Especificación de Requerimientos de Software

Santiago Videla

19 de abril de 2010

Índice

1. Introducción

1.1. Propósito

El propósito de este documento es la especificación de requerimientos de software en el marco 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"**. Los requerimientos del software son provistos por integrantes de FuDePAN en su carácter de autores intelectuales de la solución que se pretende implementar y colaboradores de dicha tesis.

A continuación se enumeran las personas involucradas en el desarrollo de la tesis y que por lo tanto, representan la principal audiencia del presente documento.

- Dra. Laura Alonso Alemany: Directora de tesis, FaMAF
- Daniel Guston: Colaborador de tesis, FuDePAN
- Santiago Videla: Tesista, FaMAF

1.2. Alcance

El producto que se especifica en este documento se denomina **"vac-o"** y su principal objetivo es encontrar secuencias genómicas de patógenos que mantengan funcionalidad como vacunas con una probabilidad mínima de sufrir reversión a la virulencia. El producto final debe proveer al usuario, la capacidad de hacer uso del software mediante extensiones o *plugins* que implementen las características particulares para cada patógeno.

La principal responsabilidad de vac-o es la de proveer a las extensiones, un motor combinatorio de secuencias genómicas utilizando diferentes estrategias computacionales que conduzcan a la optimización de los cálculos. Por otro lado, las extensiones son responsables de evaluar las secuencias generadas asignando un puntaje a cada una con el fin de que vac-o genere el listado de secuencias resultante.

En su versión inicial, vac-o incluye una extensión para la vacuna Sabin contra la poliomielitis que también se especifica en este documento y podría ser tomada como guía para futuras extensiones.

1.3. Descripción general del documento

La estructura de este documento sigue las recomendaciones de la "Guía para la especifiación de requerimientos de la IEEE" (IEEE Std 830-1998).

En la sección 2 se presenta una descripción general de vac-o, sus principales funcionalidades, interfaces y perfiles de usuarios.

En la sección 3 se detallan los requerimientos funcionales específicos de vac-o y los principales atributos que debe cumplir el software. Además se incluye la especificación de una extensión para la vacuna Sabin contra la poliomielitis.

2. Descripción General

2.1. Perspectiva del Producto

Al momento de la confección de este documento, no existen productos de software que brinden las funcionalidades que se pretenden implementar. En este sentido, vac-o representa una innovación en el área de las bioinformaticas, cuyo principal objetivo es la optimización de vacunas basadas en virus atenuados.

Uno de los problemas de estas vacunas es su potencialidad de revertir a la virulencia. En sucesivas replicaciones, los virus atenuados pueden acumular mutaciones que les devuelvan el carácter patogénico y de esa manera podrían producir enfermedad en el vacunado o sus contactos.

Para lograr su objetivo, vac-o debe ser capaz de encontrar las variantes del virus con menor probabilidad de revertir a la patogenia. Partiendo de la secuencia genómica que se encuentra en la cepa vacunal, se deben buscar las modificaciones de esta secuencia que cumplan las siguientes propiedades:

- Que originen una respuesta inmune protectora contra el virus salvaje.
- Que tengan muy baja probabilidad de mutar hasta convertirse en un virus patógeno.

Entre todas las secuencias encontradas, vac-o debe construir una lista o *ranking* de secuencias con sus respectivos puntajes. Para lograr esto, se deberá implementar un *plugin* que brinde la información y las características propias del virus. En particular, cada *plugin* debe especificar las regiones combinatorias de interés y ser capaz de asignar un puntaje a cada secuencia comparándola con la secuencia original.

En la figura 1 se presenta el flujo de trabajo y funcionamiento general de vac-o. Notar que este esquema es puramente conceptual y no hace ninguna referencia a los detalles de implementación. La intención es clarificar al lector los diferentes “actores” y sus responsabilidades.

2.1.1. Interfaces del Sistema

No se registran requerimientos.

2.1.2. Interfaces de Usuario

La interfaz con el usuario final consiste de dos formularios y un listado con las secuencias resultantes. Inicialmente, el usuario de vac-o deberá indicar la ubicación o *path* de la extensión que desea usar. El siguiente paso, sera completar un formulario con los datos requeridos por la extensión elegida. Finalmente, se da comienzo a la ejecución y se recibe como resultado, la lista de secuencias con sus respectivos puntajes.

2.1.3. Interfaces de Hardware

No se registran requerimientos.

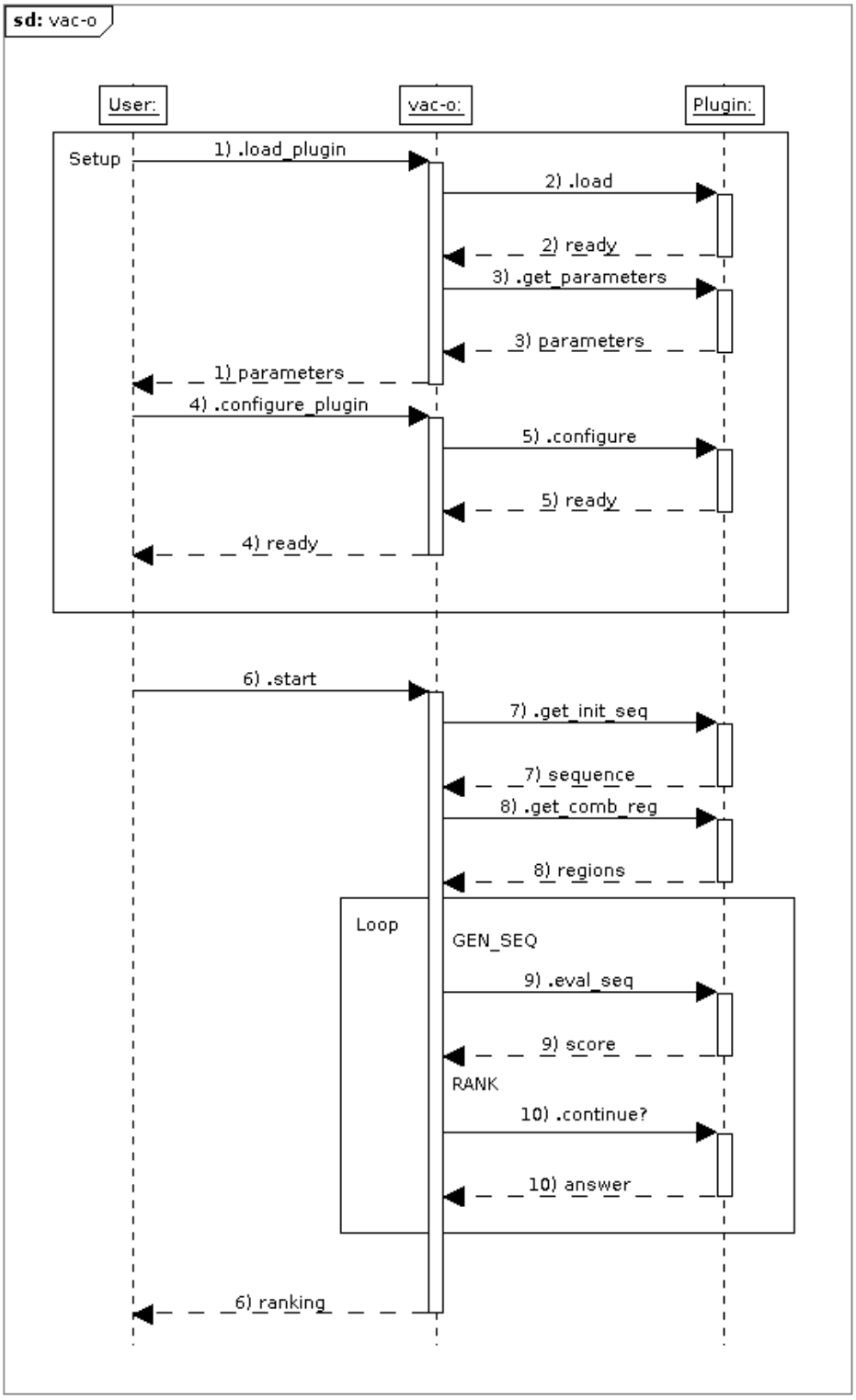

Figura 1: Flujo de trabajo de vac-o

2.1.4. Interfaces de Software

- BioPP: Se debe utilizar la librería BioPP para realizar el *folding* de secuencias genómicas, predicción de la estructura secundaria y cálculo de distancias.
- Boost.Python: Se debe utilizar Boost.Python para brindar a los desarrolladores de extensiones, la posibilidad de hacerlo utilizando el lenguaje de programación Python.
- QT: Se debe utilizar QT para el desarrollo de la interfaz de usuario.

2.1.5. Interfaces de Comunicaciones

No se registran requerimientos.

2.1.6. Restricciones de Memoria

No se registran restricciones de memoria para la ejecución del software. No obstante, se debe tener en cuenta que dada la naturaleza y la complejidad del problema, el tiempo de calculo estará directamente relacionado con la memoria disponible.

2.1.7. Operaciones

No se registran requerimientos.

2.1.8. Requerimientos de Instalación

No se registran requerimientos.

2.2. Funciones del Producto

2.2.1. Configuración inicial

Inicialmente, vac-o presenta al usuario final una pantalla con un formulario donde el usuario debe indicar la ubicación de la extensión que desea usar. A continuación, vac-o carga la extensión en memoria y presenta al usuario otro formulario con los campos requeridos por la extensión cargada. Una vez que los datos son validados, vac-o configura la extensión con los valores enviados por el usuario y queda listo para comenzar la ejecución cuando el usuario lo indique.

2.2.2. Configuración de vac-o

Antes de dar comienzo a la generación de secuencias genómicas, vac-o debe solicitar a la extensión cargada los siguientes valores:

- Secuencia inicial: La secuencia genómica que se encuentra en la cepa vacunal y se desea mejorar.
- Regiones combinatorias: Las regiones de la secuencia inicial que resultan de interes, indicando sus posiciones de inicio y fin, y que tipo de regiones son cada una (restricciones en base a código genético, o en base a estructura secundaria).

- Estrategias de busqueda: La estrategia que se debe utilizar para la generación o busqueda de nuevas secuencias.

2.2.3. Generación de secuencias

Luego de las configuraciones basicas, vac-o esta en condiciones de generar nuevas secuencias genómicas que cumplan las restricciones que hallan sido impuestas. Para esto, se deben calcular las posibles "mutaciones" de cada region combinatoria (teniendo en cuenta el tipo de región y sus posibles intersecciones) y de acuerdo a la estrategia de busqueda, construir nuevas secuencias que seran evaluadas posteriormente por la extensión. Dado que el espacio de busqueda podría ser eventualmente muy grande, el criterio de parada para la generación de secuencias también debe ser provisto por la extensión.

Notar que esta funcionalidad es el "corazon" de vac-o y es donde se encuentra la mayor complejidad del problema a resolver.

Supongamos $N \geqslant 1$ y $R_1, \cdots, R_N$ regiones combinatorias. Luego, tendremos:

$$\begin{array}{c} M_{1\cdot 1}, ..., M_{1\cdot j_1} \text{ mutaciones de } R_1 \\ M_{2\cdot 1}, ..., M_{2\cdot j_2} \text{ mutaciones de } R_2 \\ \cdots \\ M_{N\cdot 1}, ..., M_{N\cdot j_N} \text{ mutaciones de } R_N \end{array}$$

con $j_1, \cdots, j_N \geqslant 1$. Es decir, que el número de posibles secuencias que vac-o "deberia" evaluar sera igual al producto $j_1 \times j_2 \times \cdots \times j_N$.

Haciendo uso de diferentes estrategias de busqueda, combinadas con las funciones de evaluación provistas por la extensión, vac-o intentara optimizar esta generación de secuencias.

2.2.4. Evaluación de secuencias

De acuerdo a los puntajes asignados por la extensión a cada una de las secuencias generadas, vac-o debe construir un listado o *ranking* de secuencias que sera dado al usuario como resultado final de la ejecución.

2.3. Características de Usuarios

Se identifican 3 tipos de usuarios de vac-o:

- Final: que solo interactúa a través de la interfaz gráfica.
- Extensionista: que posee los conocimientos de programación suficientes como para implementar extensiones de vac-o. A los fines de ampliar los potenciales usuarios dentro de esta categoría, es que se ofrece la posibilidad de desarrollar extensiones usando el lenguaje Python.
- Contribuidor: que contribuye al código fuente de vac-o realizando mejoras o desarrollando nuevas funcionalidades.

2.4. Restricciones

El producto debe ser desarrollado utilizando el lenguaje de programación C++ y bajo la licencia de software GPLv3.

2.5. Suposiciones y Dependencias

- Sistema operativo: GNU/Linux

2.6. Trabajo Futuro

Probablemente una de las principales mejoras a la primer versión de vac-o, sera realizar las modificaciones necesarias para permitir la ejecución del software en paralelo y de esta manera reducir los tiempos de cálculo. Por otro lado, se debera profundizar en el desarrollo de extensiones para diferentes tipos de vacunas y virus.

3. Requerimientos

3.1. Funciones del Sistema

3.1.1. Configuración inicial

El objetivo de esta función es la carga en memoria de la extensión que se desea utilizar para ejecutar vac-o.

1. **Carga de la extensión:** El sistema recibe la ubicación del archivo *.so* y lo carga en memoria. Si la carga es exitosa, se devuelven los parametros requeridos por la extensión (*[(nombre, tipo, validaciones, defecto)]*) para su posterior configuración. En caso de que la carga de la extensión fracase, se devuelve un mensaje de error.

2. **Validación de valores:** La interfaz de usuario (QT) es responsable de la validación de los datos ingresados por el usuario para la configuración de la extensión. Para esto, se deberan usar los diferentes *criterios de validación* para cada parametro y solo cuando la validación sea exitosa, los valores son enviados al sistema. Caso contrario, se debe presentar un mensaje de error al usuario.

3. **Configuración de la extensión:** El sistema recibe la lista de parametros requeridos por la extensión (*[(nombre, valor)]*) y se asignan los valores en la extensión. Finalmente, se devuelve un mensaje de exito indicando que vac-o esta listo para comenzar la ejecución.

3.1.2. Configuración de vac-o

El objetivo de esta función es solicitar a la extensión cargada en memoria, la información necesaria para comenzar la ejecución del motor combinatorio.

1. **Secuencia inicial:** El sistema solicita a la extensión cargada en memoria, la secuencia de ARN que se desea optimizar. La extensión debe devolver la secuencia en formato FASTA y el sistema la almacena en memoria.

2. **Regiones combinatorias:** El sistema solicita a la extensión cargada en memoria, las regiones combinatorias de interes para la optimización. La extensión debe devolver las regiones (*[(inicio, fin, tipo, evaluador)]*) y el sistema inicializa cada region en memoria. Para cada region, el parametro *tipo* debe ser uno de los siguientes:

- Estructura secundaria (SS): Las posibles mutaciones de la region deben conservar la estructura secundaria.
- Codigo genetico (GC): Las posibles mutaciones de la region deben resultar silentes en términos de expresión aminocídica.
- Personalizada (CU): La extensión debe proporcionar la implementación de la region combinatoria.

El parametro *evaluador* debe ser una función $f : Secuencia \rightarrow (0, 1)$ que sera utilizada por el sistema para determinar la "bondad" de las diferentes mutaciones de cada region.

3. **Estrategia de busqueda:** El sistema solicita a la extensión un *umbral* de "bondad" y debe implementar diferentes estrategias de optimización, asegurando que se cumpla lo siguiente:

   Sea n el numero de regiones combinatorias, s_i la secuencia seleccionada de la region i y $f_i : Secuencia \rightarrow (0, 1)$ la función de evaluación de la region i, con $i = 1, ..., n$.

   $$\prod_{i=1}^{n} f_i(s_i) \geq umbral$$

4. **Librerias externas:** El sistema solicita a la extensión, la libreria con la que se deben generar las mutaciones de cada region combinatoria.

3.1.3. Generación de secuencias

El objetivo de esta función es la construcción de nuevas secuencias genómicas que cumplan con las restricciones impuestas.

1. **Generar mutaciones de region combinatoria:** Para cada region combinatoria, el sistema debe ser capaz de generar todas las posibles mutaciones que cumplan con las restricciones impuestas por el tipo de region y considerando las posibles intersecciones con las demas regiones.

2. **Generar secuencias:** A partir de las mutaciones de cada region combinatoria, se deben construir nuevas secuencia genómicas, reemplazando cada region por su mutación en la secuencia inicial y asegurar que la secuencia resultante consereve la estructura secundaria de la secuencia inicial. El criterio de terminación en la generación de secuencias debe ser provisto por la extensión.

3. **Validación de secuencias:** Para cada secuencia generada por el sistema, se deben aplicar pruebas de "calidad". El sistema solicita a la extensión el tipo de prueba, que sera uno de los siguientes:

   - Mutaciones sistemáticas.
   - Mutaciones al azar.

   y el tipo de criterio, que sera uno de los siguientes:

   - Comparación estructural por regiones.
   - Conteo y comparación de nucleótidos apareados/desapareados.

Luego, la validación consiste en aplicar el criterio de calidad a las N mutaciones obtenidas. Las secuencias que no pasen las pruebas de calidad seran descartadas.

3.1.4. Evaluación de secuencias

El objetivo de esta función es la evaluación y posterior construcción de un *ranking* de secuencias genómicas.

1. **Asignar puntaje a una secuencia:** El sistema solicita a la extensión cargada en memoria, un puntaje para una secuencia dada. La extensión debe evaluar la secuencia recibida comparandola con la secuencia inicial y devolver un puntaje.
2. **Construir *ranking* de secuencias:** El sistema debe construir un *ranking* de secuencias basandose en los puntajes asignados por la extensión. Este listado de secuencias es además, la salida final de la ejecución.

3.2. Restricciones de Rendimiento

No se registran requerimientos.

3.3. Base de Datos

No se registran requerimientos.

3.4. Restricciones de Diseño

3.4.1. Cumplimiento de Estándares

El producto debe cumplir con los siguientes principios y patrones de diseño de la programación orientada a objetos. Los primeros 5, son también conocidos por el acrónimo "**SOLID**".

- **S**ingle responsibility principle (SRP)
- **O**pen/closed principle (OCP)
- **L**iskov substitution principle (LSP)
- **I**nterface segregation principle (ISP)
- **D**ependency inversion principle (DIP)
- Law of Demeter (LoD)

Además el producto debe cumplir con el estandar ANSI C++ y el "coding style" provisto por FuDePAN.

3.5. Atributos del Software

Se requiere que el código del producto tenga un 80 % de covertura con pruebas automatizadas.

Apendices

A. Definiciones, Acrónimos y Abreviaturas

- **vac-o**: Combinatory Vaccine Optimizer.
- **FaMAF**: Facultad de Matemática, Astronomía y Física.
- **UNC**: Universidad Nacional de Córdoba.
- **FuDePAN**: Fundación para el Desarrollo de la Programación en Ácidos Nucleicos.
- **API**: Application Programming Interface.
- **GPL**: General Public License.
- **IEEE**: Institute of Electrical and Electronics Engineers
- **SOLID**: acrónimo nemotécnico introducido por Robert C. Martin en la década de 2000, que representa cinco patrones básicos de programación y diseño orientado a objetos.
- **LoD**: Law of Demeter, principio de diseño orientado a objetos para lograr "bajo acoplamiento".

B. Referencias

- C++: Lenguaje de programación.
  `http://www.cplusplus.com`
- Python: Lenguaje de programación interpretado.
  `http://www.python.org`
- BioPP: Librería C++ para biología molecular
  `http://code.google.com/p/biopp`
- Boost.Python: Librería C++ para la interacción con Python.
  `http://www.boost.org/doc/libs/1_42_0/libs/python/doc/index.html`
- QT: Librería C++ para el desarrollo de interfaces gráficas.
  `http://qt.nokia.com/products`
- GPL: General Public License.
  `http://www.gnu.org/licenses/gpl.html`
- IEEE STD 830-1998: Guía para la especifiación de requerimientos.
  `http://standards.ieee.org/reading/ieee/std_public/description/se/830-1998_desc.html`
- SOLID: "Design Principles and Design Patterns", Robert C. Martin.
  `http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf`
- FASTA: Formato basado en texto, utilizado para representar secuencias genómicas.
  `http://es.wikipedia.org/wiki/Formato_FASTA`