

# **RNAemo**

## Especificación de Diseño de Software

Franco Gaspar Riberi

20 de junio de 2012

# Índice

<b>1. Introducción</b>	<b>3</b>
1.1. Propósito . . . . .	3
1.2. Descripción general del documento . . . . .	3
<b>2. Consideraciones de diseño</b>	<b>3</b>
2.1. Objetivos . . . . .	3
2.2. Metodología . . . . .	4
2.3. Herramientas y convenciones . . . . .	4
<b>3. Arquitectura del sistema</b>	<b>4</b>
<b>4. Diseño de alto nivel</b>	<b>7</b>
4.1. Interfaces - Responsabilidades - Colaboradores . . . . .	7
4.1.1. IValidator . . . . .	7
4.1.2. IFold . . . . .	8
4.1.3. ISequence . . . . .	8
4.1.4. IData . . . . .	8
<b>5. Diseño de bajo nivel</b>	<b>10</b>
5.1. Paquetes y clases concretas . . . . .	10
5.1.1. Validator . . . . .	10
5.1.2. RNAmDB . . . . .	10
5.1.3. Matching . . . . .	11
5.1.4. Generator . . . . .	11
5.1.5. fideo . . . . .	11

## 1. Introducción

### 1.1. Propósito

El propósito de este documento es la especificación de diseño de software correspondiente a la primera versión del producto ***RNAemo***.

La confección de este documento se contextualiza dentro del desarrollo de la tesis de grado de la carrera Licenciatura en Ciencias de la Computación de la UNRC, ***RNAemo***, a cargo de Franco Gaspar Riberi, con la dirección de la Lic. Laura Tardivo (UNRC) y las colaboraciones de Daniel Gutson, el Lic. Guillermo Biset y el Dr. Roberto Daniel Rabinovich (**FuDePAN**).

El documento está dirigido a las personas involucradas en el desarrollo de la tesis como así también a todos 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 2 se mencionan los objetivos, la metodología adoptada y las dependencias del diseño.

En la sección 3 se exhibe la arquitectura general del sistema con sus principales componentes e interacciones.

En la sección 4 se presenta el diseño de alto nivel del sistema, sus interfaces y paquetes principales.

En la sección 5 se observa el diseño de bajo nivel del sistema, esto involucra las clases concretas y sus relaciones para cada paquete.

## 2. Consideraciones de diseño

### 2.1. Objetivos

Se pretende lograr un diseño del sistema que cumpla con los principios fundamentales del diseño orientado a objetos, comúnmente conocidos por el acrónimo “**SOLID**” [1].

En particular, se pretenden respetar los principios **SRP** (*Single Responsibility Principle*), **OCP** (*Open-Closed Principle*) y **DIP** (*Dependency Inversion Principle*) debido a su importancia para obtener un sistema fácilmente extensible y configurable con el fin de satisfacer las necesidades de los usuarios.

## 2.2. Metodología

La metodología empleada para realizar el análisis y descripción del diseño se denomina “*Diseño dirigido por responsabilidades*”[2].

Esta técnica se enfoca en *qué* acciones (responsabilidades) deben ser cubiertas por el sistema y que objetos serán los responsables de llevarlas a cabo. *Cómo* se realizara cada acción, queda en un segunda plano.

## 2.3. Herramientas y convenciones

Se utiliza UML[3] como lenguaje de modelado, ArgoUML[4] como herramienta para la confección de diagramas, y Dia[5] para la edición de diagramas de propósito general. Además se adopta la convención de nombrar a las interfaces anteponiendo una letra “*I*” al nombre de la clase concreta que la implementa.

Por ejemplo, interface: “*IPersona*” → clase concreta: “*Persona*”.

## 3. Arquitectura del sistema

La arquitectura del sistema y la interacción entre los diversos componentes que la conforman se exhiben en la figura 1. A continuación se describe brevemente cada uno de los módulos que comprenden el sistema.

- **MasterOfPuppets:** corresponde al componente principal en términos de ejecución del sistema. Comprenderá la inicialización e invocación de los demás componentes. Representa el módulo encargado de contabilizar y generar las tablas y gráficos que se esperan como salida de este software.
- **Generador:** representa el módulo encargado de la generación de secuencias humanizadas. Dada una secuencia original, genera la secuencia humanizada de la misma. Dicho módulo es externo a este desarrollo, y para ello se empleará el software *GeneDesign*<sup>1</sup>.
- **Matching:** representa el componente encargado de realizar el matching por complemento y el cálculo de score entre secuencias de RNA<sub>m</sub> y secuencias de small-RNA<sub>s</sub>.
- **RNAmDB:** representa el componente que provee al sistema las cadenas correspondientes para su uso.

---

<sup>1</sup>Descarga de: <http://www.xmarks.com/site/slam.bs.jhmi.edu/gd/>

- **DataBase:** representa la base de datos de small-RNA<sub>s</sub> (particularmente *mi*RNA).
- **BLAST:** corresponde a un módulo externo el cual permite el alineamiento de secuencias. Básicamente, compara una secuencia con una gran cantidad de secuencias administradas en una base de datos.
- **fideo:** corresponde a una librería parcialmente ya implementada. Representa el componente que provee al sistema la funcionalidad de “*folding*”. Se deberá agregar la implementación de UNAFold y MFold.
- **Validator:** representa el módulo encargado de realizar el control de calidad del sistema. Es el componente encargado de decidir, si una secuencia se encuentra en el marco de lectura correcto o no.
- **Outputs:** refiere a los archivos con formato CSV que serán obtenidos como resultado del presente producto.

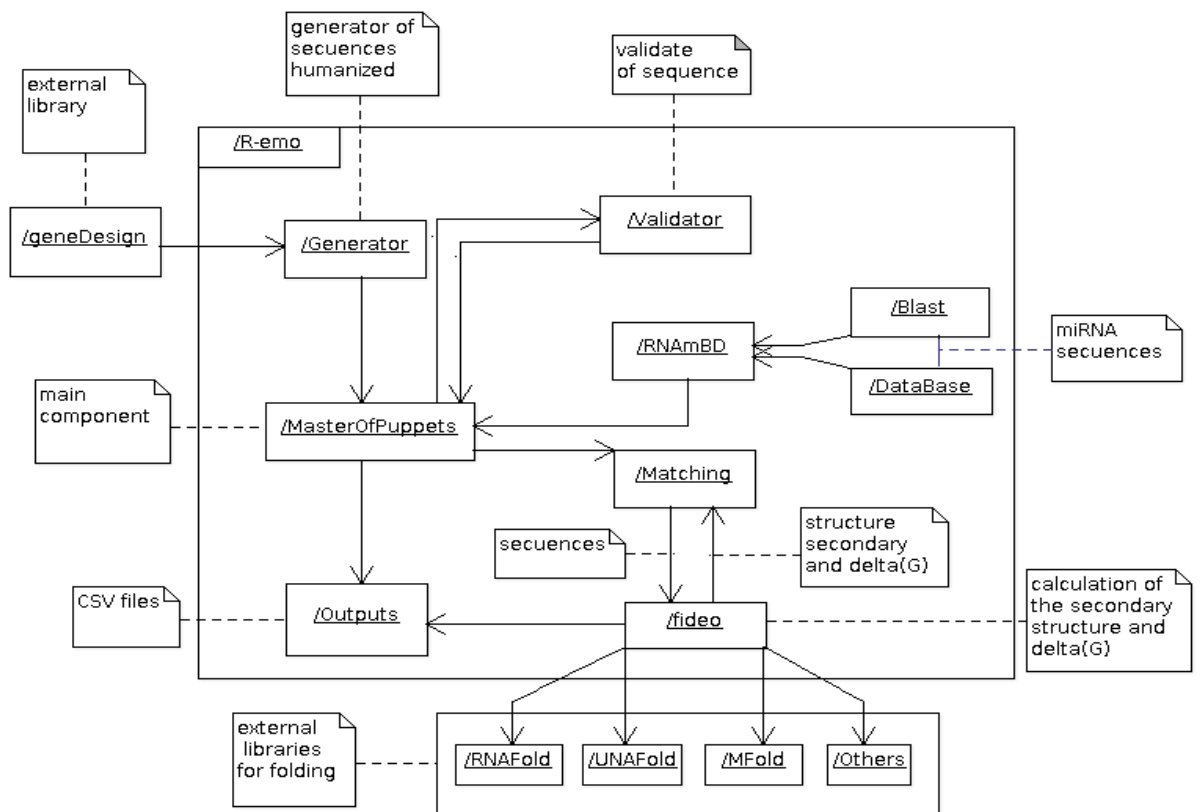


Figura 1: UML - Arquitectura del Sistema

## 4. Diseño de alto nivel

### 4.1. 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 exhibe el diagrama de interfaces correspondiente.

Finalmente, en la figura 3 se presenta el diagrama de secuencia correspondiente a la comunicación entre las principales entidades del sistema.

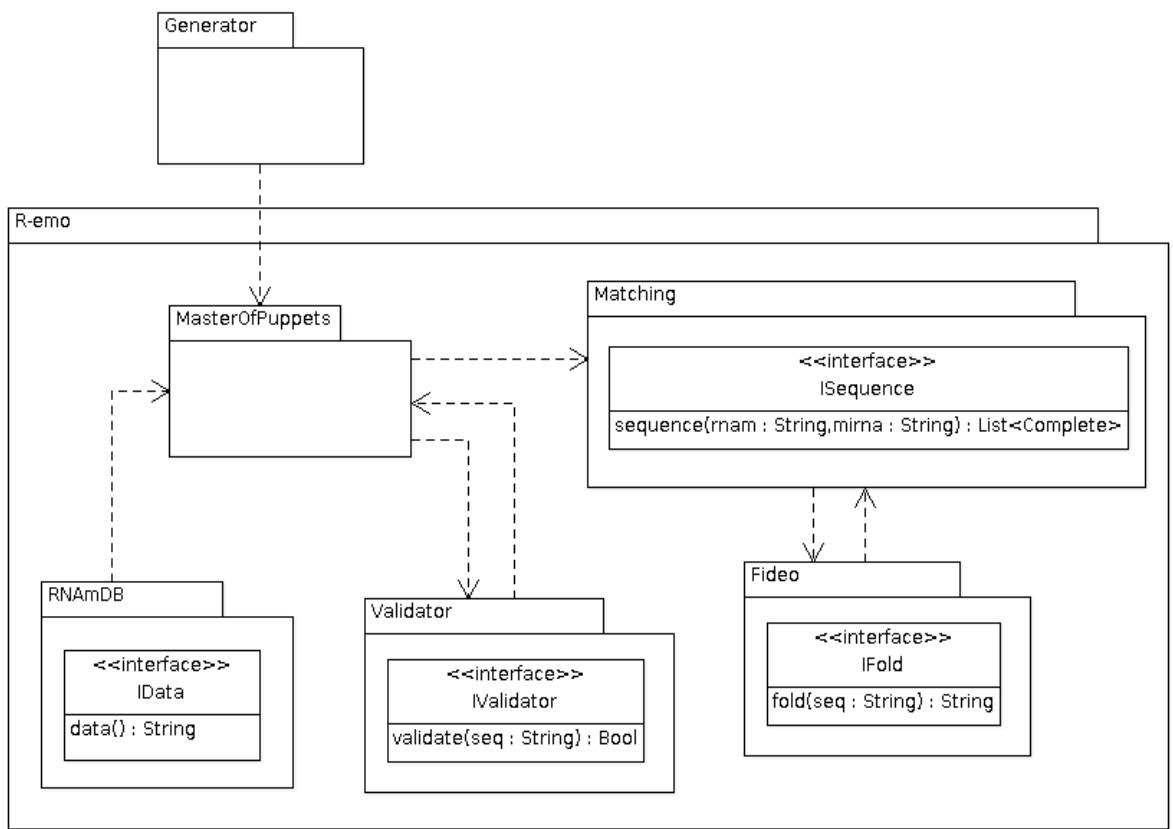


Figura 2: UML - Interfaces

#### 4.1.1. IValidator

**Responsabilidad:** Realizar el control de calidad para las secuencias. Determinar si una secuencia aprueba o no el control de calidad.

**Colaboradores:**

#### 4.1.2. IFold

**Responsabilidad:** Proveer al sistema el “*folding*” de secuencias de RNA.

**Colaboradores:**

1. Vienna Package (RNAFold).
2. UNAFold.
3. MFold.
4. Otros.

#### 4.1.3. ISequence

**Responsabilidad:** Provee al sistema el “*matching*” de secuencias.

**Colaboradores:**

#### 4.1.4. IData

**Responsabilidad:** Provee al sistema las secuencias de small-RNA<sub>s</sub>.

**Colaboradores:**

1. DataBase
2. BLAST



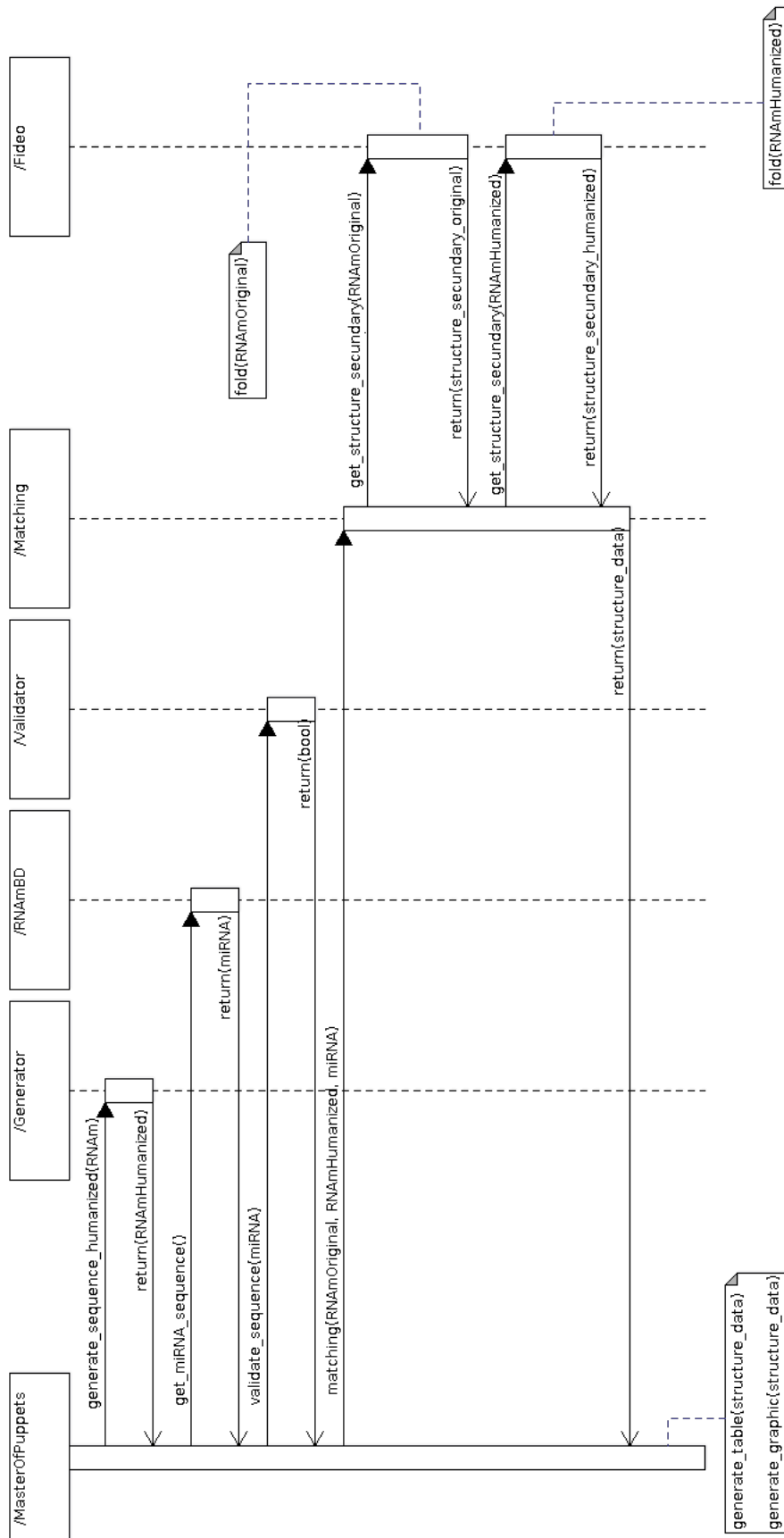


Figura 3: UML - Pasaje de mensajes

## 5. Diseño de bajo nivel

### 5.1. Paquetes y clases concretas

En esta sección se presentan las clases concretas que implementan las interfaces presentadas en la sección 4. Para mayor claridad, se dividieron los diagramas UML por paquetes.

#### 5.1.1. Validator

En la figura 4 se exhibe el diagrama de clases correspondiente al componente **validator**. Este paquete representa el módulo “Validator” que se observa en la figura 1 en la sección 3.

La responsabilidad de este componente es realizar un control de calidad sobre las secuencias de small-RNA<sub>s</sub> determinando si las mismas se encuentran en un marco de lectura válido. Para garantizar esta calidad, es necesario traducir la cadena de nucleótidos a una cadena de aminoácidos y verificar que no queden nucleótidos libres, conocidos como *codones stops*. Para ello se utilizará la librería *BioPP*.

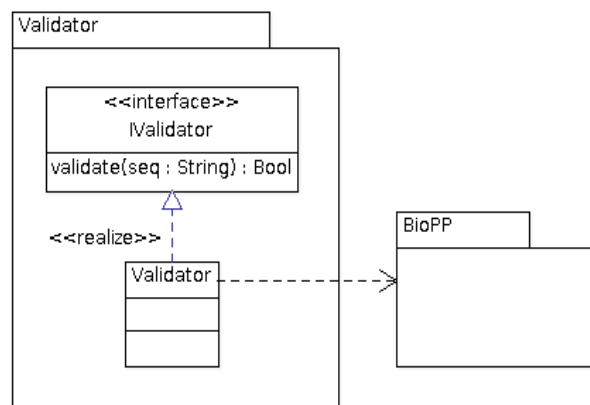


Figura 4: UML - Validator

#### 5.1.2. RNAmDB

**complete**

### 5.1.3. Matching

En la figura 5 se observa el diagrama de clases para el paquete **Matching**. Este paquete representa el componente **Matching** correspondiente a la figura 1 de la sección 3.

La responsabilidad de este componente es realizar el matching entre secuencias y calcular el score de matching. Para ello, se generarán diversas cadenas por cada posición del  $\text{RNA}_m$  resultantes de matching diferentes.

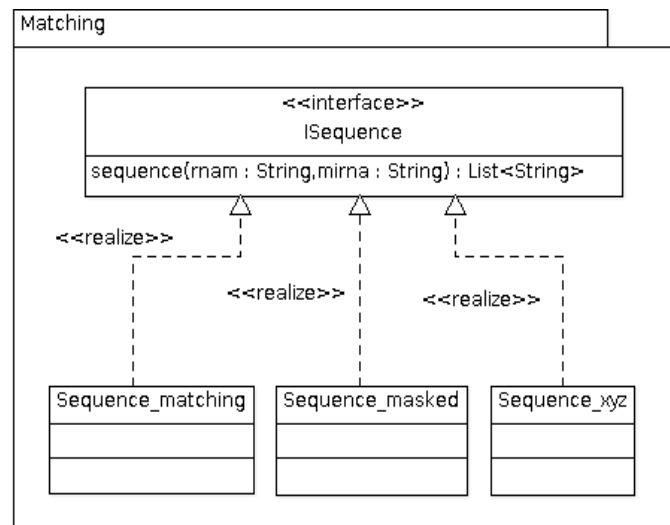


Figura 5: UML - Matching

### 5.1.4. Generator

**complete**

### 5.1.5. fideo

En la figura 6 se observa el diagrama de clases para el paquete **fideo**. Este paquete representa el componente **fideo** correspondiente a la figura 1 de la sección 3.

La responsabilidad de este componente es brindar al sistema el servicio de “*folding*” sobre secuencias de  $\text{RNA}_m$ . Para cumplir con esta responsabilidad, se ofrece al sistema el acceso a librerías externas de manera transparente y permitiendo utilizar diferentes librerías para acceder a diferentes servicios.

Se contemplan los siguientes tipos de folding:

- RNAFold.
- UNAFold.
- MFold.

La importancia de este paquete y las interfaces que contiene radica en que permite abstraerse del uso de una u otra librería.

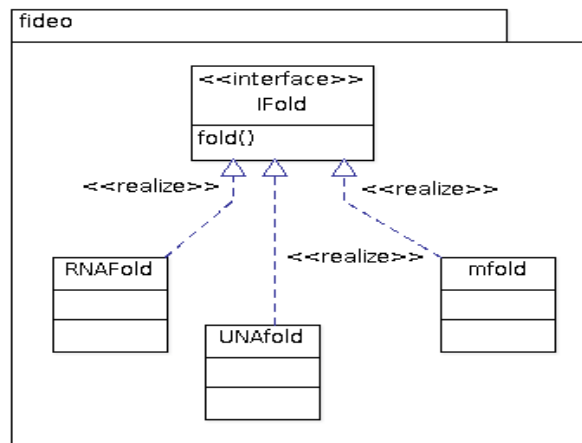


Figura 6: UML - fideo

## Referencias

- [1] “*Design Principles and Design Patterns.*” ROBERT C. MARTIN, 2000.  
<http://www.objectmentor.com>
- [2] “*Object Design: Roles, Responsibilities.*” REBECCA WIRFS-BROCK AND  
ALAN MCKEAN AND COLLABORATIONS, Addison-Wesley, 2003.
- [3] “*Unified Modeling Language.*” <http://www.uml.org/>
- [4] “*ArgoUML.*” <http://argouml.tigris.org/>
- [5] “*Dia.*” <http://live.gnome.org/Dia>