instant l'environnement est dans un état qui vérifie certaines contraintes

  - des étapes étendues (chemins):
    - contraintes qui s'appliquent jusqu'à ce qu'une condition soit réalisée

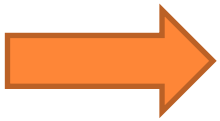
Scénario

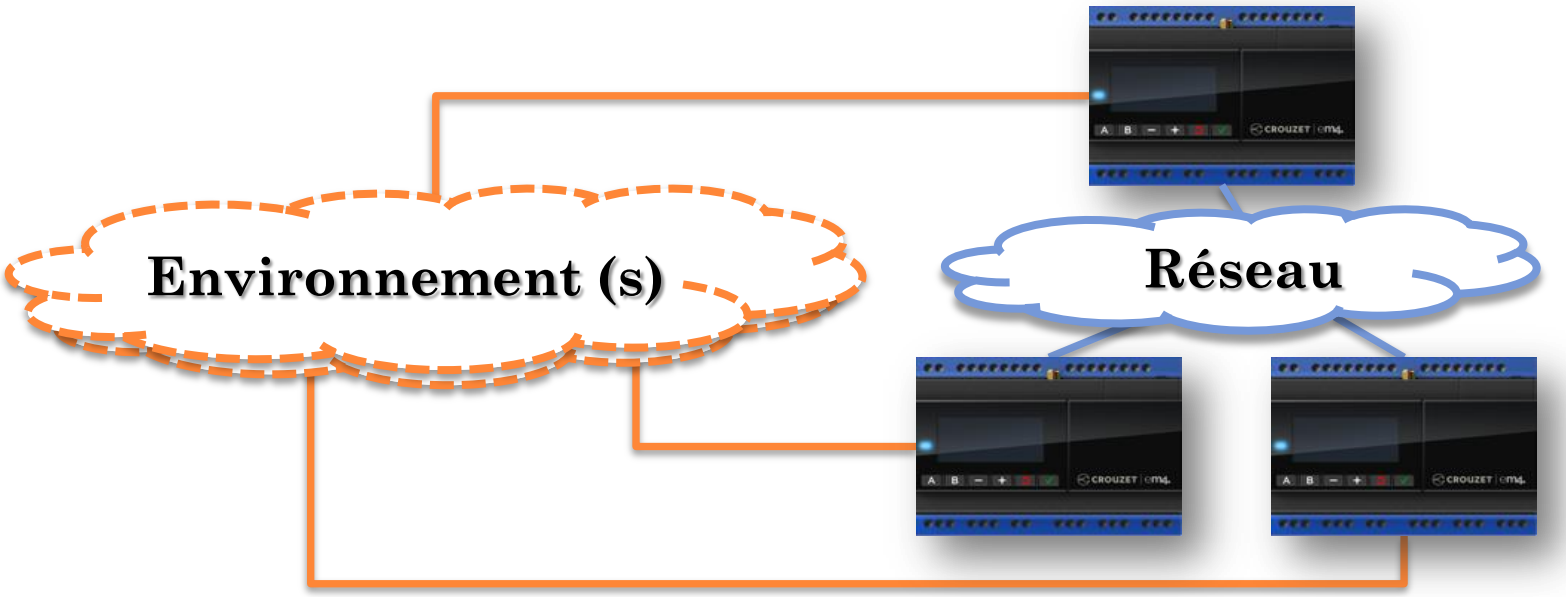
```
scenario UserWarmThenCold
        time t
begin
        { Tamb = 30; Tuser = 8 }
        | [ Tamb = pre(Tamb) - 1 (Tamb = Tuser) ]
        | { Tuser = 22 ; t.start}
        | [ Tamb = pre(Tamb) + (if t % 20s = 0 then 1 else 0)
        (Tamb = Tuser) ]
end
```

1. Étape ponctuelle
2. Étape étendue
   - contraintes
   - condition de sortie
3. Étape ponctuelle
4. Étape étendue

Résolution des contraintes

- une exécution est spécifiée par
  - un profil
  - un scenario
- à chaque instant, un ensemble de contraintes sont à résoudre
  - celles du profil
  - celles de l'étape courante du scenario

 compilation du programme de test vers Prolog



Validation Asynchrone

GALS: Systèmes distribués concurrents

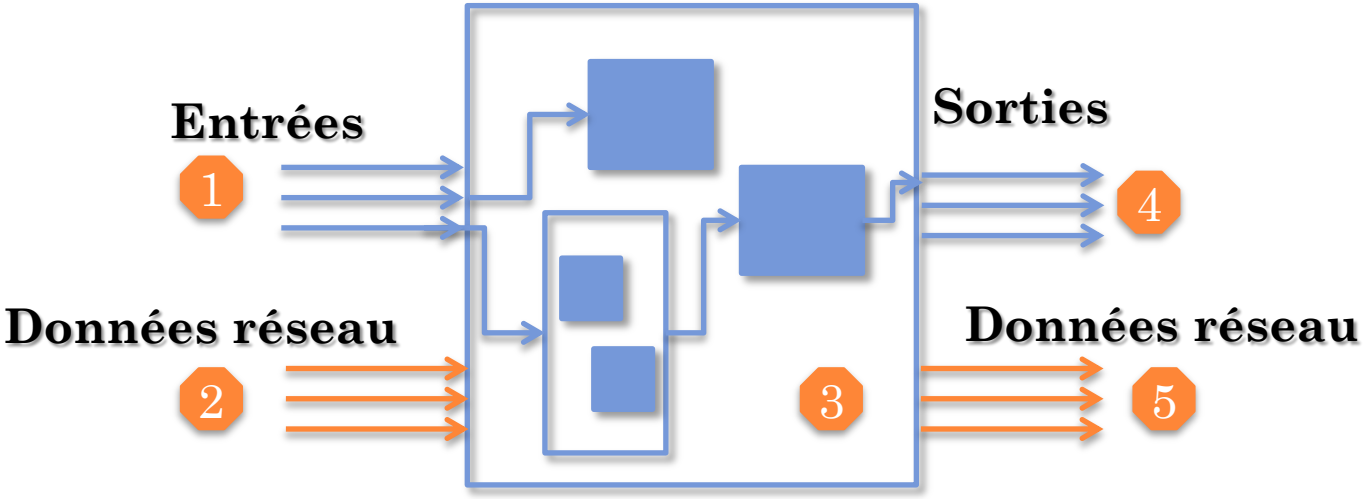
- Modélisation explicite
  - Comportement individuel des systèmes synchrones distribués
  - Protocole et topologie du réseau de communication
  - Contraintes de l'environnement extérieur
- Parallélisme asynchrone
  - Pas d'hypothèse sur la fréquence relative des systèmes synchrones
  - Pas d'hypothèse sur les délais de communication
  - Non-déterminisme

GRL: Gals Representation Language

- Syntaxe textuelle conviviale, sémantique formelle
- Description expressive et concise
  - Modèle synchrone inspiré des langages a flot de données (ex., FBD)
  - Modèle asynchrone inspiré du langage LNT:
    - basé sur les algèbres de process
    - étendu avec données et des structures de contrôle

Blocs: systèmes synchrones

- Programmes déterministes
  - Structures de contrôle classiques (langages algorithmiques)
  - Instanciations hiérarchiques de blocs
- Exécution synchrone atomique



BLOCS: SYSTÈMES SYNCHRONES

```
block  AileronSystem  (in spo : bool; out cpo : nat)
                          {receive lock; up; down :bool; send  apo : nat} is

    perm pos : nat := 0
    if (not(lock) and spo) then
       if up then pos := pos + 1
       elsif down then pos := pos + 1 end if
    end if;
    cpo := pos; apo := pos

end block
```

Environnements et Médiums

- Médiums
  - Modéliser le réseau de communication
  - Comportement passif guidé par l'envoi des données ou la réception des données par un bloc
- Environnements
  - Modéliser des contraintes physiques et logiques sur les blocs
  - Comportement passif guidé par la lecture ou l'ecriture d'une entrée ou d'une sortie d'un bloc
- Non-déterminisme pour modéliser des comportements complexes

Médiums: Réseau de Communication

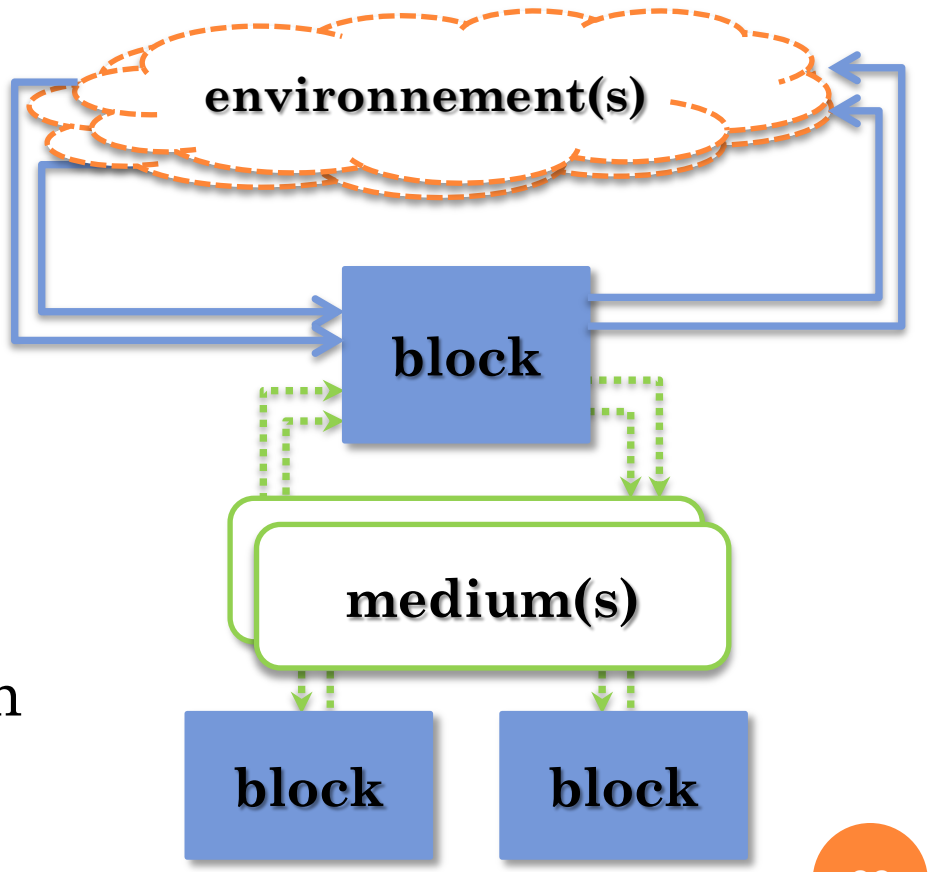
```
medium AccessScheduler {receive apo : nat | send lock; up; down : bool |
                        receive lp; up; dp : bool | send app : nat |
                        receive ls; us; ds : bool | send aps : nat} is

perm lock_bu : bool := true; up_bu ; down_bu : bool := false; apo_bu : nat := 0
    select
      on lp, up, dp  -> lock bu := lp; up bu := up; down bu := dp
      [] on ls, us,  ds  -> lock bu := ls; up bu := us; down bu := ds
      [] on apo -> apo bu := apo
      [] on ?app -> app := apo bu
      [] on ?aps -> aps := apo bu
      [] on ?lock; ?up; ?down  ->  lock := lock bu ; up := up bu ; down := down bu
    end select

end medium
```

Systèmes: Composition des réseaux

- Définir les acteurs:
  - Blocs
  - Environnements
  - médiums
- Décrire les interactions entre les acteurs
- Exécution cyclique des blocs induisant l'exécution des médiums et environnements



Systèmes: Composition des réseaux

```
system FlightControlSystem (in op, os : nat, sp : bool; out alrm : bool) is
    allocate FBWComputer as Primary , FBWComputer  as Secundary,
                AileronSystem as Aileron, FCDConcentrator as Concentrator,
                AileronControl [10] as Control, AccessScheduler as Scheduler
    temp tp : bool, app : int, lp, up, dp : bool, ts : bool, aps : int, ls, us, ds : bool,
          spo, cpo, apo : nat, lock, up, down : bool
    network -- blocs synchrones
        Primary (op; tp) {app; ?lp; ?up; ?dpg},
        Secondary (os; ts) {aps; ?ls; ?us; ?dsg},
        Aileron (spo; ?cpo) {lock; up; down; ?apo}
    constrainedby -- environnements
        Concentrator (sp | ?tp | safeSecundary | ?ts | ?alrm),
        Control (cpo | ?spo)
    connectedby -- mediums
        Scheduler {lp; up; dp | ?app | ls; us; ds | ?aps | apo | ?lock; ?up; ?down}
end system
```

Conclusion et travaux futurs

Validation synchrone

- SPTL pour la génération automatique des séquences de test, basée sur:
  - la description des contraintes dans l'environnement du système
  - les profiles des utilisateurs
- Implémentation du prototype du générateur du test et expérimentation sur des systèmes complexes (Réseaux de CLP)

Validation asynchrone

- GRL comme langage pivot entre les outils de conception industriels des GALS (Réseaux de CLP) et les outils de validation automatique de CADP

- Conception des applications CLP dans l'internet des objets et validation proposée en mode SaaS

Interdépendance des deux types de validation

- Scénarios de tests décrits au niveau synchrone peuvent être rejoués sur le modèle asynchrone
- Des séquences générées automatiquement à partir du modèle asynchrone peuvent alimenter la génération de tests au niveau synchrone