UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Génie des systèmes interactifs

Mathieu RAYNAL

mathieu.raynal@irit.fr

https://www.irit.fr/~Mathieu.Raynal/

Fonctionnement par événements

- Evénement = Action effectuée par l'utilisateur

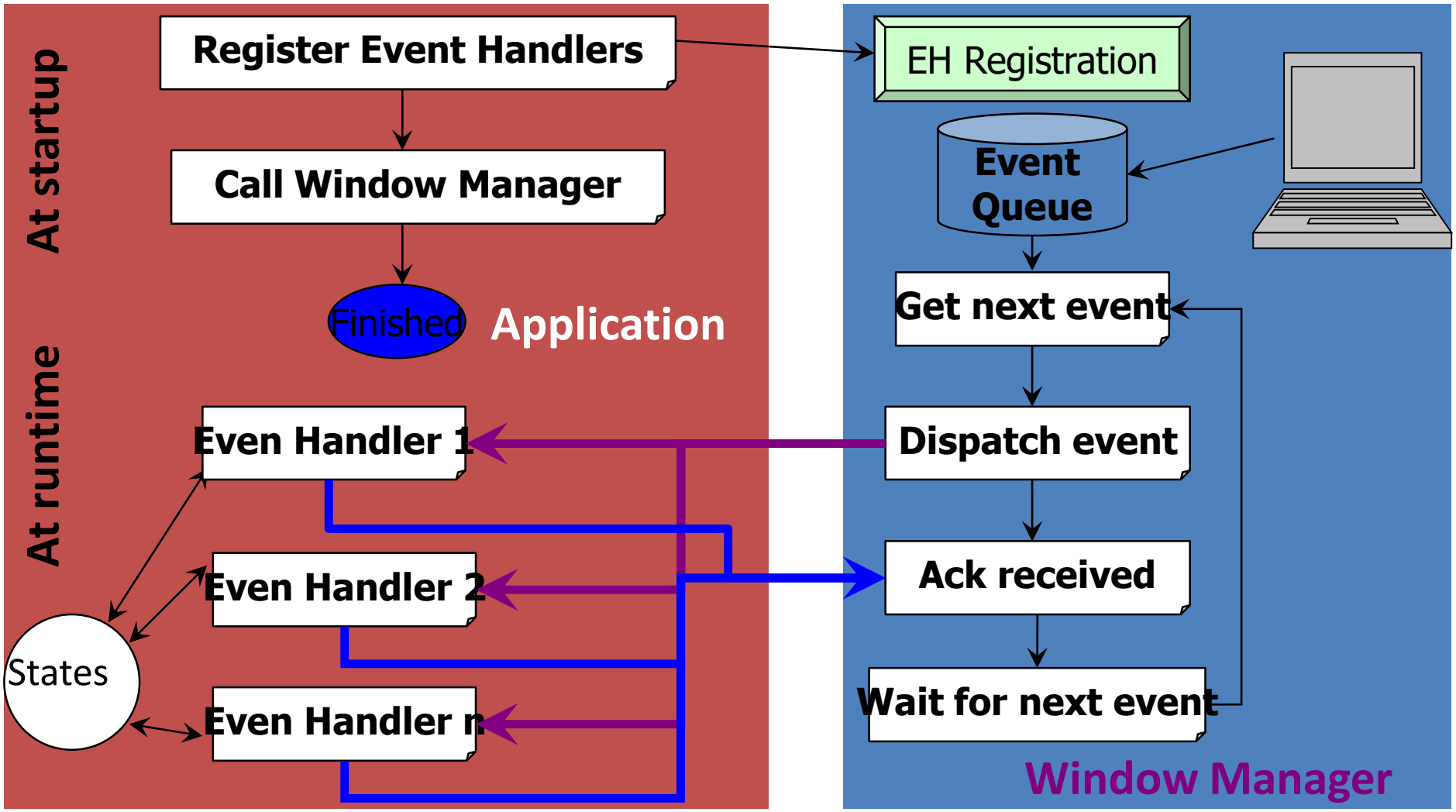
- Les sources
  - Matériel
    - Clavier,
    - Souris,
    - Ecran tactile,
    - …
  - Logiciel (composants graphiques)
    - Fenêtre,
    - Bouton,
    - Liste,
    - …

Les événements

- Deux niveaux d'événements
  - Bas niveau
    - Déplacer la souris
    - Appuyer sur un bouton d'un dispositif
    - Taper sur une touche du clavier
  - Sémantique
    - Appuyer sur un bouton de l'interface
    - Prendre ou perdre le focus
    - Entrer ou sortir d'un composant

- Les événements sont associés aux composants graphiques
  - Un composant peut être lié à plusieurs événements
  - Un type d'événement peut se produire sur différents types de composants

Gestion des événements



Structure application par événement

- La boucle d'événement (main event loop)
  - reçoit chaque événement produit par l'utilisateur

- Les gestionnaires d'événements (event handler)
  - procédures associées à chaque couple
    - Composant graphique
    - Action sur le composant
  - appelés par la main event loop dès qu'une action a été réalisée.
  - ont tous la même structure

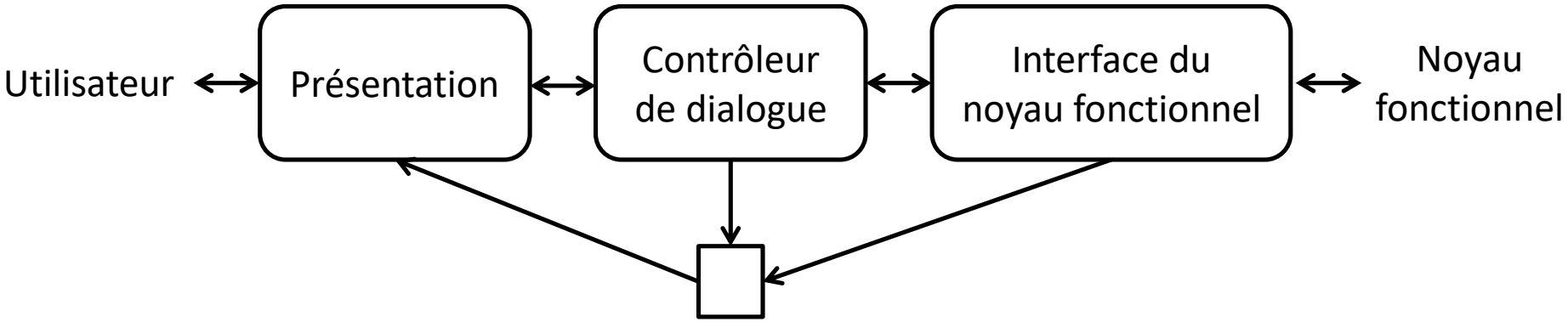
Les gestionnaires d'événements

- Un gestionnaire d'événements gère un type d'événement
  - Un événement peut être envoyé à plusieurs gestionnaires d'événements
  - Plusieurs événements peuvent être envoyé au même gestionnaire d'événements
- Attention : en JAVA/SWING, les gestionnaires d'événements ne sont pas les Listeners !
  - Ce sont les méthodes des Listeners

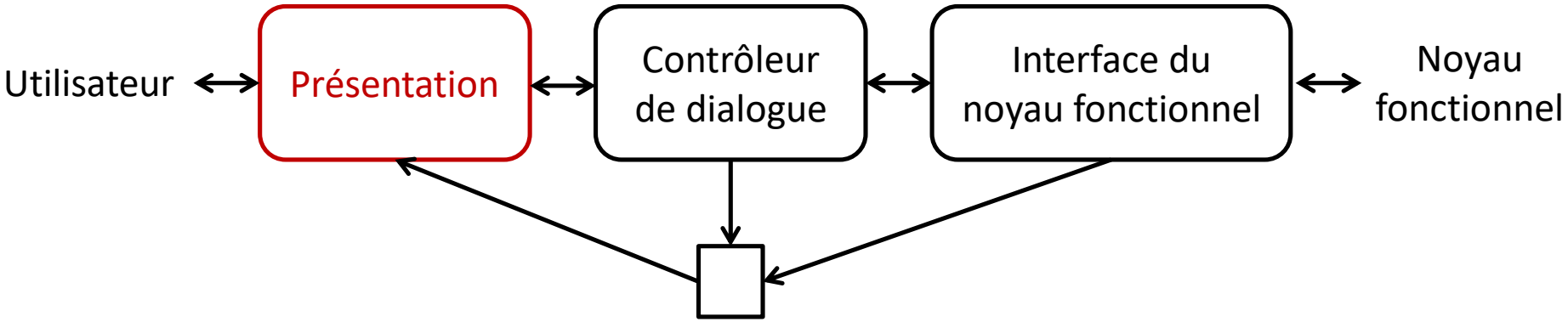
UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Comment faire pour que tous les gestionnaires d'événements communiquent entre eux ?

Modèle de Seeheim

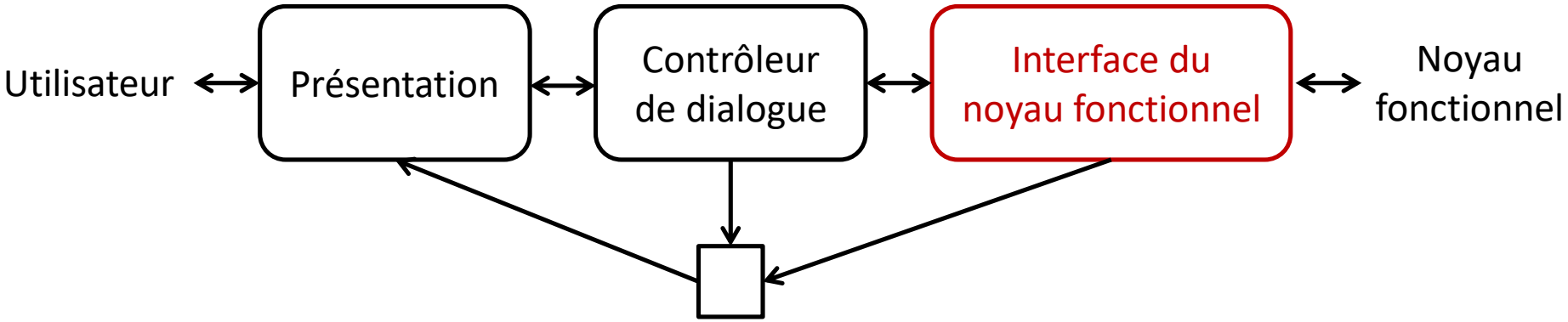


Modèle de Seeheim



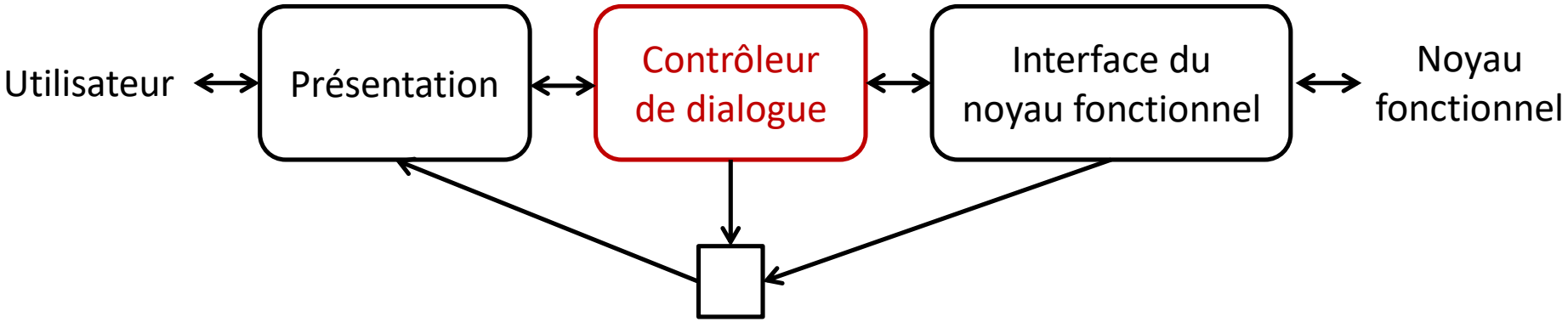
- Ce que l'utilisateur voit de l'application

Modèle de Seeheim



- Les fonctions réalisées par l'application
- Les données manipulées par l'application

Modèle de Seeheim

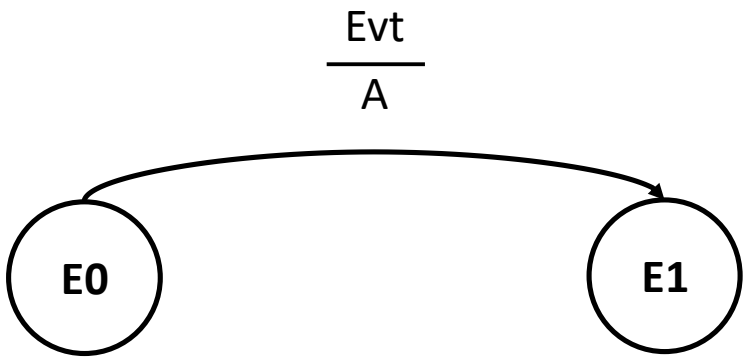


- Le contrôleur de dialogue définit :
  - ce que l'utilisateur a la possibilité de faire
  - Comment l'utilisateur peut agir sur la présentation
  - L'influence de son action sur ce qu'il pourra faire ensuite

→ Comment modéliser ce contrôleur de dialogue ?

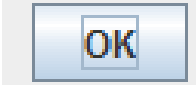
Modélisation du contrôleur de dialogue

- Sous forme d'automate de comportement
  - État du système {E0, E1}
    - Représente ce que le système permet de faire à un instant t
  - Événements {Evt}
    - Se produit à partir d'un état
    - Engendre généralement une action du système
    - Peut modifier l'état du système
  - Actions {A}
    - Ce que le système fait



Exemple : Modéliser le fonctionnement d'un bouton

- 3 états différentes
  - Par défaut



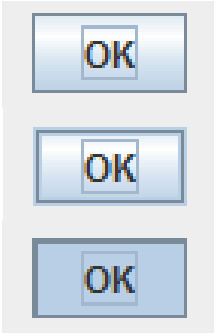
  - Survolé par le pointeur de la souris

  - Lors de l'appui sur le bouton de la souris
    - Possible que lorsque le pointeur est au dessus du bouton

Exemple : Modélisation du bouton

- Liste des événements
  - E : Entrer dans le bouton avec le pointeur de la souris
  - S : Sortir du bouton avec le pointeur de la souris
  - P : Presser le bouton de la souris
  - R : Relâcher le bouton de la souris

- Liste des actions
  - A0 : Afficher style par défaut
  - A1 : Afficher style survolé
  - A2 : Afficher style enfoncé
  - A3 : Lancer événement

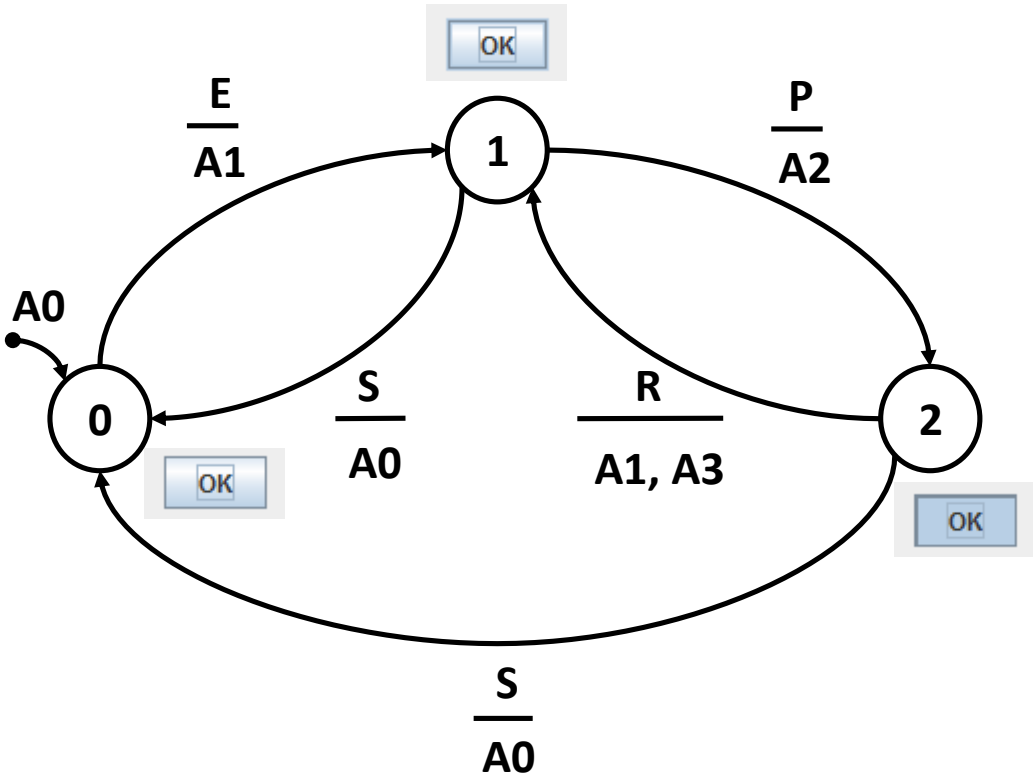


Exemple : Modélisation du bouton

- Liste des événements
  - E : Entrer dans le bouton avec le pointeur de la souris
  - S : Sortir du bouton avec le pointeur de la souris
  - P : Presser le bouton de la souris
  - R : Relâcher le bouton de la souris

- Liste des actions
  - A0 : Afficher style par défaut
  - A1 : Afficher style survolé
  - A2 : Afficher style enfoncé
  - A3 : Lancer événement

- Automate de comportement



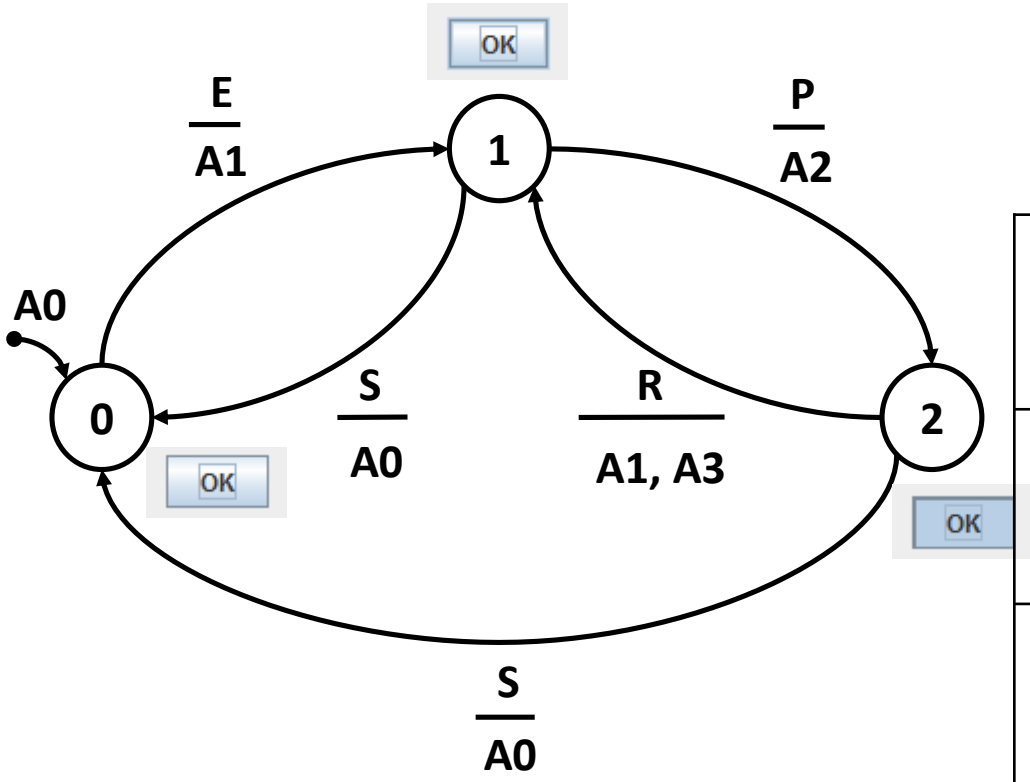
Comment passer de l'automate à l'implémentation ?

- Matrice états / événements
  - Une ligne par état du système
  - Une colonne par événement exploitable

- Chaque cellule correspond à ce que doit faire le système
  - lorsque l'événement, correspondant à cette colonne, se produit
  - Et que le système est dans l'état correspondant à cette ligne

Exemple : Modélisation du bouton

- Matrice états / événements



Init :
e = 0
A0

	E	P	S	R
0	e=1 A1			
1		e=2 A2	e=0 A0	
2			e=0 A0	e=1 A1, A3

Exercice 1

- 4 Boutons
  - Au départ, le bouton 1 est en gras
  - Si l'on clique sur le bouton en gras, le suivant devient en gras à sa place
  - De manière cyclique

Exercice 1 : A faire

1. Liste des états
2. Liste des événements
3. Liste des actions
4. Automate de comportement
5. Matrice états / événements

Exercice 1 : Constat

- Problème de conception
  - A chaque état, 3 boutons sont cliquables alors qu'ils ne permettent aucune action …

- Il faut empêcher l'utilisateur de produire un événement lorsque ce dernier n'a pas lieu d'être à un état donné du système

  → Il faut désactiver le composant qui permet cet événement

Gestion des événements

- A chaque changement d'état, il faut vérifier les événements qui pouvaient se produire sur l'état « source » (S) et ceux qui peuvent se produire sur l'état « destination » (D)

  - Si un événement était actif à l'état S et est inactif à l'état D

    → On désactive le composant sur lequel cet événement se produit

  - Si un événement était inactif à l'état S et est actif à l'état D

    → On active le composant sur lequel cet événement se produit

Exercice 2

- Modifier l'automate et compléter la matrice états / événements de l'exercice 1 pour résoudre ce problème de conception

- Résultat attendu

Exercice 3

- 4 Boutons
  - Au départ, les boutons 1 et 2 sont actifs
  - Si l'on clique au moins une fois sur les deux boutons actifs, les deux suivants deviennent actifs à leur place
  - De manière cyclique

Exercice 3 : A faire

1. Liste des états
2. Liste des événements
3. Liste des actions
4. Automate de comportement
5. Matrice états / événements

Exercice 3 : Constat

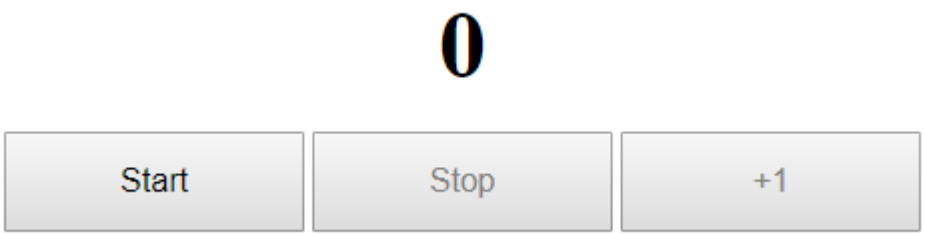
- Problème de conception
  - Trois états du système ont le même aspect graphique
    - Comment l'utilisateur sait dans quel état il se trouve ?
    - Ce qu'il peut faire ?
    - Les conséquences d'une de ses actions ?
- Il doit y avoir un retour à l'utilisateur à chaque changement d'état du système

Exercice 4 : Compteur

- A l'état initial,
  - le compteur affiche 0
  - L'utilisateur peut activer le compteur avec le bouton « Start »

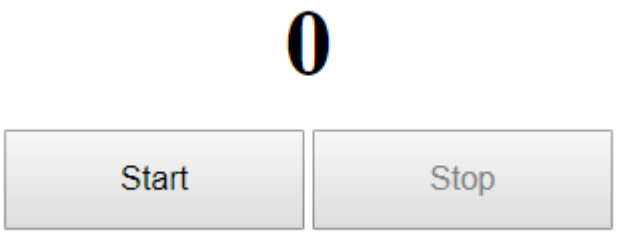
- Une fois le compteur activé
  - L'utilisateur peut l'arrêter à tout moment en appuyant le bouton « Stop », le compteur revient à l'état initial en conservant sa valeur

  - L'utilisateur peut avancer le compteur de 1 en appuyant sur le bouton « +1 »
    - Si le compteur atteint la valeur 3, le prochain clic sur le bouton « +1 » affiche « plouf » à la place de la valeur. L'utilisateur ne peut plus appuyer que sur le bouton « Start »



Exercice 5 : Chrono

- Le fonctionnement est identique au compteur
- Après l'appui sur le bouton « Start », le compteur est incrémenté de 1 toutes les secondes
- Les conditions d'arrêt sont les mêmes que pour le compteur



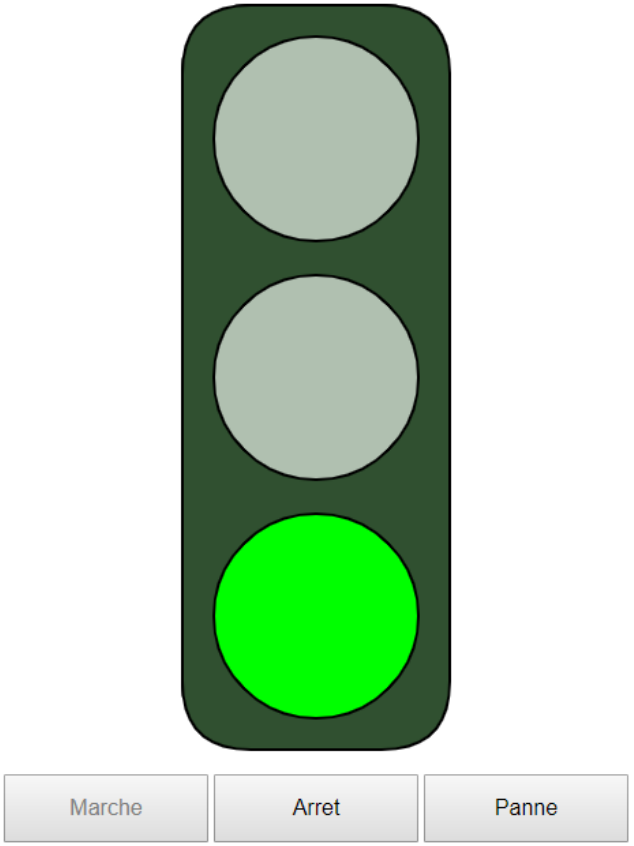
Exercice 6 : Chrono double sens

- Lorsque le chrono est en train d'incrémenter la valeur du compteur, l'utilisateur a la possibilité d'appuyer sur le bouton « << ».
  - S'il appuie sur ce bouton, le compteur décrémentera

- Inversement si le compteur est en train de décrémenter la valeur du compteur, l'utilisateur a la possibilité d'appuyer sur le bouton « >> ».
  - S'il appuie sur ce bouton, le compteur incrémentera

Exercice 7 : Feu tricolore français

- Au départ, le feu est à l'arrêt
  - Tout est éteint
- Dans le cas de la panne
  - Le feu clignote au orange
- Dans le cas de la marche
  - Il passe successivement au vert, à l'orange et au rouge de manière cyclique



Passage à l'implémentation !

- La valeur de l'état est gérée par une variable de type entier

- Chaque gestionnaire d'événements
  - Correspond à une colonne de la matrice
  - Contient un switch sur la valeur de l'état