Université Abdelhamid Mehri Constantine 2- Algérie
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique Fondamentale et ses Applications

1ère Année Master Réseaux et Systèmes Distribués

TP ALGORITHMES DISTRIBUÉS (ALDI)

TP N°4 : Implémentation d'un algorithme distribué en utilisant la plateforme JADE

Année universitaire : 2021/2022

Compétences à acquérir

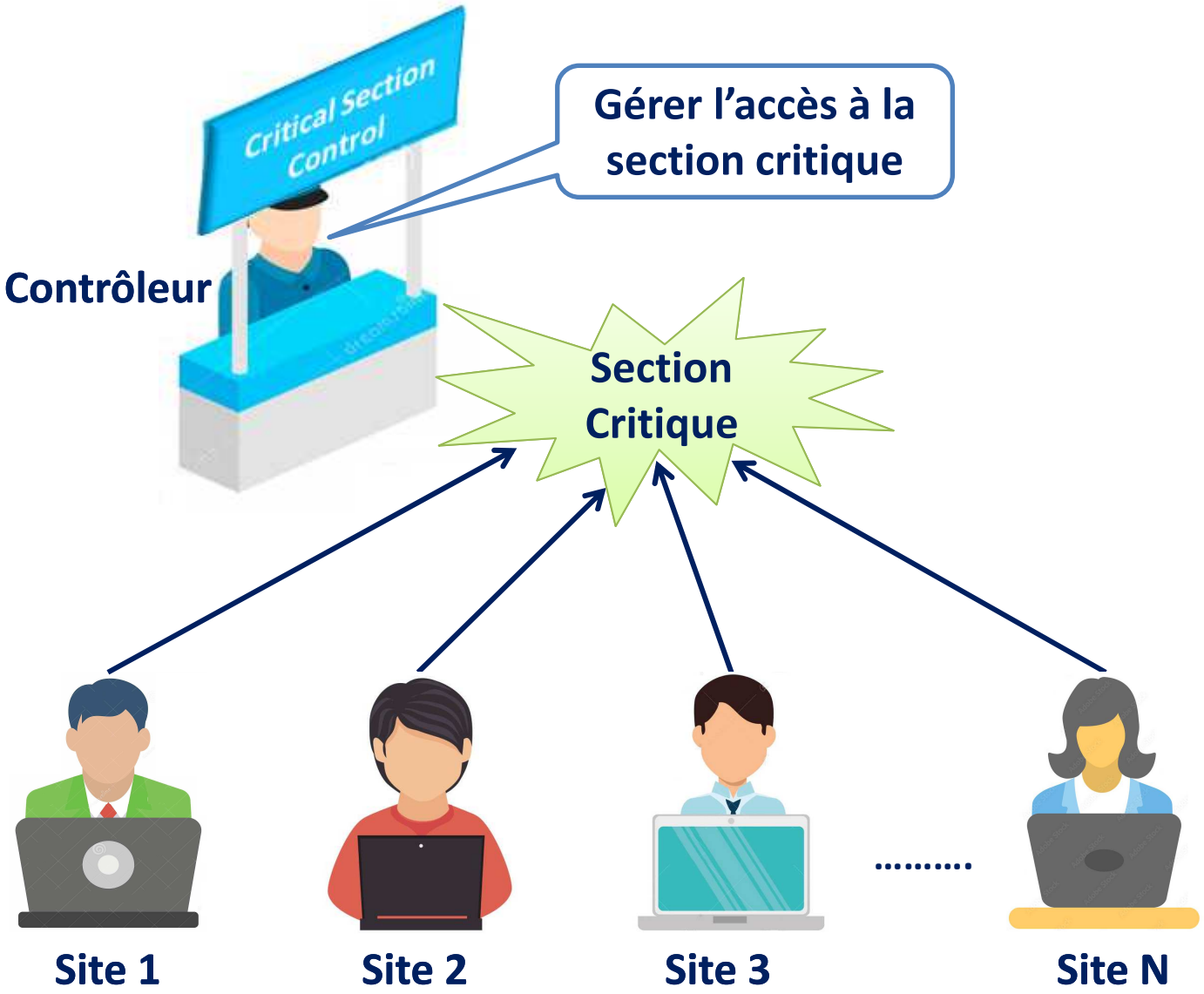
Suite à ce TP, vous serez en mesure de :

- Découvrir les étapes à suivre afin d'implémenter les algorithmes distribués en utilisant la plateforme JADE.
- Tester l'implémentation d'un algorithme distribué.
- Implémenter d'autres algorithmes distribués.

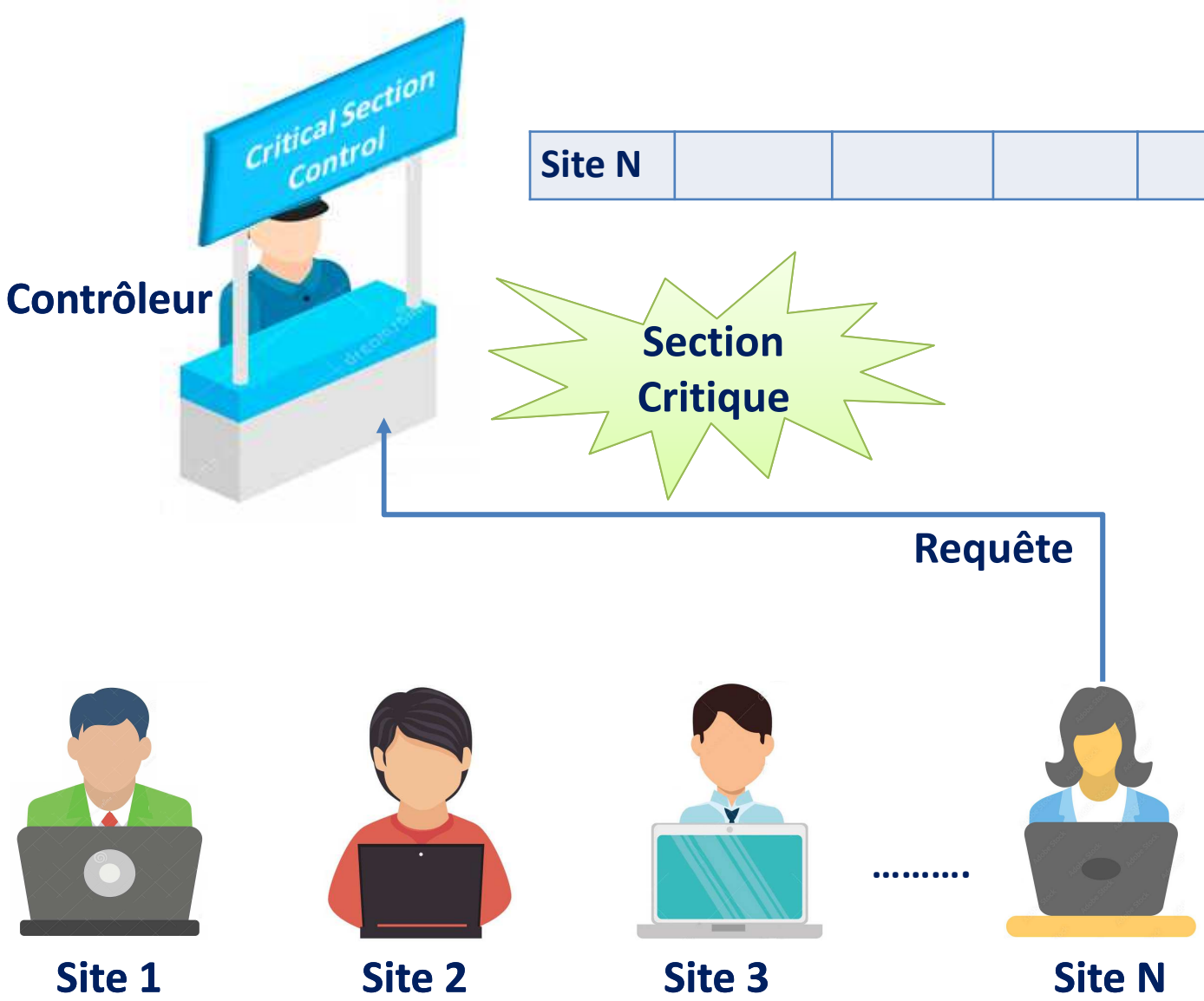
Démarche d'implémentation d'un algorithme distribué sous la plateforme JADE

1. Identifier les types d'agents qui seront utilisés dans l'algorithme distribué.
2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements.
3. Identifier les différents messages échangés entre les agents et donner leur contenu.
4. Identifier les arguments qui seront passés à chaque agent.
5. Implémenter la classe de chaque agent (la méthode setup et les différents comportements).
6. Tester les classes implémentées.
7. Analyser le contenu de la console après exécution, s'il y a des anomalies alors il faut revoir l'étape 5.

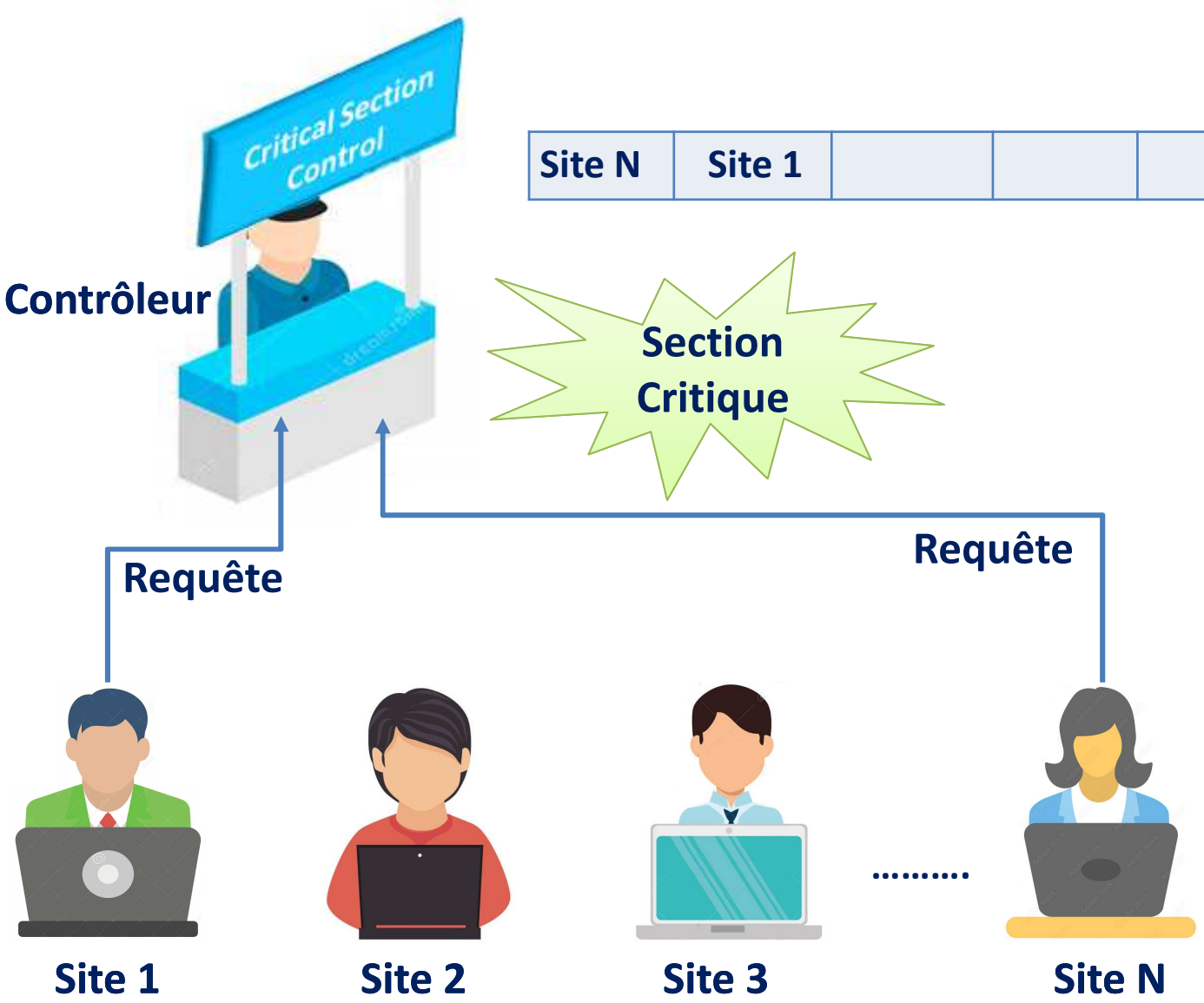
Algorithme à implémenter



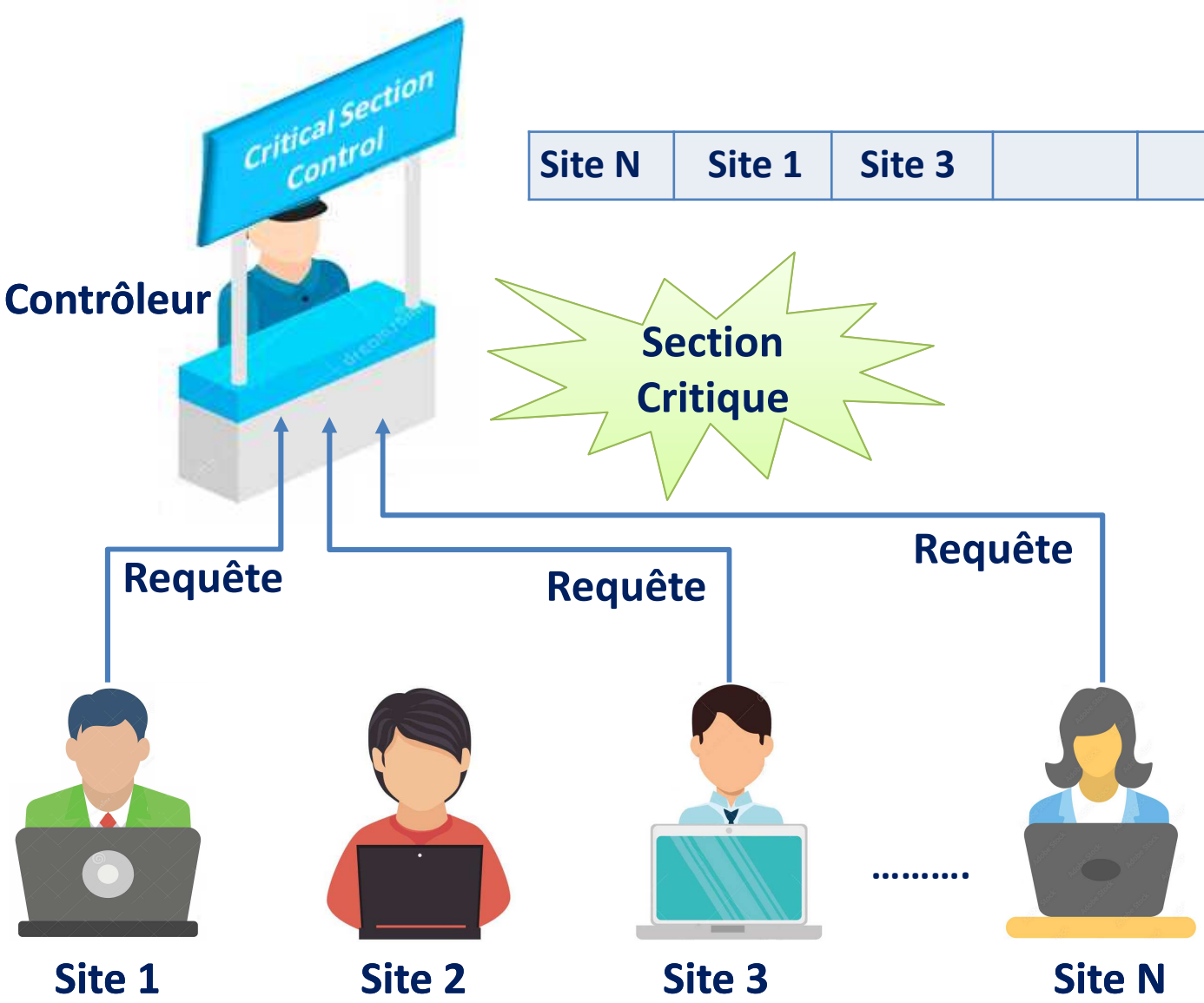
Algorithme à implémenter



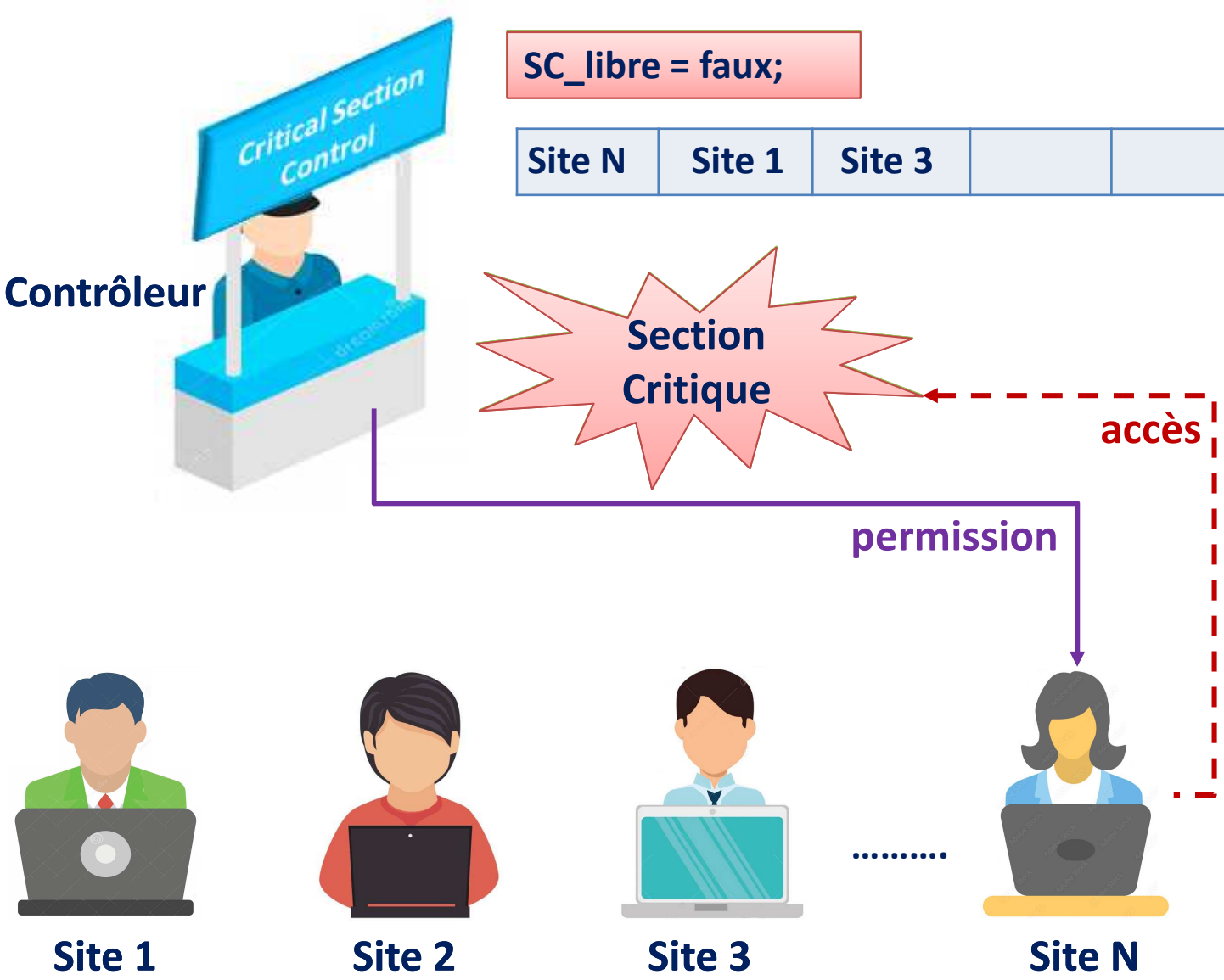
Algorithme à implémenter



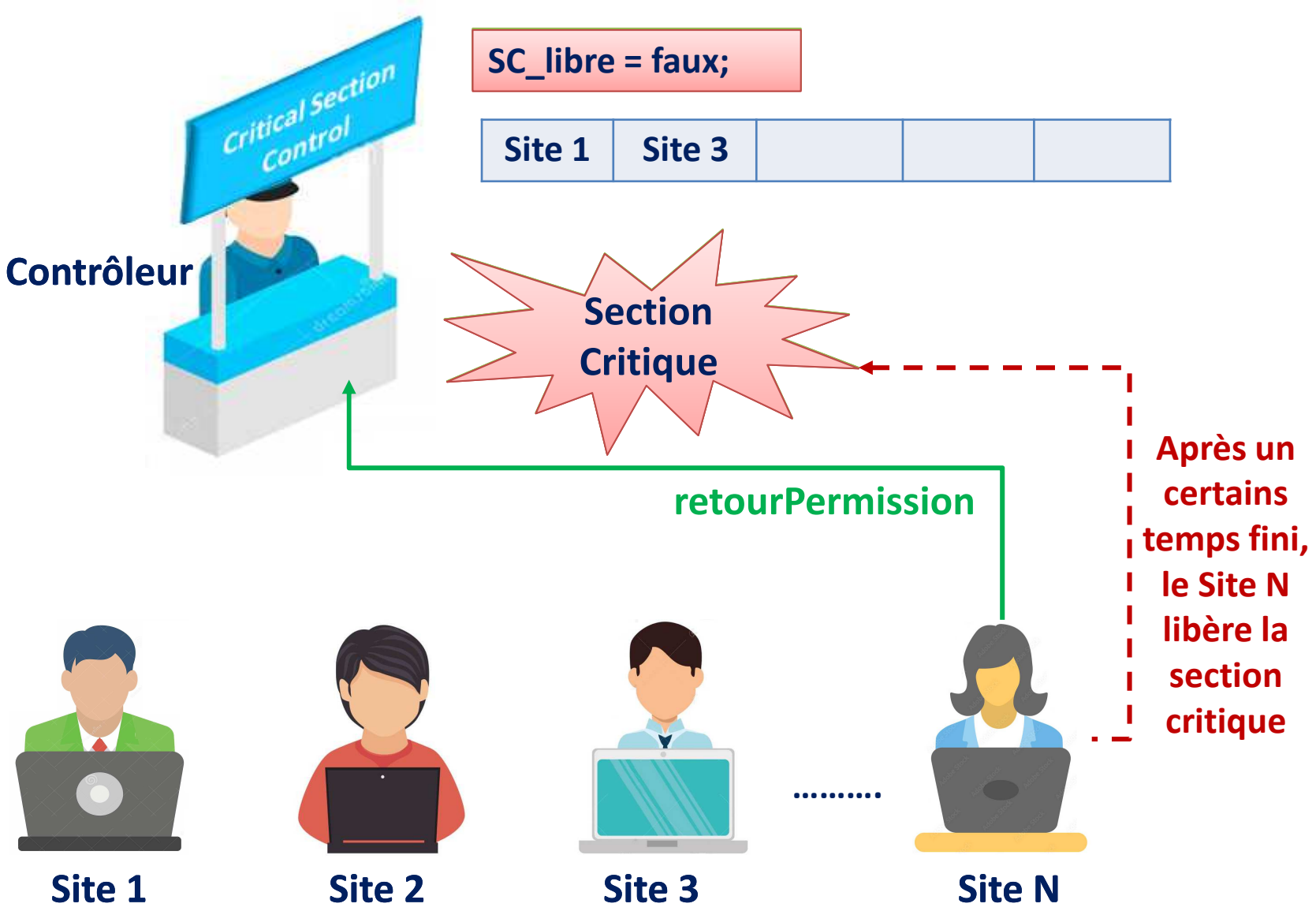
Algorithme à implémenter



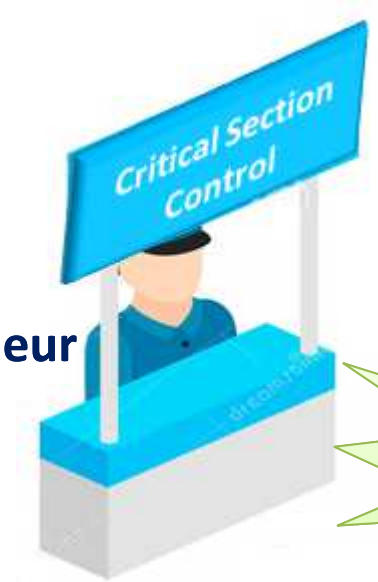
Algorithme à implémenter



Algorithme à implémenter



Algorithme à implémenter



Contrôleur

Variables du Contrôleur :

- **SC_libre : booléen initialisé à vrai;**
- **file_requêtes : tableau contenant les identités des sites qui ont envoyé des requêtes pour rentrer en section critique, il est initialisé à Ø;**

Section Critique

Site 1

Site 2

Site 3

……

Site N

Variables du site i :

- **étati = {dehors, demandeur, dedans} initialisé à dehors;**
- **nomContrôleur : Nom du site Contrôleur;**
- **permission_aquise : booléen initialisé à faux;**

Algorithme à implémenter

Algorithme exécuté par chaque site i :

Lors d'un appel à acquérir

étati = demandeur;

Envoyer (**requête(i)**) au **Contrôleur**;

Attendre (**permission_aquise == vrai)**;

étati = dedans;

Lors d'un appel à libérer

étati = dehors;

Envoyer (**retour_permission**) au **Contrôleur**;

permission_aquise = faux;

Lors de la réception de permission

permission_aquise = vrai;

Algorithme à implémenter

Algorithme exécuté par le contrôleur :

Lors de la réception de requête(j)
  Insérer **'j'** dans **file_requêtes**;
  Si (SC_libre == vrai) **alors** //Section critique est libre
    k = retirer le 1er élément de file_requêtes;
    Envoyer (**permission**) au site **k**;
    SC_libre = faux; //Section critique devient occupée
  Fin Si

Lors de la réception de retour_permision
  SC_libre = vrai; //**Section critique devient libre**
  Si (file_requetes est non vide) **alors**
    k = retirer le 1er élément de file_requêtes;
    Envoyer (**permission)** au site **k**;
    SC_libre = faux; //**Section critique devient occupée**
  Fin Si

Implémentation de l'algorithme

1. **Identifier les types d'agents qui seront utilisés dans l'algorithme distribué :**

   Dans cet algorithme, nous avons deux types d'agents :

   - Un agent (site) qui désire accéder à une section critique.
   - Un agent (contrôleur) qui gère l'accès à la section critique.

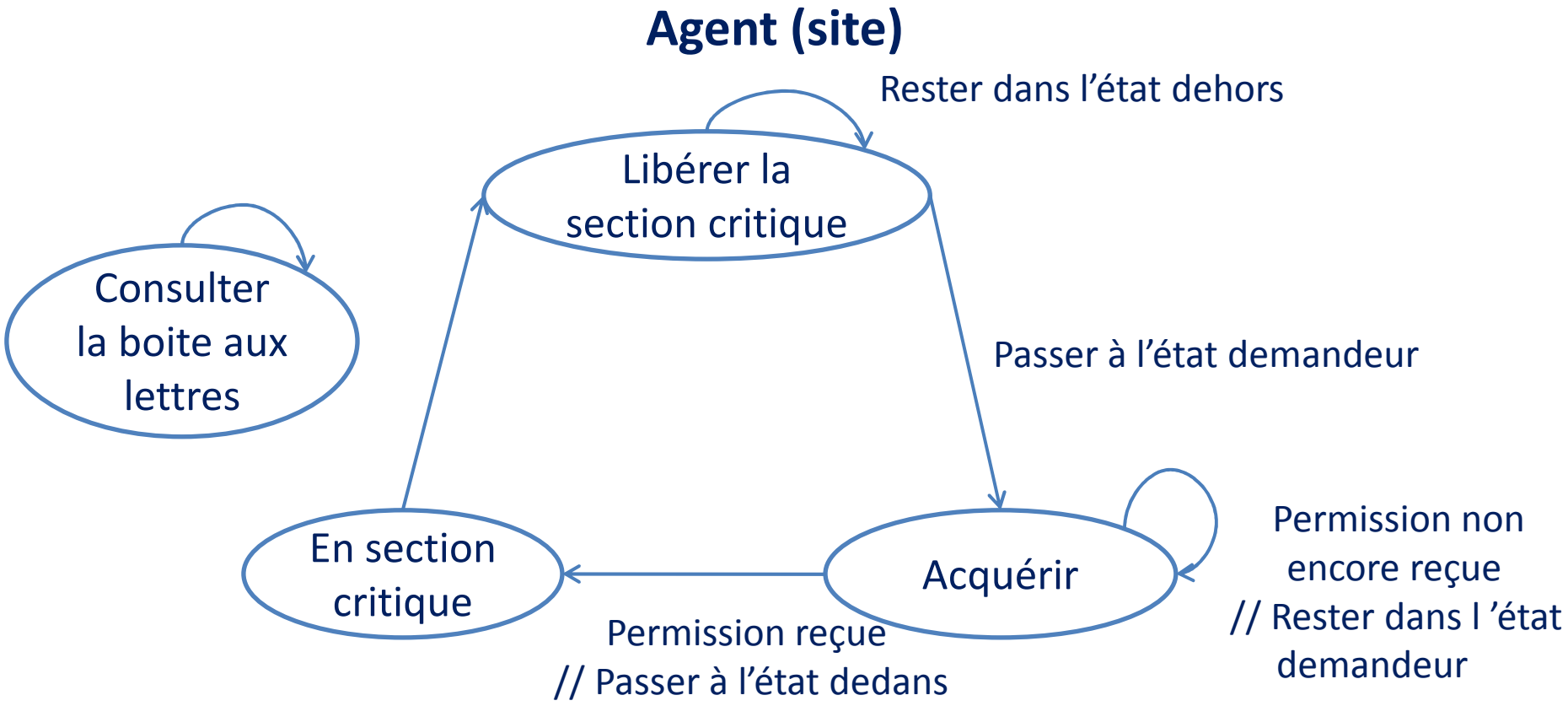
Implémentation de l'algorithme

2. **Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :**

Agent (site)	Agent (contrôleur)
• Acquérir (demandeur) • En section critique (dedans) • Libérer la section critique (dehors) • Consulter la boite aux lettres	• Consulter la boite aux lettres

Implémentation de l'algorithme

2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :



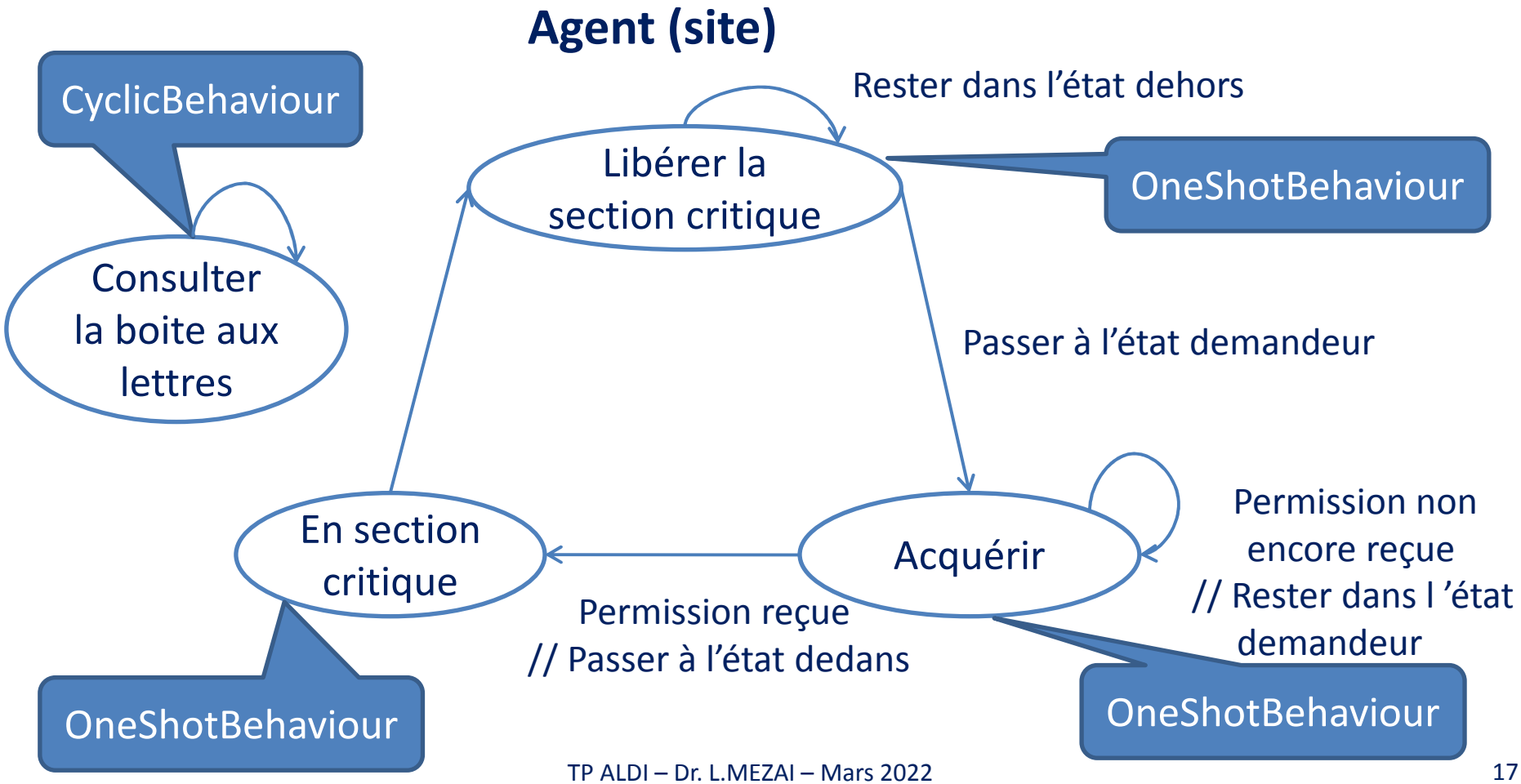
Implémentation de l'algorithme

2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :

Agent (contrôleur)

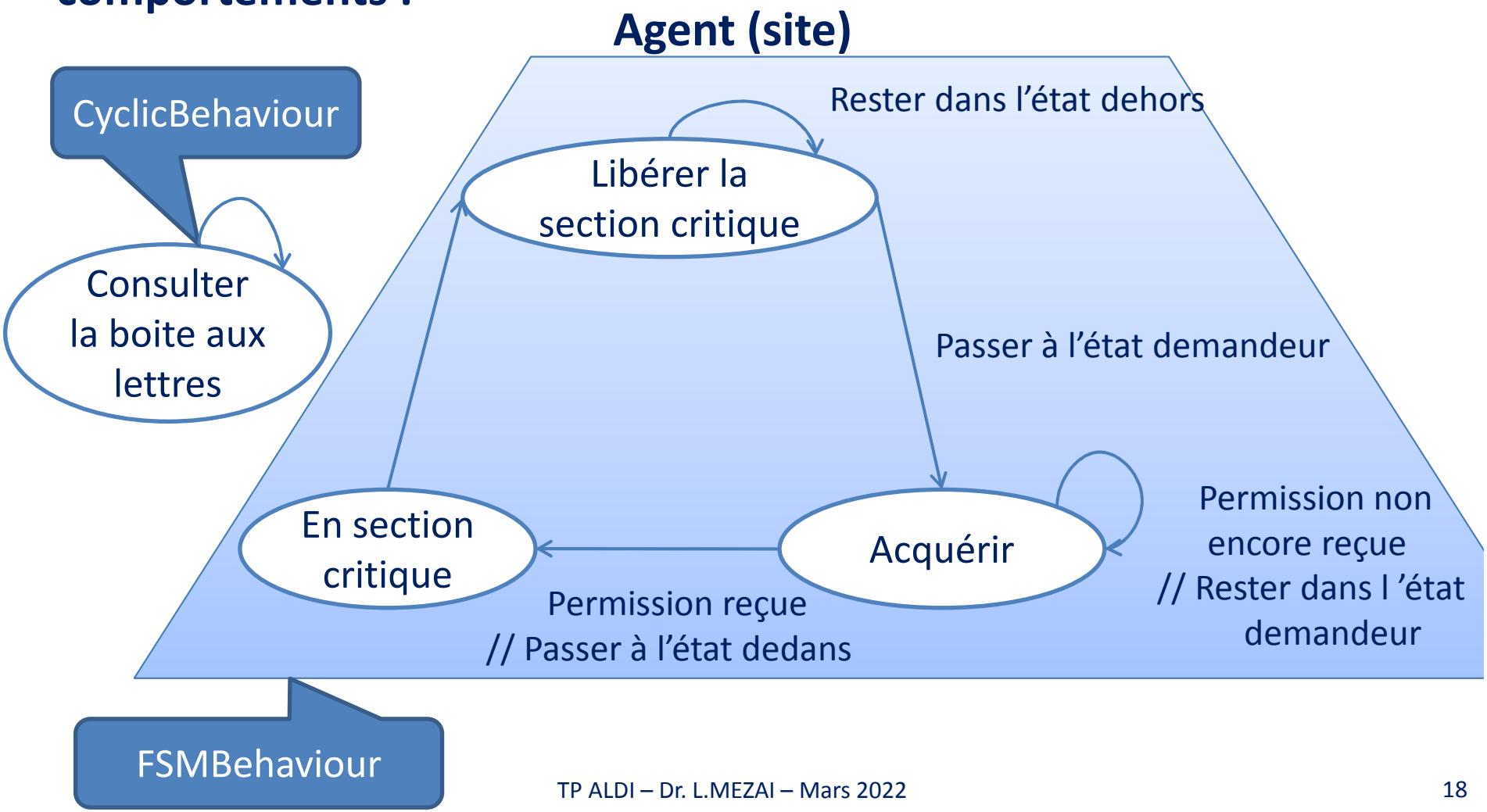
Implémentation de l'algorithme

2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :



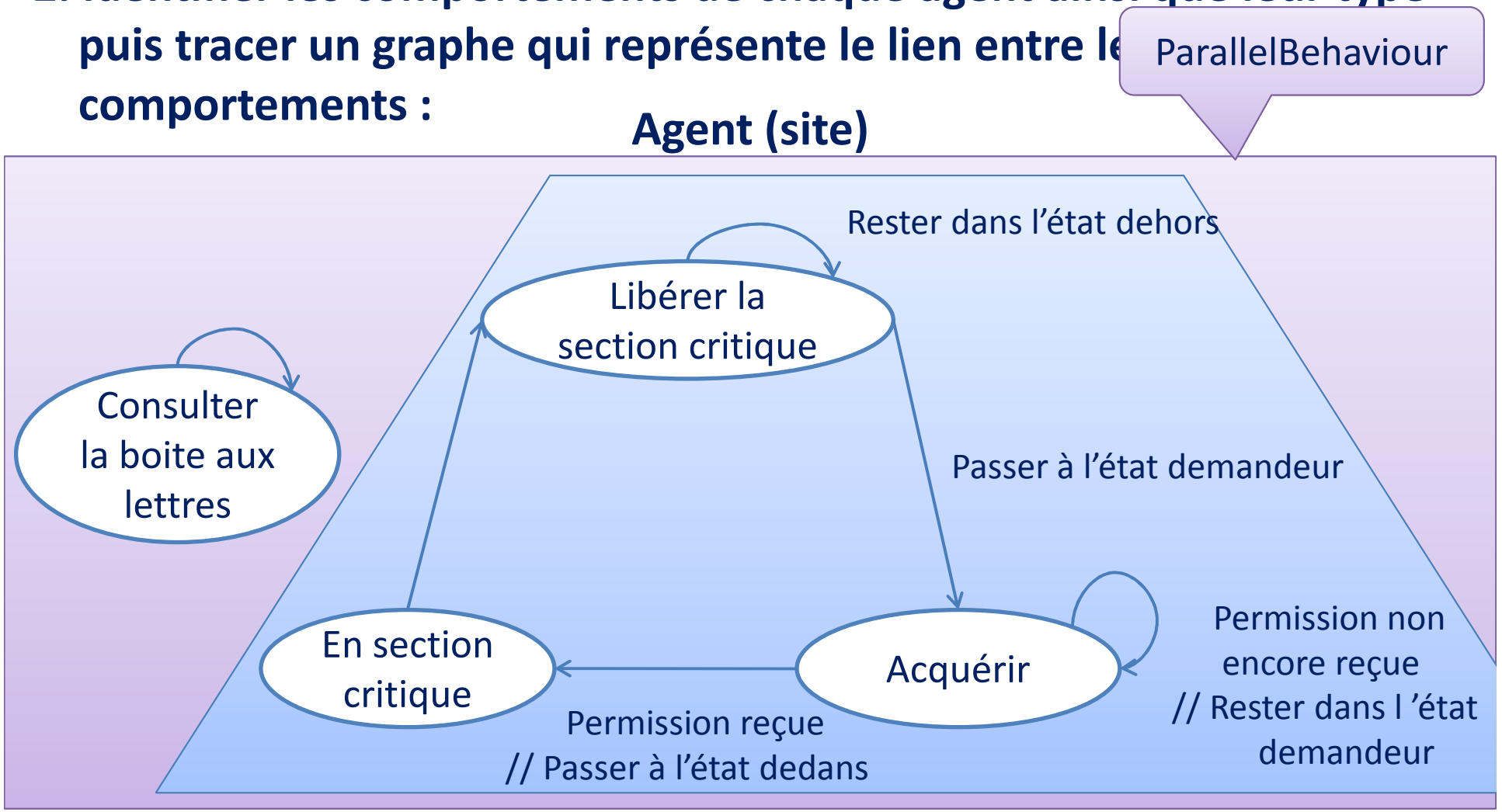
Implémentation de l'algorithme

2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :



Implémentation de l'algorithme

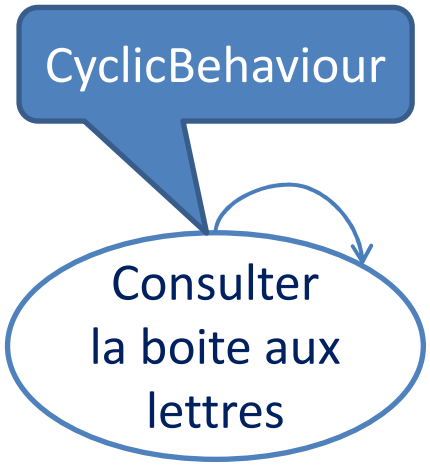
2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre le comportements :



Démarche d'implémentation

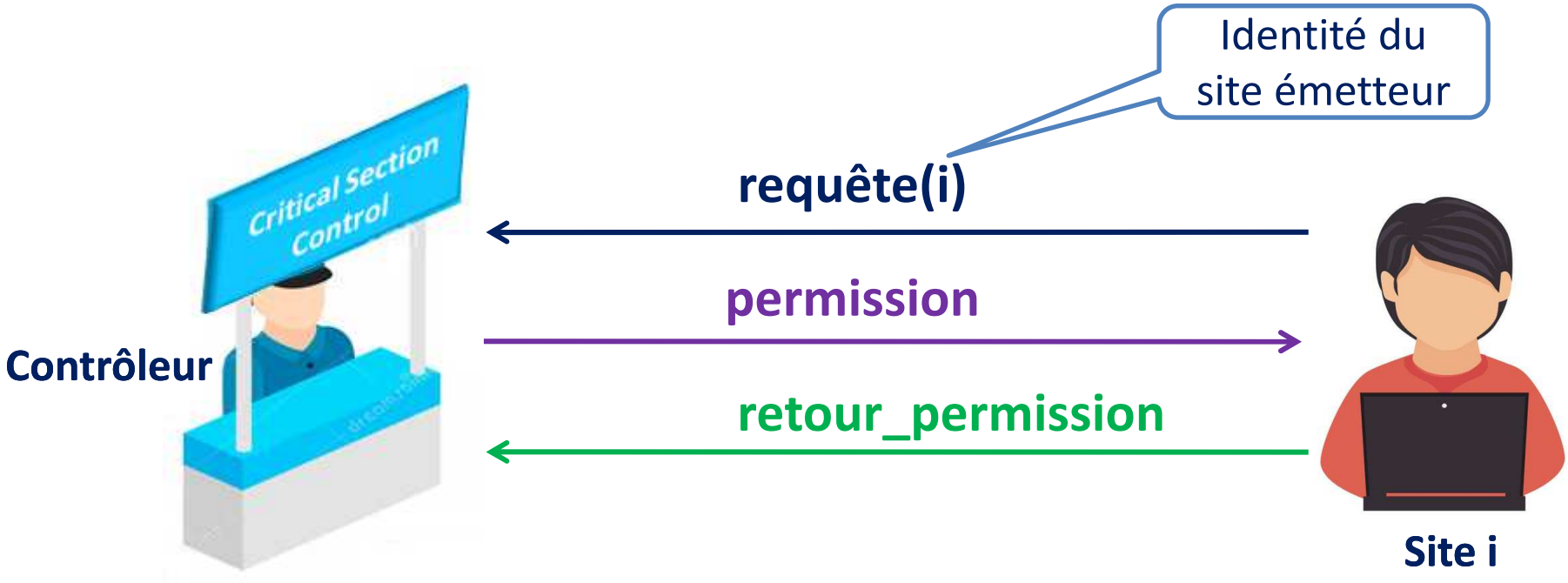
2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :

Agent (contrôleur)



Implémentation de l'algorithme

3. Identifier les différents messages échangés entre les agents et donner leur contenu :



Implémentation de l'algorithme

4. Identifier les arguments qui seront passés à chaque agent :

- Chaque agent (site) doit connaitre :

  ✓le nom de l'agent contrôleur

  ⇒La liste des arguments sera la forme suivante :

  (nom_Contrôleur)

- L'agent contrôleur n'a pas besoin des paramètres.

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent (la méthode setup et les différents comportements)

- Nous allons implémenter 2 classes :
  1. **Classe Site**
  2. **Classe Contrôleur**

Implémentation de l'algorithme

5.1. Implémenter la classe Site

Algorithme exécuté par chaque site i :

Variables du site i :

- étati = {dehors, demandeur, dedans} initialisé à dehors;
- nomControleur : Nom du site Contrôleur;
- permission_aquise : booléen initialisé à faux;

```
public class Site extends Agent{
  String etat = "dehors";
  String nomControleur;
  boolean permission_acquise = false;
```

Implémentation de l'algorithme

5.1. Implémenter la classe Site - méthode setup() :

```
public void setup(){
   System.out.println("Agent " + getLocalName());
   Object [] args = getArguments();
   if (args != null)
      nomControleur = args[0].toString();
   FSMBehaviour fsm = new FSMBehaviour(this);
   fsm.registerFirstState(new liberer(), "dehors");
   fsm.registerState(new acquerir(), "demandeur");
   fsm.registerState(new enSC(), "dedans");
```

Implémentation de l'algorithme

5.1. Implémenter la classe Site - méthode setup() :

```
fsm.registerTransition("dehors", "dehors", 0);
fsm.registerTransition("dehors", "demandeur", 1);
fsm.registerTransition("demandeur", "demandeur", 0);
fsm.registerTransition("demandeur", "dedans", 1);
fsm.registerDefaultTransition("dedans", "dehors");

ParallelBehaviour parallel = new
ParallelBehaviour(ParallelBehaviour.WHEN_ALL);
parallel.addSubBehaviour(fsm);
parallel.addSubBehaviour(new consulterBoite());

addBehaviour(parallel);
}//fin de la méthode setup
```

Implémentation de l'algorithme

5.1. Implémenter la classe Site - les comportements :

```
public class acquerir extends OneShotBehaviour {
  int valTransition;
  public void action(){
    if (etat.equals("dehors")) {
      etat = "demandeur";
      ACLMessage msgEnvoi = new ACLMessage(ACLMessage.INFORM);
      msgEnvoi.addReceiver(new AID(nomControleur,AID.ISLOCALNAME));
      msgEnvoi.setContent("requete");
      send(msgEnvoi);
      valTransition = 0;
    }//if
      if (permission_acquise == true) {
        valTransition = 1;
        etat = "dedans";
      } //if
  } //action
  public int onEnd(){
     return valTransition;
  }//onEnd
```

Lors d'un appel à acquérir

étati = demandeur;

Envoyer (**requête(i)**) au **Contrôleur**;

Attendre (**permission_aquise == vrai**);

étati = dedans;

Implémentation de l'algorithme

5.1. Implémenter la classe Site - les comportements :

```
public class liberer extends OneShotBehaviour {
  public void action(){
    if (etat.equals("dedans")){
      etat = "dehors";
      ACLMessage msgEnvoi  = new ACLMessage (ACLMessage.INFORM);
      msgEnvoi.addReceiver(new AID(nomControleur,AID.ISLOCALNAME));
      msgEnvoi.setContent("retourPermission");
      send(msgEnvoi);
      permission_acquise = false;
    } //if
  } //action
  public int onEnd(){
    int valTransition = (int) (Math.random() * 2);
    //0 rester dans l'état dehors
    //1 passer à l'état demandeur
    return valTransition;
  }//onEnd
} //liberer
```

Lors d'un appel à libérer
étati = dehors;
Envoyer (**retour_permission**) au **Contrôleur**;
permission_aquise = faux;

Implémentation de l'algorithme

5.1. Implémenter la classe Site - les comportements :

```
public class enSC extends OneShotBehaviour {
  public void action(){
    System.out.println("Agent " + getLocalName() + " je suis en SC");
    block((int) (Math.random() * 1000)); //rester en section critique
                                         // pendant une durée finie
  }
}
```

Implémentation de l'algorithme

5.1. Implémenter la classe Site - les comportements :

```
public class consulterBoite extends CyclicBehaviour {
  public void action(){
    ACLMessage msgRecu = receive();
    if (msgRecu != null){
      String msgContenu = msgRecu.getContent();
      if (msgContenu.equals("permission")){
        permission_acquise = true;
      }//if
    }//if
  }//action
}//consulterBoite
```

Lors de la réception de permission

permission_aquise = vrai;

Implémentation de l'algorithme

5.2. Implémenter la classe Controleur

Algorithme exécuté par le contrôleur :

Variables du site i :

- SC_libre : booléen initialisé à vrai;
- file_requetes : tableau contenant les identités des sites qui ont envoyé des requêtes pour rentrer en section critique, il est initialisé à Ø;

```
public class Controleur extends Agent{
    boolean SC_libre = true;
    ArrayList file_requetes = new ArrayList();
```

Implémentation de l'algorithme

5.2. Implémenter la classe Controleur - méthode setup() :

```
public void setup(){
   System.out.println("Agent " + getLocalName());
   addBehaviour(new consulterBoite());
}//setup
```

Implémentation de l'algorithme

5.2. Implémenter la classe Controleur - les comportements :

```
public class consulterBoite extends CyclicBehaviour {
  public void action(){
    ACLMessage msgRecu = receive();
    if (msgRecu != null){
      String msgContenu = msgRecu.getContent();
      if (msgContenu.equals("requete")){
        file_requetes.add(msgRecu.getSender().getLocalName());
        if (SC_libre == true){// SC libre
          String nomSite = (String)file_requetes.get(0);
          file_requetes.remove(0);
          ACLMessage msgEnvoi  = new ACLMessage (ACLMessage.INFORM);
          msgEnvoi.addReceiver(new AID(nomSite,AID.ISLOCALNAME));
          msgEnvoi.setContent("permission");
          send(msgEnvoi);
          SC_libre = false;
        } //if SC_libre
      }//if requete
```

Lors de la réception de requête(j)
Insérer 'j' dans **file_requêtes**;
Si (SC_libre == vrai) **alors**
k = retirer le 1er élément de file_requêtes;
Envoyer (**permission**) au site **k**;
SC_libre = faux;
Fin Si

Implémentation de l'algorithme

5.2. Implémenter la classe Controleur - les comportements :

```
        else{
         if (msgContenu.equals("retourPermission")){
          SC_libre = true;
          if (file_requetes.size() != 0){// file des requetes non vide
           String nomSite = (String)file_requetes.get(0);
           file_requetes.remove(0);
           ACLMessage msgEnvoi  = new ACLMessage (ACLMessage.INFORM);
           msgEnvoi.addReceiver(new AID(nomSite,AID.ISLOCALNAME));
           msgEnvoi.setContent("permission");
           send(msgEnvoi);
           SC_libre = false;
          }//if file_requetes
         }//if retourPermission
        }// else
      } // if msgRecu!=null
    } // action
  } //consulter boite
} // Controleur
```

Lors de la réception de retour_permision

SC_libre = vrai;

Si (file_requetes est non vide) **alors**

k = retirer le 1er élément de file_requêtes;

Envoyer (**permission**) au site **k**;

SC_libre = faux;

Implémentation de l'algorithme

6. Tester les classes implémentées (faire une exécution avec un seul contrôleur et 3 sites).
7. Analyser le contenu de la console après exécution, s'il y a des anomalies alors il faut revoir l'étape 5.