# Décorateur

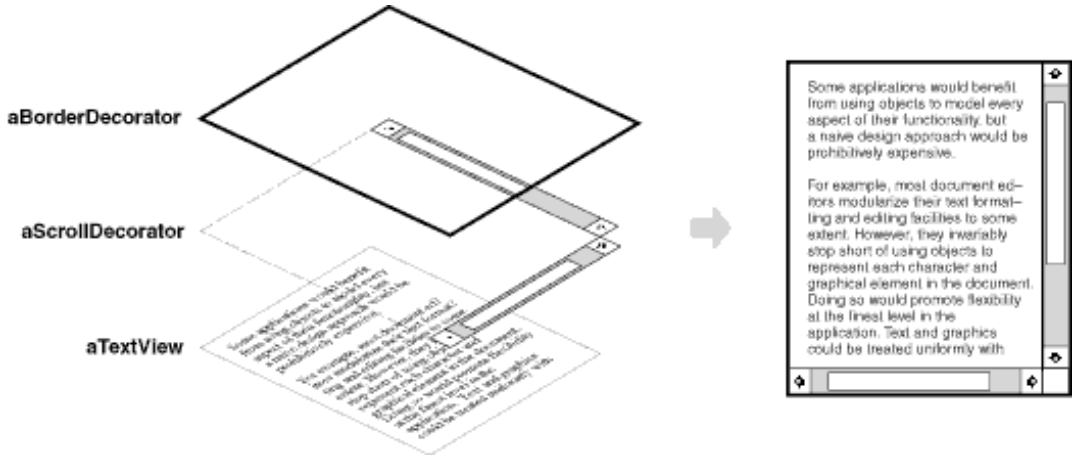
[Gamma 95]

# Objectif

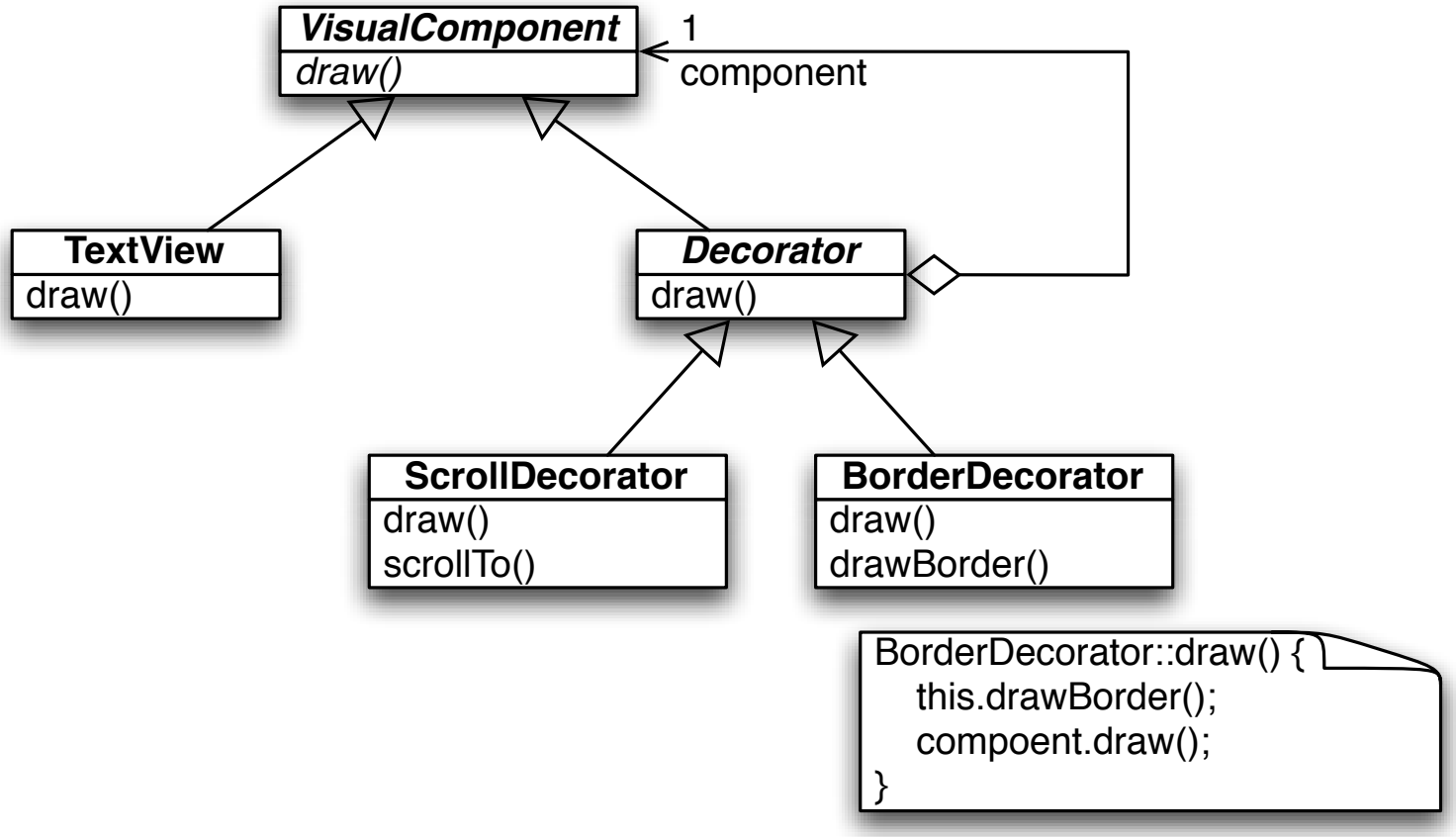
- Ajouter dynamiquement un comportement à un objet
- Fournir une alternative flexible à l'héritage

# Motivation

- Parfois, on souhaite ajouter un comportement à seulement certains objets d'une classe



# Motivation - Exemple (1/2)



# Motivation - Exemple (2/2)

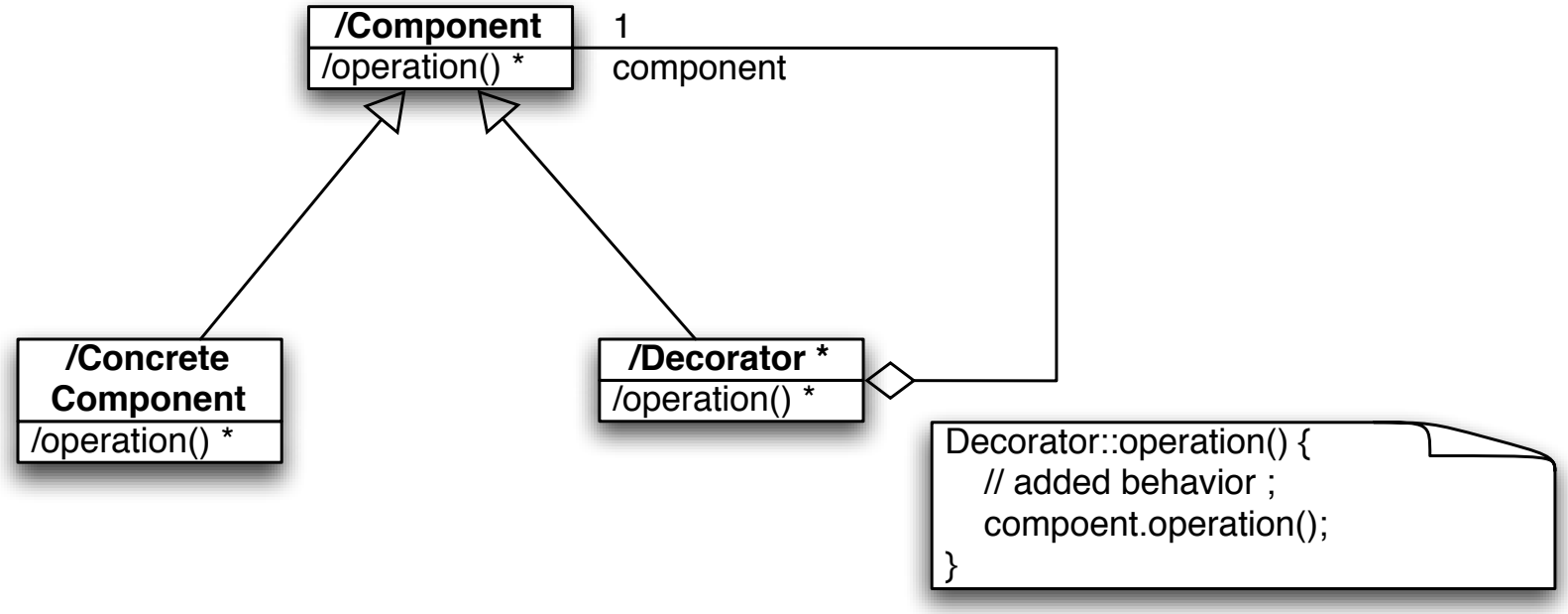
# Applicabilité (1/2)

- Utiliser un décorateur quand on souhaite:
  - ajouter un comportement à un objet dynamiquement et de manière transparente
  - ajouter un comportement qui peut être retiré

# Applicabilité (2/2)

- Utiliser un décorateur quand la création de sous-classes est impraticable

# Structure


/Component
/operation() *
1
component
/Concrete Component
/operation() *
/Decorator *
/operation() *
Decorator::operation() {
// added behavior ;
compoent.operation();
}

# Conséquences

- Plus flexible que l'héritage.
- Empêche l'utilisation de classes complexes dans le haut d'une hiérarchie de classes.
- Un composants et ses décorateurs ne sont pas identiques.
- Explosion de petits objets.

# Compromis d'implémentation

- Super-classe commune pour assurer la même interface.
- Omission du décorateur abstrait.
- Les classes «composant» restent légères.

# Décorateur