FuL
Especificación de Requerimientos de Software

Alejandro Kondrasky

1 de febrero de 2012

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 descrito 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 interfaces del producto, tanto las del usuario como interfaces 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

Proveer 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

Proveer 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.

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

Proveer un 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 intercambiará el conocimiento entre la extensión y FuL. El SDK otorga las librerías y las herramientas necesarias tanto para la construcción de la extensión en sí.

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

2.2.5. Optimización de la KB *

Proveer la capacidad de optimizar la KB. Esta característica permite la detección de redundancias y obtener una versión minimal de la KB original en caso de no encontrarse redundancias, 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[2]. Restricciones al *scope* de dicha KB están *por ser determinadas (TBD)*[3].

[2]Por ejemplo:

- Virus life in diagrams - Hans-Wolfgang Ackermann, Laurent Berthiaume, Michel Tremblay.
- Fields Virology - Bernard N. Fields, David Mahan Knipe.

[3]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 interfaz consiste en un command line, una *CLI (Command Line Interface)* y un lenguaje formal, descriptos a continuación.

3.2.1. Command Line

Provee un comando del linea a través del cual configurar y acceder a la CLI de FuL, al cual el usuario le otorgara los siguientes parámetros :

- Un archivo de configuración XML donde se especifique el conjunto de extensiones que FuL utilizara en esta ejecución. Además este archivo puede contener configuración para las variables de FuL y las extensiones a registrar.
  Este parámetro es obligatorio, pero se considera valido un XML que no presente configuración ni registrado de extensiones alguna.
- La KB inicial la cual FuL utilizara durante toda la sesión. Este parámetro es opcional, en caso de no otorgarse FuL iniciara con una KB vacía.

A medida que se cargan dichos elementos se mostraran por consola los mensajes pertinentes al usuario, en caso de error se mostrara el mensaje correspondiente y se anulara o continuara con su ejecución dependiendo de la gravedad del error[4]. Si no hay ningún error se pondrá en funcionamiento la CLI.

3.2.2. CLI

Es una interfaz CLI, similar a un prompt, la cual permite ejecutar los siguientes comandos[5]:

1. ***import*** *<delta.kb> :* Permite introducir el conocimiento presente en *delta.kb* a la KB actualmente cargada en FuL. En caso de que delta.kb contenga conocimiento invalido[6], FuL abortara el proceso de importación e informara al usuario mediante la CLI del error con un nivel de detalle adecuado. Si la ejecución del comando a sido exitosa se mostrara en la CLI el mensaje *IMPORT OK*.

2. **save** *<file.kb> :* Permite guardar la KB que FuL tiene en memoria dentro de *file.kb*. Si la ejecución del comando a sido exitosa se mostrara en la CLI

[4] La clasificación de nivel de gravedad de los errores esta TBD, la cual sera definida una vez de que estos sean clasificados y estudiados en la fase de análisis y/o diseño.

[5] Los elementos presentes en un comando englobados con <> son parámetros que el argumento requiere para ser ejecutado. Los nombres de los parámetros son representativos y pueden diferir de los que tenga el producto final.

[6] Tanto por ser sintácticamente incorrecto respecto del lenguaje formal o por contradecir el ya presente en la memoria de FuL.

el mensaje *SAVED*. En caso de error informara al usuario mediante la CLI de dicho error.

3. **query** *<query-string> :* Permite realizar la consulta *query-string* a FuL, el cual devolverá una respuesta acorde con la KB y extensiones cargadas previamente[7]. En caso de que la consulta sea invalida respecto del lenguaje formal entonces devolverá el mensaje de error correspondiente. Si no se puede llegar a una respuesta definida entonces FuL devolverá la lista de elementos requeridos para que FuL de una respuesta definida.

4. **execute** *<script.ful> :* Permite ejecutar *script.ful* el cual contiene una secuencia de comandos validos. Cada linea se interpreta como un comando y en caso de error en alguno de los comandos se detendrá la ejecución del script.

5. **show** *<variable> :* Permite ver el valor de *variable*, la cual debe pertenecer a las variables registradas por FuL o las extensiones. En caso de que *variable* no pertenezca se devolverá el mensaje *ID-10-T error: unrecognized variable*.

6. **set** *<variable> <value> :* Permite asignar *value* a *variable* , siendo *variable* una variable registrada por FuL o las extensiones y *value* un valor valido para esta. En caso de que *variable* no pertenezca al conjunto de variables registradas se devolverá el mensaje *ID-10-T error: unrecognized variable*. En caso de que *value* sea un valor invalido para *variable* se devolverá el mensaje *ID-10-T error: invalid value for <variable>*.

7. **quit** *:* Permite salir de la CLI y cerrar FuL. En caso de que no se haya guardado la KB que FuL tiene en memoria, se mostrara el mensaje *WARNING: memory state not saved. Sure to quit? (y/n)*. Si el usuario lo confirma entonces se saldrá de la CLI y cerrara FuL , sino abortara el comando.

Adicionalmente se podrán ejecutar los comandos registrados por las extensiones, las cuales definirán el comportamiento de los mismos. En caso de que el comando introducido no este registrado por las extensiones cargadas o no sea parte de los anteriormente mencionados entonces se devolverá el siguiente mensaje *ID-10-T error: unrecognized command*.

3.2.3. 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[8] , el cual tendrá la propiedad de ser almacenable en una KB y ser interpretado por una extensión.

3.3. Interfaces de Hardware

No hay requerimientos especificados.

[7] Esta respuesta estará compuesta por una explicación del resultado obtenido y las lineas de pensamiento que se utilizaron para arribar a el.

[8] Pero no exclusivamente.

3.4. Interfaces de Software

Se deberán utilizar las siguientes librerías :

- fudepan-build : Build system para proyectos de FuDePAN.
- FXP : Parser XML para C++, requiere eXpat.
- getoptpp : Versión C++ de la función getopt que permite parsear opciones command line y manejar variables de entorno. Fácil de usar y altamente compatible con STL.
- MiLi : Colección de librerías útiles de C++.
- unplugged: Librería que provee las herramientas genéricas necesarias para el soporte de extensiones en Linux y Windows.

3.5. Interfaces de Comunicación

No hay requerimientos especificados.

4. Requerimientos Funcionales

Nomenclatura para los requerimientos

- ERROR_EXIT *: **La CLI deberá informar al usuario de los errores ocurridos y se finalizara dicha operación.***
- *<parámetro> :* Es un parámetro ingresado al comando de dicha linea, siendo *parámetro* el valor del mismo.

4.1. REQ_01 Ejecutar FuL como command-line

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario ejecutar FuL como si fuera un comando de linea. Este tiene como parámetros una KB inicial y un archivo de configuración de extensiones contenidos en archivos de formato kb y xml respectivamente.
- ***Prioridad :*** Alta
- ***Descripción :***

  1. *Ejecución del comando en la terminal :* El usuario ejecuta en la terminal el comando ful <config.xml> <database.kb> , siendo *:*
     - *config.xml :* El archivo XML que contiene la configuración de las extensiones. Esta podrá tener una lista de las extensiones a agregar, las cuales podrán estar acompañadas por una asignación de valores a las variables que ellas otorgan al usuario para que este manipule. Este parámetro es obligatorio.
     - *database.kb :* La KB inicial con la cual FuL procesara los comandos posteriores provenientes de la CLI. El conocimiento deberá estar expresado en el lenguaje formal de FuL. Este parámetro es opcional.

     En caso de que no se otorguen los parámetros requeridos se devolverá el mensaje *ID-10-T error: invalid parameters* y se finalizara la ejecución de FuL.
  2. *Cargar configuración de FuL:* Se deberá procesar la configuración de las variables de FuL presente en *config.xml*. Las variables de FuL no presentes en dicho archivo tomara el valor default definido por FuL. Para esto deberá utilizarse FXP[9]. En caso de que error ERROR_EXIT.
  3. *Cargar las extensiones :* Se deberá cargar las extensiones presentes en *config.xml* y se conectan con FuL, aplicando para cada una de las extensiones la configuración de estas presente en el mismo documento. También se registraran los comandos que cada extensión otorgue. Para esto se deberá utilizar unplugged[10]. En caso de no encontrarse la configuración de dicha extensión, no encontrarse la extensión propiamente dicha u algún otro tipo de error ERROR_EXIT.

[9] Leer 3.4 en la página anterior.
[10] Leer 3.4 en la página anterior.

4. *Cargar la KB :* Se deberá procesar el conocimiento existente en la KB por parte de FuL, teniendo en cuenta su configuración y las de las extensiones que fueron cargadas.
   En caso de encontrarse conocimiento dirigido a una extensión determinada, se deberá procesar dicho conocimiento de la manera que dicha extensión lo indique. En caso de no encontraste dicha extensión ERROR_EXIT.
   En caso de que no se haya otorgado una KB, se pasara al paso siguiente.
5. *Puesta en funcionamiento de la CLI :* Se deberá poner en funcionamiento la CLI de FuL con la configuración, extensiones y KB obtenidas en los paso previos. Para los pasos posteriores leer 4.2.

4.2. REQ_02 Ejecución de comandos en la CLI

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario interactuar con la CLI mediante la introducción de comandos[11] y la devolución de respuestas.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Introducir un comando :* El usuario podrá introducir un comando con el formato <comm> <p_1> ... <p_n> siendo *comm* el comando a ejecutar y *<p_1> ... <p_n>* un conjunto de n parámetros que *comm* permite y/o requiere[12]. Si así sucede se proceda a los siguientes pasos:
   a) *Validación del comando :* Se deberá chequear que *comm* sea un comando registrado en FuL. En caso de no serlo ERROR_EXIT.
   b) *Validación de los parámetros :*
      1) Si $comm =$ "*quit*" entonces, si $n = 0$, se deberá realizar el paso 3. Caso contrario se deberá devolver el mensaje *ID-10-T error: invalid parameters.*
      2) Si $comm \epsilon$ {"*import*","*save*","*query*","*execute*","*show*","*set*"}, leer el requerimiento correspondiente.
      3) Sino se cumple ninguno de los anteriores se procederá deslindar la ejecución de dicho comando a la extensión bajo la cual fue registrado[13], con los parámetros *<p_1> ... <p_n>*.
2. *Esperar :* Se esperara hasta que el usuario introduzca un nuevo comando, en cuyo caso se volverá al paso 1.
3. *Finalizar :* Si la KB que FuL tiene en memoria no fue guardada en un archivo se le deberá mostrar al usuario el mensaje *WARNING: memory state not saved. Sure to quit? (y/n).* Si el usuario elige *y* entonces se finaliza la ejecución de FuL, devolviendo al usuario a la terminal desde donde se ejecuto el command-line de FuL. Si elije *n*

[11] Los cuales son especificados en este requerimiento o pertenecen al conjunto de comandos registrados por extensiones. Leer 4.1 en la página anterior.
[12] Puede ser $n = 0$, en cuyo caso no habrá parámetro alguno.
[13] Leer 4.1 en la página anterior.

entonces se volverá al paso 2.
Si la KB que FuL tiene en memoria fue guardada en un archivo entonces se finaliza la ejecución de CLI y de FuL, devolviendo al usuario a la terminal desde donde se ejecuto el command-line de FuL.

4.3. REQ_03 Ejecución del comando IMPORT

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario importar el conocimiento presente en un archivo <*delta.kb*> y agregarlo a la KB de FuL.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 1$ y $p\_1 = delta.kb$ deberá ser un archivo que contendrá conocimiento, sintáctica y semánticamente valido, interpretable por FuL o por las extensiones registradas en el. También se deberá chequear que dicho conocimiento no es contradictorio con el presente en la KB de FuL. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se finaliza la ejecución del comando.
2. *Registrado del conocimiento :* Se deberá registrar el conocimiento presente en *delta.kb* en la KB de FuL, junto con todo el conocimiento que se pueda derivar al agregar dicho archivo. Finalizado el registrado se deberá mostrar al usuario, mediante la CLI, el mensaje *IMPORT OK* y se finaliza la ejecución del comando

4.4. REQ_04 Ejecución del comando SAVE

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario guardar el conocimiento presente en en la KB en un archivo <*database.kb*>.
- ***Prioridad :*** Alta
- ***Descripción :***

1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 1$ y $p\_1 = database.kb$ deberá ser un nombre valido para de un archivo[14]. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se pasara al paso final.
2. *Guardado :* Se guardara en el archivo la KB presente en la memoria de FuL. Una vez guardada se mostrara el mensaje *SAVED*.
3. *Finalizada la ejecución del comando :* Se finaliza la ejecución del comando.

[14] Ya sea que este exista o no en el sistema de archivos de la sistema donde se esta ejecutando FuL.

4.5. REQ_05 Ejecución del comando QUERY

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario realizar una consulta a FuL a través de la CLI, la cual este dará una respuesta respecto de su KB y las extensiones registradas en dicha sesión.
- ***Prioridad :*** Alta
- ***Descripción :***

  1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 1$ y $p\_1 = consulta$ deberá ser una consulta interpretable por FuL, la cual respete las restricciones de su lenguaje formal. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se pasara al paso final.
  2. *Procesado de la Consulta :* Se deberá procesar la consulta con respecto a la conocimiento presente en la KB de FuL y las extensiones registradas.
  3. *Retorno de las conclusiones obtenidas :* Se retorna mediante la CLI las conclusiones obtenidas y una explicación del significado de las mismas.
  4. *Finalizada la ejecución del comando :* Se finaliza la ejecución del comando.

4.6. REQ_06 Ejecución del comando EXECUTE

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario ejecutar un script en FuL a través de la CLI, la cual este dará una respuesta respecto de su KB y las extensiones registradas en dicha sesión.
- ***Prioridad :*** Baja
- ***Descripción :***

  1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 1$ y $p\_1 = script.ful$ deberá ser un archivo el cual contenga comandos validos para FuL y las extensiones registradas. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se pasara al paso final.
  2. *Procesado de ejecución del script :* Se deberá ejecutar cada uno de los comandos presentes en script.ful[15].
  3. *Finalizada la ejecución del comando :* Se finaliza la ejecución del comando.

[15] Leer 4.2 en la página 11.

4.7. REQ_07 Ejecución del comando SHOW

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario ver el valor de una variable registrada en FuL.
- ***Prioridad :*** Baja
- ***Descripción :***

  1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 1$ y $p\_1 = variable$ deberá ser una variable registrada en FuL. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se pasara al paso final.
  2. *Obtención del valor de la variable:* Se deberá mostrar a través de la CLI el valor de la variable solicitada.
  3. *Finalizada la ejecución del comando :* Se finaliza la ejecución del comando.

4.8. REQ_08 Ejecución del comando SET

- ***Objetivo :*** El objetivo de este requerimiento es permitirle al usuario darle un valor a una variable registrada en FuL.
- ***Prioridad :*** Baja
- ***Descripción :***

  1. *Validación de parámetros :* Se deberá validar que la cantidad de parámetros $n = 2$, $p\_1 = variable$ deberá ser una variable registrada en FuL y $p\_1 = value$ deberá ser un valor valido para dicha variable. De no cumplirse alguna de estas condiciones se deberá devolver el mensaje de error correspondiente a través de la CLI y se pasara al paso final.
  2. *Registrado del valor en la variable:* Se deberá registrar *value* en *variable*.
  3. *Finalizada la ejecución del comando :* Se finaliza la ejecución del comando.

4.9. REQ_09 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 pregunta objetivo.

5. Requerimientos No Funcionales

5.1. Requerimientos de Diseño

Deben cumplirse los siguientes requerimientos de diseño :

- El producto DEBERÁ cumplir con los siguientes principios 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)
- 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 código del producto DEBERÁ:

- compilar sin advertencias, o las advertencias aceptadas DEBERÁN estar documentadas.
- 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.
- cumplir con el estándar ANSI C++ y el coding-guideline definido por FuDePAN.

El software DEBERÁ:

- tener al menos un 85 % de cobertura con pruebas automatizadas.

5.2.1. Restricciones de Memoria

Deben cumplirse las siguientes restricciones de memoria :

- No deberá haber problemas de memoria 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[16] 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.

5.2.2. 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 correrá deberán estar basadas en GNU/Linux.

5.3. Licencias

El código fuente de FuL estará licenciado como GPL v3. 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[17].

[16] Tanto solida como volátil.

[17] 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
script	Secuencia de comandos interpretables por la CLI de FuL
TBD	To Be Determinated (Por ser Determinado)