**Proxy Design Pattern**

**Korte beschrijving:**

Het proxy-patroon is een ontwerppatroon die relaties tussen klassen en entiteiten creëert. Proxy’s kunnen de efficiënte en functionaliteit verbeteren van het programma. Je hebt vijf verschillende manieren die je kan gebruiken zo heb je de cache, smart ,virtual, remote en protection proxy. De cache proxy verbetert de prestaties van de leden van het onderliggend object wanneer ze langlopende taken uitvoeren die zelden veranderende resultaten opleveren. De ***protection*** proxy voegt een beveiliging laag toe aan het onderliggende object. De ***smart*** proxy voegt extra functionaliteit toe aan het oproepen naar de leden van het echte object. De ***remote*** proxy biedt een lokaal object dat verwijst naar een onderwerpobject op een andere locatie.

**1.Type design pattern (creationele ,gedrag, architectorisch ,..):**

De proxy design pattern is een structurele pattern.

**2.Waarom deze design pattern wordt gebruikt(intentie):**

Omdat je via de Proxy pattern sneller en veiliger kan communiceren met andere klassen en entiteiten. En omdat het de functionaliteit ook verbetert van het programma.

**3.Wanneer(in welke situatie kan je ze toepassen met enkele praktische voorbeelden om dit te illustreren:**

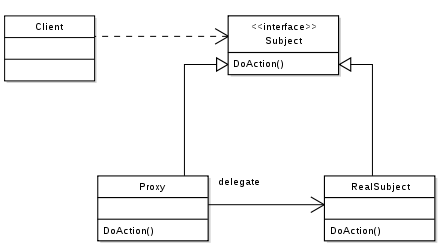
bijvoorbeeld de protection proxy wordt gebruikt om de toegang tot een bron te beheren op basis van toegangsrechten.

praktisch voorbeeld is je wilt geld afhalen bij de bank en je doet het via de bankautomaat. De bankautomaat werkt zogezegd als tussenpersoon voor de bank en die geeft dan door aan de bank. Jij kan zelf dan zien op het scherm van de bankautomaat hoeveel je nog hebt staan enz. en die geeft dat dan door naar de bank en omgekeerd.

**4.Welke gevolgen geeft de toepassing van deze pattern:**

De functionaliteit verbeteren van het programma.

**5.Structuur(met een UML diagram van de pattern (image):**



**6.Hoe wordt ze geïmplementeerd in c#:**

je maakt een extra klasse aan die als tussenpersoon dient tussen uw andere klassen.