Especificación de Diseño de Software

Santiago Videla

20 de septiembre de 2010

Índice

1. Introducción

1.1. Propósito

El propósito de este documento es la especificación de diseño de software para la primer versión del producto "vac-o".

La confección de este documento se enmarca dentro del desarrollo de la tesis de grado de la carrera Lic. en Cs. de la Computación de la FaMAF - UNC, **"Diseño de vacunas atenuadas con menor probabilidad de sufrir reversión a la virulencia"** a cargo de Santiago Videla, con la dirección de la Dra. Laura Alonso i Alemany (FaMAF) y la colaboración de Daniel Gutson (FuDePAN).

El documento esta dirigido a las personas involucradas en el desarrollo de la tesis como así también a los colaboradores de FuDePAN que eventualmente podrían participar en las etapas de desarrollo y mantenimiento del software.

1.2. Descripción general del documento

En la sección 3 se mencionan los objetivos, la metodología adoptada y las dependencias del diseño.

En la sección 4 se muestra la arquitectura del sistema con sus principales componentes e interacciones.

En la sección 5 se presenta el diseño de alto nivel del sistema, sus interfaces y paquetes principales.

En la sección 6 se presenta el diseño de bajo nivel del sistema, las clases concretas y sus relaciones para cada paquete.

2. Descripción General

3. Consideraciones de Diseño

3.1. Objetivos

Principalmente se pretende lograr un diseño que cumpla con los principios fundamentales del diseño orientado a objetos, comúnmente conocidos por el acrónimo "SOLID" [1].

3.2. Metodología

La metodología utilizada para realizar el análisis y descripción del diseño se denomina "Diseño dirigido por responsabilidades" [2]. Esta técnica se enfoca en *que* acciones (responsabilidades) deben ser cubiertas por el sistema y que objetos serán los responsables de llevarlas a cabo. *Como* se realizara cada acción, queda en un segunda plano.

3.3. Dependencias

Se asume para la confección de este diseño, el acceso a diferentes librerías externas que serán fundamentales para el correcto funcionamiento del sistema en su conjunto. Respetando la metodología adoptada, no se hará referencia directa

a una u otra librería, sino a los servicios que la misma debe ser capaz de proveer al sistema.

4. Arquitectura del Sistema

En la Figura 1 se presenta un diagrama de la arquitectura del sistema y la interacción entre sus componentes principales. A continuación se da una breve descripción de cada uno de ellos:

- Main: Representa el componente principal en términos de ejecución del sistema. Comprende principalmente, la inicialización y configuración de otros componentes.
- Plugin: Representa la implementación de las características propias de la vacuna que se desea optimizar. Brinda información de configuración inicial como así también los criterios para evaluar una secuencia.
- CombinatoryEngine: Representa el motor combinatorio del sistema encargado de encontrar, nuevas secuencias que sean candidatas a optimizar la atenuación de la vacuna.
- QAEngine: Representa el motor de control de calidad del sistema encargado de decidir, si una secuencia candidata pasa o no dicho control.
- Ranker: Representa el componente encargado de evaluar y mantener un "ranking" de secuencias.
- libRNA: Representa el componente que provee al sistema funcionalidades para la manipulación de secuencias de ARN ("folding" directo e inverso, entre otras)

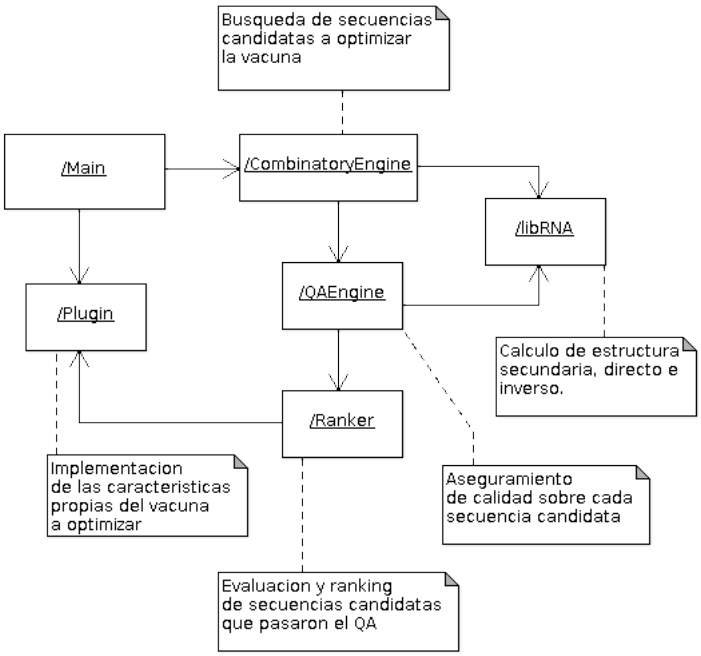

Figura 1: UML - Arquitectura

5. Diseño de alto nivel

5.1. Interfaces - Responsabilidades - Colaboradores

En esta sección se presentan las principales interfaces que intervienen en el sistema, sus respectivas responsabilidades y colaboradores. En la Figura 2 se puede ver el diagrama UML correspondiente.

5.1.1. IPluginAdmin

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

1. Cargar extensión.

5.1.2. 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 regiónes 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.

5.1.3. ILibRNAAdmin

Responsabilidad: Administrar la librería externa utilizada para la manipulación de secuencias ARN.

1. Cargar librería

5.1.4. ILibRNA

Responsabilidad: Proveer al sistema funcionalidades para la manipulación de secuencias ARN.

1. Folding directo
2. Folding inverso

Colaboradores:

1. Vienna Package, UNAFold, otros.

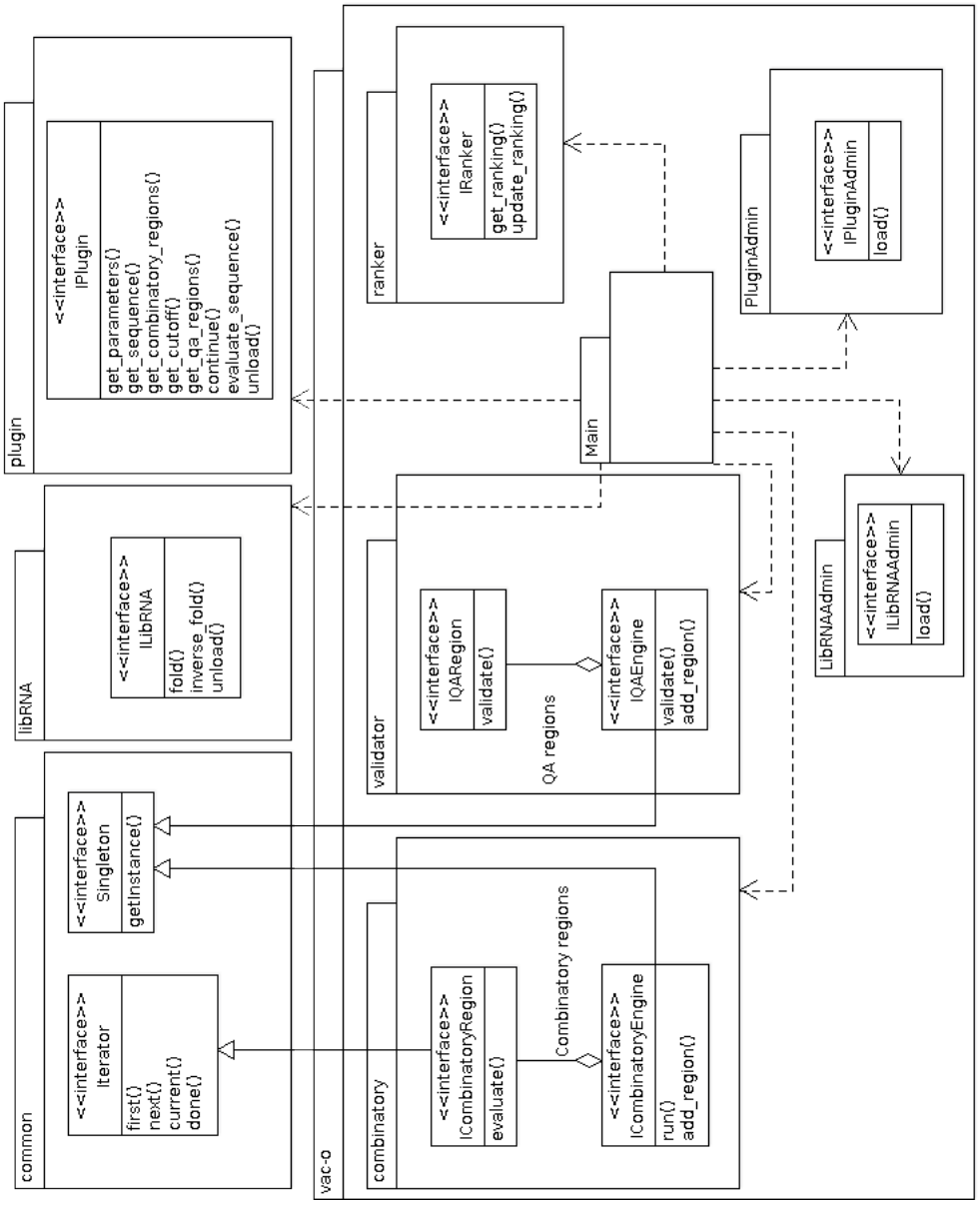

Figura 2: UML - Interfaces

5.1.5. 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. ILibRNA: Diseño de secuencias que mantengan una propiedad determinada (estructura secundaria).

5.1.6. 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.

5.1.7. 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. ILibRNA: Predicción de estructura secundaria.

5.1.8. IQAEngine

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

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

Colaboradores:

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

5.1.9. 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.

6. Diseño de bajo nivel

6.1. Clases concretas

En esta sección se presentan las clases concretas que implementan las interfaces presentadas en la sección 5. Para mayor claridad, se dividieron los diagramas UML por paquetes.

6.1.1. Combinatory

En la Figura 3 se puede ver el diagrama de clases para el paquete "Combinatory".

6.1.2. Validator

En la Figura 4 se puede ver el diagrama de clases para el paquete "Validator".

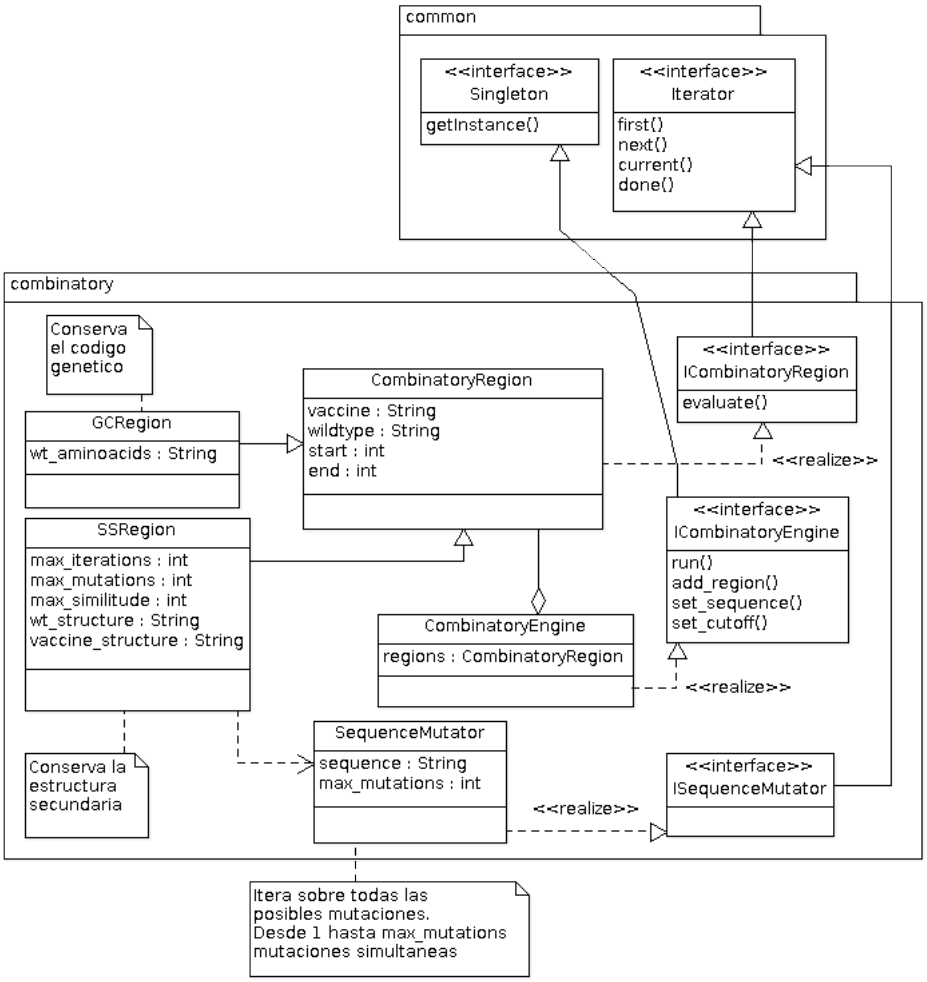

Figura 3: UML - Combinatory

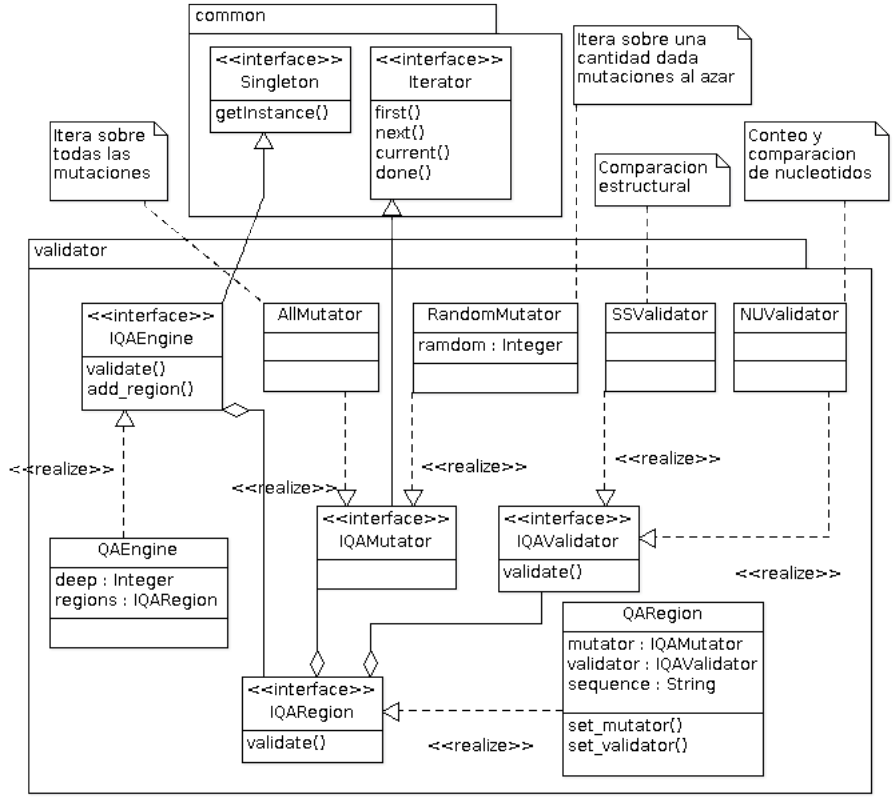

Figura 4: UML - Validator

Referencias

[1] Design Principles and Design Patterns. Robert C. Martin, www.objectmentor.com, 2000.

[2] Rebecca Wirfs-Brock and Alan McKean. Object Design: Roles, Responsibilities and Collaborations, Addison-Wesley, 2003