COURS 3 : DIAGRAMME D'ÉTAT-TRANSITIONS

Nga NGUYEN, EISTI

Abstractions logicielles

- Un problème complexe peut être « mappé » à un model d'abstraction bien connu :
  - Finite State Machine (FSM)
  - PID contrôleur
  - Réseaux neurones
  - Système linéaire des équations différentielles
- Plus simple à résoudre, à comprendre, à debugger, à modifier, …

Finite State Machine

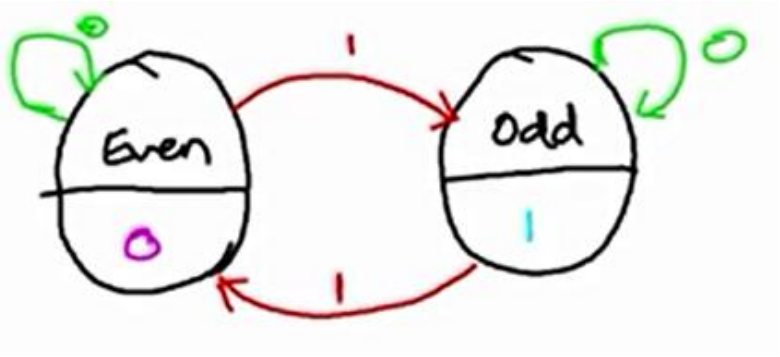
1. Ensemble d'entrées
2. Ensemble de sorties
3. Ensemble d'états, avec un état initial
4. Transitions d'état :

   nextState = f(input, currentState)
5. Détermination de sortie
   - Moore : *output = h(currentState)*
   - Mealy : *output = h(input, currentState)*

Exemple : **Odd's ones detector**

- *Source : UTAustinX: UT.6.02x Embedded Systems - Shape the World (edX)*

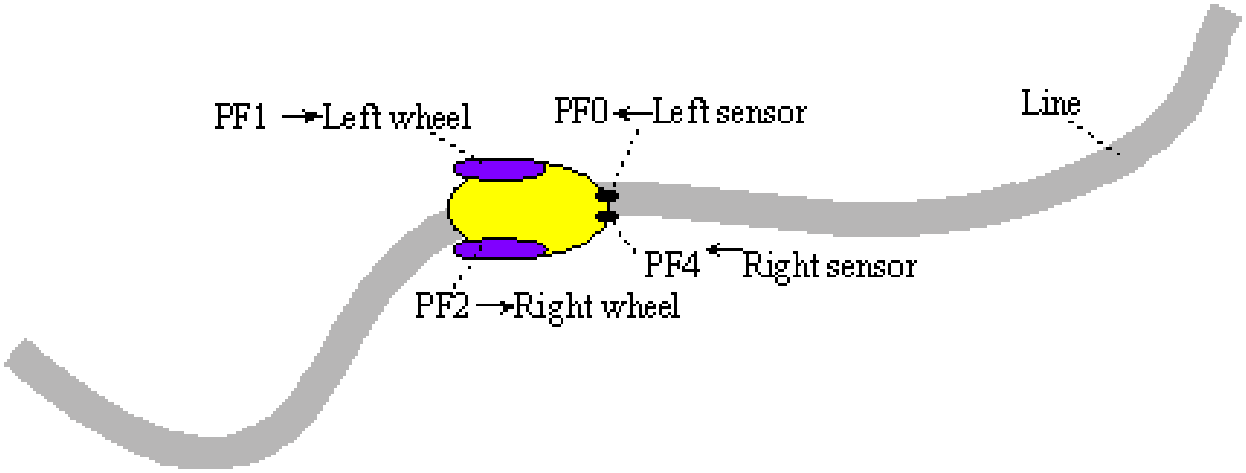
Exemple : **Odd's ones detector**




```
/* FSM data structure (flash EEPROM) */
struct state {
    unsigned char out;
    unsigned long wait;
    unsigned char next[2];
}
typedef struct state SType;
#define Even 0
#define Odd 1
SType fsm[2] = {
{0, 100, {Even, Odd}},
{1, 100, {Odd, Even}}};
```

```
/* FSM execution code (ROM) */
while (1) {
    GPIO_PORTF_DATA_R = fsm[cState].out<<2;          // output based on current state
    SysTick_Wait10ms(fsm[cState].wait);             // wait for time relevant to state
    input = (~GPIO_PORTF_DATA_R & 0x10)>>4;         // get input : NotPressed/Pressed
    cState = Fsm[cState].next[input];               // change state based on input and current state
 }
```

Autre exemple : **Line tracking robot**

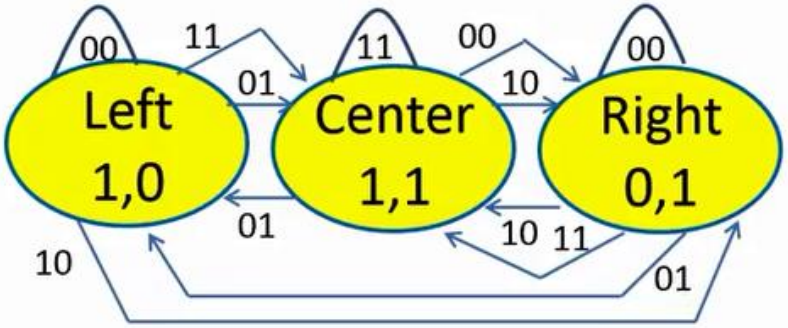


- *Source : UTAustinX: UT.6.02x Embedded Systems - Shape the World (edX)*

Autre exemple : **Line tracking robot**

State Transition Table

State	Motor	In=0,0	In=0,1	In=1,0	In=1,1
Center	1,1	Right	Left	Right	Center
Left	1,0	Left	Center	Right	Center
Right	0,1	Right	Left	Center	Center




```
StateType fsm[3]={
  {0x03, 1, { Right,  Left,   Right,  Center }},
  {0x02, 1, { Left,   Center, Right,  Center }},
  {0x01, 1, { Right,  Left,   Center, Center }}
};
```

Motors respond in 100ms, so run FSM every 10ms

Machines d'états : SysML

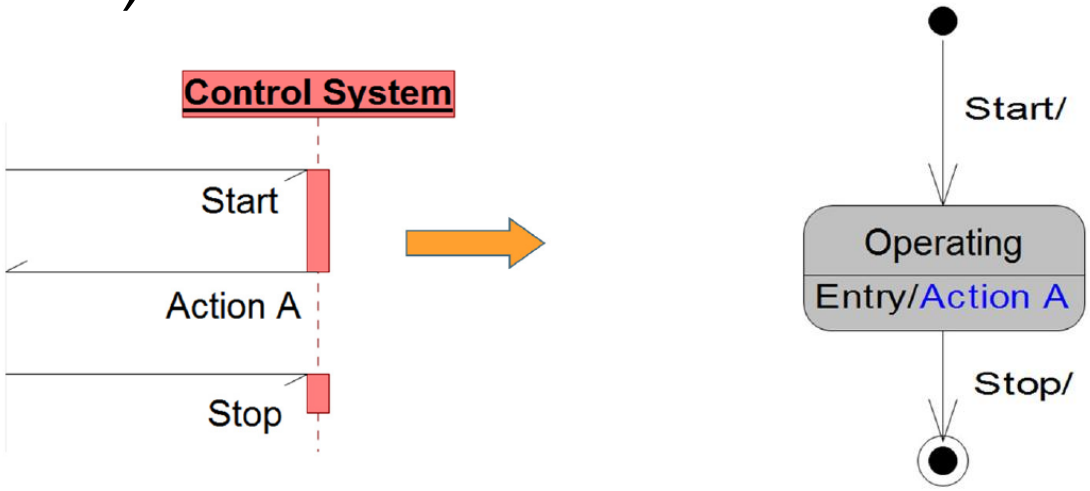
- Le comportement de la plupart de systèmes embarqués est basé sur les états:
  - états séquentiels
  - états concurrents
- Machine d'état :
  - niveau système
  - niveau composant

Les éléments de modélisation

- Etats (initial, final, ….)
- Transitions :
  - Evénements
  - Gardes
  - Effets

Etat – transition

- "A state models a situation during which some (usually implicit) invariant conditions holds" (UML)



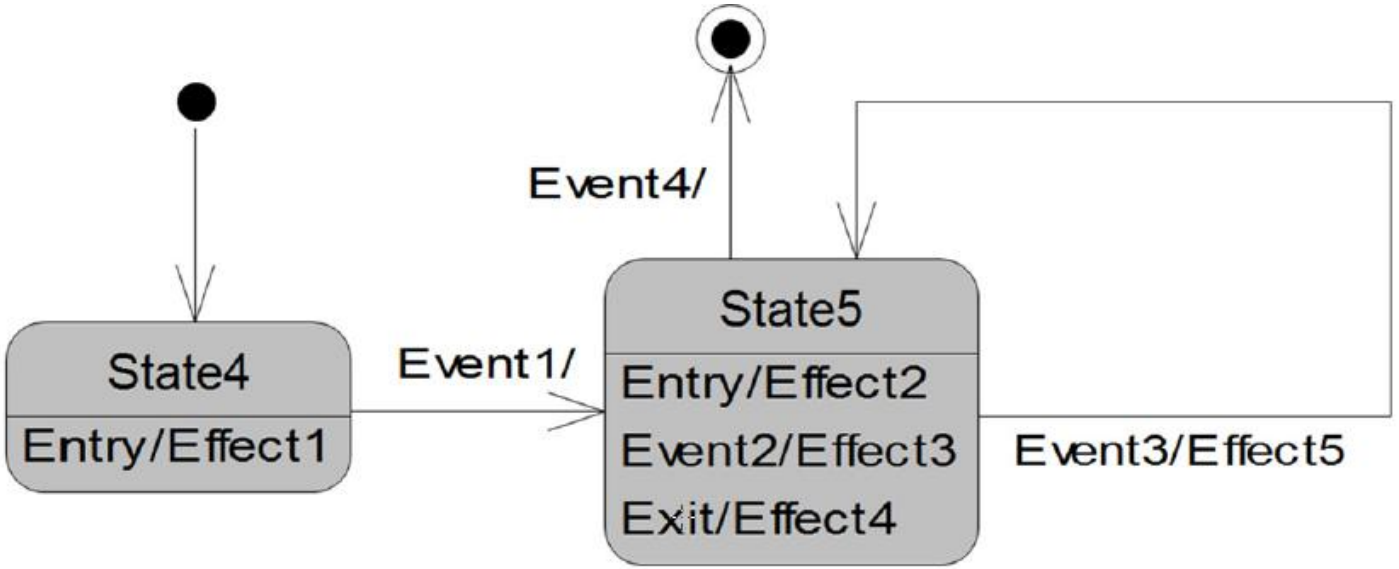
- Transition : relation directionnelle entre un état source et un état destinataire

Événements

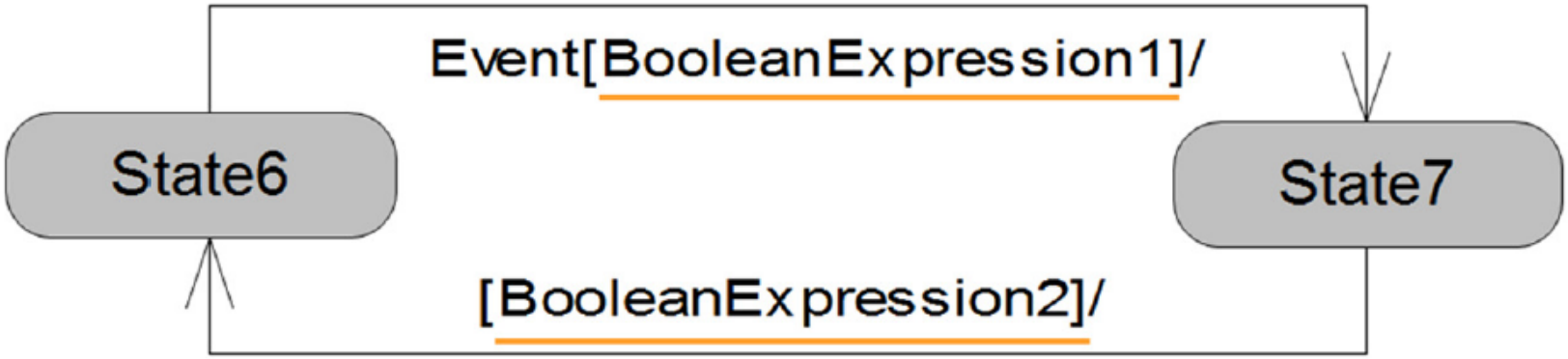
- Réception d'un signal :
  - event name /
- Réception d'un appel d'opération
  - operation name /
- Temps :
  - after(time expression) /
- Changement :
  - when(Boolean expression) /
- Entrée d'un état
  - entry /
- Sortie d'un état :
  - exit /

Effets

- Un effet représente un élément comportemental (cas d'utilisation, opération ou activité)



Condition de garde

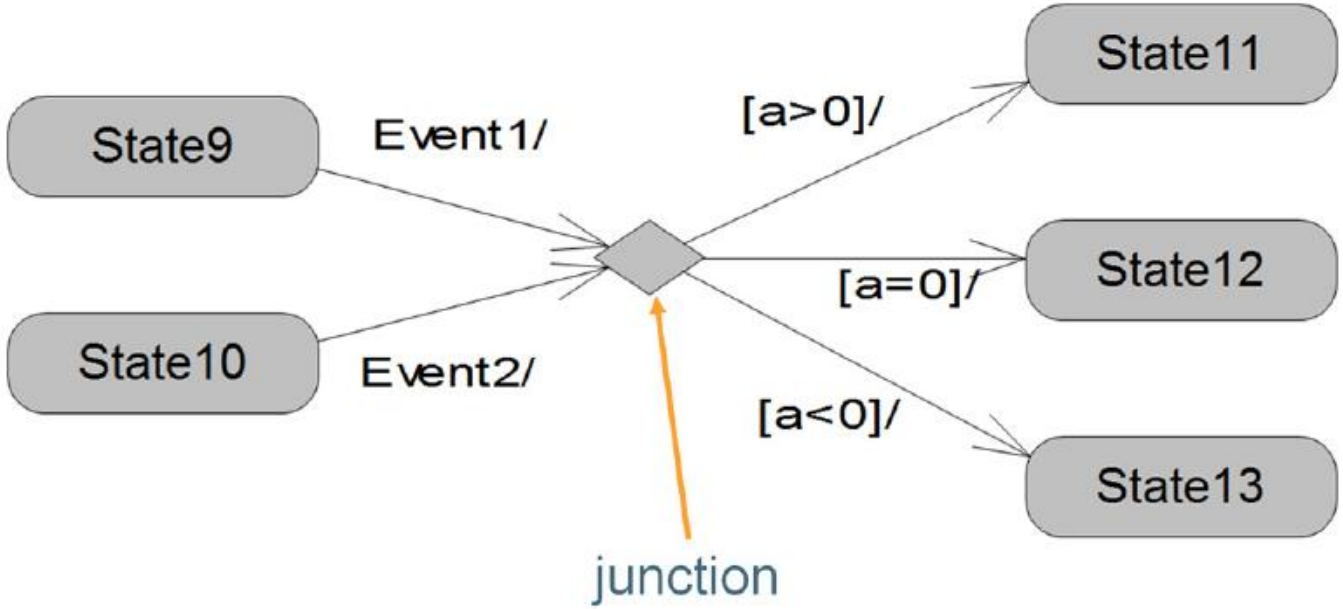


D'autres éléments de modélisation

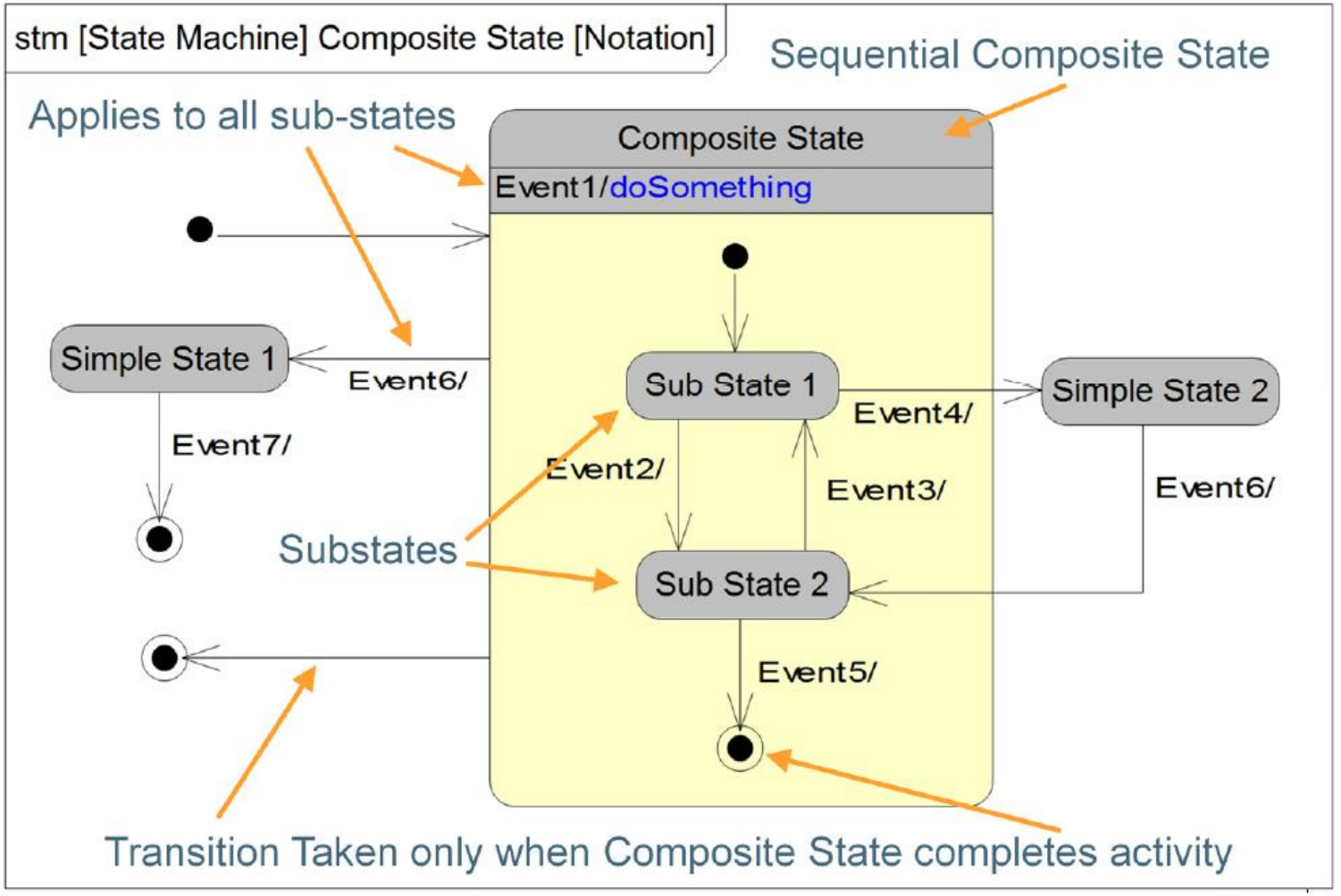
- Jonctions
- Etats composites
- Etats concurrents
- Etats historiques

Jonctions

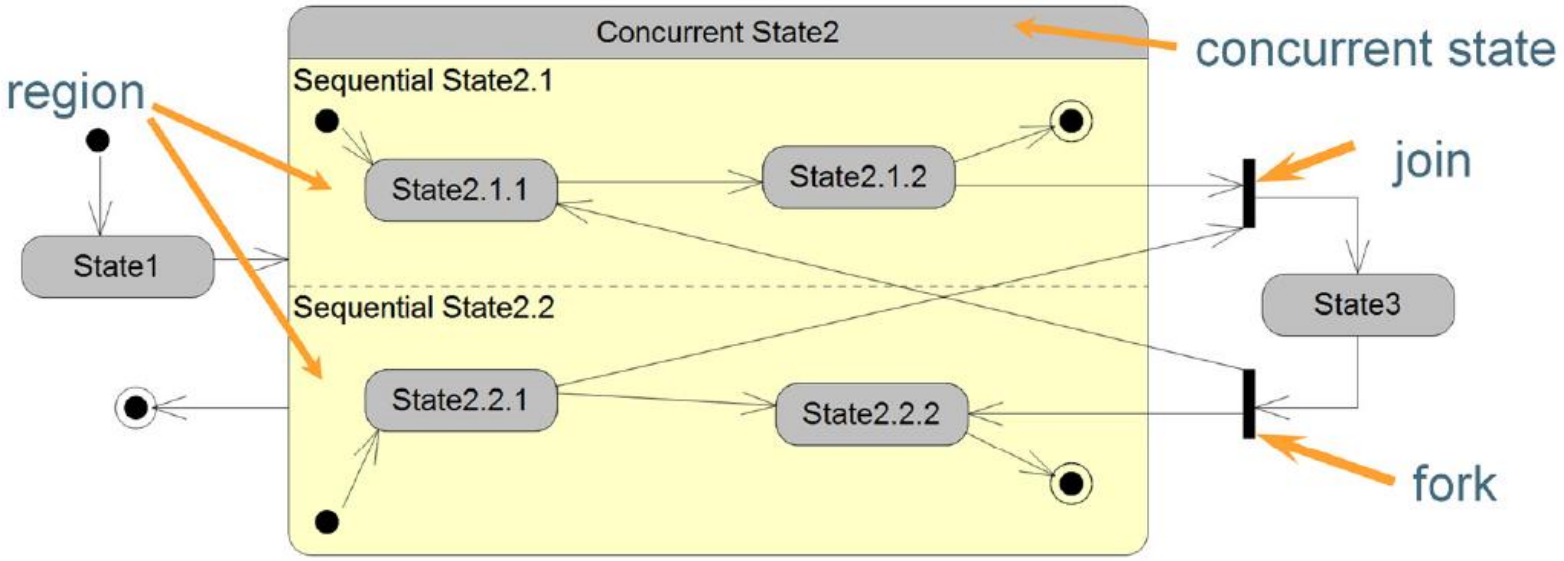
- Permet de fusionner des transitions d'entrée multiples en une transition de sortie
- Permet de diviser une seule transition d'entrée en plusieurs transitions de sortie conditionnelles



Etats composites : séquentiels

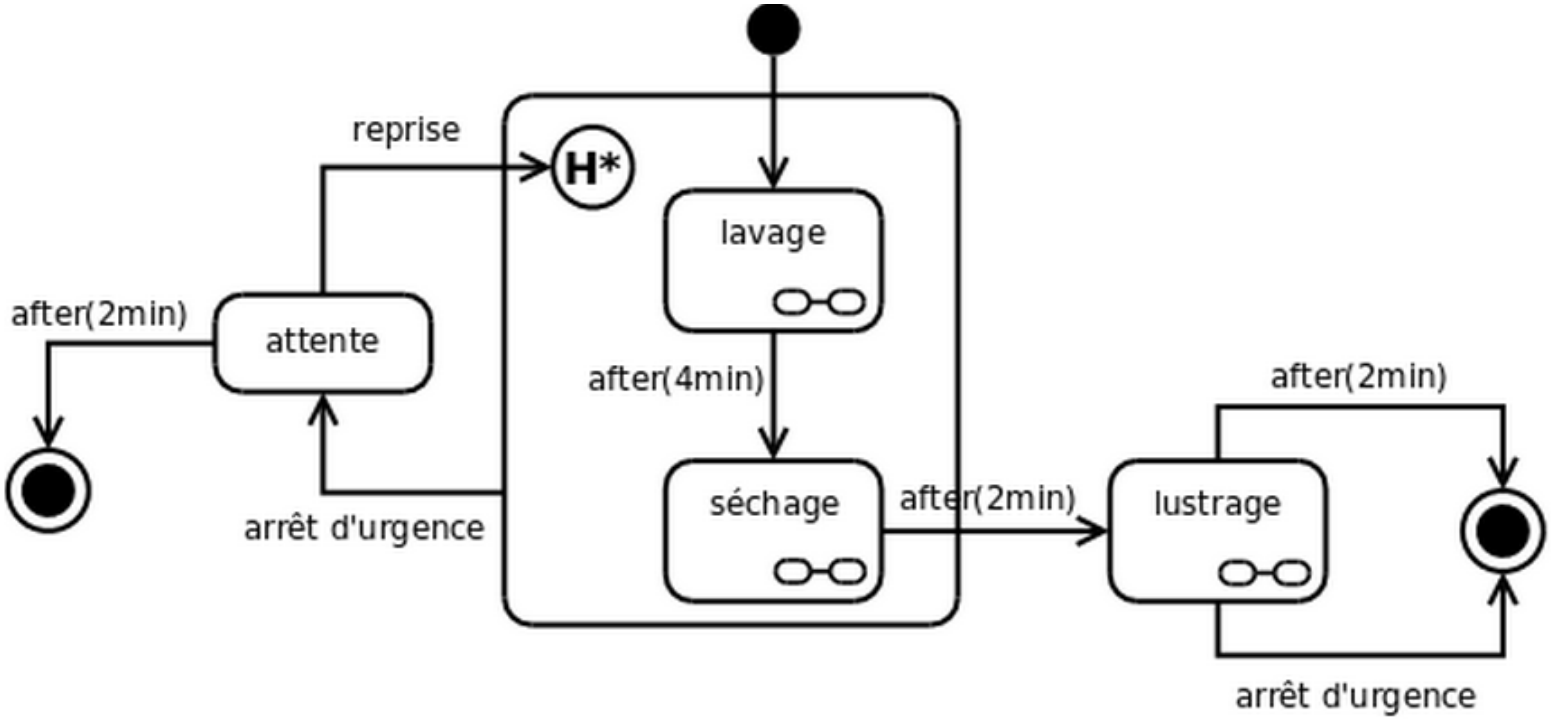


Etats composites : concurrents



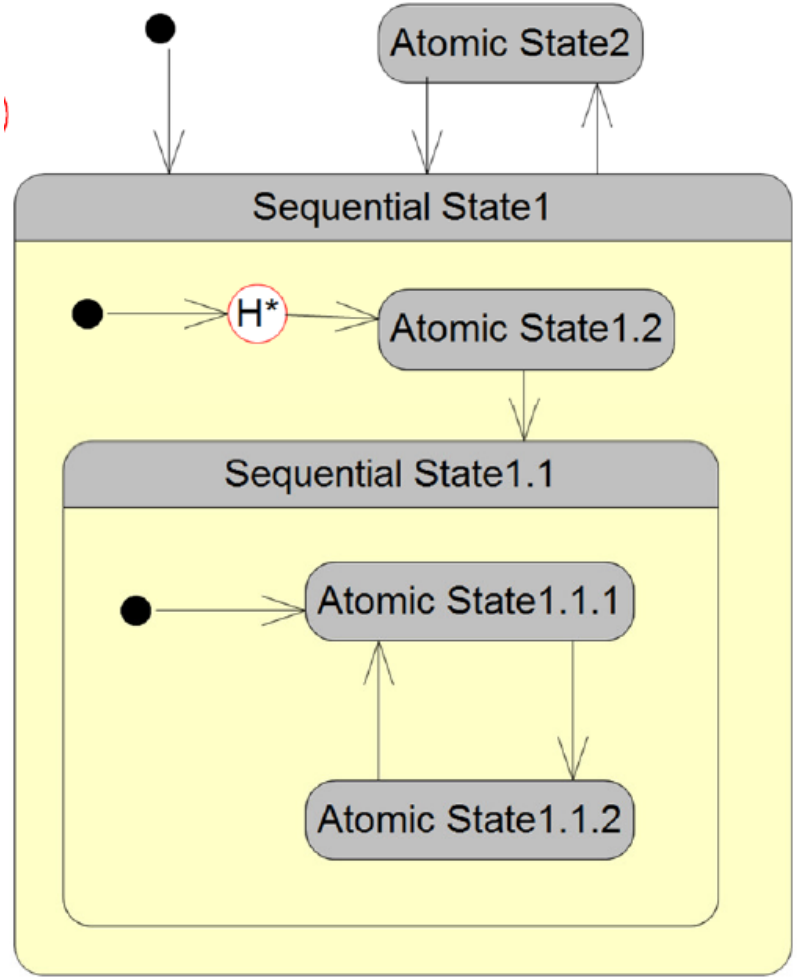
Etat historique

- Mémorise le dernier sous-état actif d'un état composite.



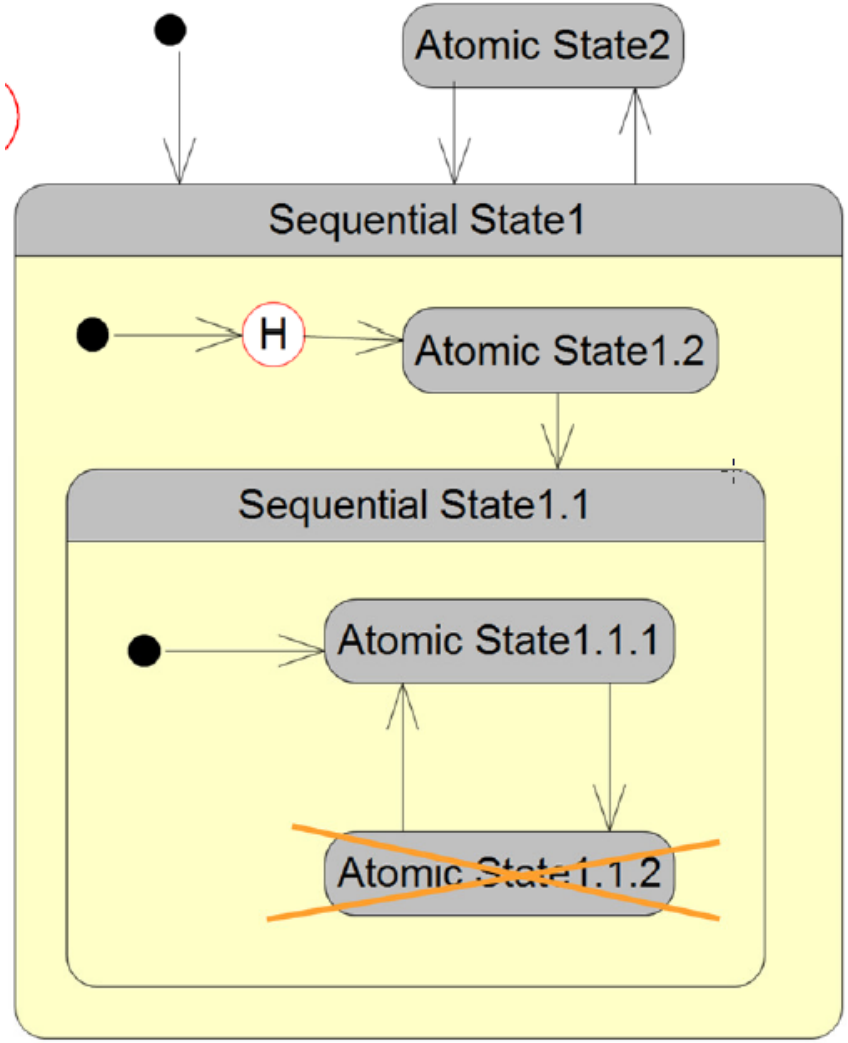
Etat historique profond

- Permet d'atteindre le dernier état visité dans la région, quel que soit sont niveau d'imbrication



Etat historique plat

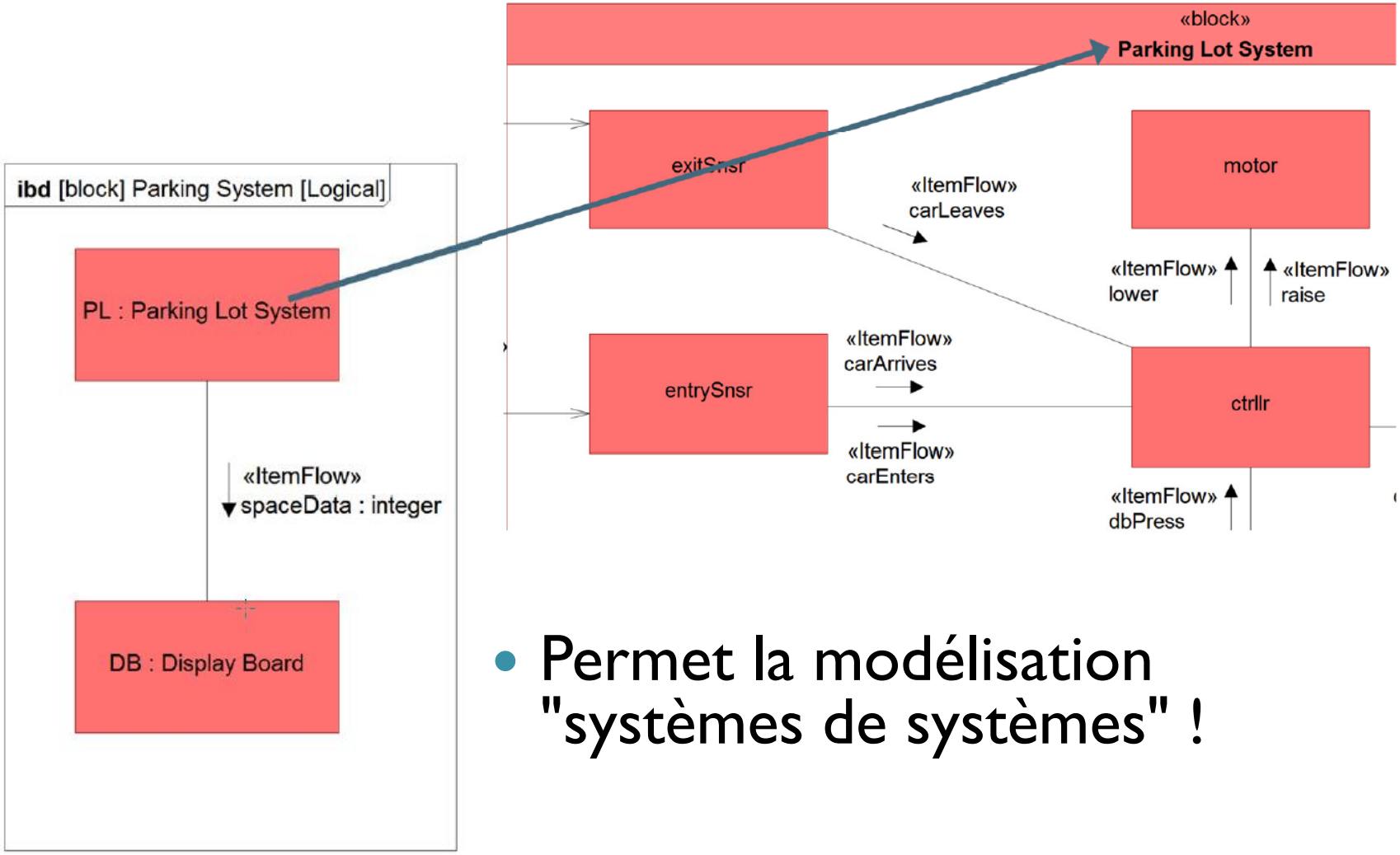
- Limite l'accès aux états de son niveau d'imbrication.



MBSE avec SysML

- Gestion de niveaux d'abstraction
- Modélisation de flot
- Spécification d'interface
- Traçabilité des exigences

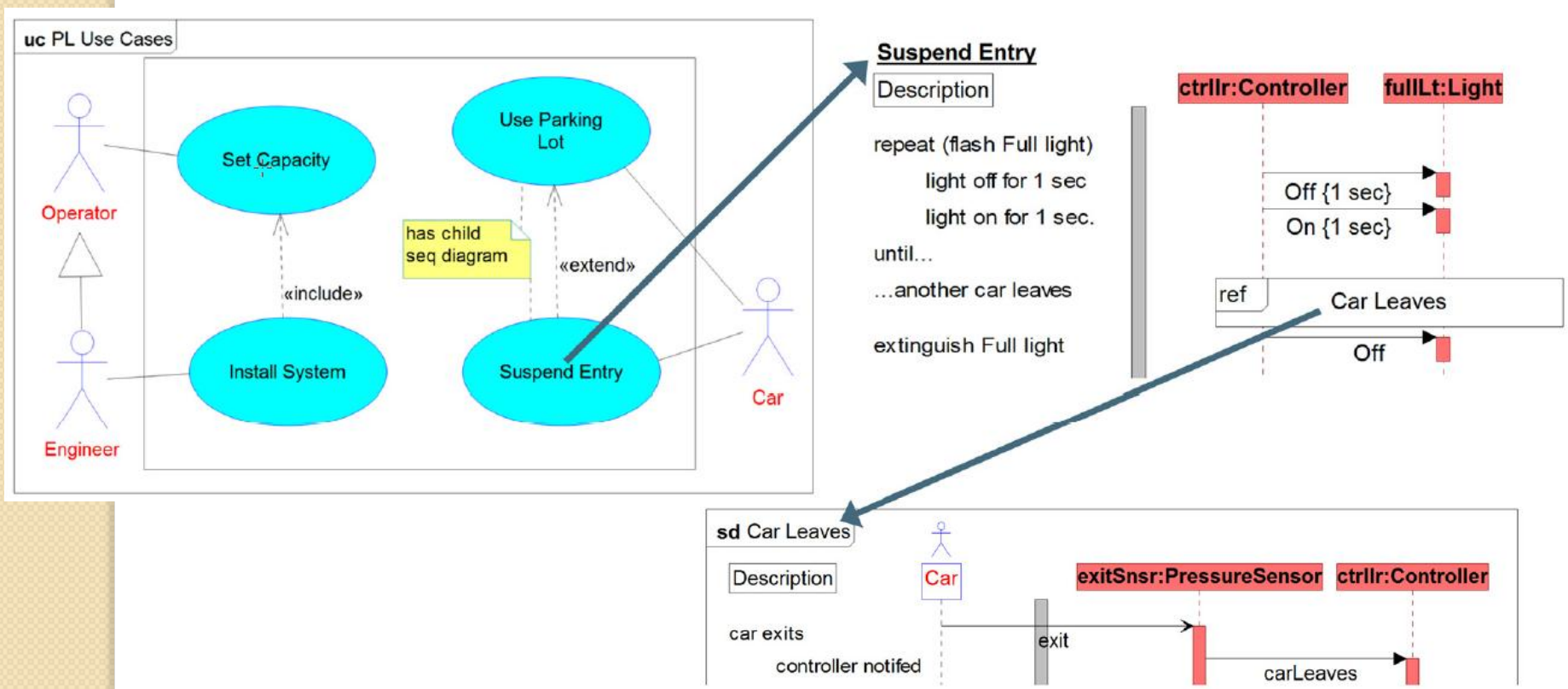
Gestion de niveaux d'abstraction : structures



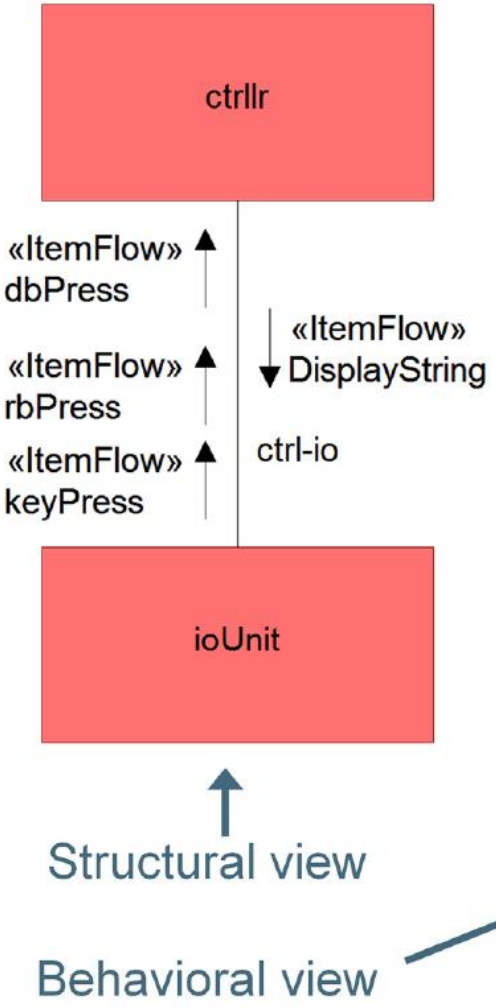
- Permet la modélisation "systèmes de systèmes" !

Gestion de niveaux d'abstraction : comportement

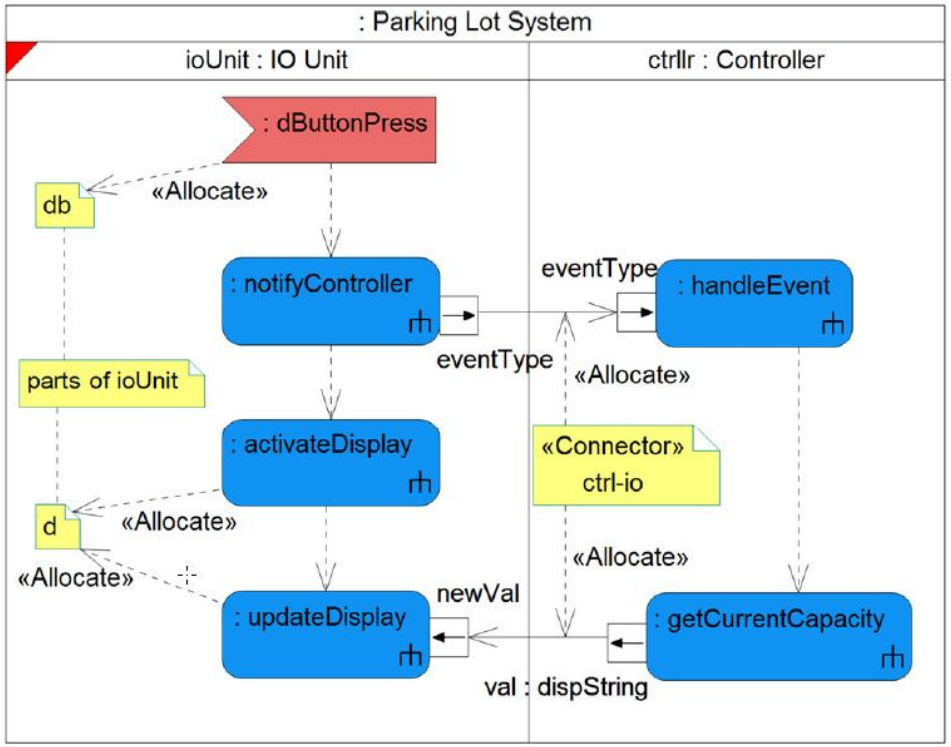
- Les cas d'utilisation définissent les services au niveau système
- Diagrammes de séquence/activités détaillent les cas d'utilisation
- Les modèles d'activité ou d'interaction peuvent être décomposés



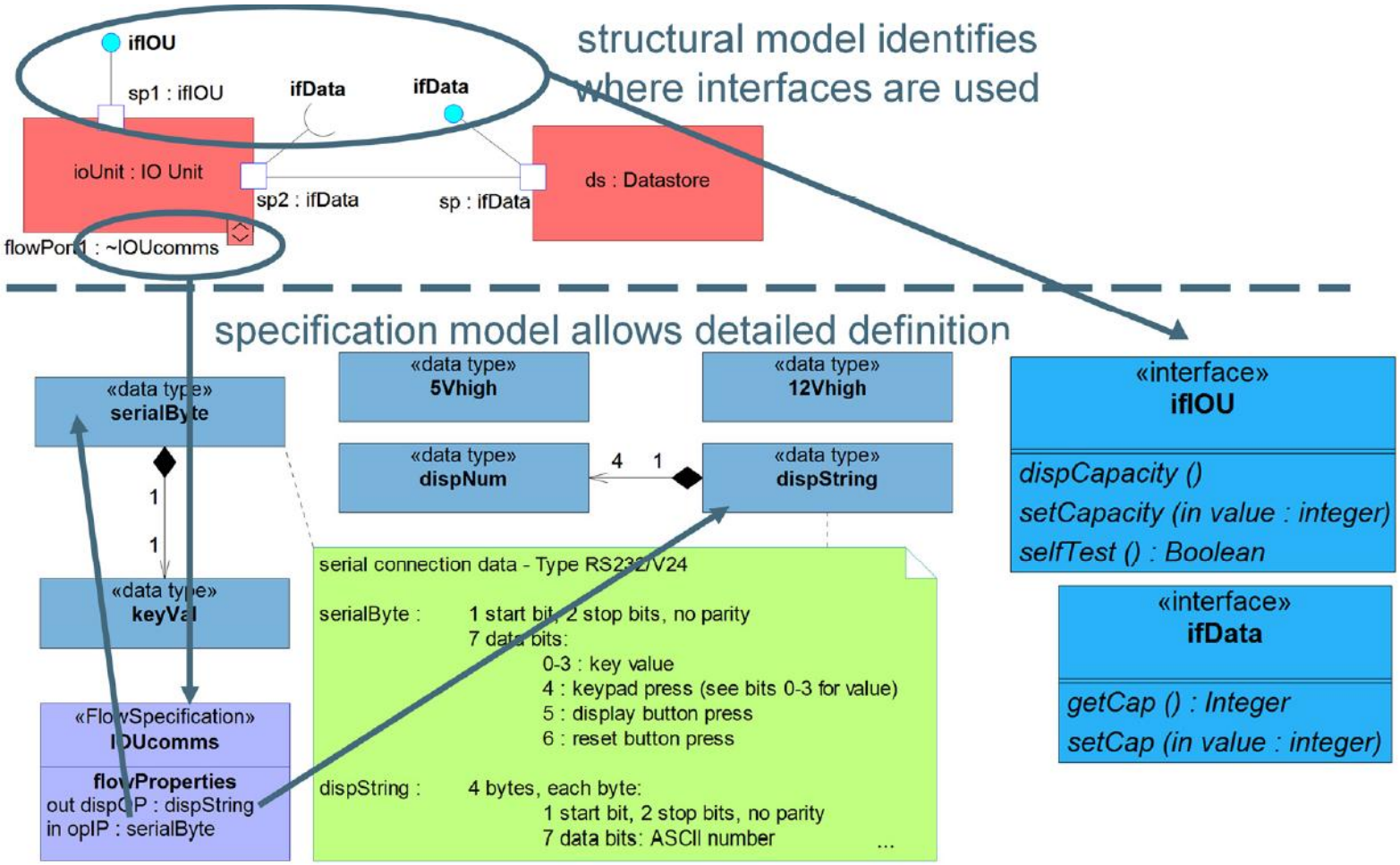
Modélisation de flot



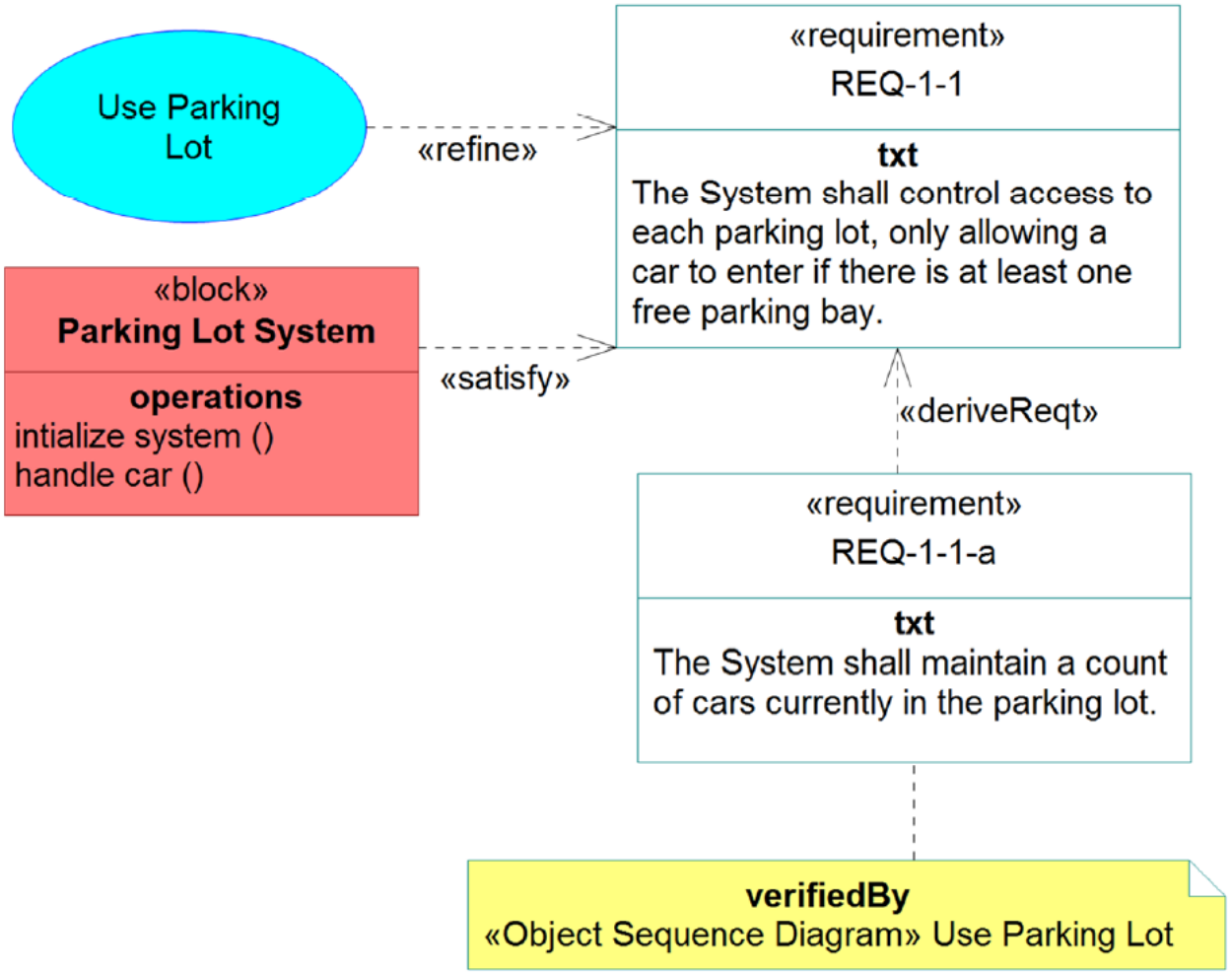
- Données, messages, événements, énergie, fluides, …



Spécification d'interface



Traçabilité des exigences



Cohérence entre les diagrammes comportementaux