Patrons de conception (Design Patterns)

Gerson Sunyé

Université de Nantes

http://sunye.free.fr

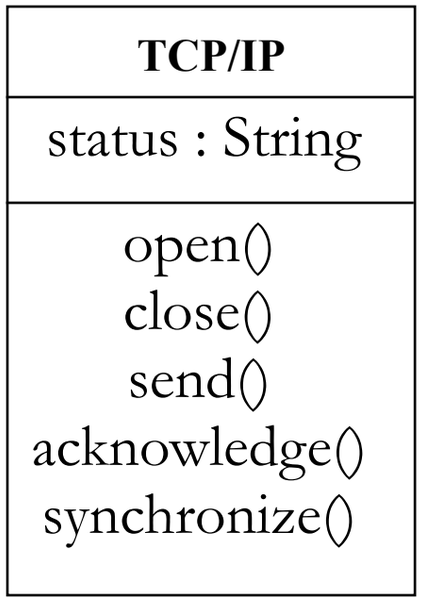
Plan

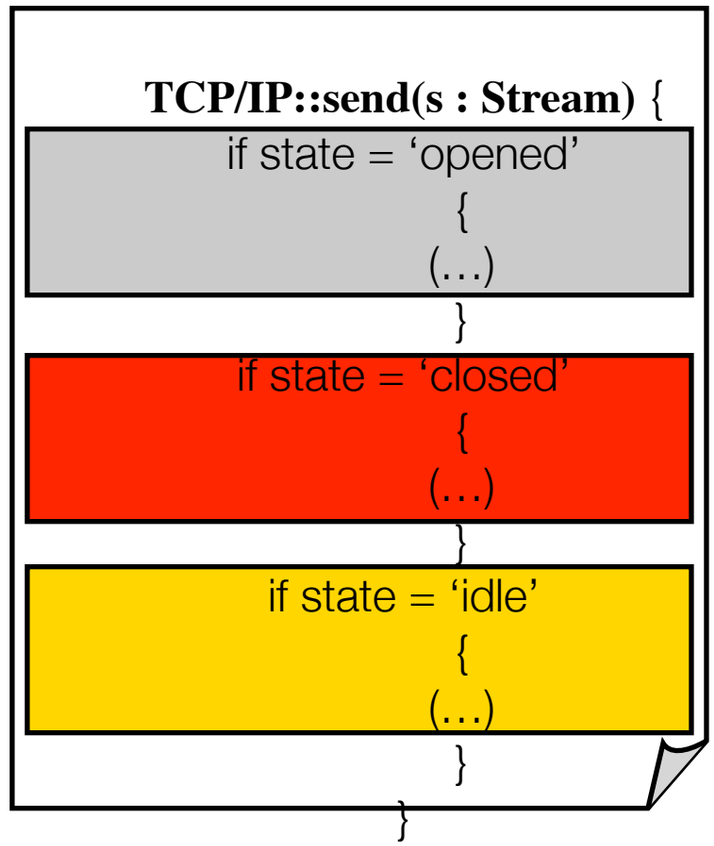
- Un exemple.
- Définition.
- Le canevas de description.
- Le pattern State.
- Références.

Les patterns par l’exemple

Le pattern state

Implémentation du protocole TCP/IP





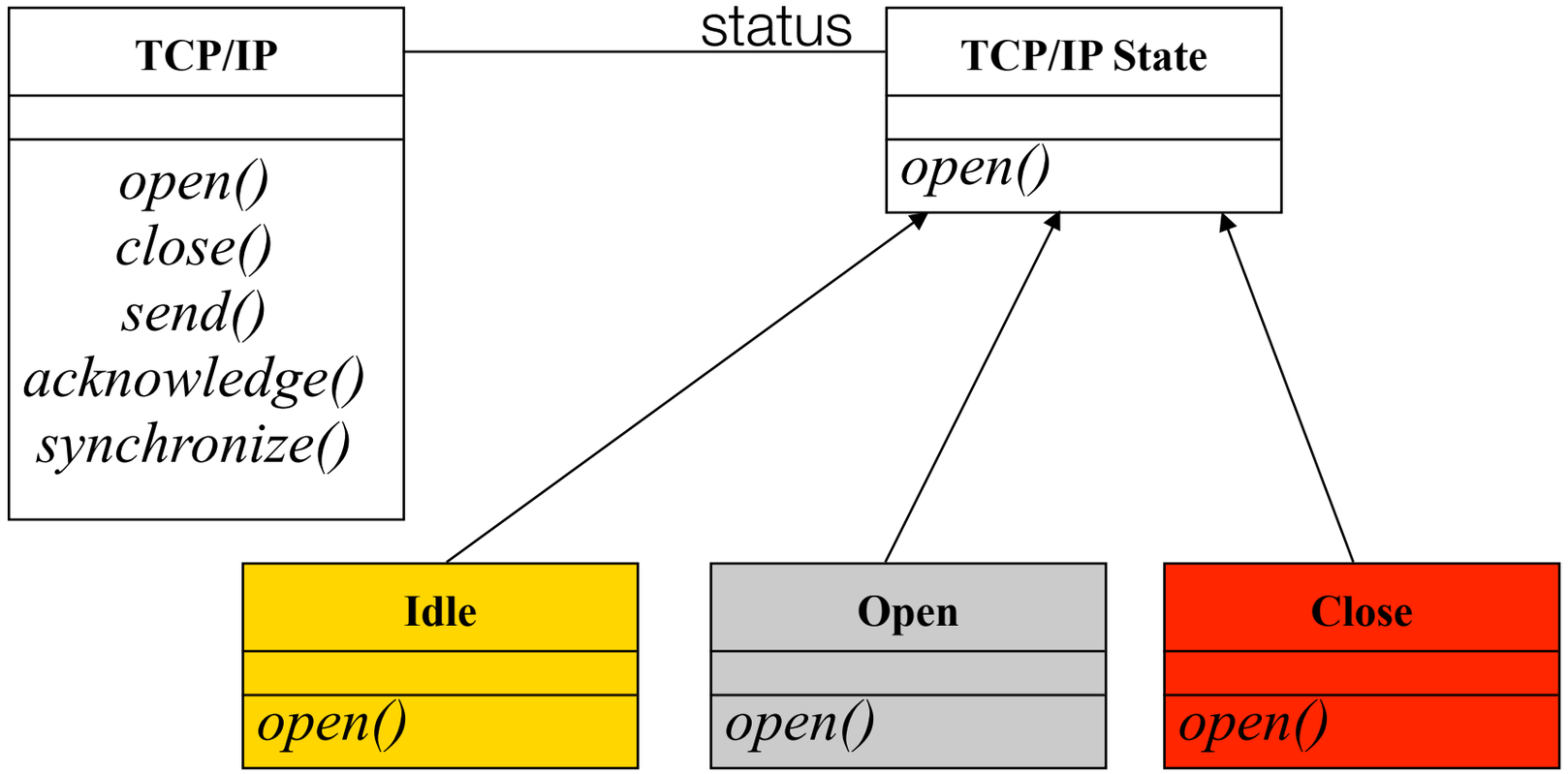
Problème

- Comment éviter que l’état de la connexion soit vérifié à chaque fois qu’un paquet est envoyé?

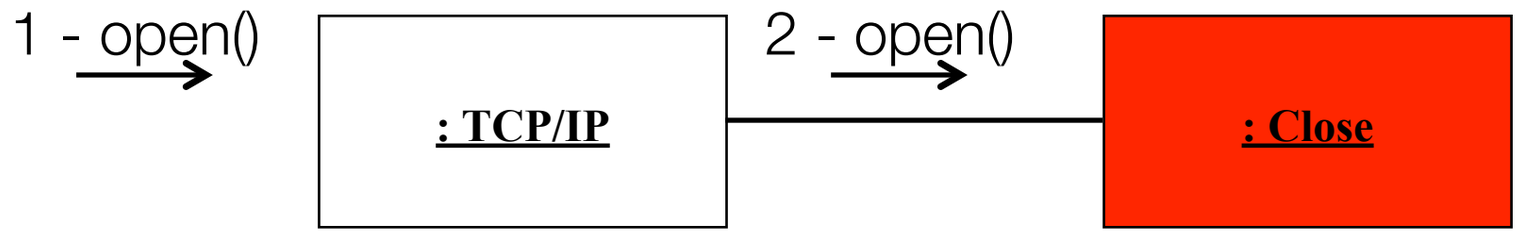
Solution

- Isoler les comportements dépendants des différents états de connexion dans des classes différentes.

En d’autres termes:

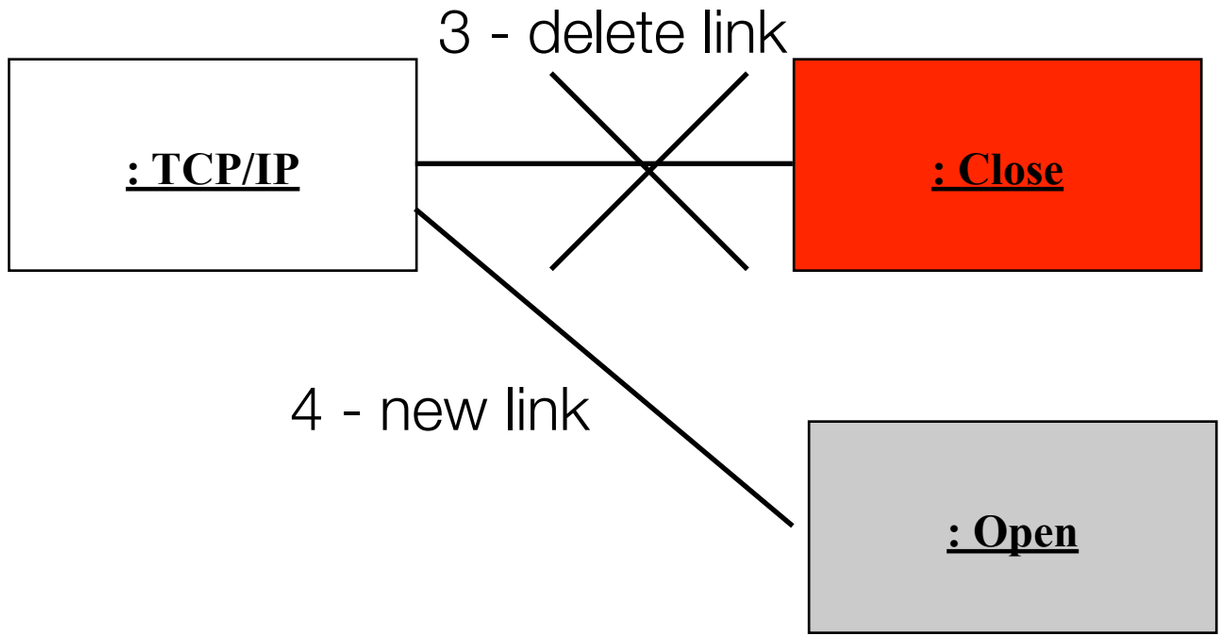


Exemple de connexion




```
TCP/IP::open() {
    this.status.open();
}
```

Exemple de connexion



Conséquences

- Chaque instance de « TCP/IP » est associé à une instance d’une sous-classe de « TCP/IP State »

- La vérification de l’état n’est plus nécessaire.

Observations

- Cette solution est utilisée dans la plupart des implémentations du protocole TCP/IP sur Unix.

- Une solution similaire est utilisée dans plusieurs éditeurs graphiques (comportement d’un outil selon le type de figure sélectionnée).

Design patterns

Définition

Motivation

- Comment cataloguer les solutions de design connues par les concepteurs experts?

- Et surtout, comment les rendre accessibles aux débutants?

Solution

- Cataloguer les solutions éprouvées de conception.

- Les décrire sous la forme de « patterns »

Historique (1/5)

- Christopher Alexander:
  - Notes on the Synthesis of Form, Harvard University Press, 1964.
  - Oregon Experiment, Oxford University Press, 1975.
  - A Pattern Language: Towns, Buildings}, Construction, Oxford University Press, 1977.
  - Timeless Way of Building, Oxford University Press, 1979.

Historique (2/5)

- Dans [TTWoB] Alexander propose un paradigme pour l'architecture, basé sur trois concepts:
  - The Quality (a.k.a. ``the Quality Without a Name'').
  - The Gate.
  - The Way (a.k.a. ``the Timeless Way'').

Historique (3/5)

- Using Pattern Languages for Object-Oriented Programs. Ward Cunningham and Kent Beck. OOPSLA'87.

Historique (4/5)

- Advanced C++ Programming Styles and Idioms. Jim Coplien, 1991.
- Workshops OOPSLA de 90 à 92.

Historique (5/5)

- Hillside Group (Conférences PLoP): Kent Beck, Grady Booch,
- Richard Gabriel et al. OOPSLA 1993, 94.
- Design Patterns [GoF]. 1995.

Définition

- Définition générale:
    - « Un pattern est une solution éprouvée d’un problème dans un contexte ».
- Christopher Alexander:
    - « Chaque pattern est une règle en trois parties, qui exprime la relation entre un certain contexte, un problème et une solution ».

Définition

- James Coplien:

« J’aime bien faire la relation entre cette définition et les patrons de couture.

Je pourrais vous expliquer comment faire un habit en spécifiant la route que les ciseaux doivent suivre à travers le tissu en terme d’angles et longueur de coupe. Ou alors, je peux vous donner un patron. (…) »

Définition

« (…) En lisant la spécification, vous n’auriez aucune idée de ce qui était en train d’être fait, ou si vous aviez fait ce qu’il fallait, avant la fin.

Le pattern annonce le produit: il est la règle pour réaliser une chose, et aussi, dans différents aspects, la chose elle-même. »

Canevas de description

Les canevas

- Un pattern est décrit selon un canevas précis, sous forme littéraire.
- Ils ne sont que de la « documentation ».
- Il existe différents canevas de description de patterns:
  - Alexander, Coplien, GoF, Portland, Cockburn, etc.

Les composants essentiels

- Contexte
- Nom
- Problème
- Forces
- Solution
- Auteur et date

Contexte

- La place du pattern dans un système.

Nom

- Un label significatif qui traduit le principe du pattern.
- Le nom a tendance à être basé sur la solution.
- Le nom devient une partie du vocabulaire du domaine.

Problème

- Habituellement, le problème est posé comme une question, ou comme un énoncé.
- C'est la première chose que l'on regarde.
- La compréhension du problème vient avec l'analyse des forces.

Forces

- Le cœur d'un pattern.
- La totalité du pattern est un champ de forces ou des compromis.
- Les patterns individuels encapsulent des forces en relation.
- Elles permettent de traiter un pattern à la fois.

Solution

- Les solutions équilibrent les forces.

- Les solutions s'appliquent au système comme un tout: elles sont transformables et non fixes.

Auteur et date

- L’auteur d’un pattern est rarement son inventeur.

Le pattern State

Nom

- State (État).

- Alias: Objects for States, Enveloppe-Letter.

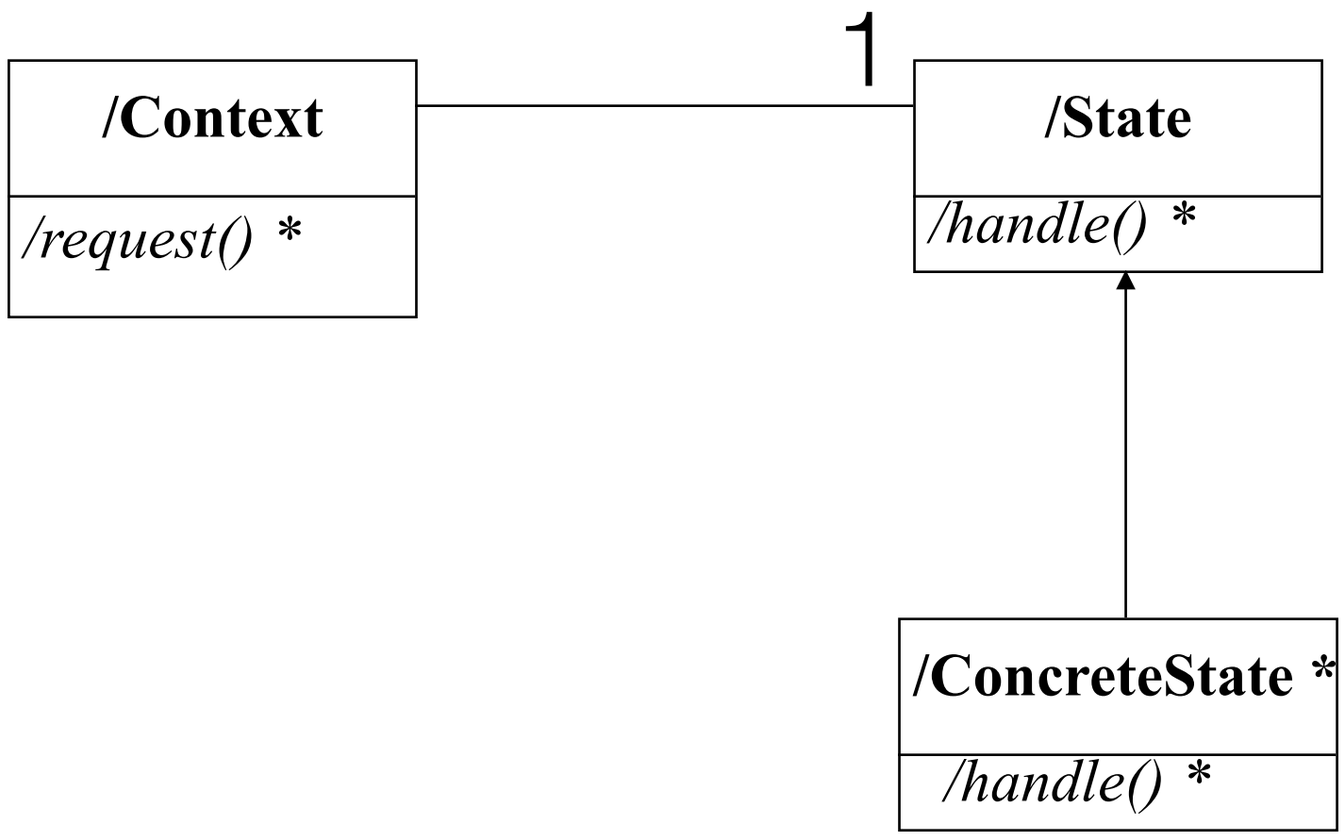
Problème

- Comment traiter un objet dont le comportement est fortement dépendant de son état, sans vérifier, à chaque appel d'une opération, l'état actuel?

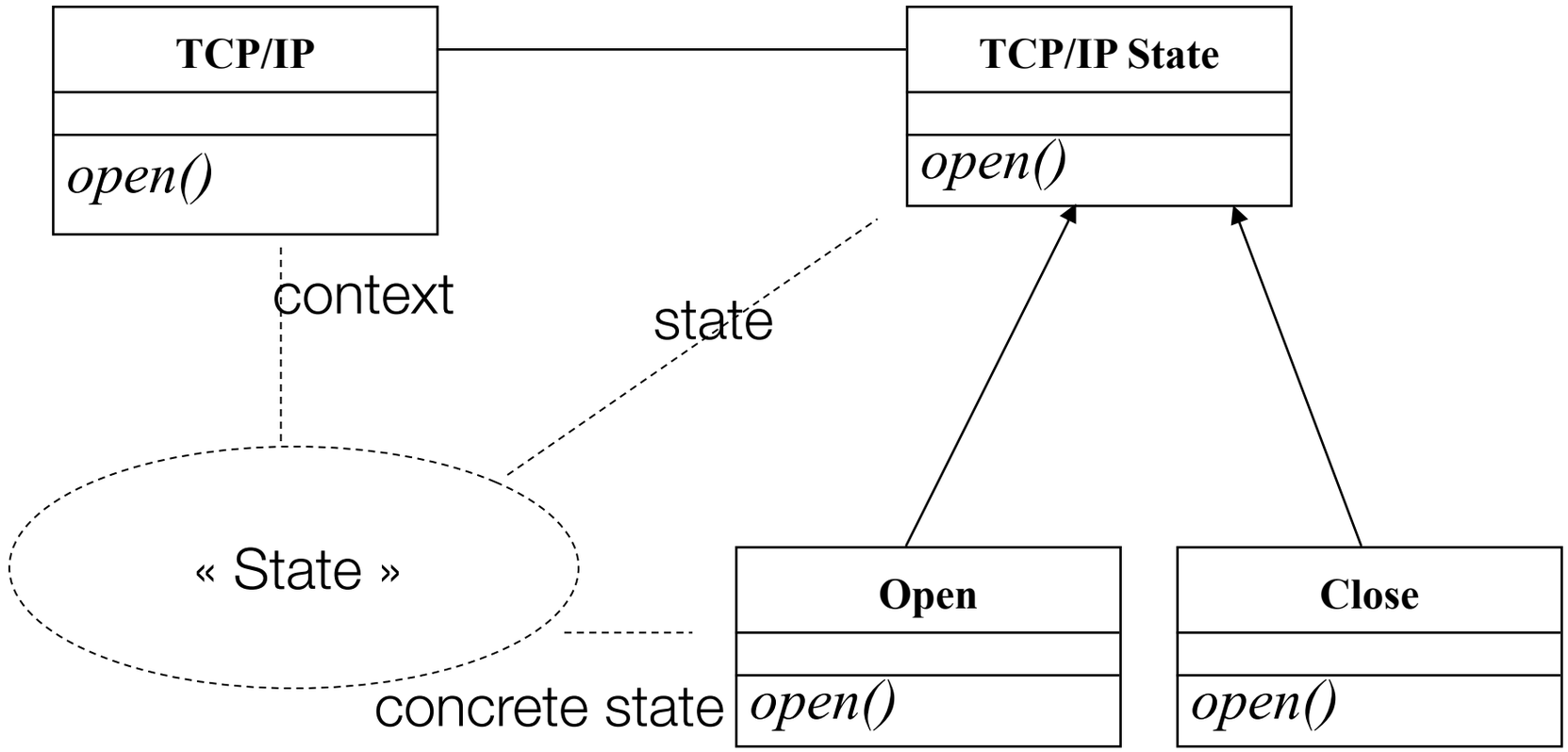
Contexte

- Le comportement d’un objet dépend de son état.
- Différentes opérations contiennent des opérateurs conditionnels sur l’état de l’objet.

Solution (1/2)



Solution (2/2)


TCP/IP
open()
TCP/IP State
open()
context
state
« State »
concrete state
Open
open()
Close
open()

Conséquences

- Le comportement spécifique à un état est isolé.
- Les transitions d’état son explicites.
- Les objets-états peuvent être partagés.
- Qui définit les transitions d’état?
- Création et destruction des états.
- Utilisation de l’héritage dynamique.

Sources

Bibliographie

- « Software Patterns ». James Coplien. SIGS Books. New York, 1996.
- « Smalltalk Patterns: Best Practices ». Kent Beck. Prentice Hall, 1997, 256 pp., ISBN 0-13-476904-X.
- « Design Patterns: Elements of Reusable Object-Oriented Software ». Erich Gamma, Richard Helm,Ralph Johnson, and John Vlissides. Addison Wesley. October 1994.

Bibliographie

- La série « Pattern Languages of Program Design »
  - Coplien & Schmidt
  - Vlissides, Coplien, & Kerth
  - Martin, Riehle, Buschmann
  - Harrison, Foote, Rohnert

Sites Internet

- Patterns homepage: http://hillside.net/patterns/
- Portland Pattern repository: http://c2.com/ppr/index.html
- Cetus Links: htpp://www.objenv.com/cetus/oo_patterns.html
- Patterns FAQ: http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html