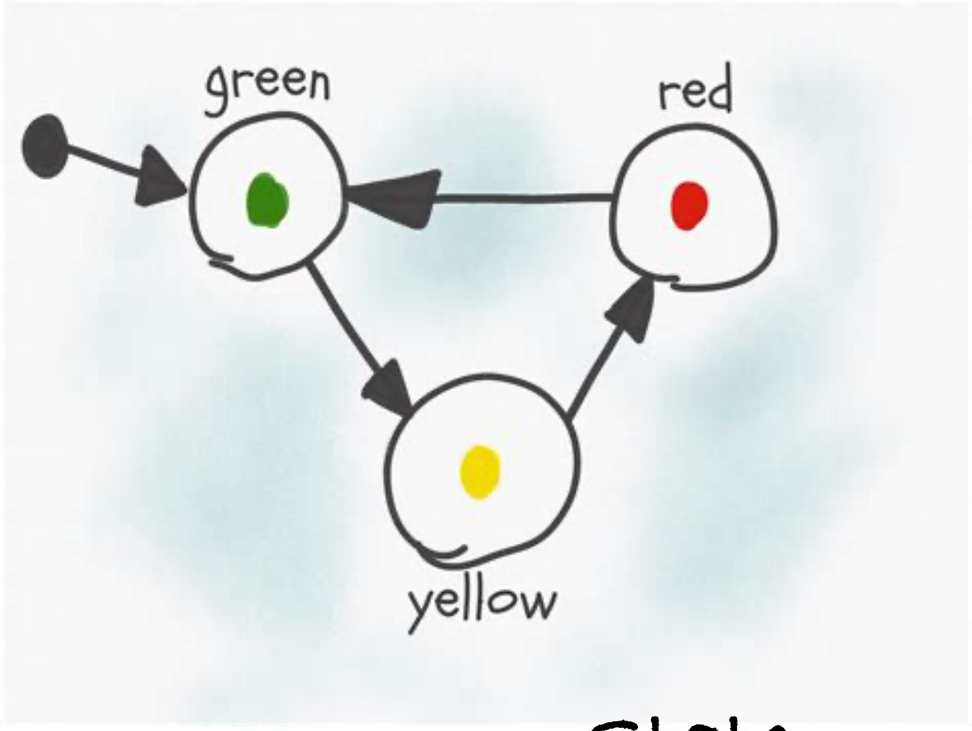


State

https://refactoring.guru/design-patterns/state

Tema 3: Disseny

Anna Puig

Enginyeria Informàtica

Facultat de Matemàtiques i Informàtica,

Universitat de Barcelona

Curs 2021/2022

UNIVERSITAT DE BARCELONA

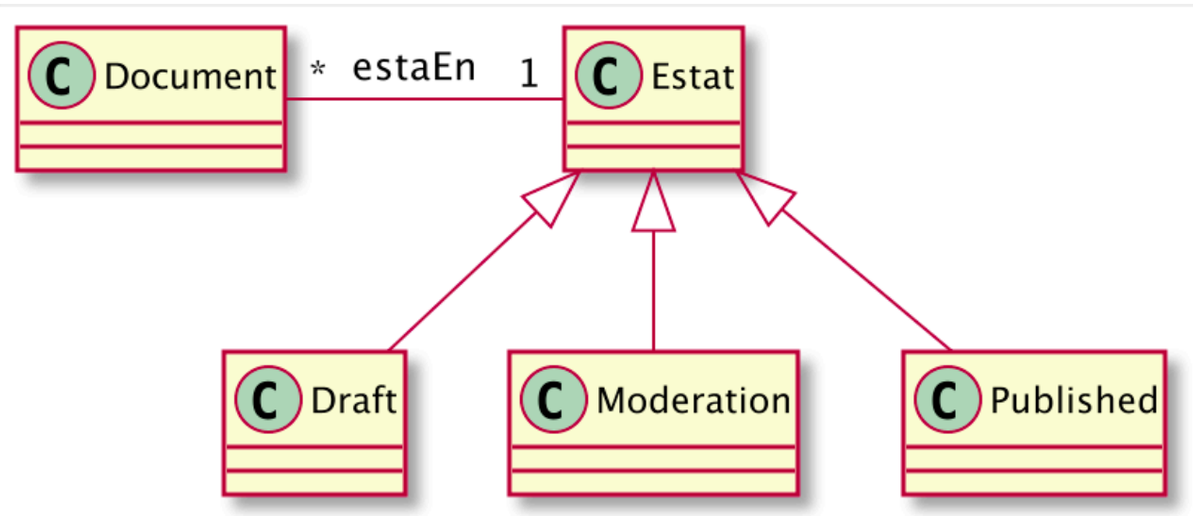
Temari

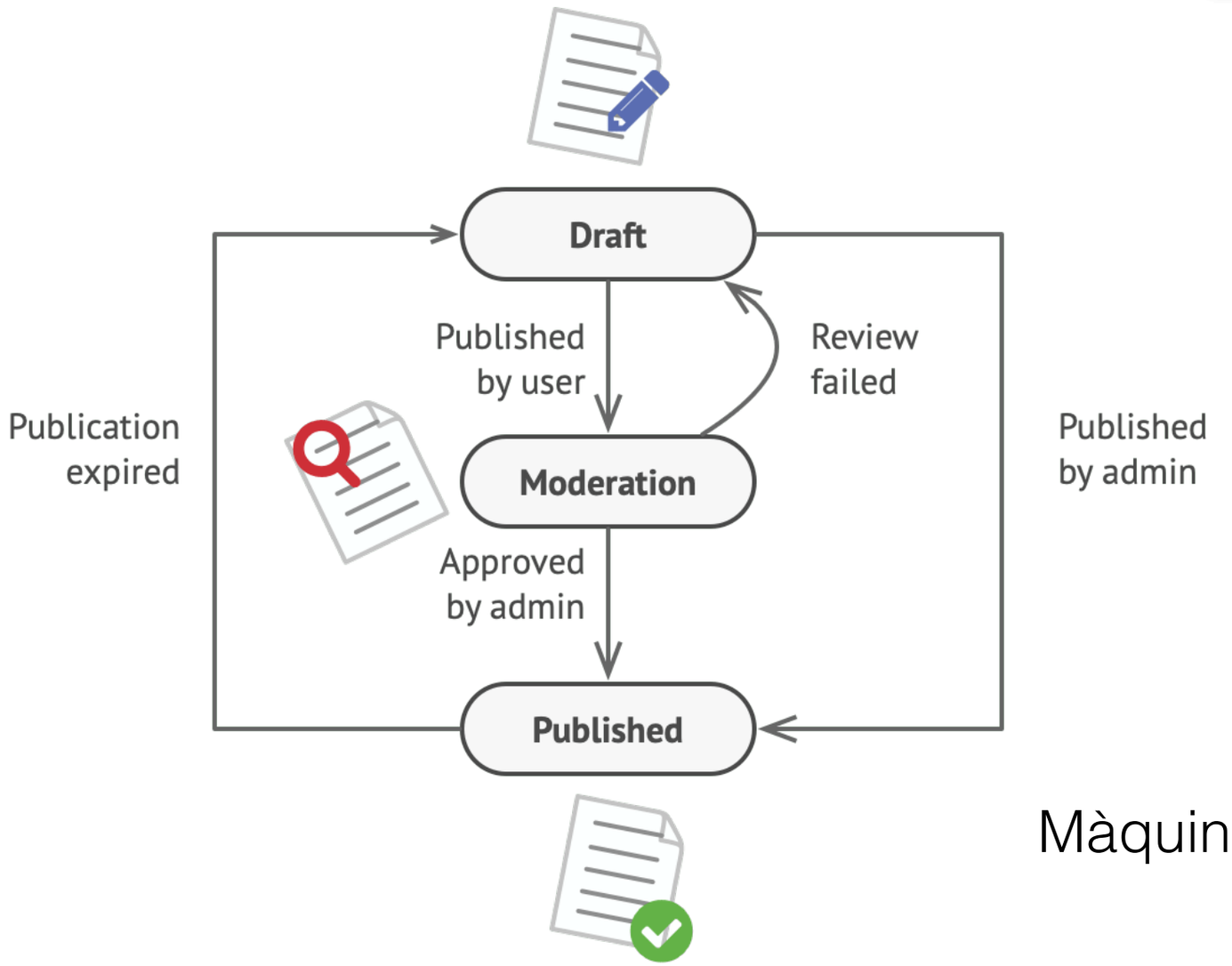
3.4. Patrons de disseny

Propòsit → **Àmbit ↓**	**CREACIÓ**	**ESTRUCTURA**	**COMPORTAMENT**
CLASSE	• Factory method	• class Adapter	• Interpreter • Template method
OBJECTE	• Abstract Factory • Builder • Prototype • Singleton • Object pool	• Object Adapter • Bridge • Composite • Decorator • Facade • Flyweight • Proxy	• Chain of Responsability • Command • Iterator • Mediator • Memento • Observer • **State** • Strategy • Visitor

Exemple patró State

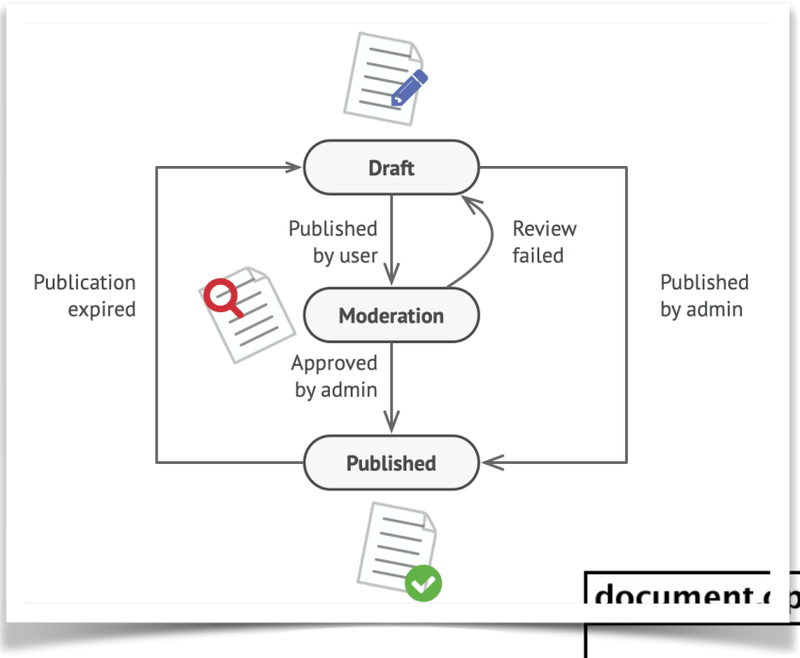
- Edició d'un document té 3 estats:
  Draft, Moderat, Publicat

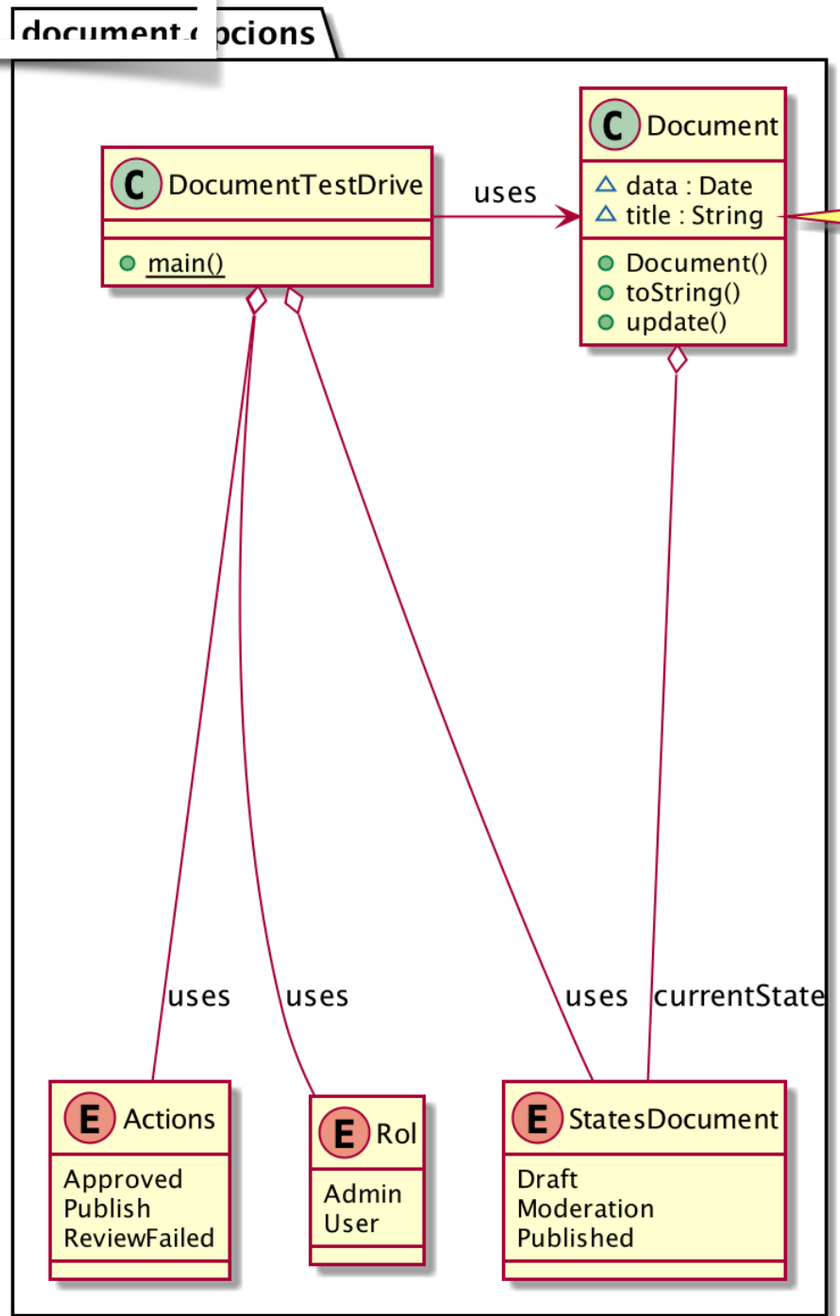




Màquina d'estats (State Machine)

Primera implementació



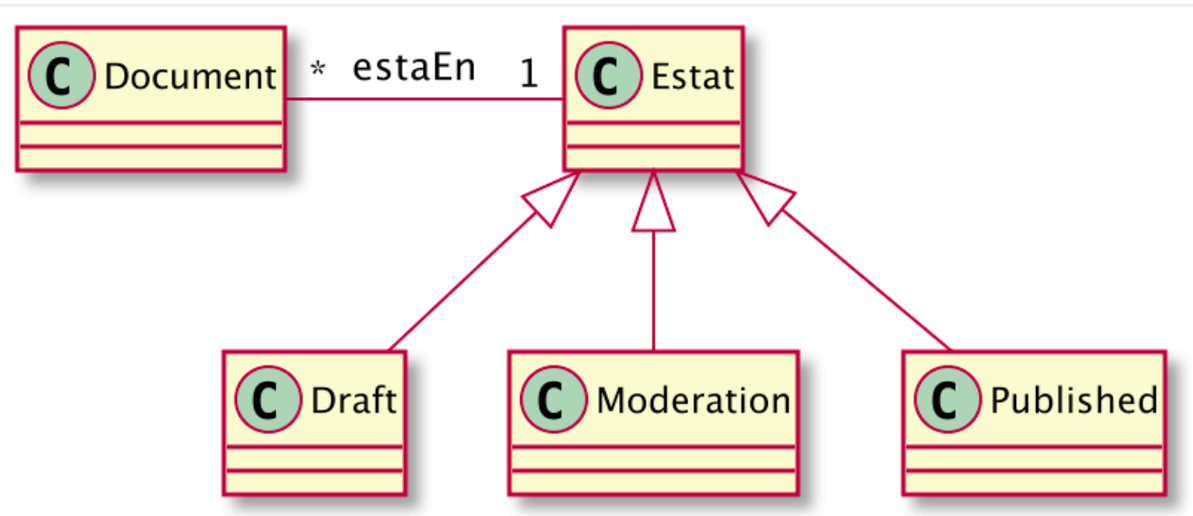


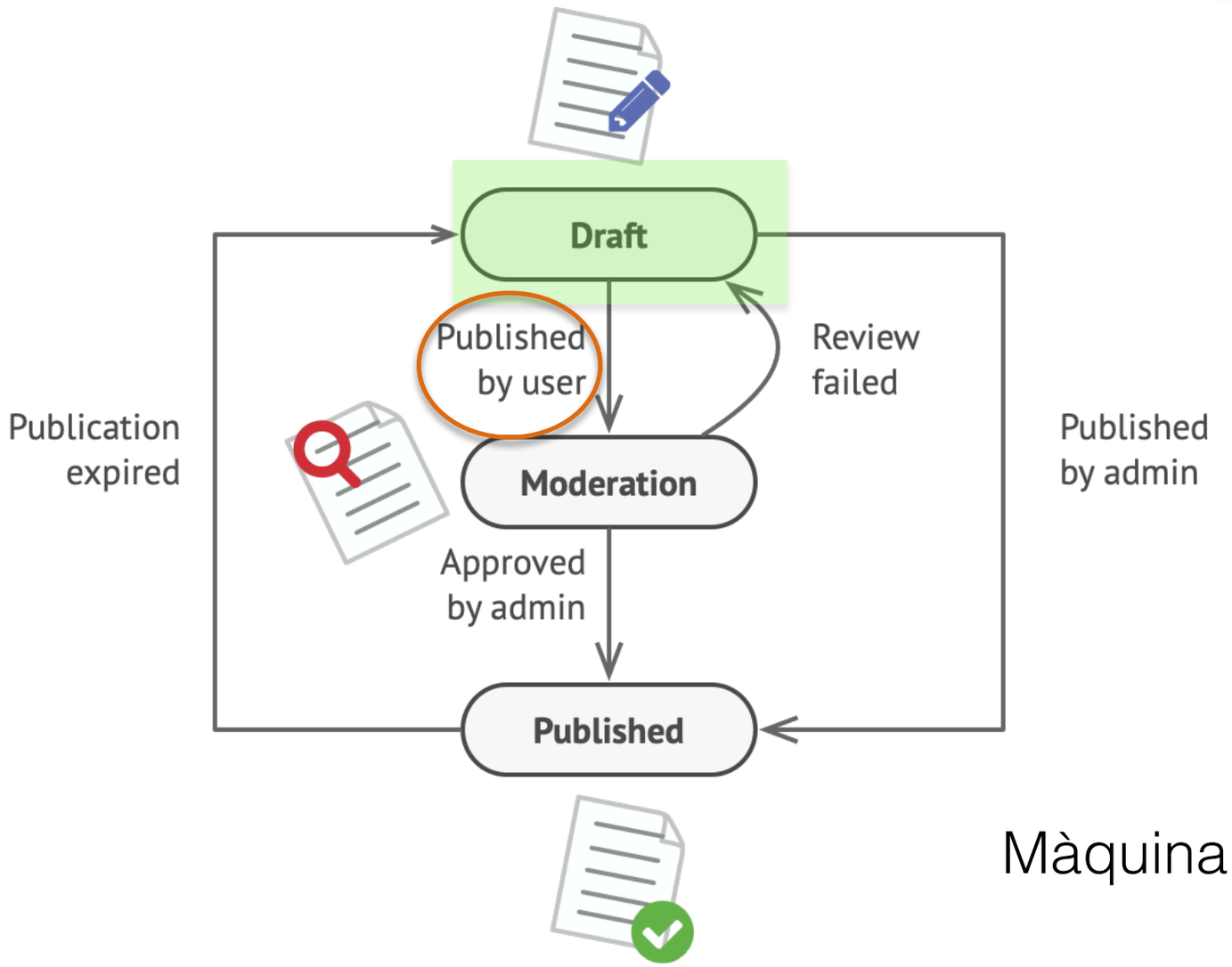

```
StatesDocument currentState;
Date data;
String title;

public Document(String title, Date data) {
   this.title = title;
   this.data = data;
   this.currentState = StatesDocument.Draft;
}

public void update(Rol role, Actions action ) {
   System.out.println("The role: "+role + " is doing "+action);
   switch (currentState) {
      case Draft:
         if (action == Actions.Publish) {
            if (role == Rol.User)
               currentState = StatesDocument.Moderation;
            else if (role == Rol.Admin)
               currentState = StatesDocument.Published;
         }
         break;
      case Moderation:
         if (role == Rol.Admin) {
            if (action == Actions.Approved)
               currentState = StatesDocument.Published;
            else if (action == Actions.ReviewFailed)
               currentState = StatesDocument.Draft;
         }
         break;
      case Published:
         Calendar obj = Calendar.getInstance();
         Date currentDate = obj.getTime();
         System.out.println("Current Date and time: "+currentDate);
         if (data.after(currentDate)) {
            currentState = StatesDocument.Draft;
         }
         break;
   }
}
```

Quins principis SOLID trenca?

Exemple patró State

- Edició d'un document té 3 estats:
  Draft, Moderat, Publicat

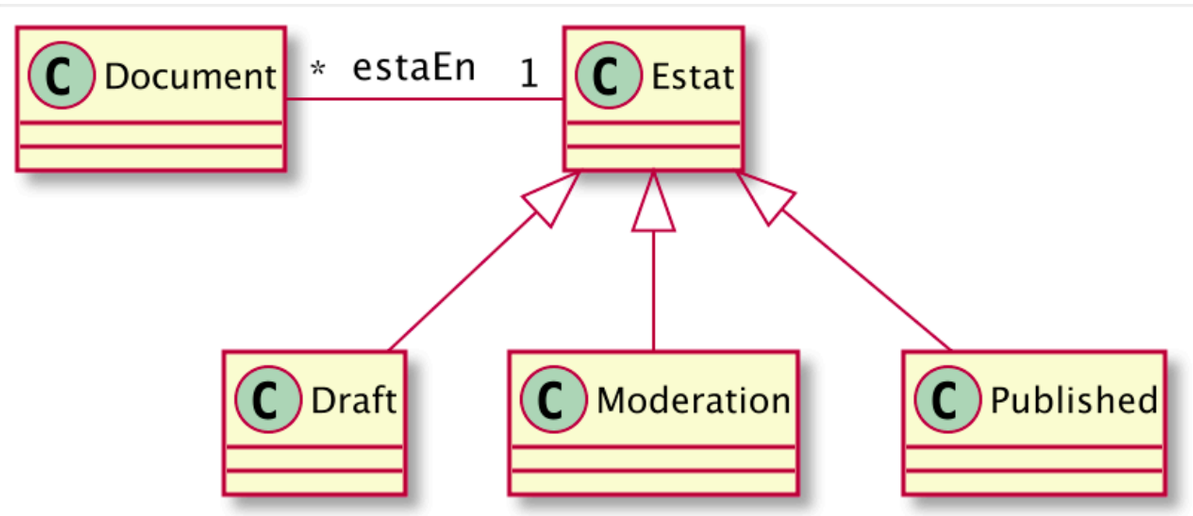


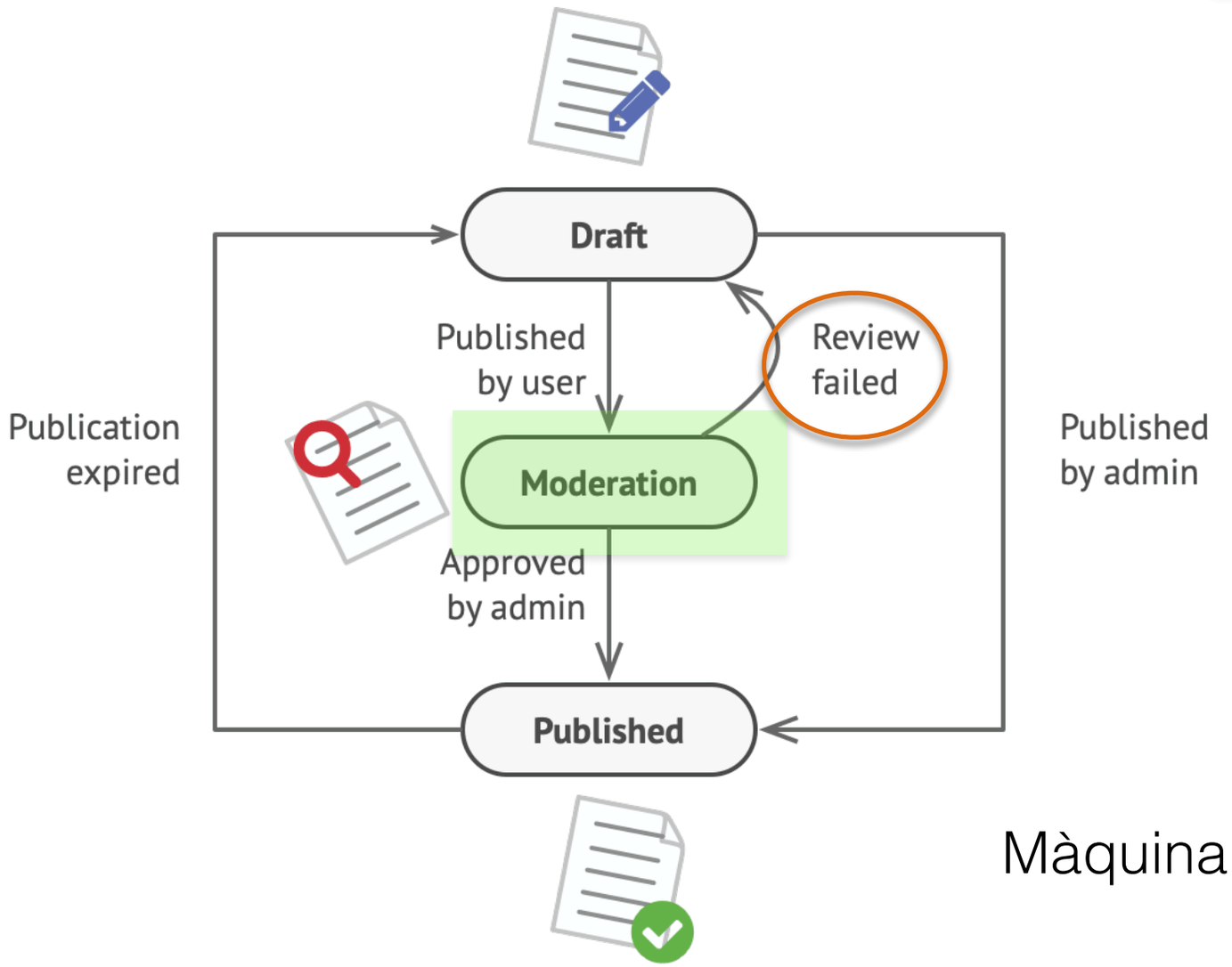


Màquina d'estats (State Machine)

Exemple patró State

- Edició d'un document té 3 estats:
  Draft, Moderat, Publicat

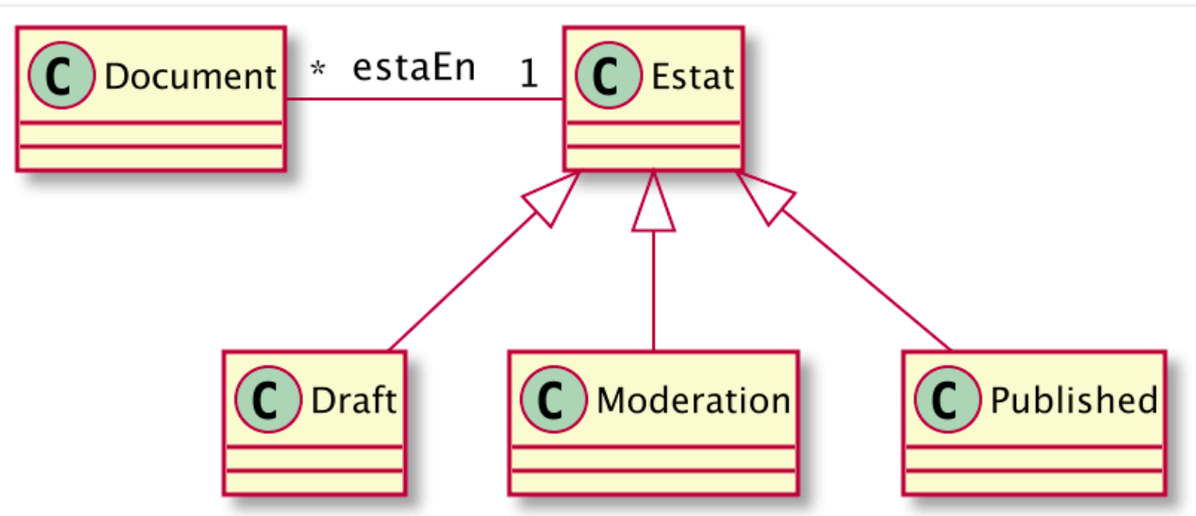


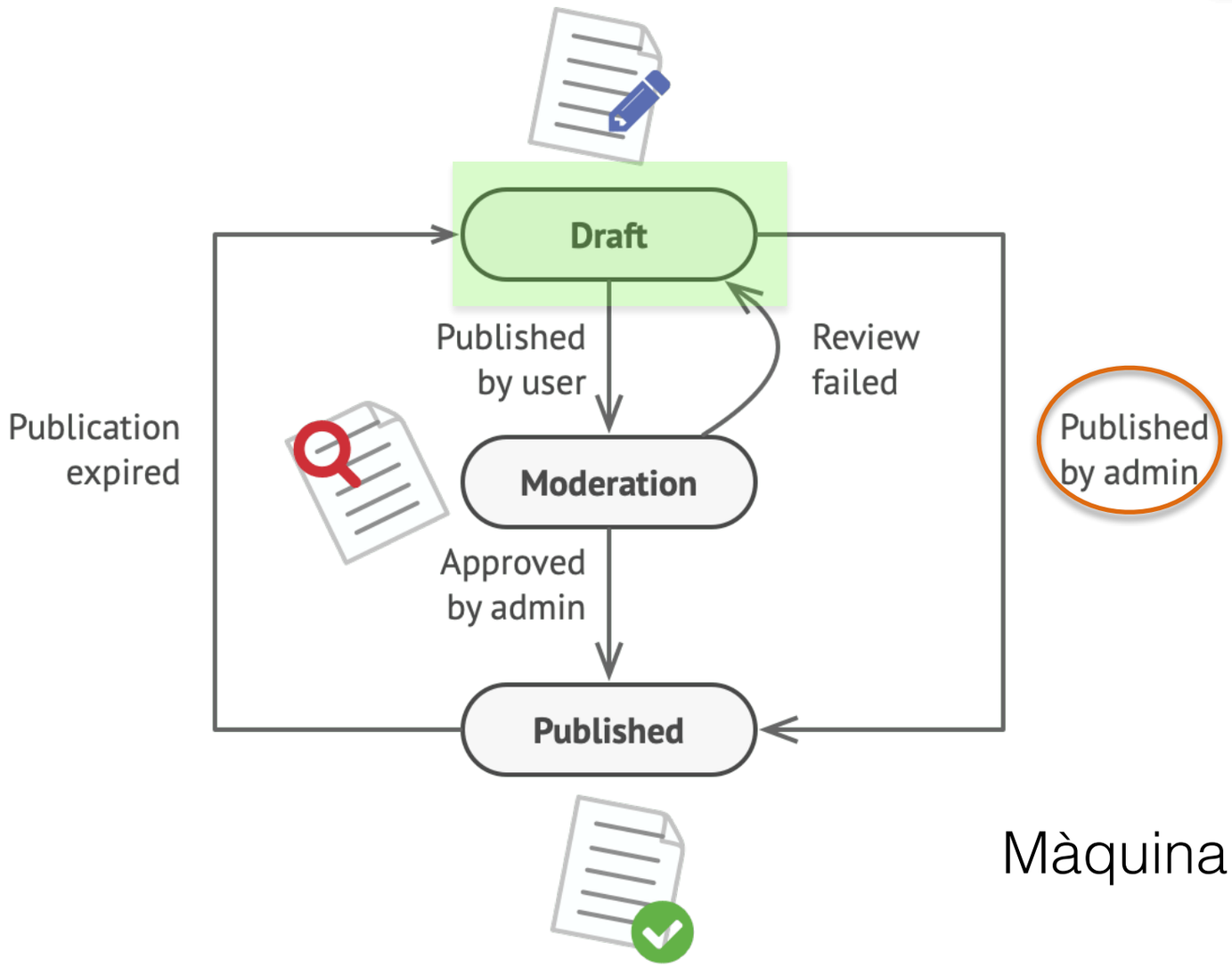


Màquina d'estats (State Machine)

Exemple patró State

- Edició d'un document té 3 estats:
  Draft, Moderat, Publicat

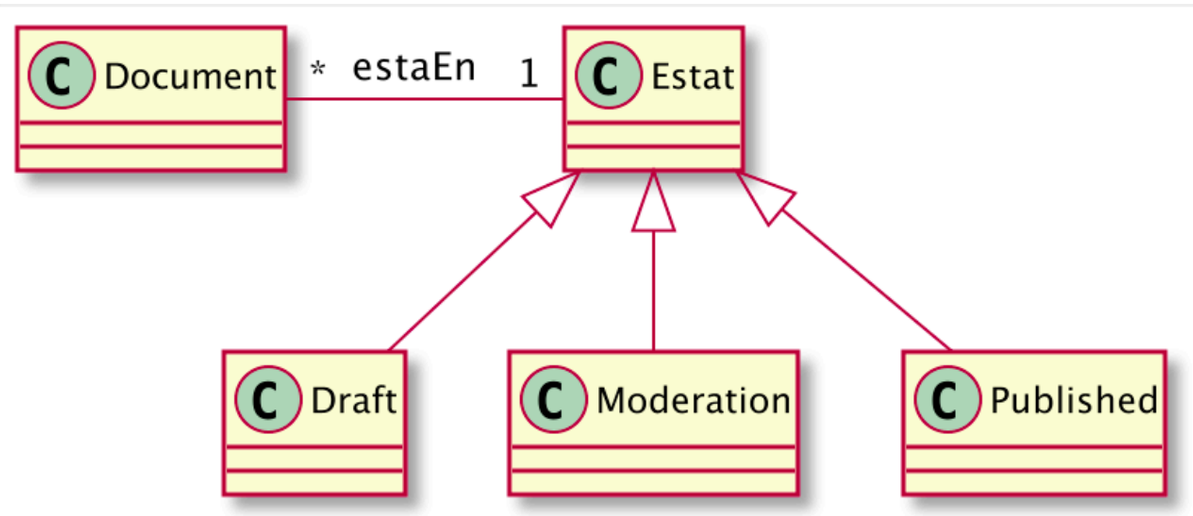


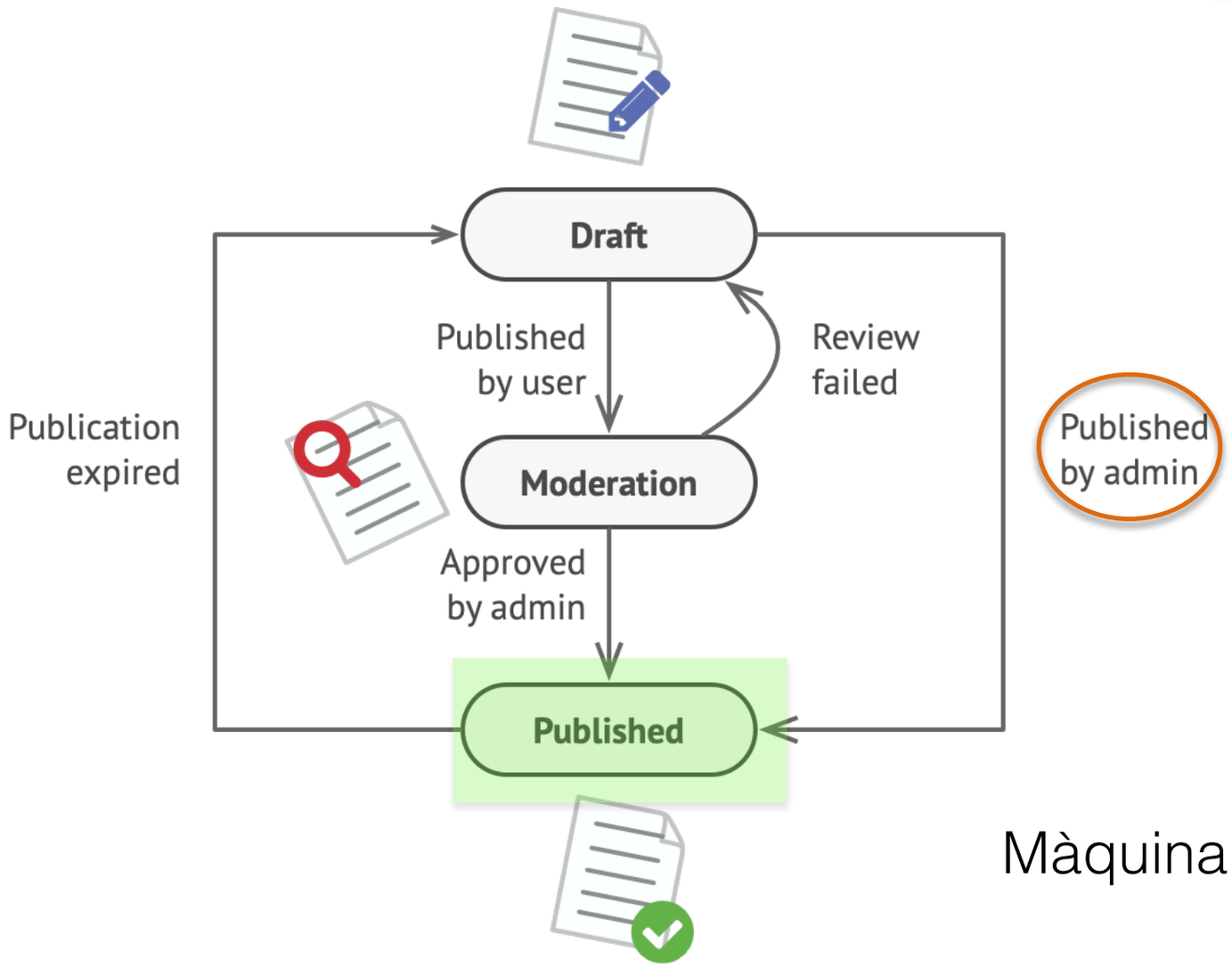


Màquina d'estats (State Machine)

Exemple patró State

- Edició d'un document té 3 estats:
  Draft, Moderat, Publicat





Màquina d'estats (State Machine)

State

- **State** – Patró de disseny de comportament que permet a un objecte alterar el seu comportament quan el seu estat intern canvia. Sembla com si l'objecte canviés la seva classe.

Patró State

Nom del patró: State

Context: Comportament

Problema:

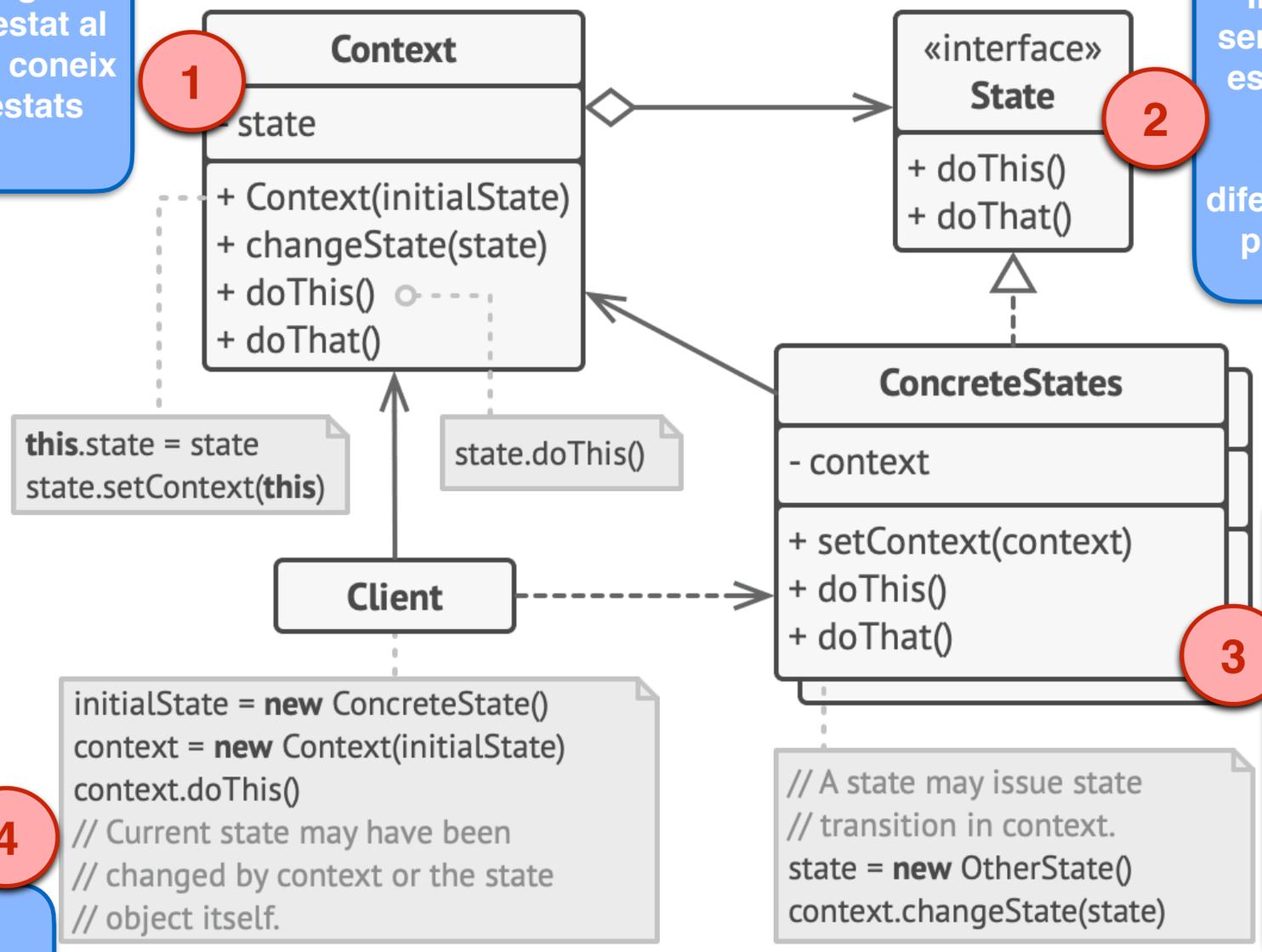
- Permetre a un objecte modificar el seu comportament quan el seu estat intern canvia. L'objecte sembla que canviï la seva pròpia classe.

Solució:

- Es crea una interfície **State** que encapsula tots els possibles estats
- Les subclasses de la interfície representen cada possible estat
- Es crea una classe **Context** que té un mètode per delegar a State el canvi d'estat i la seva gestió
  - Context manté una instància d'una de les classes derivades que defineix l'estat actual

Patró State

Context té la referència a l'estat actual i delega la funció de canvi d'estat al propi estat. Només coneix la interfície i no estats concrets

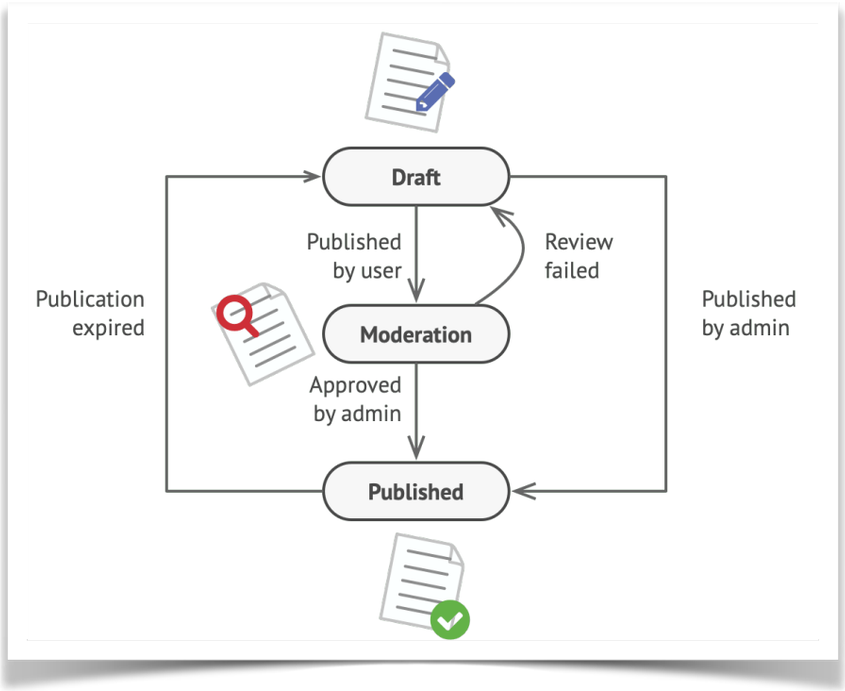


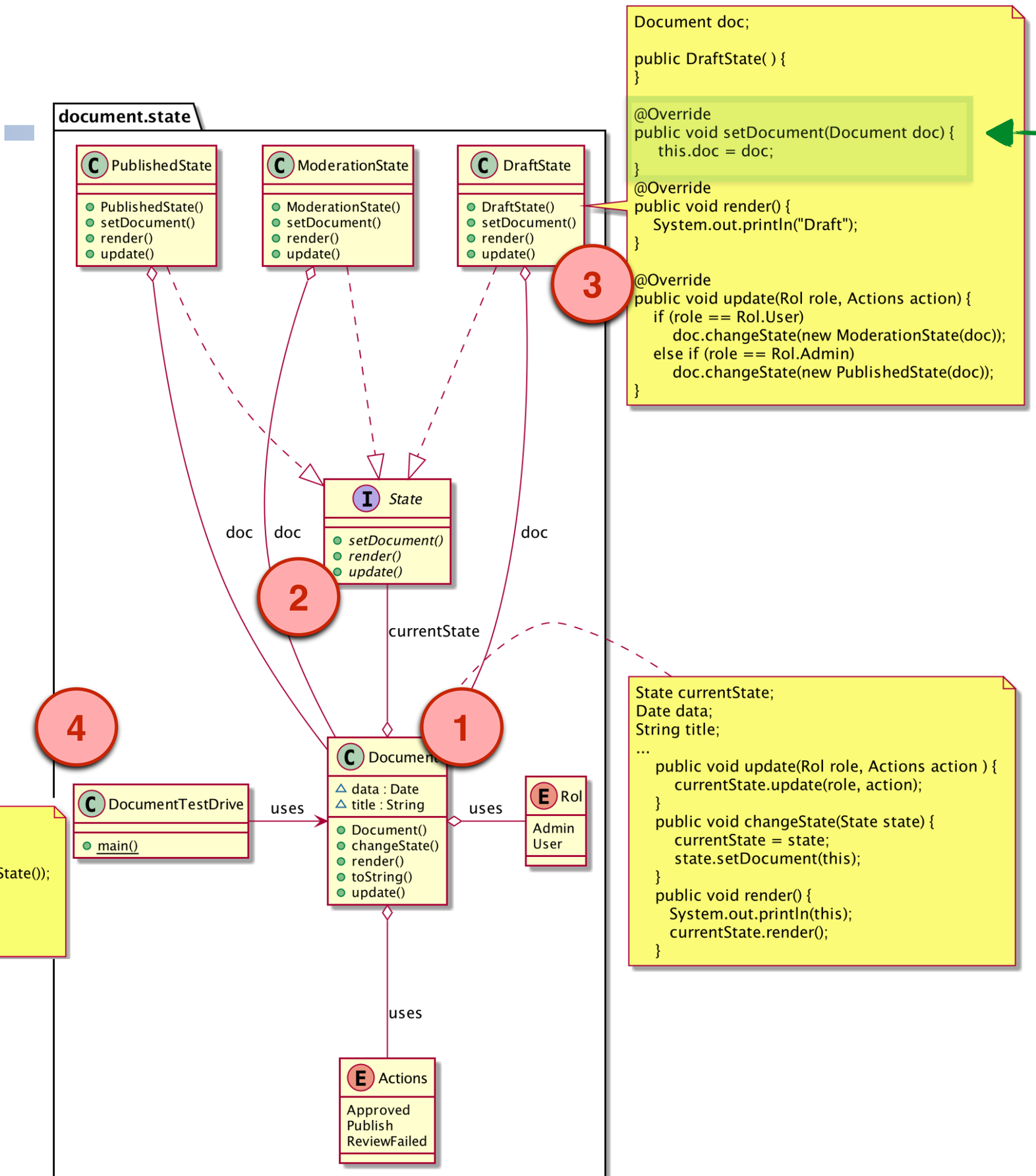
Interfície que declara com seran les accions a fer per un estat. En general té el mètde de update() però pot teir d'altres noms o implicar diferents funcions a l'estat que provoquin un canvi d'estat

Estats concrets que fan el update segons el que ha notificat la classe de Context. Aquests estats concrets tenen un atribut de tipus la classe de Context per que normalment necessiten valors d'aquesta classe Context per determinar el canvi a fer. A més necessiten la classe de Context per a fer el canvi de estat

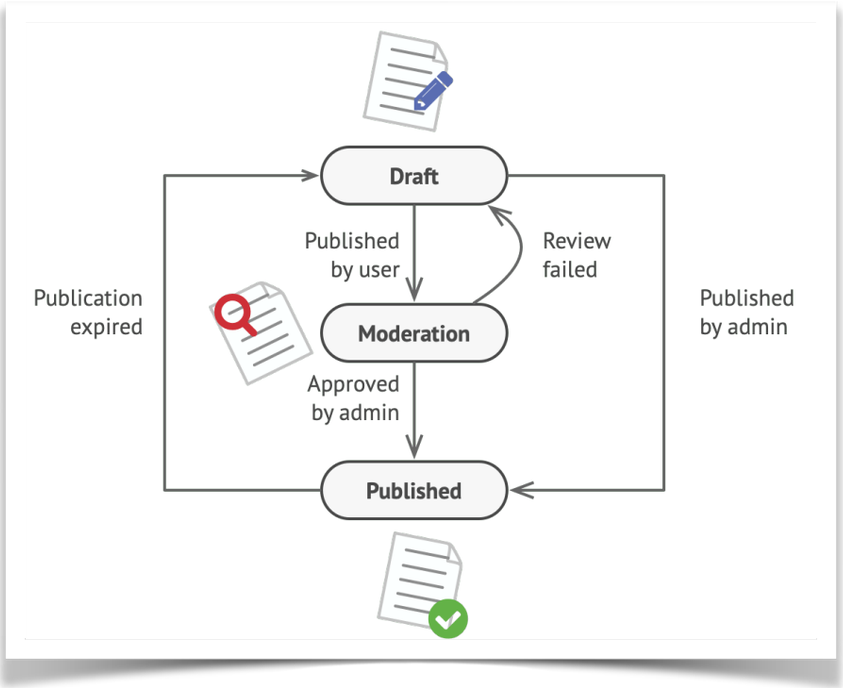
Client és la classe que usarà el context per a poder inicialitzar la màquina d'estats i per a notificar-li els canvis

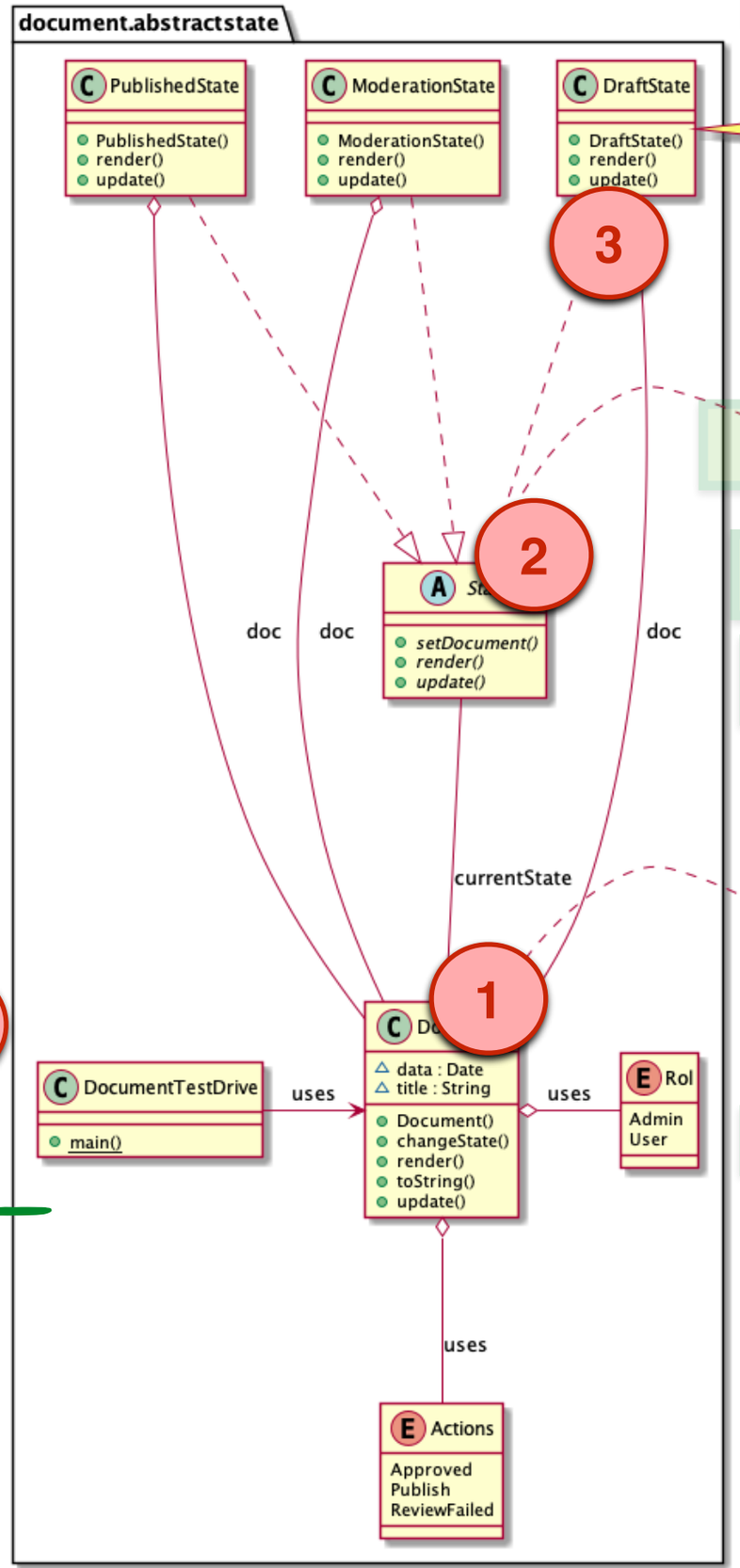
Aplicació del patró State





Aplicació del patró State usant Abstract class






```
public class DraftState extends State {

    public DraftState(Document document) {
        super(document);
    }
    @Override
    public void render() {
        System.out.println("Draft");
    }

    @Override
    public void update(Rol role, Actions action) {
        if (role == Rol.User)
            doc.changeState(new ModerationState(doc));
        else if (role == Rol.Admin)
            doc.changeState(new PublishedState(doc));
    }
}
```

```
public abstract class State {

    protected Document doc;

    protected State(Document doc) {
        this.doc = doc;
    }

    public void setDocument(Document doc) {
        this.doc = doc;
    }

    abstract public void render();
    abstract public void update(Rol rol, Actions action);
}
```

```
State currentState;
Date data;
String title;
...
    public void update(Rol role, Actions action ) {
        currentState.update(role, action);
    }
    public void changeState(State state) {
        currentState = state;
    }
    public void render() {
        System.out.println(this);
        currentState.render();
    }
```

4

```
public static void main(String[] args) {
        ...
    date = formatter.parse(date_string);
    doc = new Document("Prova de canvis d'estat", date);
    doc.changeState(new DraftState(doc));
    doc.render();
    doc.update(Rol.User, Actions.Publish);
    ....
```

Patró State

Nom del patró: State

Context: Comportament

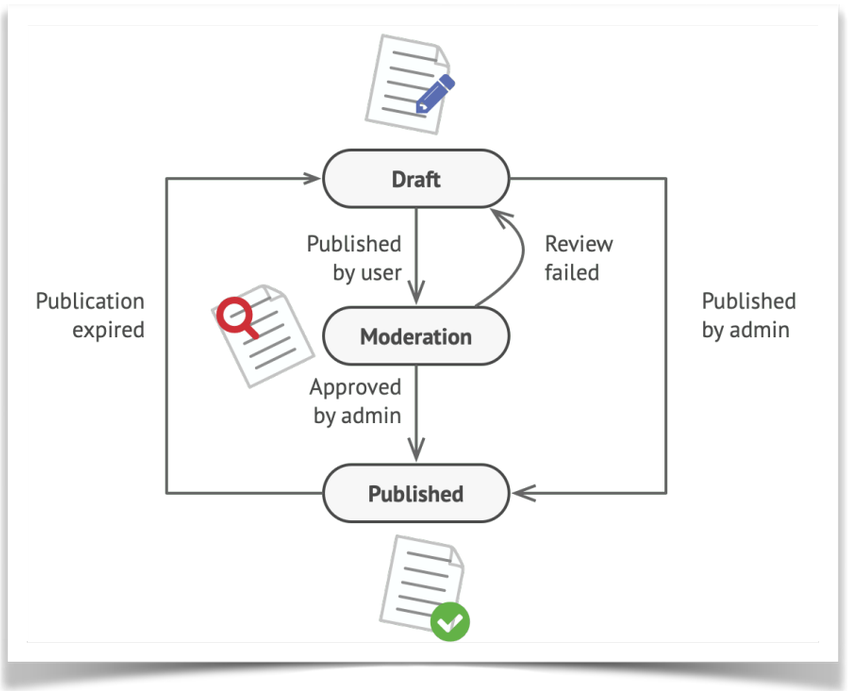
Pros:

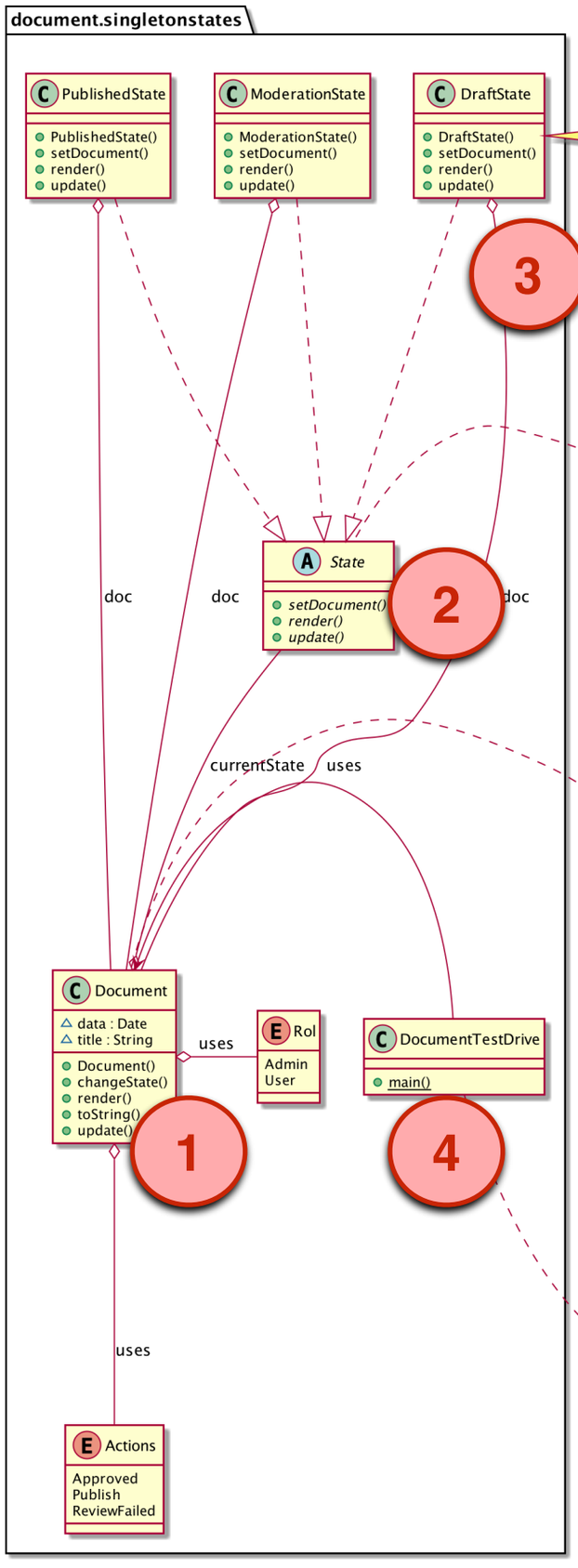
- La lògica específica dels estats està localitzada en classes que representen els estats
- Facilita l'escalabilitat del sistema
- Single Responsability Principle ✓
- Open-Closed Principle ✓

Cons:

- Problemes de memòria quan es canvia molt d'estats.
- De vegades, si els estats son pocs i fixes, el patró State pot produir masses classes i complicar molt el disseny final

Aplicació del patró State usant Singletons als estats





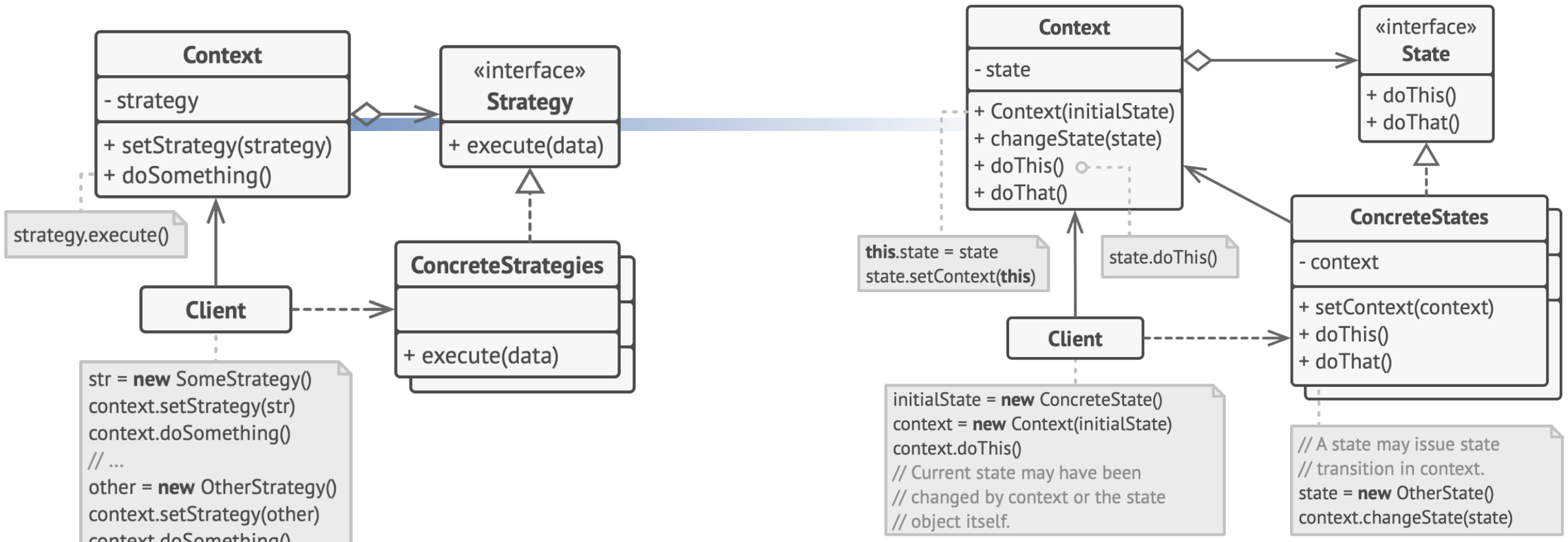

```
public class DraftState extends State {

   private static State state;
   private DraftState() {
   }

   protected static State getInstance() {
      if (state==null) {
         state = new DraftState();
      }
      return state;
   }

   @Override
   public void render() {
      System.out.println("Draft");
   }

   @Override
   public void update(Rol role, Actions action) {
      if (role == Rol.User)
         doc.changeState( ModerationState.getInstance());
      else if (role == Rol.Admin)
         doc.changeState( PublishedState.getInstance());
   }
}
```

```
public abstract class State {

      protected Document doc;

      public void setDocument(Document doc) {
            this.doc = doc;
      }

      abstract public void render();
      abstract public void update(Rol rol, Actions action);
}
```

```
public class Document {
   State currentState;
   Date data;
   String title;

   public Document(String title, Date data) {
      this.title = title;
      this.data = data;
   }

   public void update(Rol role, Actions action ) {
      currentState.update(role, action);
   }
   public void changeState(State state) {
      currentState = state;
      currentState.setDocument(this);
   }
   public void render() {
     System.out.println(this);
     currentState.render();
   }
   public String toString() {
      return "The document: "+ this.title + " is in " ;
   }
}
```

```
public static void main(String[] args) {
      ...
      date = formatter.parse(date_string);
      doc = new Document("Prova de canvis d'estat", date);
      doc.changeState(DraftState.getInstance());
      doc.render();
      doc.update(Rol.User, Actions.Publish);
      ....
```

State versus Strategy



Strategy

Diferents mètodes posats cadascun d'ells en una classe diferent.

El client especifica l'objecte Strategy que composa el Context. Normalment només hi ha una estratègia definida per a un context.

State

Conjunt de comportaments encapsulats en estats.

El context delega a un dels estats en qualsevol moment. Al llarg del temps, l'estat actual canvia a un altre estat per reflexar l'estat intern del context.

El client coneix poc o gens sobre els objectes d'estat.