FuL
Especificación de Requerimientos de Software

Alejandro Kondrasky

10 de octubre de 2011

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 FaMAF - UNC denominada ***Procesador Lógico para diseño de experimentos de Virología***.

Los requerimientos son provistos por integrantes de FuDePAN en su carácter de autores intelectuales de la solución a implementar y colaboradores de dicha tesis.

1.2. Convenciones del Documento

Las palabras clave *DEBE*, *NO DEBE*, REQUERIDO, DEBERÁ, NO DEBERÁ, DEBERÍA, NO DEBERÍA, RECOMENDADO, PUEDE Y OPCIONAL en este documento son interpretadas como esta descripto en el documento *RFC 2119*.

1.3. Audiencia Esperada

A continuación se enumeran las personas involucradas en el desarrollo de la tesis, los cuales representan la principal audiencia de este documento :

- Dr. Carlos Areces: Director de tesis, FaMAF
- Daniel Gutson: Colaborador de tesis, FuDePAN
- Alejandro Kondrasky: Tesista, FaMAF

1.4. Alcance del Producto

El producto especificado en este documento se denomina *FuDePAN Logic Processor (FuL)* y su principal objetivo es, dada una *Base de Conocimiento (KB)* del área de virología, organizar, analizar, chequear incongruencias en ella y poder utilizarla para asistir en la planificación de experimentos . A la vez que introducimos nuevos conocimientos a la KB, ésta deberá pasar por los procesos previamente nombrados.

El producto final debe proveer al usuario la capacidad de agregar extensiones capaces de interpretar el conocimiento en la KB de maneras apropiadas y de introducir nuevo conocimiento mediante un lenguaje formal definido para ello.

La principal responsabilidad de FuL es procesar el conocimiento presente en la KB, utilizando las extensiones para interpretarlo, para así obtener una planificación del experimento solicitado. Las extensiones son utilizadas para interpretar ciertos tipos de conocimiento, procesarlo utilizando sus KB internas y devolver nuevo conocimiento a FuL.

En su versión inicial, FuL incluye un *planner* y una extensión de *Lógicas Descriptivas (DL)*.

1.5. Estructura del Documento

La estructura de este documento sigue las recomendaciones de *Guía para la especificación de requerimientos de la IEEE (IEEE Std 830-1998)*. Contiene las siguientes secciones :

- *Sección 2:* Provee una descripción general de los aspectos generales del producto, como la perspectiva de este, características principales.
- *Sección 3:* Describe las interfases del producto, tanto las del usuario como interfases de software.
- *Sección 4:* Organiza y describe los requerimientos funcionales del producto.
- *Sección 5:* Organiza y describe los requerimientos no funcionales.

2. Descripción General

2.1. Perspectiva del Producto

Este producto trata de proveer a la comunidad científica una herramienta para la organización, interpretación, verificación y exploración del conocimiento en el área de virología, para así poder encontrar incongruencias y conclusiones derivadas automáticamente de dicho conocimiento.

Otra de sus funciones principales es la asistencia informática para planificar experimentos equivalentes a preguntas del tipo $A \Longrightarrow B$.

2.2. Características del Producto

Las secciones marcadas con * son opcionales o serán incluidas en trabajos futuros.

2.2.1. Introducir nuevo conocimiento a la KB

Permite, mediante el lenguaje formal otorgado por FuL[1], introducir conocimiento a la KB el cual sera chequeado para encontrar contradicciones con esta, para luego ser aceptado o mostrar las encontradas.

2.2.2. Procesamiento de la KB

Provee la capacidad de procesar la información presente en la KB para así obtener nuevas conclusiones del conocimiento disponible en ella .

2.2.3. Asistir en la planificación de experimentos

Provee la capacidad de responder preguntas objetivo y, en caso de no ser capaz de responder dicha pregunta, devolver preguntas relevantes para la que fue introducida. Se dice que son relevantes ya que deben ser contestadas para poder llegar a una respuesta para la pregunta objetivo. Dichas preguntas objetivo serán metas lógicas del tipo $A \Longrightarrow B$.

Estas preguntas devueltas son útiles para el usuario en el desarrollo de sus experimentos, ya que sirven para la confección de arboles de hipótesis y planeamiento de escenarios.

2.2.4. API y SDK para Extensiones

Provee una API y un SDK para la creación de extensiones, los cuales serán utilizados para interpretar el conocimiento que le es otorgado a dicha base de datos.

El API define la forma en la que se intercambiara el conocimiento entre la extensión y FuL. El SDK otorga las librerías necesarias tanto para la construcción de la extensión en si, como también la definición de su lenguaje formal de representación de conocimiento correspondiente.

[1] Dicho lenguaje será traducible a lenguajes de representación de datos.

2.2.5. Optimización de la KB*

Tendrá la capacidad de optimizar la KB. Esta proveerá la eliminación de redundancias y devolverá una versión minimál de la KB original, de la cual se puede deducir todo el conocimiento presente en la original.

2.2.6. KB inicial con conocimientos de virología*

Proveer una KB inicial, la cual contenga conocimientos básicos de virología tales como lo que se encuentran el libros de estudio de nivel básico a intermedio. Restricciones al *scope* de dicha KB están *por ser determinadas (TBD)*[2].

[2]Se propuso restringirlo a los arenavirus y los retrovirus. Se decidirá acorde con las necesidades de FuDePAN y los tiempos disponibles.

3. Interfaces

3.1. Interfaces de Sistema

Será capaz de correr al menos en sistemas GNU/Linux, así que por este hecho sólo se utilizaran librerías compatibles con sistemas GNU/Linux.

3.2. Interfaces de Usuario

La interfase consiste en una *CLI (Command Line Interface)* y un lenguaje formal, descriptos a continuación.

3.2.1. CLI

Este sera tratable como un comando de terminal, al cual se le podrán pasar al menos parámetros para realizar las siguientes operaciones[3]:

1. *Introducir un archivo con conocimiento en la KB (V):* Parsea y chequea consistencia de dicho conocimiento para luego chequear contradicciones de este con la KB. En caso de éxito se agregara a la KB, caso contrario sera rechazado en su totalidad.

2. *Procesar el conocimiento en la KB (V):* Será la opción por defecto en la cual solo procesará el conocimiento existente en la KB para obtener nuevas conclusiones a los posibles experimentos.

3. *Realizar un consulta de una meta lógica de tipo* $A \Longrightarrow B$ *(V):* Chequea la valides de dicha consulta. En caso de ser valida FuL responderá con un valor booleano o con una lista de metas lógicas de tipo $A \Longrightarrow B$ para los cuales no tiene respuesta. En caso de ser invalida mostrara el mensaje de error correspondiente.

4. *Agregar una nueva extensión a FuL (V):* El usuario introducirá el nombre de archivo de la extensión que desea agregar. En caso de que dicho archivo no sea valido o surja algún error sera informado del mismo.

5. *Configurar parámetros relacionados con FuL, su KB y extensiones :* TBD[4]

3.2.2. Lenguaje Formal

Es el lenguaje a través del cual el usuario deberá utilizar para introducir conocimiento y realizar consultas a FuL. También es con el cual se presentaran los resultados de las operaciones solicitadas. Dicho lenguaje sera capaz de representar conocimiento relacionado con el área de Virología[5] , el cual tendrá la propiedad de ser almacenable en una KB y ser interpretado por una extensión.

[3] Las operaciones marcadas con *(V)* tendrán la opción ser puestas en modo verborrágico, lo que hará que muestren por pantalla mayor información relacionada con su procesamiento.

[4] Dependerá de factores relacionados con el diseño e implementación de la KB, extensiones y otros aspectos relacionados con FuL tales su instalación,etc. Deberá ser resuelto en las etapas anteriormente mencionadas.

[5] Pero no exclusivamente.

3.3. Interfaces de Hardware

No hay requerimientos especificados.

3.4. Interfaces de Software

El producto tendrá los siguientes módulos :

1. *fulmop :* Este es producto principal, el cual se encarga de articular los otros componentes, tales como la KB, fulplanner y las extensiones para realizar las operaciones pedidas por el usuario.
2. *fulplanner :* Es el encargado de organizar el proceso de razonamiento.
3. *fulapi :* Es la API definida para interactuar con las extensiones creadas por terceros.
4. *fuldl :* Extensión que permite interpretar conocimiento entendible por lenguajes DL. Actúa de la misma manera que cualquier otra extensión, recibiendo conocimiento a ser interpretado y devolviendo nuevo conocimiento respecto de este.

3.5. Interfaces de Comunicación

No hay requerimientos especificados.

4. Requerimientos Funcionales

Nomenclatura

- ERROR_EXIT : La CLI deberá informar al usuario de los errores ocurridos y se finalizara dicha operación.

4.1. REQ Introducir Conocimiento en la KB

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario introducir conocimiento en la KB.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Apertura :* La CLI deberá proveer una opción para introducirle como parámetro el nombre del archivo a ser abierto. En caso de no poder ser abierto, ERROR_EXIT.
2. *Lectura, validación sintáctica y gramatical :* Se leerá el contenido del archivo. En caso de que su contenido no corresponda con el lenguaje formal especificado por FuL, ERROR_EXIT.
3. *Validación de la semántica interna del archivo :* Se deberá validar que cada una de las afirmaciones en el archivo no se contradicen entre si. En caso de encontrarse alguna contradicción, ERROR_EXIT.
4. *Validación de la semántica del archivo respecto de la KB de FuL :* Se deberá validar las afirmaciones en el archivo con las existentes en la KB de FuL. En caso de encontrarse contradicciones entre dichos conjuntos, ERROR_EXIT.
5. *Registrado de las afirmaciones :* Se deberá registrar las afirmaciones presentes en el archivo en la KB. Finalizado el registrado se deberá mostrar al usuario, mediante la CLI, un aviso del registrado exitoso de las afirmaciones contenidas en el archivo.

4.2. REQ Procesar el Conocimiento en la KB

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario procesar el conocimiento existente en la KB de FuL para obtener nuevo conocimiento, el cual sera agregado a esta.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Obtención del conjunto de inicio :* fulmop deberá obtener el conjunto de estados iniciales, utilizando el conocimiento existente en la KB y las extensiones.
2. *Creación de las lineas de pensamiento :* Utilizando dicho conjunto, fulplanner deberá crear las lineas de pensamiento correspondientes a cada uno.

3. *Expansión de las lineas de pensamiento :* fulmop, utilizando su KB y las extensiones, deberá expandir los estados de cada una de las lineas de pensamiento creadas por fulplanner. Se deberá eliminar todo ciclo presente en las lineas de pensamiento. Cada ciclo deberá eliminarse evitando eliminar otras lineas de pensamiento presentes en este que no sean cíclicas. Las lineas de pensamiento deberán ser extendidas hasta el punto de que no sea posible extenderlas mas.
4. *Registrado del nuevo conocimiento en la KB :* fulmop deberá registrar en la KB el nuevo conocimiento derivable de las lineas de pensamiento obtenidas.
5. *Mostrar el nuevo conocimiento obtenido :* El nuevo conocimiento registrado en la KB deberá ser devuelto al usuario mediante la CLI, ya sea devolviendo las lineas de pensamiento obtenidas y/o una explicación del significado de las mismas.

4.3. REQ Consultar Metas Lógicas de Tipo $A \Longrightarrow B$

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario consultar las conclusiones a las que FuL puede llegar para una meta lógica de tipo $A \Longrightarrow B$.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Ingreso del Experimento :* La CLI deberá proveer una opción para introducir la meta lógica de tipo $A \Longrightarrow B$ a ser consultado.
2. *Lectura, validación sintáctica y gramatical :* Una vez introducida la meta lógica se validara que su sintaxis corresponda con la del lenguaje formal definido en FuL. En caso contrario, ERROR_EXIT.
3. *Validación de definiciones :* Si la meta lógica esta bien definida respecto del lenguaje formal entonces validaremos las definiciones A y B, utilizando la KB y las extensiones. En caso de no sean validas, ERROR_EXIT.
4. *Iniciado de fulplanner :* Si A y B son definiciones validas, entonces se procederá a iniciar fulplanner con la definición A como su estado inicial.
5. *Expansión a nuevos estados :* Luego fulmop deberá tomar los estados que no han sido expandidos y, mediante KB y las extensiones, obtener los nuevos estados a los que estos pueden ser expandidos.
6. *Agregación de los nuevos estados :* fulplanner conectara a los estados a expandir los nuevos estados obtenidos para estos. Si se obtuvieron nuevos estados para alguno de los estados a expandir entonces se procederá al paso siguiente. En caso contrario se procederá al paso final.
7. *Eliminación de ciclos :* fulplanner deberá eliminar todo ciclo[6] presente en la planificación. Cada ciclo deberá eliminarse evitando eliminar otras lineas de pensamiento presentes en este que no sean cíclicas.

[6] Léase como la presencia de $A \rightarrow A$ en alguna parte de la planificación.

8. *Validación de la presencia de B en la planificación :* Si existe en la planificación el estado B tal que es alcanzado desde el estado raíz A por una linea de pensamiento afirmativa[7] entonces se procederá con el paso final. En caso contrario se procederá al siguiente.
9. *Verificación de presencia de estados a expandir :* Si existen en la planificación estados a expandir entonces se deberá realizar el paso 5.
10. *Retorno de la planificación obtenida :* Se retorna mediante la CLI la planificación obtenida, una explicación del significado de la misma y/o el resultado de la operación realizada.

4.4. REQ Prueba: Experimento Junin

- ***Objetivo :*** El objetivo de este requerimiento es demostrar que FuL es capaz de validar las conclusiones obtenidas en el experimento Junin realizado por la fundación FuDePAN.
- ***Prioridad :*** Alta
- ***Descripción :*** FuL deberá ser capaz de validar las conclusiones obtenidas en el experimento. Para esto se deberá proveer a FuL de una KB que represente el conocimiento utilizado para llegar a dichas conclusiones. Deberá llevar acabo esta prueba mediante una consulta de meta lógica.

4.5. REQ Agregar una Nueva Extensión

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario agregar una nueva extensión a FuL, la cual podrá ser usada por fulmop en las otras operaciones representadas en los requerimientos previos.
- ***Prioridad :*** Baja
- ***Descripción :***

  1. Ingreso : La CLI deberá proveer una opción para introducir el nombre del archivo que contiene la extensión a ser agregada, formato del archivo TBD[8].
  2. Validación : FuL deberá validar que la extensión introducida cumple con las interfases definidas en su API y que sus dependencias han sido satisfechas.
  3. Instalación : Una vez validada, FuL instalara dicha extensión en el entorno de trabajo de FuL. Por ultimo avisara al usuario, mediante la CLI, que la instalación se a realizado exitosamente.

[7] No contiene falsedades ni hipótesis en ella.
[8] Esto deberá ser definido en el paso de diseño.

4.6. REQ Configurar FuL

- ***Objetivo :*** El objetivo de este requerimiento es proveer al usuario operaciones para configurar los distintos parámetros relacionados con el funcionamiento de FuL.
- ***Prioridad :*** Baja
- ***Descripción :*** Deberá ser capaz de proveer operaciones para configurar los distintos aspectos de FuL, tales como parámetros relacionados con la KB, extensiones, fulplanner y fulmop. Estos aspectos son y el grado de configurabilidad de cada uno sera TBD[9].

[9]Ya que dependerá del diseño e implementación de los elementos anteriormente mencionados.

5. Requerimientos No Funcionales

5.1. Requerimientos de Diseño

Deben cumplirse los siguientes requerimientos de diseño :

- En caso de utilizar C++ en su implementación, deberá cumplir los estándares ANSI C++, conforme con el coding-guideline definido por FuDePAN.
- Deberá ser capaz de construirse utilizando herramientas GNU.
- El código deberá ser compilado sin advertencias, o las que sean permitidas deberán ser documentadas.

5.2. Atributos del Software

El producto debera tener los siguientes atributos :

- Deberá ser testeado el 85 % del código fuente presente en este producto.
- Deberá definirse un lenguaje formal para la representación del conocimiento del área de Virología, el cual deberá ser de fácil interpretación y utilización tanto por personas calificadas de dicha área como por FuL. Su semántica deberá ser capaz de representar el conocimiento relacionado con el área de Virología.

5.3. Restricciones de Memoria

Deben cumplirse las siguientes restricciones de memoria :

- No deberá haber problemas de memorial o leaks, que estén directamente relacionados con el proyecto.
- En caso de que exista código C/C++ en el producto, se deberá utilizar el comando valgrind −−leak−check=full para corroborar la existencia de leaks.
- Las dependencias externas que puedan llegar a ser usadas en producto no deberán ser tenidas en cuenta a la hora de chequear problemas de memoria o leaks.

Es importante hacer notar que se requerirá capacidad de almacenamiento[10] para el registro del conocimiento y la manipulación del mismo. Por lo tanto el desempeño de FuL estará directamente relacionado con la maquina en la que se corra.

5.4. Entorno de Funcionamiento

FuL esta pensado para ser utilizado como una herramienta de consulta de conocimiento del área de Virología y para la planificación de experimentos relacionados con esta.

Las plataformas en las que deberá correr sera en las basadas en GNU/Linux.

[10] Tanto solida como volátil.

5.5. Licencias

El código fuente de FuL estará licenciado como GPLv3. Toda la documentación generada para el desarrollo de FuL sera licenciada bajo Creative Commons para proteger tanto el texto como las ideas presentes en este documento[11].

[11] Este ultimo punto podría modificarse a futuro para utilizar licencias mas seguras para la protección de ideas.

Nomenclatura

CLI	Command Line Interface
DL	Description Logics (Lógicas Descriptivas)
FuL	FuDePAN Logic Processor
KB	Knowledge Base (Base de Conocimiento)
planner	COMPLETAR
scope	Alcance de cierto objeto
TBD	To Be Determinated (Por ser Determinado)