Modélisation & Vérification

Modéliser pour Vérifier

Asma BERRIRI

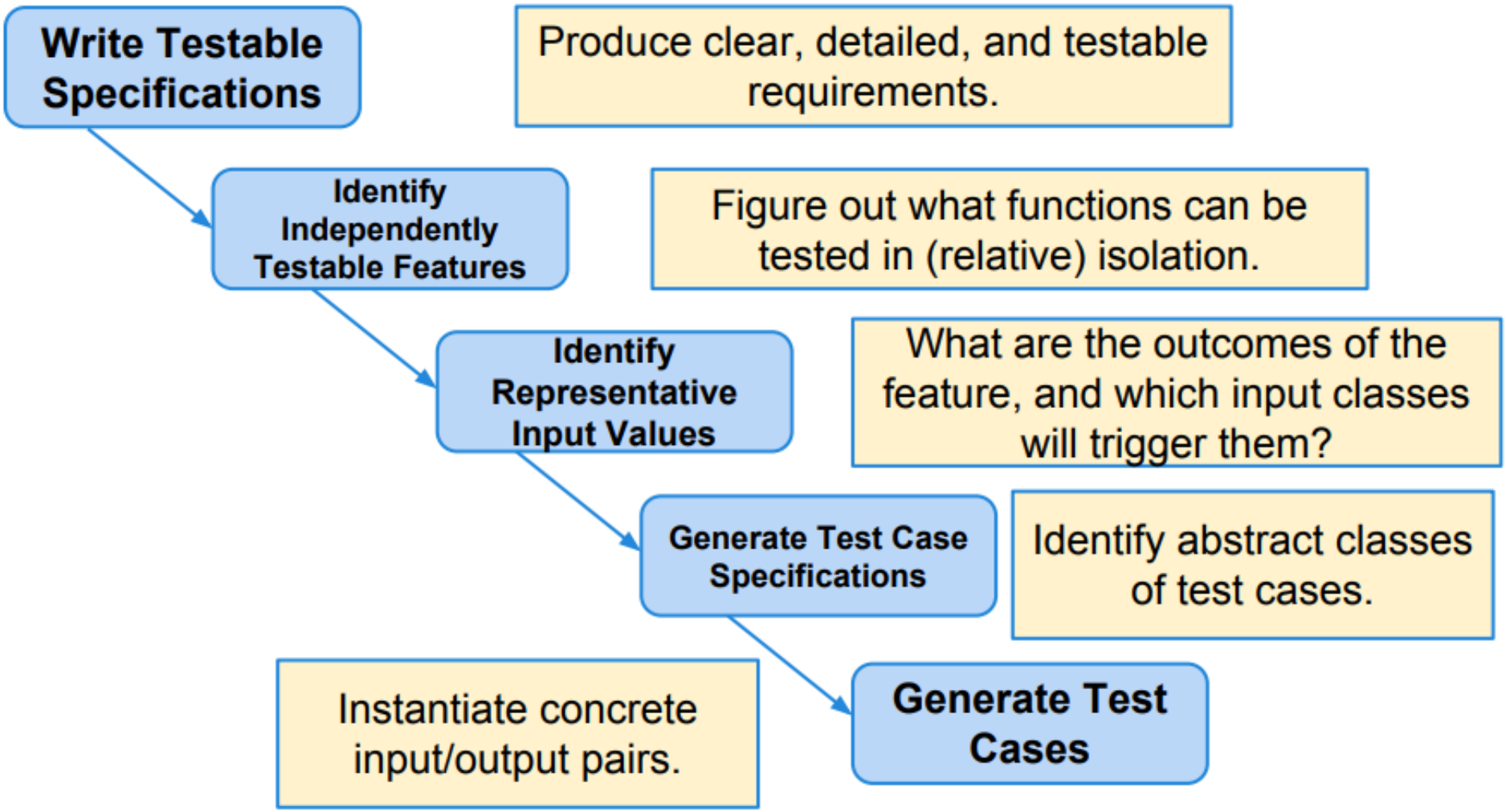
asma.berriri@universite-paris-saclay.fr

Page web du cours :
https://sites.google.com/view/asmaberriri/home

Outline

POLYTECH PARIS-SACLAY

Création de tests basés sur les exigences



POLYTECH PARIS-SACLAY

Création de tests basés sur les exigences

- Ce processus est efficace pour identifier les partitions indépendantes pour chaque entrée

-> Nous laissant avec un grand nombre de spécifications de test

- Les humains doivent encore identifier les contraintes sur les combinaisons de choix d'entrées et identifier un sous-ensemble de spécifications de test importantes
- Une approche alternative - construire un modèle à partir de la spécification et dériver des tests à partir de la structure du modèle

POLYTECH PARIS-SACLAY

Modèles et analyse logicielle

- Avant et pendant la construction des produits, les ingénieurs analysent les modèles pour répondre aux questions de conception
- Les modèles logiciels capturent les différentes manières dont le logiciel se comporte pendant l'exécution

Modélisation du comportement

- Abstraction - simplifier le problème en identifiant et en se concentrant uniquement sur les aspects importants

-> En supprimant les détails inutiles, des analyses extrêmement puissantes peuvent être effectuées

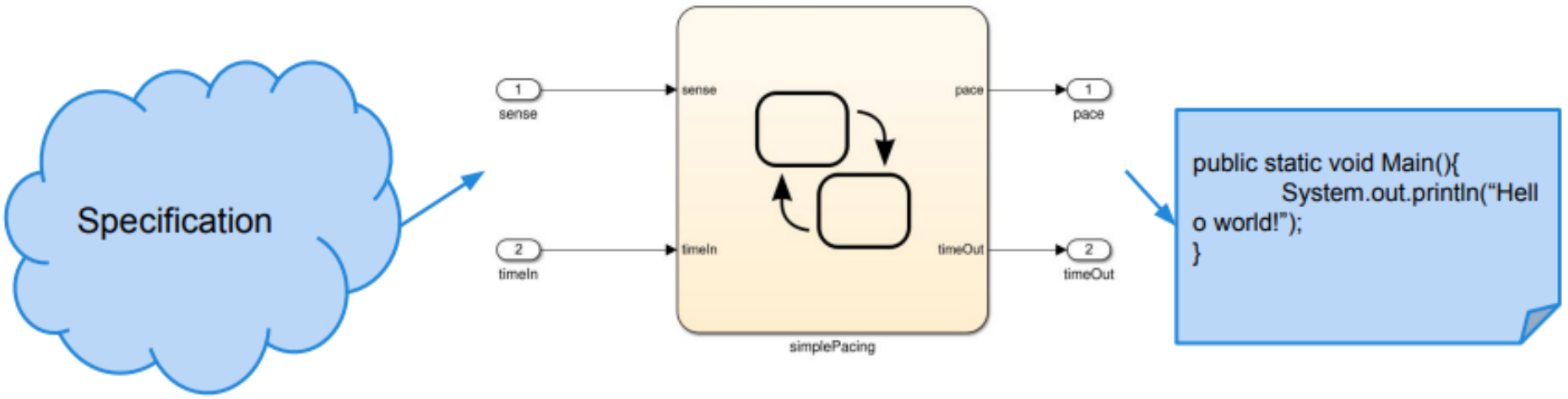
- Peut être extrait des spécifications et des plans de conception

-> Illustrez le comportement prévu du système

-> Prennent souvent la forme de machines à états

- Les événements font réagir le système, modifiant son état interne

Que pouvons-nous faire avec ce modèle ?



If the model satisfies the specification...

And If the model is well-formed, consistent, and complete.

And If the model accurately represents the program.

Ensuite, nous pouvons dériver des cas de test du modèle qui peuvent être appliqués au programme. Si le modèle et le programme ne concordent pas, il y a une faute.

Que pouvons-nous faire avec ce modèle ?

Développement piloté par les modèles (Model-Driven)

- Modèles souvent créés lors de l'analyse des exigences
- Permet d'affiner les exigences
- Prouver que les propriétés s'appliquent au modèle
- Vérification de l'état fini (classe suivante) - utilisée pour analyser les exigences, planifier le développement, créer des cas de test
- Peut générer du code à partir de modèles
- Peut créer des tests en utilisant le modèle
- Les modèles décrivent la structure de l'espace d'entrée
  - Ils identifient ce qui se passera lorsque les types d'entrée sont appliqués au système
- Cette structure peut être exploitée
  - Identifiez les partitions d'entrée
  - Identifier les contraintes sur les entrées
  - Identifier les combinaisons d'entrées significatives
- Peut dériver et satisfaire des métriques de couverture pour certains types de modèles

Modéliser pour vérifier

Modélisation: Abstraction du système concret et simulation de ce modèle

- Spécifications algébriques
- Graphiques de contrôle/flux de données
- Modélisation des propriétés attendues du système (Spécification basée sur la logique)
- Spécification basée sue machine à états finis
- Spécification basée sur la grammaire

Systèmes de transitions

- Exécution du programme sous forme de séquence d'états transformés par des actions
- "Comportement" est: état -> action -> transitions d'état
- L'ensemble de tous les comportements réels possibles est souvent infini
  - Appelé "l'espace d'état" du programme
  - Les modèles simplifient une fonctionnalité ou un espace d'états de classes en un ensemble fini d'états

POLYTECH PARIS-SACLAY

Modéliser pour vérifier

Qu'est-ce qu'une machine à états finis (FSM)?

- Est une représentation abstraite du comportement présenté par certains systèmes
- Dérivé des exigences de l'application. Par exemple, un protocole réseau pourrait être modélisé par un FSM
- Tous les aspects des exigences d'une application ne sont pas spécifiés par un FSM. Par exemple, les exigences en temps réel et les exigences de performance ne peuvent pas être spécifiées par un FSM

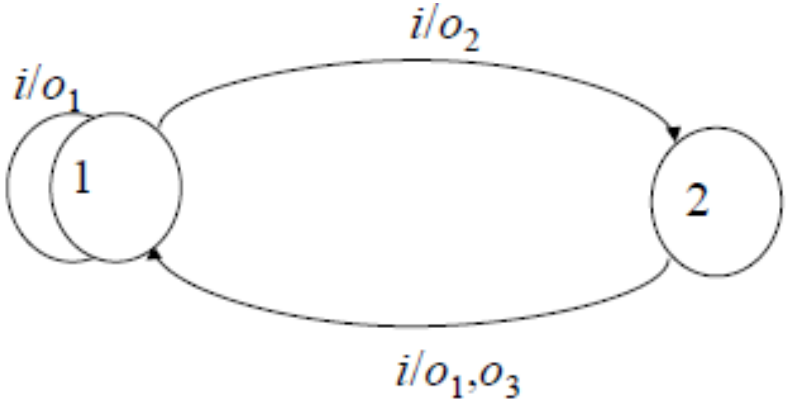
Où sont utilisées ces méthodes ?

- Test de conformité des protocoles de communication
- Test de tout système/sous-système modélisé comme une FSM, par ex. ascenseurs, composants automobiles, etc.
- La génération de tests à partir du FSM aide à tester la conformité des implémentations au modèle correspondant
- Tests de couverture basés sur la boîte blanche

POLYTECH PARIS-SACLAY

Finite State Machines with Output

- Mealy Machine (G. H. Mealy -1955) – Les sorties correspondent aux transitions entre états.
- Moore Machine (E. F. Moore -1956) – Les sorties sont déterminées uniquement par les états



Mealy machine (1)

$\mathcal{S} =< S, I, O, f, g, S_0 >$ is a *Mealy machine*

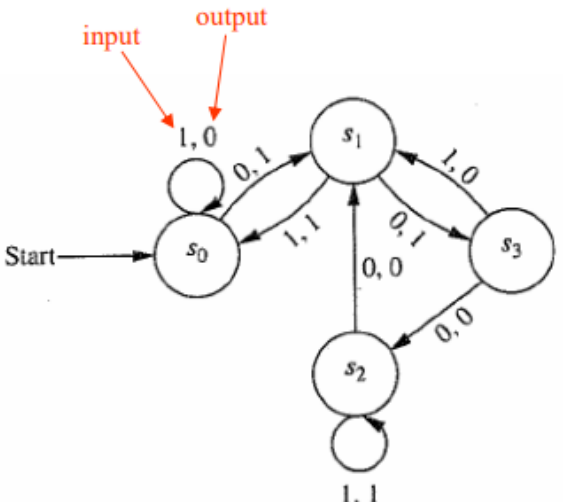
- S is a finite nonempty set of states
- S_0 the start state (a.k.a. initial state) contained in S
- I and O are finite input and output alphabets
- $f : S \times I \rightarrow S$ is a behaviour relation that maps a state/input pair to the next state
- $g : S \times I \rightarrow O$ is an output function that maps a state/input pair to an output

POLYTECH PARIS-SACLAY

Mealy machine (2)

Représentation en diagramme d'états

- Un diagramme d'état est un graphe orienté où chaque noeud est un état et chaque arête est une transition entre les états
- Chaque transition d'un état peut être déclenchée par une entrée et produire une sortie
  $Input \times CurrentState \rightarrow Output \times NextState$



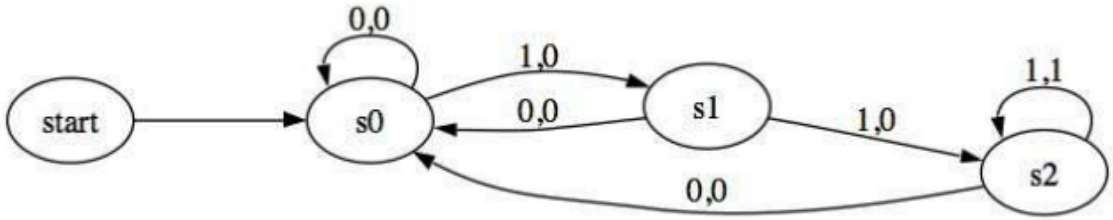
Représentation tabulaire

TABLE				
	f		g	
	Input		*Input*	
State	*0*	*1*	*0*	*1*
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0

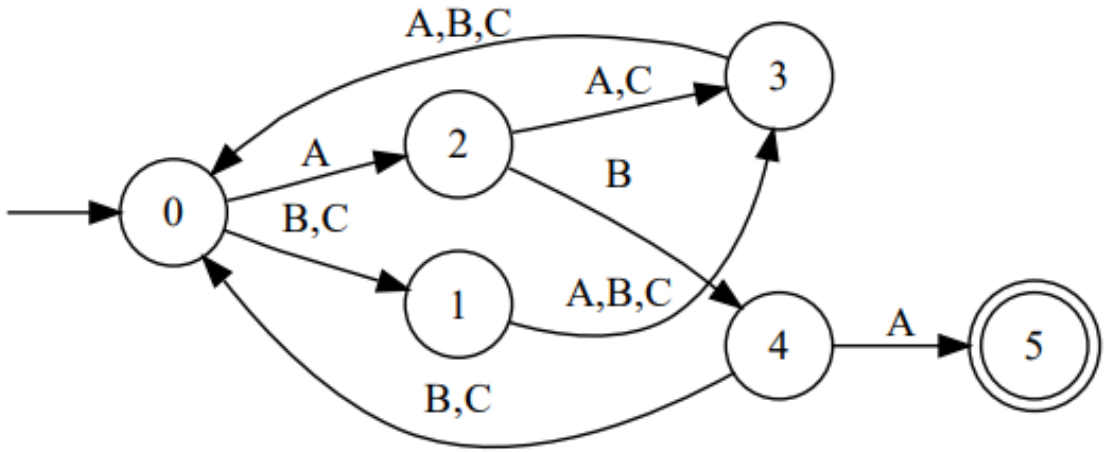
POLYTECH PARIS-SACLAY

Mealy machine (3): Examples

Une machine Mealy qui renvoie 1 si et seulement si la chaîne d'entrée lue jusqu'à présent se termine par 111. Cette machine est un outil de reconnaissance de langage



Un digicode à trois touches A, B, C. La porte s'ouvre quand ABA est saisi.Le digicode est dans son état initial après saisie d'un mauvais code



Propriétés de FSM

Un FSM $\mathcal{S}$ est

- Déterministe : pour chaque paire d'état et de valeur d'entrée, il existe un état suivant unique donné par la fonction de transition
- Non déterministe : il peut y avoir plusieurs états suivants possibles pour chaque paire d'état et de valeur d'entrée.
- Complètement spécifié si à partir de chaque état de $\mathcal{S}$ il existe une transition pour chaque symbole d'entrée
- Fortement connecté si pour chaque paire d'états (s_i, s_j) il existe une séquence d'entrée qui fait passer $\mathcal{S}$ de l'état s_i à s_j
- Minimal si le nombre d'états dans $\mathcal{S}$ est inférieur ou égal à tout autre FSM équivalent à $\mathcal{S}$

POLYTECH PARIS-SACLAY

Equivalence d'états

- V-Equivalence de deux états
  -> Soient $\mathcal{S}_1 =< I, O, S_1, S_0^1, f_1, g_1 >$ et $\mathcal{S}_2 =< I, O, S_2, S_0^2, f_2, g_2 >$ deux FSM (Mealy Machines)
  -> Soit V un ensemble de chaînes non vides sur l'alphabet d'entrée I, c'est-à-dire $V \subseteq I^+$
  -> Soit s_i et s_j, $i \neq j$, deux états des machines $\mathcal{S}_1$ et $\mathcal{S}_2$, respectivement

  s_i et s_j sont considérés comme $V - quivalents$ si $g_1(s_i, v) = g_2(s_j, v)$ pour tout v dans V. En d'autres termes, les états s_i et s_j sont considérés comme $V - quivalents$ si $\mathcal{S}_1$ et $\mathcal{S}_2$, lorsqu'ils sont excités respectivement dans les états s_i et s_j, produisent des séquences de sortie identiques.
- Equivalence de deux États
  -> Les états s_i et s_j sont dits équivalents si $g1(si, v) = g2(sj, v)$ pour tout ensemble V
    - Si s_i et s_j ne sont pas équivalents alors ils sont distinguables
  -> On écrit $s_i = s_j$ si les états s_i et s_j sont équivalents, et $s_i \neq s_j$ lorsqu'ils sont distinguables

Equivalence de deux machines

Équivalence de deux FSM

- $\mathcal{S}_1$ et $\mathcal{S}_2$ sont dites équivalentes si
  - -> Pour chaque état α dans $\mathcal{S}_1$ il existe un état α' dans $\mathcal{S}_2$ tel que α et α' soient équivalents
  - -> Pour chaque état β dans $\mathcal{S}_2$ il existe un état β' dans $\mathcal{S}_1$ tel que que β et β' sont équivalents
- Les machines qui ne sont pas équivalentes sont considérées comme distinguables
- Si $\mathcal{S}_1$ et $\mathcal{S}_2$ sont fortement connexes, alors ils sont équivalents si leurs états initiaux respectifs, s_0^1 et s_0^2, sont équivalents
- On écrit $\mathcal{S}_1 = \mathcal{S}_2$ si les machines $\mathcal{S}_1$ et $\mathcal{S}_2$ sont équivalentes, et $\mathcal{S}_1 \neq \mathcal{S}_2$ lorsqu'elles sont distinguables

K – Equivalence

- Les états $s_i \in S_1$ et $s_j \in S_2$ sont considérés comme *k – equivalents* si, lorsqu'ils sont excités par une entrée de longueur k, produisent des séquences de sortie identiques
- Les états qui ne sont pas k – *equivalents* sont considérés comme *k – distinguables*
- Il est aussi facile de voir que si deux états sont k – *distinguables* pour tout $k > 0$ alors ils sont aussi distinguables pour tout *nk*

POLYTECH PARIS-SACLAY

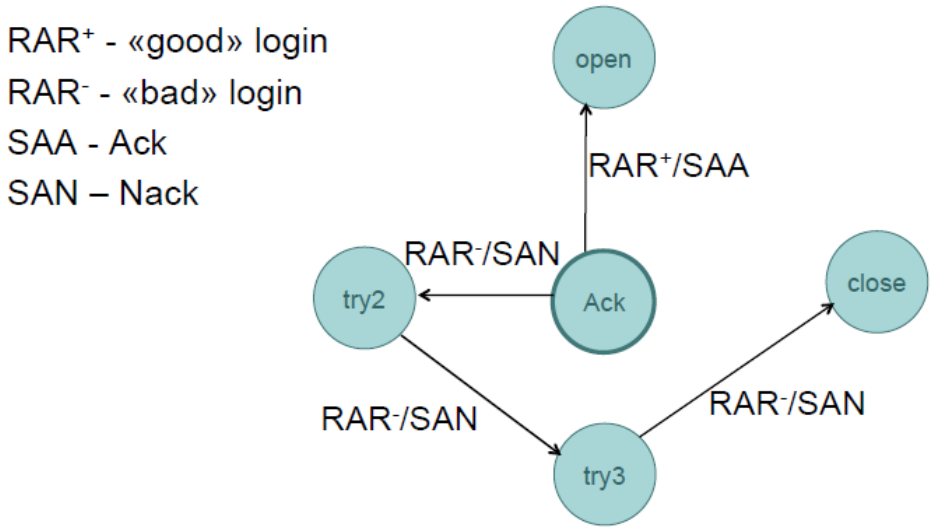
Exemple : Password Authentication Protocol (PAP)

- Deux entités partagent un mot de passe à l'avance et utilisent le mot de passe comme base d'authentification
- Considéré comme peu secure, mais cela est une autre affaire!

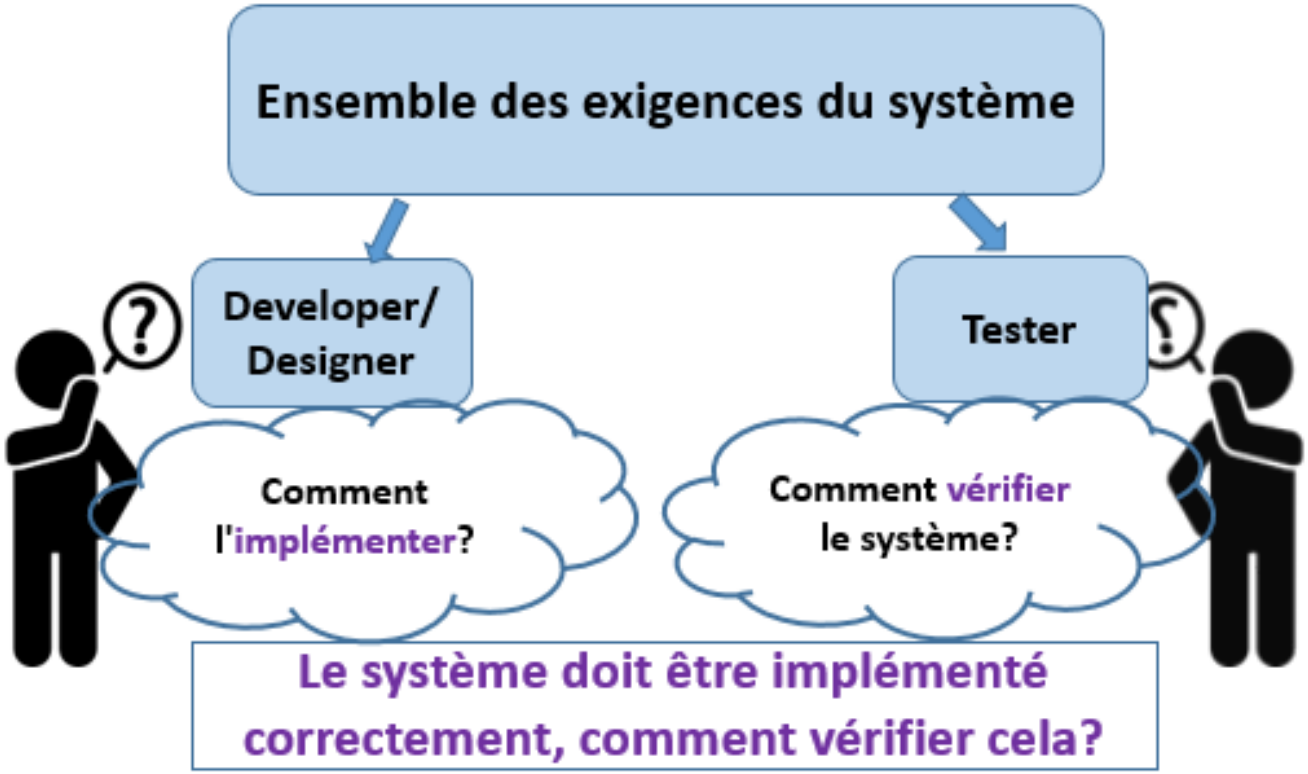
Comment ça fonctionne?

- Un client envoie un nom d'utilisateur et un mot de passe
- Le serveur envoie une authentification : *Ack* (quand OK !) ou *Nack* (quand pas OK !)

One of FSMs for PAP



Fautes dans l'implémentation?



Un FSM sert à spécifier l'exigence ou la conception correcte d'un système. Par conséquent, les tests générés à partir d'un FSM ciblent les défauts liés au FSM lui-même.

Quelles sont les fautes visées par les tests générés à l'aide d'un FSM ?
Comment générer les tests?

POLYTECH PARIS-SACLAY