Analyse et conception

Cours de 1e année ingénieur

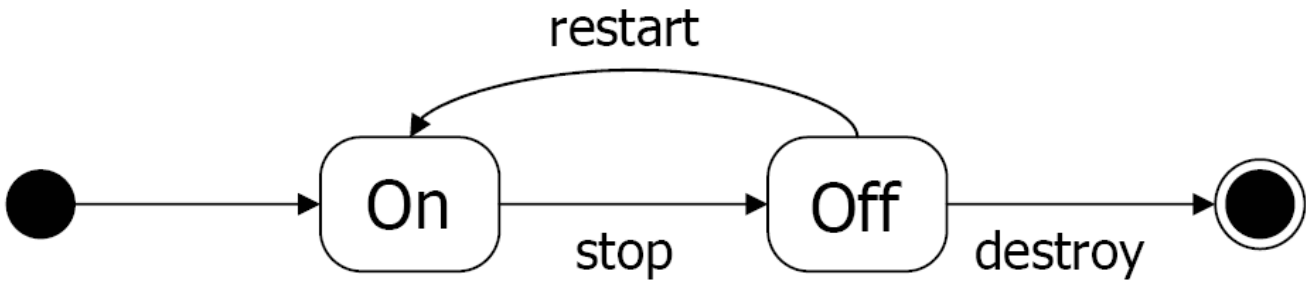
fabien.romeo@fromeo.fr
http://www.fromeo.fr

UML State Machines

- Un type de diagrammes de comportement
  - Behavioral state machines
  - Protocol state machines
- Peut être associé à un classifier UML (classe, interface, composant, use case,…)
- Issus des Statecharts de Harel
  - David Harel. Statecharts : A Visual Formalism for Complex Systems. Science of Computer Programming, 8(3) :231–274, June 1987.
- *statecharts = state-diagrams + depth + orthogonality + broadcast-communication*

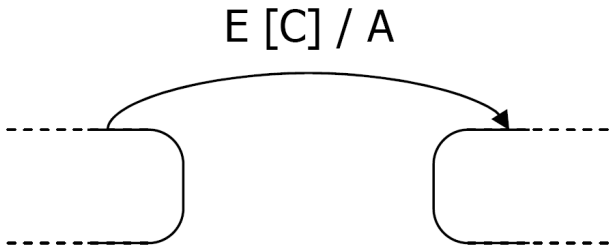
state-diagrams

Automate fini simple

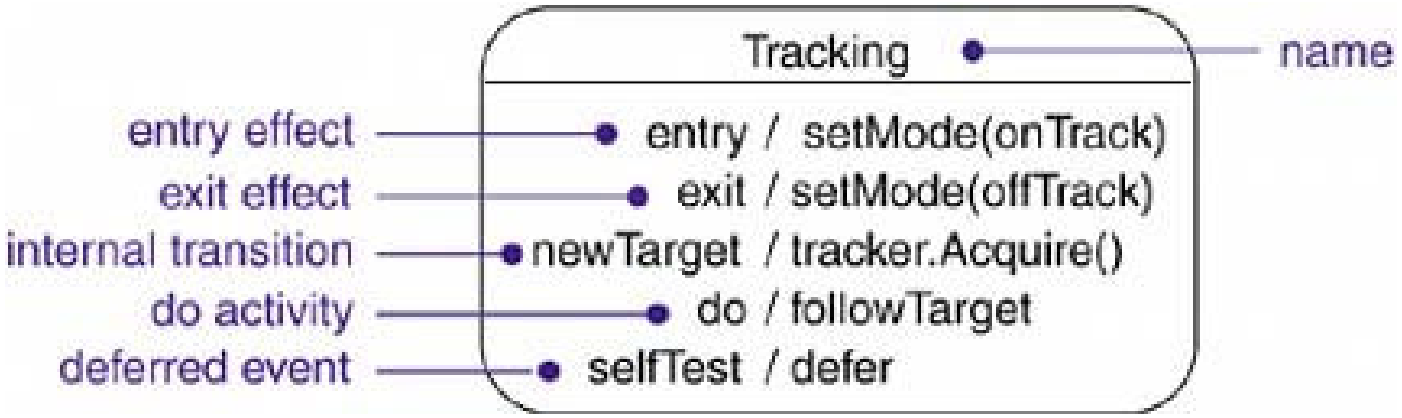


Transition

E : événement
C : condition
A : action



État, actions et activités

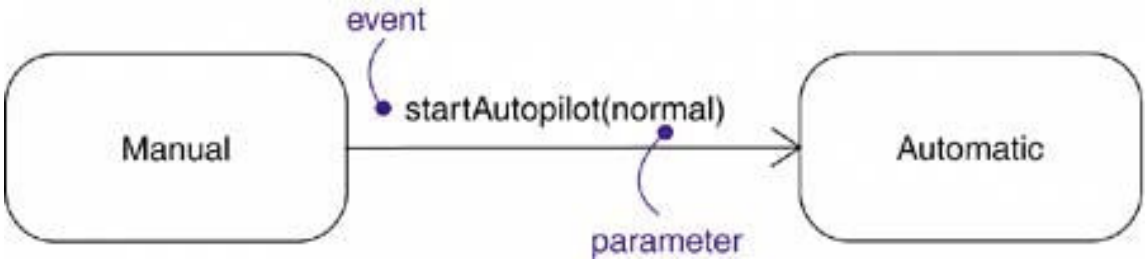


[BRJ2005]

Événements

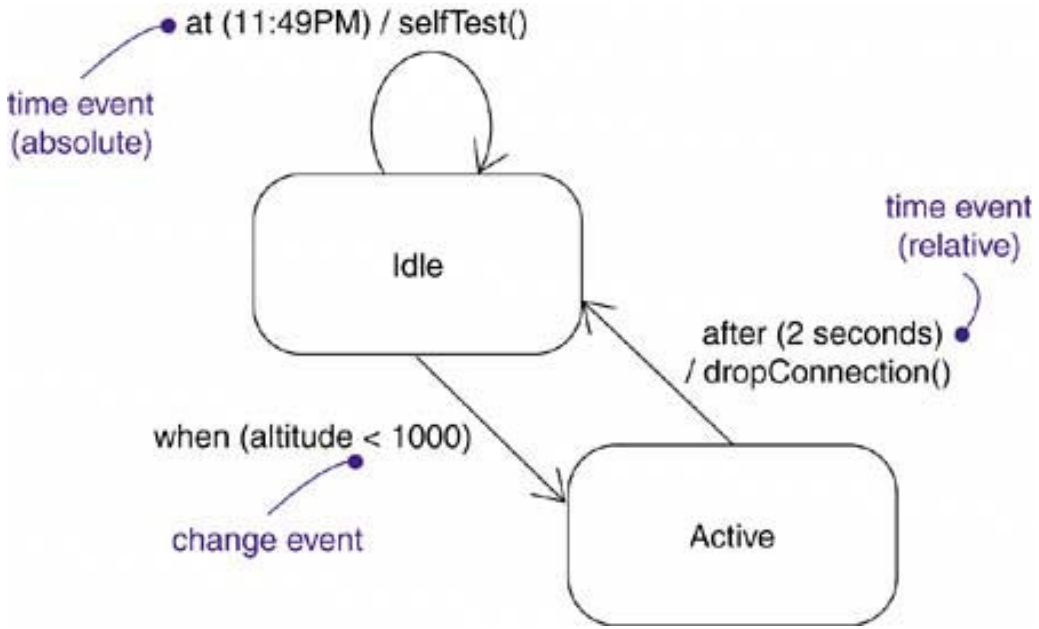
- Une transition est provoquée par un événement (une opération)
- La durée d’une transition est considérée comme nulle donc non interruptible
- 4 catégories d'événements
  - Les appels d'opérations
  - Les signaux (clic de souris, communication, etc.)
  - Les conditions (quand x>10)
  - Les délais (après 5 sec.)

Événements : exemples



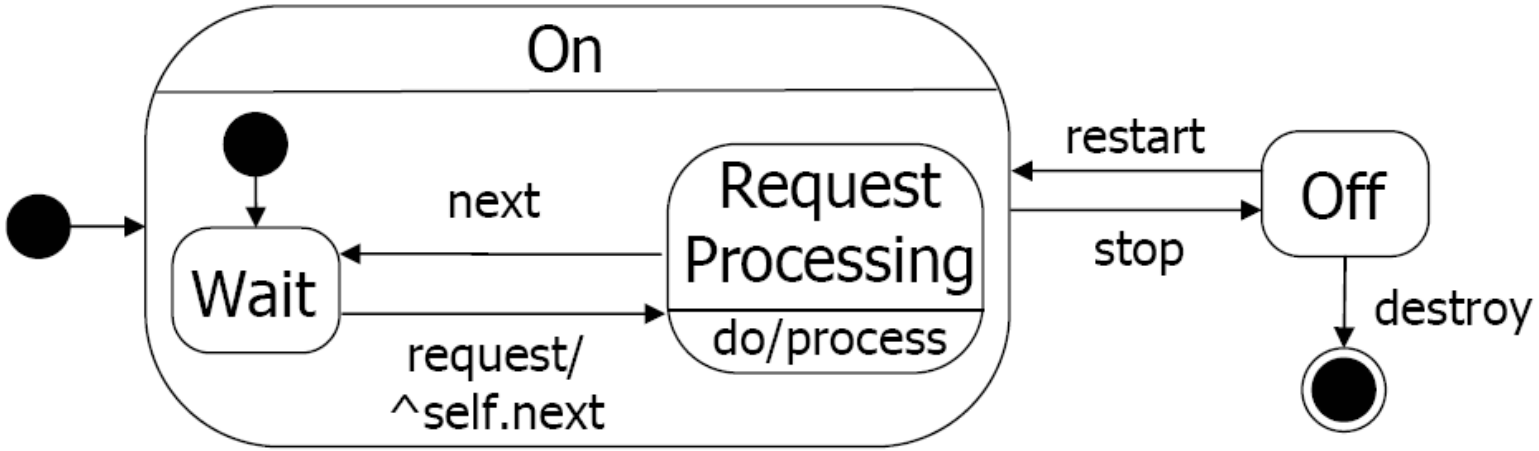
appels d'opérations

Conditions et délais



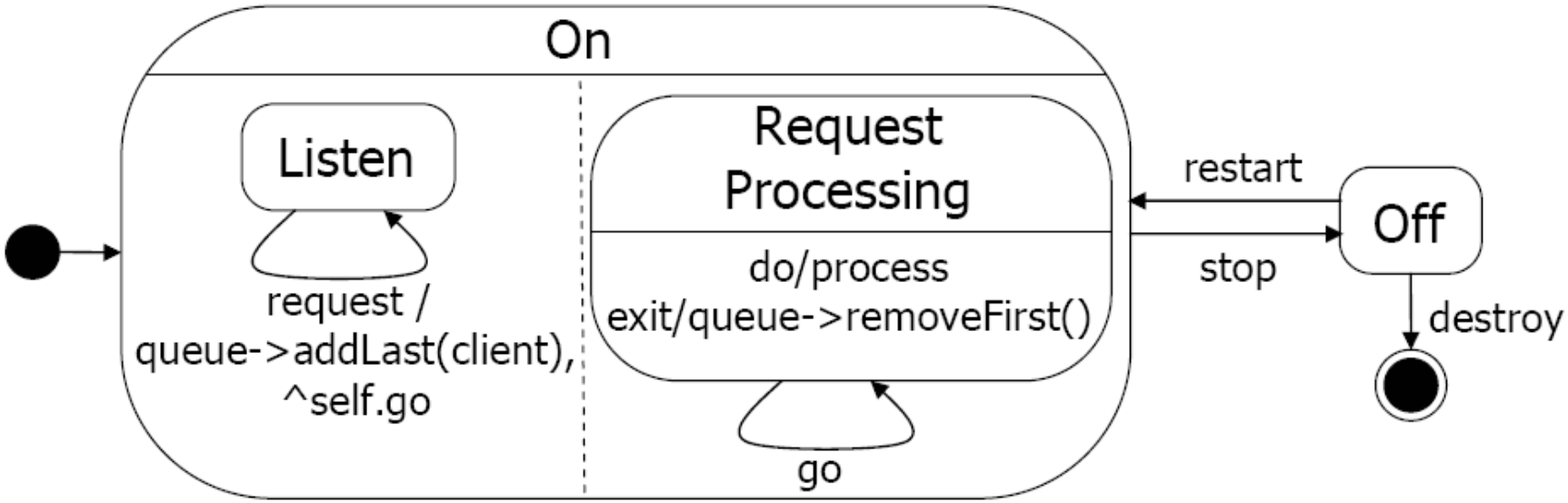
[BRJ2005]

+ *depth* (hiérarchie)



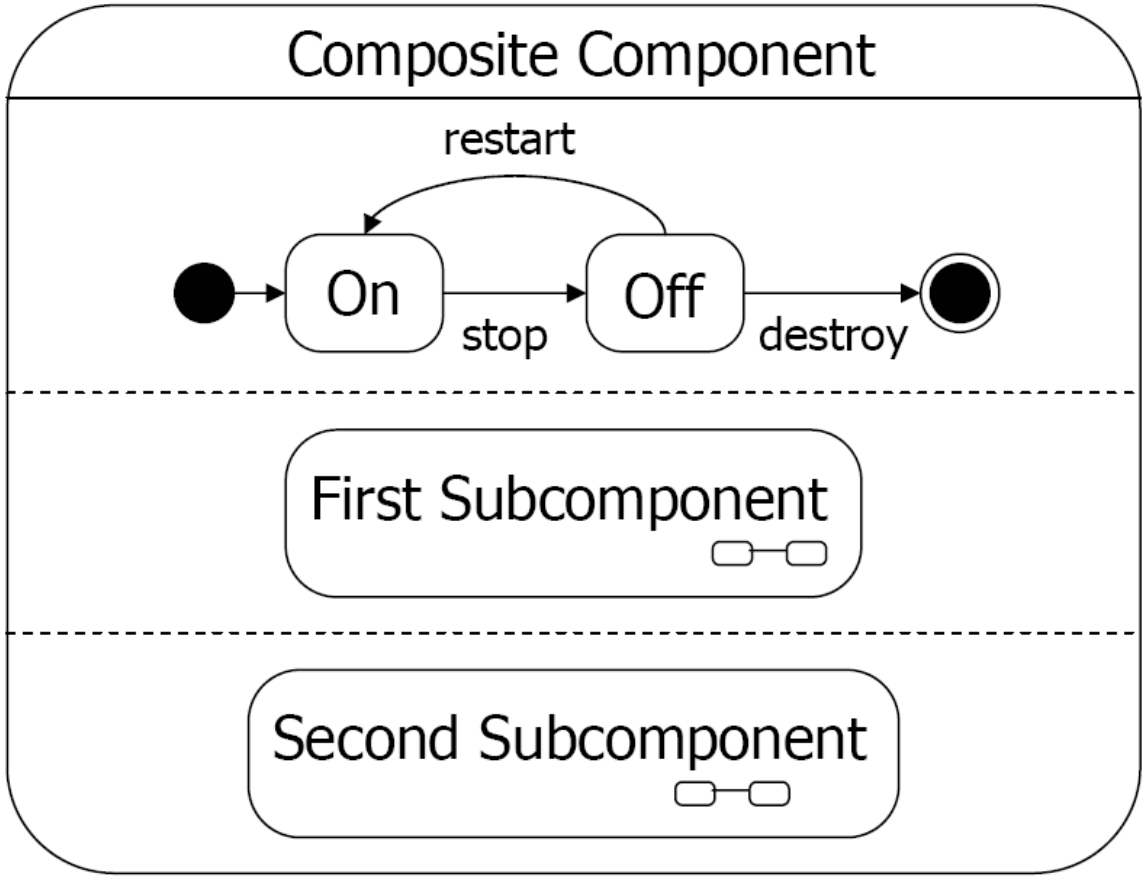
Décomposition en états mutuellements exclusifs (XOR)

+ orthogonality

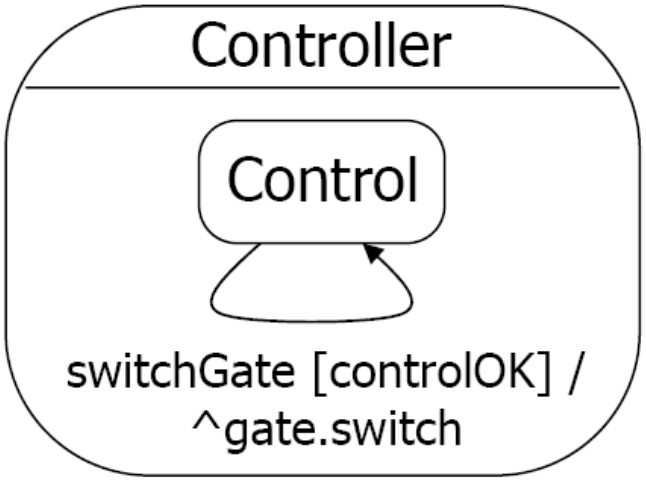


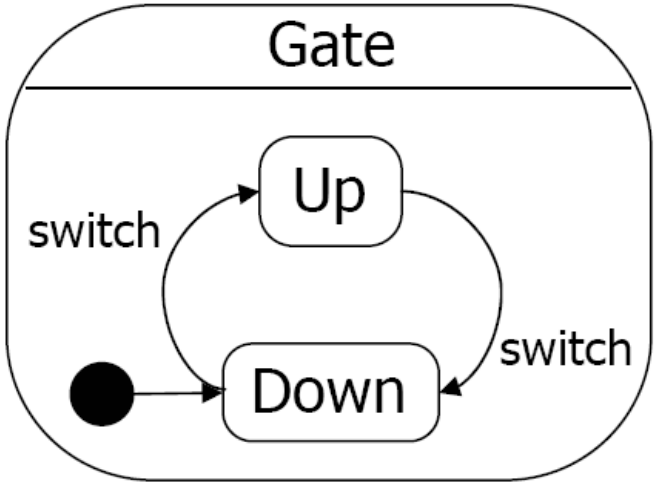
Décomposition en états concurrents (AND)

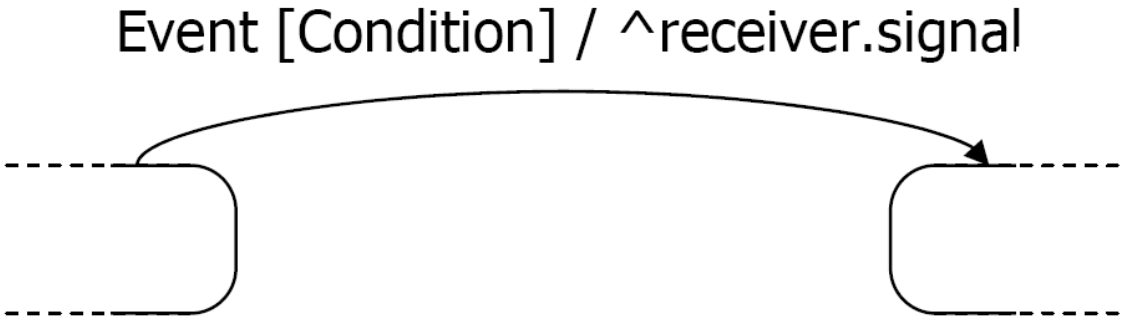
Composition de SM



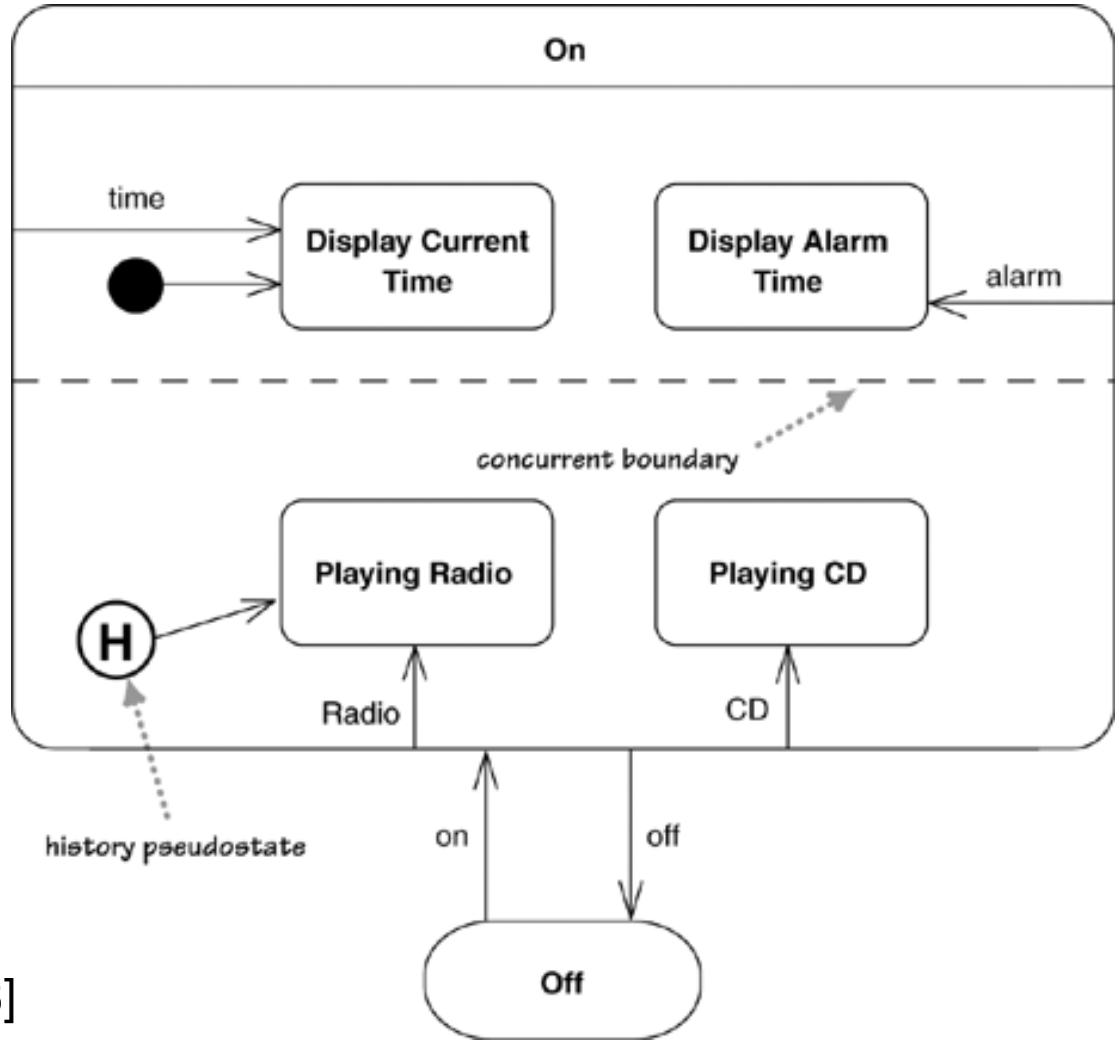
+ broadcast-communication





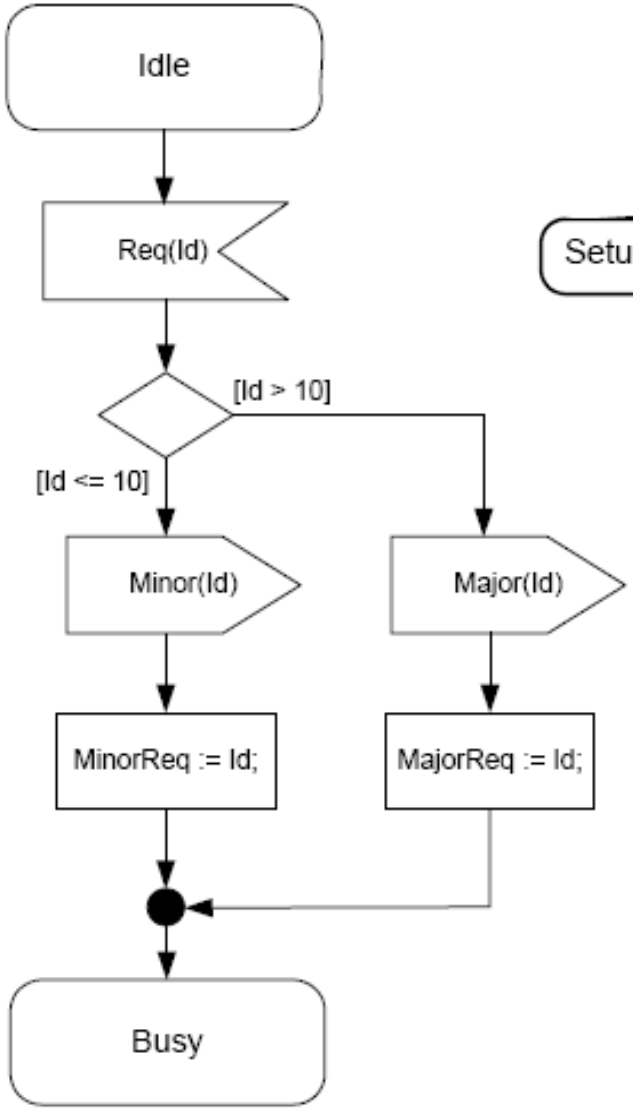


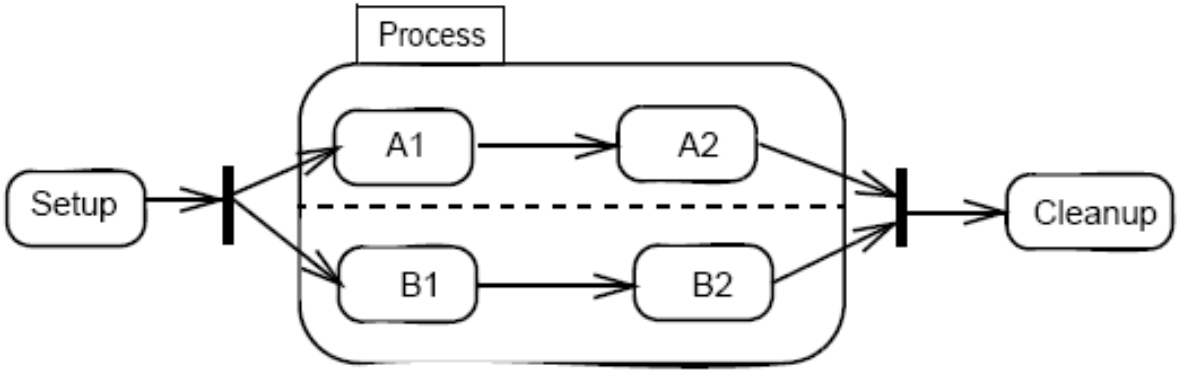
Historique (pseudo-state)



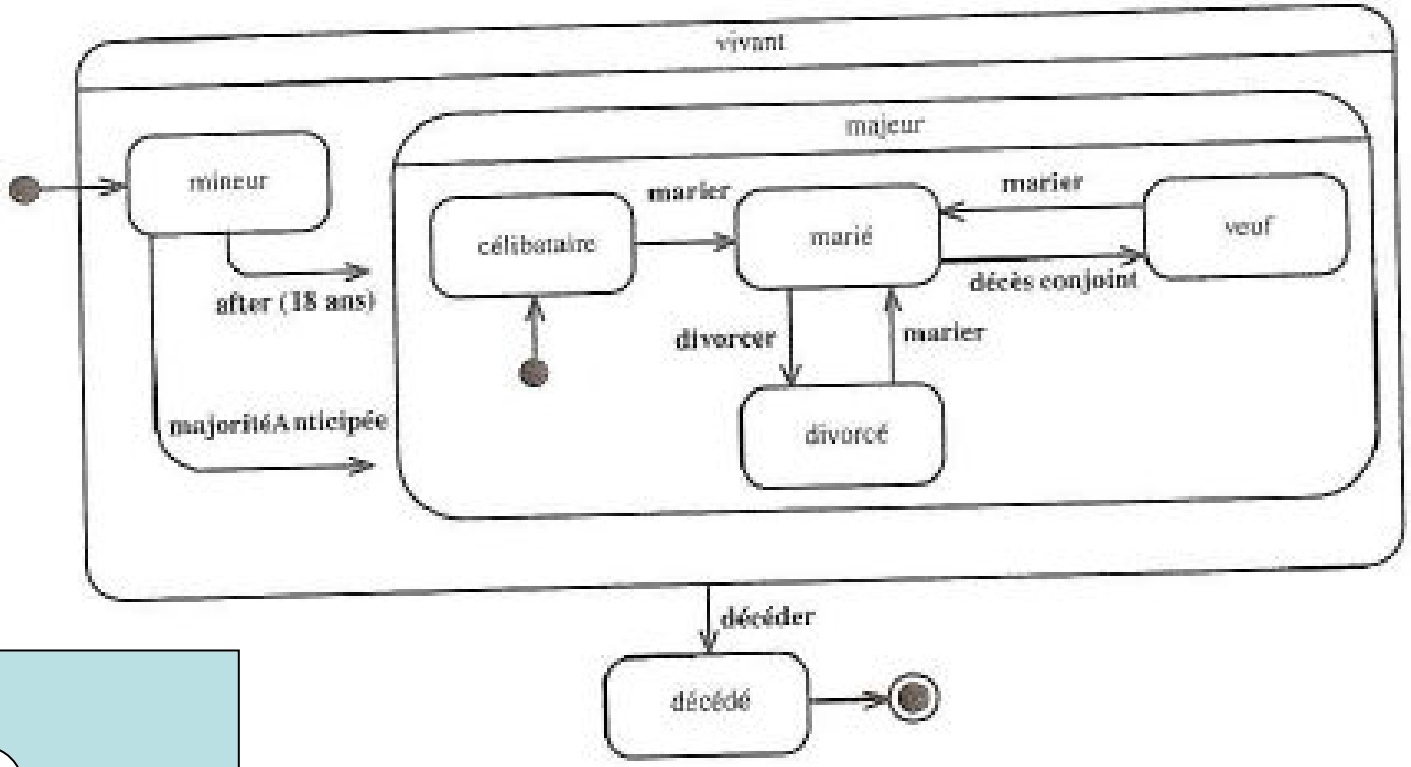
[Fowler2003]

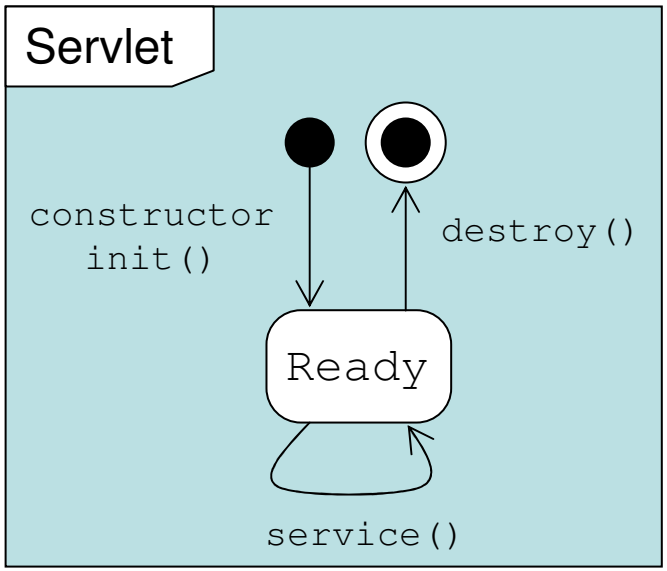
+ *activity diagrams*…(UML 2)



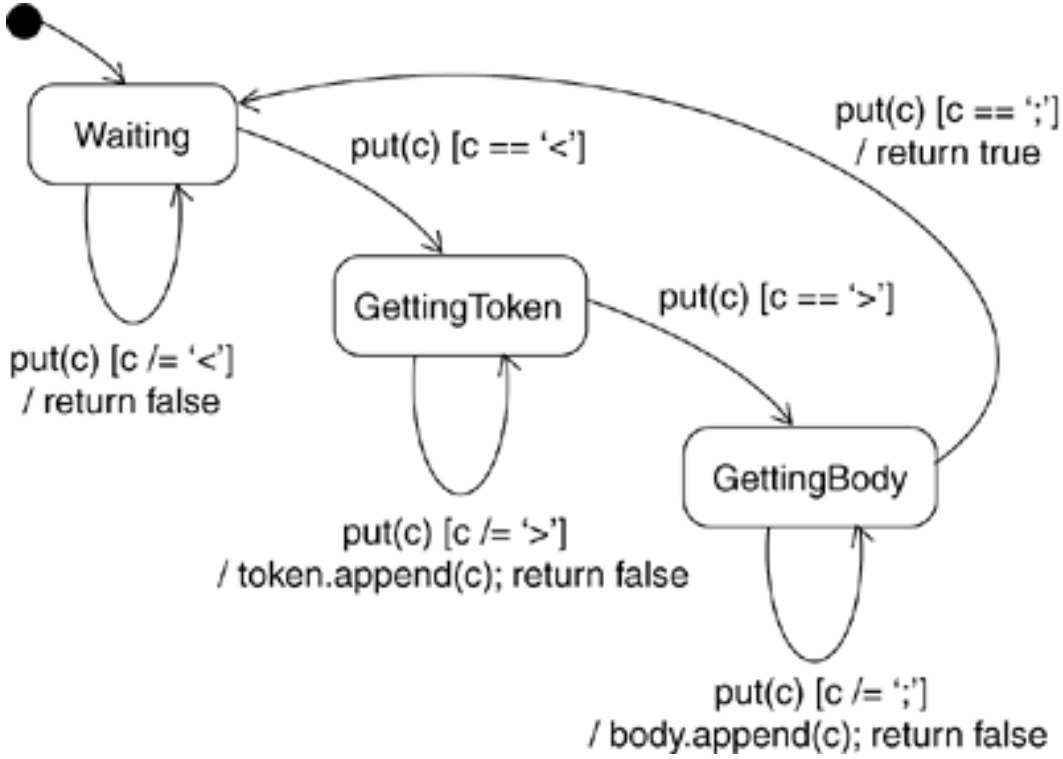


Modéliser le cycle de vie d'un objet





Modéliser des objets réactifs



[BRJ2005]

```
class MessageParser {
public
 public boolean put(char c) {
  switch (state) {
   case Waiting:
    if (c == '<') {
     state = GettingToken;
     token = new StringBuffer();
     body = new StringBuffer();
    }
    break;
   case GettingToken :
    if (c == '>') state = GettingBody;
    else token.append(c);
    break;
   case GettingBody :
    if (c == ';') state = Waiting;
    else body.append(c);
    return true;
  }
  return false;
 }
 public StringBuffer getToken() { return token; }
 public StringBuffer getBody() {  return body; }

 private final static int Waiting = 0;
 private final static int GettingToken = 1;
 private final static int GettingBody = 2;
 private int state = Waiting;
 private StringBuffer token, body;
}
```

Modéliser le comportement de systèmes complexes