Module 13:
"Proxy"

Agenda

- Introductory Example: Web Shop Products
- Challenges
- Implementing the Proxy Pattern
- Pattern: Proxy
- Overview of Proxy Pattern
- Variation: Simple Proxy
- Use Cases for Proxy

Introductory Example: Web Shop Products

```
interface IProductRepository
{
    Product GetById( int id );
    IEnumerable<Product> GetAll();
    void Add( Product product );
}
```

```
IProductRepository products = new ProductRepository();
products.Add( new Product( ...) );
IEnumerable<Product> books = products
                .GetAll()
                .Where(p => p.Category == Category.Book);
foreach (Product product in books)
{
    Console.WriteLine( product );
}
```

Challenges

- How can we control that only administrators can add new products?
  - Cannot change the source code of the web shop library component!

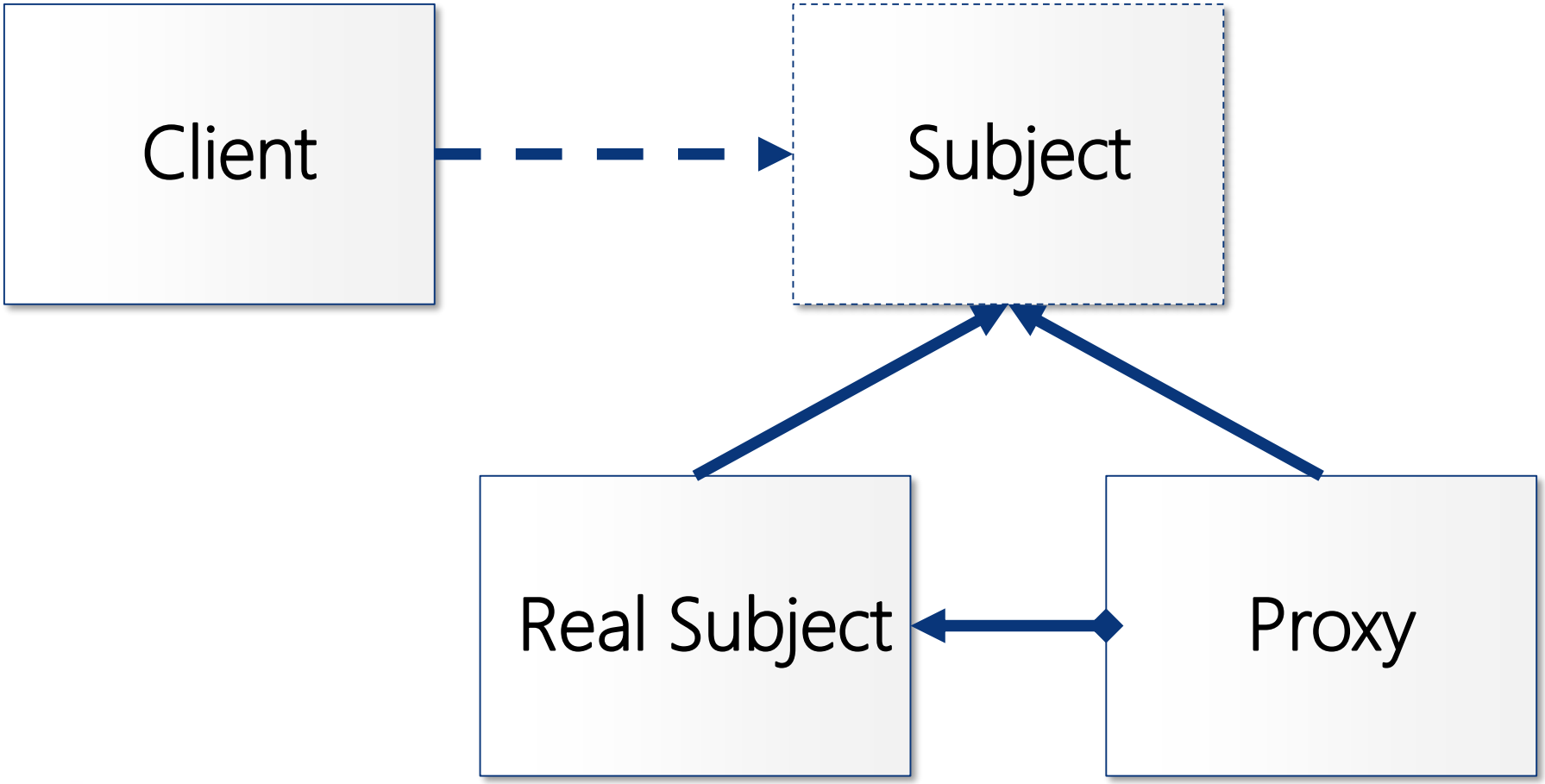
Pattern: Proxy

- *Provide a surrogate or place-holder for another object to control access to it.*

- Purpose
  - Define a substitute object with the same interface
  - Implement additional functionality or restriction in substitute object
  - Clients cannot tell whether they interact with the real object or a proxy

- Origin: Gang of Four

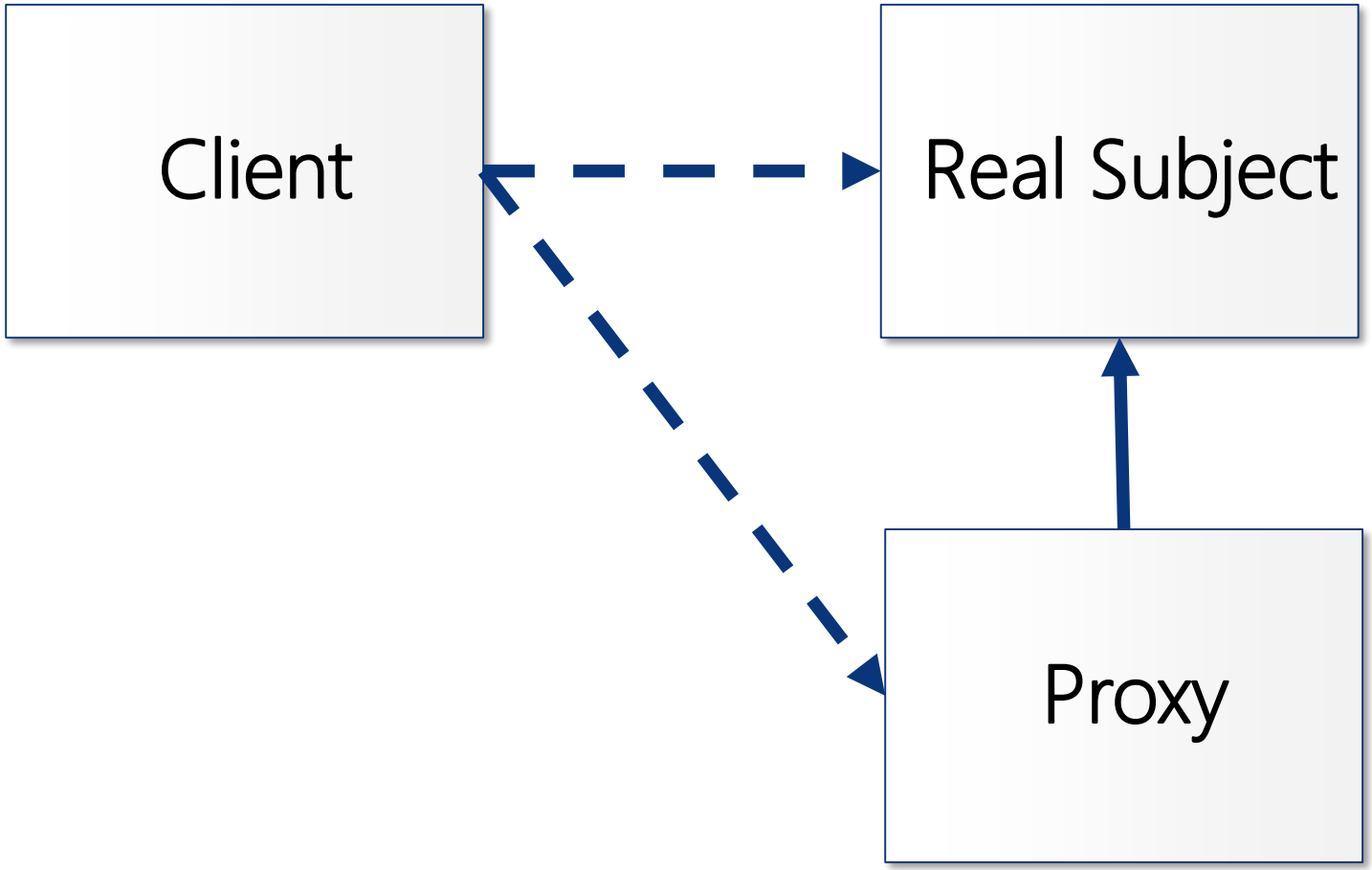
Overview of Proxy Pattern



Overview of Proxy Pattern

- Client
  - Interacts with any Subject through a general interface
- Subject
  - Interface or base class to subject functionality
- Real Subject
  - Concrete subject class implementing Subject interface
  - Provides concrete functionality
- Proxy
  - Substitute subject class implementing Subject interface
  - Implements added functionality or control restrictions to underlying Real Subject being controlled

Variation: Simple Proxy



Simple Proxy Pros and Cons

- Simple to implement
- Easier to maintain
- Only works when Real Subject is suitably "open"

- Violates Single Responsibility Principle of SOLID
- Uses inheritance instead of composition

- Fits well when there is no general interface to proxy

Proxy vs. Adapter

- Proxy
  - Simple, nice and clean to implement
  - Can be a slight burden to maintain if not autogenerated
  - Satisfies SOLID principles – Fits beautifully with Dependency Injection
  - Keeps same interface
- Adapter
  - Changes the interface

Use Cases for Proxy

- Use cases include
  - Virtual proxies
    - Lazy loading
    - Caching
  - Remote proxies
    - Distributed communication
  - Protection proxies
- Very frequently used


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Hasselvangen 243
8355 Solbjerg
Denmark