

vac-o
Documento de Diseño

Santiago Videla

16 de septiembre de 2010

Índice

1. Introducción	3
1.1. Propósito	3
1.2. Descripción general del documento	3
2. Diseño de alto nivel	4
2.1. Interfaces - Responsabilidades - Colaboradores	4
2.1.1. IPluginAdmin	4
2.1.2. IPlugin	4
2.1.3. ICombinatoryRegion	4
2.1.4. ICombinatoryEngine	6
2.1.5. IQAMutator	6
2.1.6. IQAValidator	6
2.1.7. IQARegion	6
2.1.8. IQAEngine	6
2.1.9. IRanker	7
3. Diseño de bajo nivel	8

1. Introducción

1.1. Propósito

El propósito de este documento es presentar el diseño de vac-o. La técnica adoptada para realizar el análisis y descripción del mismo se denomina “Diseño dirigido por responsabilidades” (Responsability-driven design) descrito en un *paper* de Rebecca Wirfs-Brock y Brian Wilkerson, “Object-Oriented Design: A Responsibility-Driven Approach”.

1.2. Descripción general del documento

En la sección 2 se presenta el diseño de alto nivel del sistema, sus interfaces y paquetes principales.

En la sección 3 se presenta el diseño de bajo nivel del sistema, las clases concretas y sus relaciones para cada paquete.

2. Diseño de alto nivel

2.1. Interfaces - Responsabilidades - Colaboradores

En esta sección se presentan las principales interfaces que intervienen en el sistema, sus respectivas responsabilidades y colaboradores. En la Figura 1 se puede ver el diagrama UML correspondiente.

2.1.1. IPluginAdmin

Responsabilidad: Administrar las extensiones del sistema (archivos *.so*).

1. Cargar extensión.

2.1.2. IPlugin

Responsabilidad: Brindar la información y servicios particulares para una vacuna determinada.

1. Proveer la lista de parámetros requeridos por la extensión.
2. Proveer la secuencia de ARN que se encuentra en la cepa vacunal.
3. Proveer las regiones combinatorias que se deben usar para buscar mejoras a la vacuna.
4. Proveer el umbral que se debe usar para determinar la bondad de las secuencias obtenidas de las regiones combinatorias.
5. Proveer las regiones de validación que se deben usar para realizar el control de calidad.
6. Determinar si se continua buscando secuencias o no.
7. Evaluar las secuencias candidatas.
8. Descargar la extensión.

Colaboradores:

1. **BioPP:** Calculo de distancias entre secuencias.
2. **LibPlugin:** Factories de regiones combinatorias y de validación.

2.1.3. ICombinatoryRegion

Responsabilidad: Calcular las secuencias que mantengan determinadas propiedades de una secuencia original.

1. Devolver la siguiente secuencia.
2. Evaluar la bondad de una secuencia.

Colaboradores:

1. **BioPP:** Calculo de secuencias que mantengan una propiedad determinada (estructura secundaria, código genético, etc).

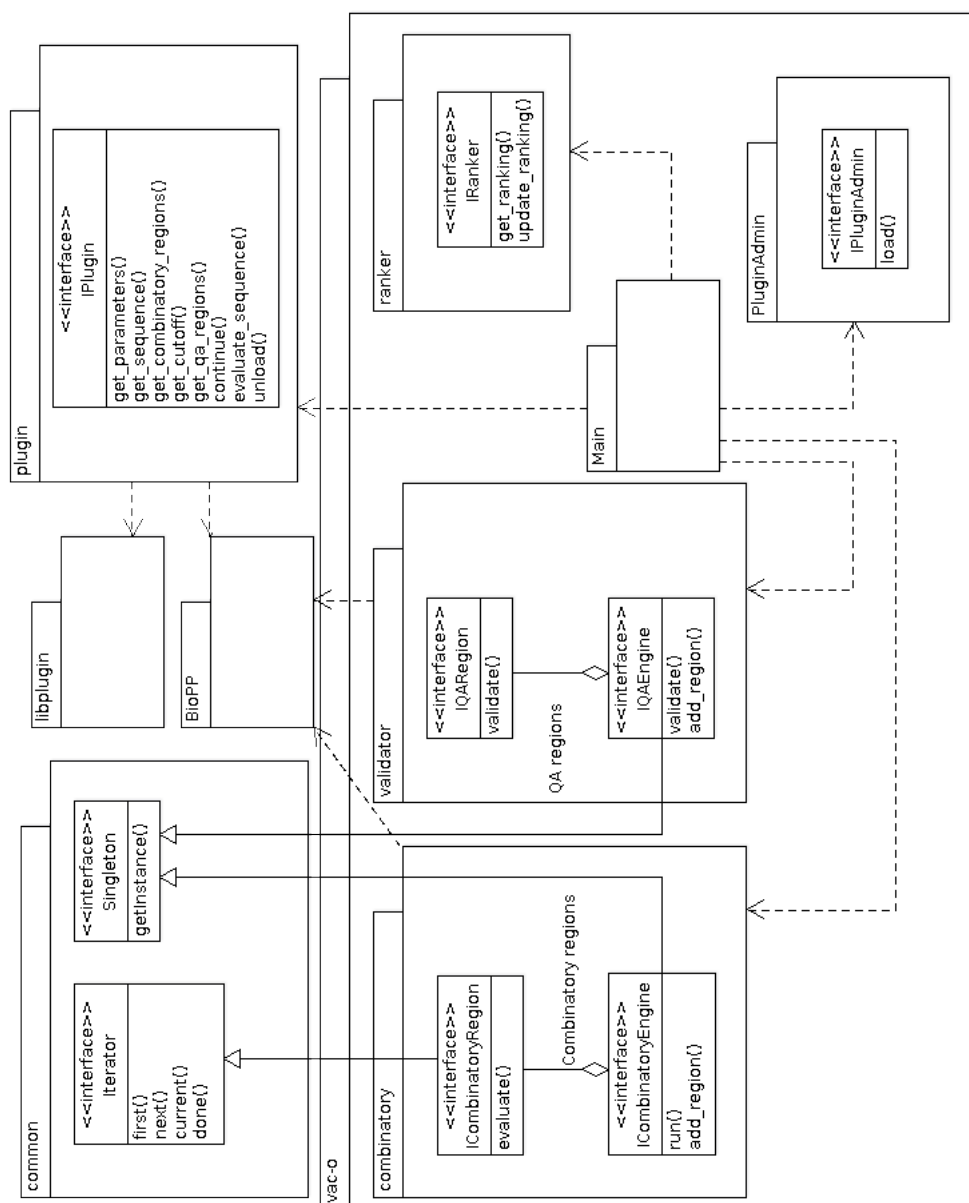


Figura 1: UML - Interfaces

2.1.4. ICombinatoryEngine

Responsabilidad: Generar secuencias candidatas a partir de las variantes generadas de cada región combinatoria.

1. Recorrer el espacio de búsqueda generado por las variantes de las regiones combinatorias.

Colaboradores:

1. **ICombinatoryRegion:** Consulta la siguiente secuencia de cada región combinatoria.

2.1.5. IQAMutator

Responsabilidad: Generar mutaciones de una secuencia utilizando algún criterio (todas las mutaciones posibles, N mutaciones aleatorias, etc).

1. Devolver la siguiente mutación.

2.1.6. IQAValidator

Responsabilidad: Decidir si una secuencia satisface o mantiene determinadas propiedades.

Colaboradores:

1. **BioPP:** Calculo y comparación de estructura secundaria o conteo de nucleótidos

2.1.7. IQARegion

Responsabilidad: Realizar el control de calidad para una región de validación.

1. Calcular y validar mutaciones acumuladas de la región hasta alcanzar la profundidad deseada.

Colaboradores:

1. **IQAMutator:** Consulta la siguiente mutación.
2. **IQAValidator:** Consulta si una mutación determinada es valida o no.

2.1.8. IQAEngine

Responsabilidad: Realizar el control de calidad para una secuencia candidata.

1. Inicializar regiones de validación.
2. Determinar si una secuencia candidata aprueba o no el control de calidad para todas sus regiones de validación.

Colaboradores:

1. **IQARregion:** Consulta si la región de validación aprueba o no el control de calidad.

2.1.9. IRanker

Responsabilidad: Mantener un *ranking* de secuencias en base al puntaje asignado a cada una.

Colaboradores:

1. **IPlugin:** Consulta el puntaje para una secuencia determinada.

3. Diseño de bajo nivel

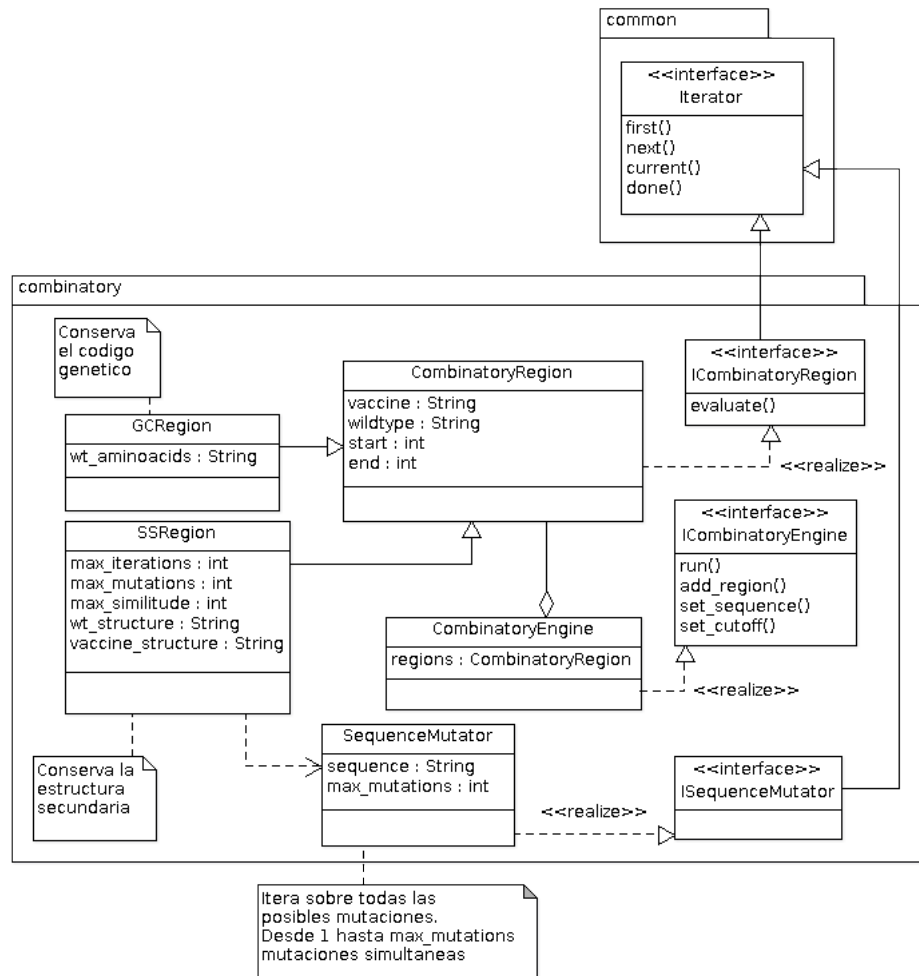


Figura 2: UML - Combinatory

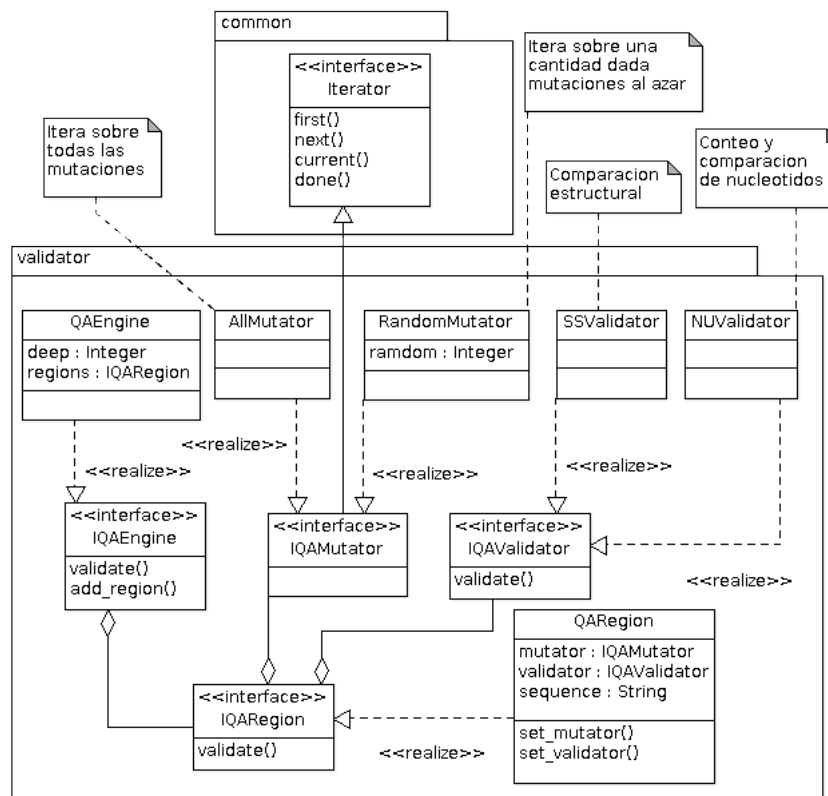


Figura 3: UML - Validator