INSTITUTO POLITÉCTICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA

Patrón de diseño

presenta:

Balderas Hernández David Vadhir

Díaz González Lizeth

Hernández García Jaime Gabriel

Meza Bravo Iván Marcelino

Ingeniería de Software

Fecha

13/04/2025

PATRÓN DE DISEÑO: FACTORY METHOD

El sistema se compone de varios módulos funcionales, entre ellos:

- Importación de datos (CU-01)
- Filtrado y clasificación de eventos sísmicos (CU-02)
- Cálculos matemáticos especializados (CU-03)
- Creación y gestión de nodos (CU-04, CU-06)
- Visualización (CU-05)
- Generación de reportes (CU-09)
- Predicción (CU-10)

Todos estos módulos trabajan con diferentes tipos de objetos, como:

- Eventos sísmicos (terremotos, réplicas, etc.)
- Nodos del grafo (por tipo: ubicación, magnitud, fecha, etc.)
- Archivos de entrada
- Estrategias de filtrado
- Algoritmos de predicción

Cada uno de estos objetos puede variar según el tipo de operación, origen o algoritmo, por lo que tener un sistema centralizado y flexible para crear objetos según su tipo es ideal.

Ventajas del Patrón Factory

Beneficio	Aplicación en el Sistema
Desacoplamiento	Cada módulo pide objetos sin saber cómo se crean internamente.
Reutilización	Las fábricas pueden ser compartidas entre diferentes partes del sistema.

Beneficio	Aplicación en el Sistema
Extensibilidad	Si agregas nuevos tipos de eventos, nodos o filtros, no necesitas modificar el código base.
Cohesión alta	Cada fábrica se encarga exclusivamente de construir un tipo de objeto.
Menor error humano	Centraliza la lógica de creación, evitando errores por mala instanciación manual.

Cómo el patrón mejora la calidad del diseño

Criterio	Mejora aportada por el patrón Factory
Mantenibilidad	Se pueden agregar nuevas operaciones sin alterar código existente.
Escalabilidad	Se puede extender con nuevas funcionalidades sin afectar a las demás.
Reusabilidad	Las fábricas y operaciones son reutilizables en distintos contextos.
Bajo acoplamiento	El sistema depende de la abstracción (Operacion), no de la implementación.
Alta cohesión	Cada clase tiene una única responsabilidad clara.

Ejemplos de Fábricas que puede tener el sistema

Fábrica	Objetos que crea
EventoFactory	EventoTerremoto, EventoReplica, EventoArtificial, etc.
NodoFactory	NodoUbicacion, NodoMagnitud, NodoFecha, etc.
FiltroFactory	FiltroPorFecha, FiltroPorMagnitud, etc.
AlgoritmoFactory	AlgoritmoTrilateracion, AlgoritmoPrediccion, etc.
ReporteFactory	ReportePDF, ReporteCSV, ReporteInteractivo, etc.

¿Por qué Factory es mejor aquí que Singleton u Observer?

- Singleton se usa para mantener una única instancia (útil para configuración o logging, pero no para objetos variables).
- Observer se usa cuando hay que reaccionar ante eventos (útil para la visualización o actualización en tiempo real, pero no para construcción).
- Factory es perfecto cuando el tipo exacto de objeto que se necesita puede variar y no quieres acoplar el sistema a clases concretas.

Diagrama de clases para método Factory