TESIS DOCTORAL

# Uso de algoritmos genéticos para la creación automática de patrones de requisitos

presentada por
Jesús Manuel Poza Carrasco

Tesis depositada en cumplimiento parcial de los requisitos para el grado de Doctor en Ciencia y Tecnología Informática

Universidad Carlos III de Madrid

Directores
Profesor Doctor D. Valentín Moreno Pelayo
Profesora Doctora Dña. Anabel Fraga Vázquez

Tutora
Profesora Doctora Dña. Anabel Fraga Vázquez

Septiembre de 2022

# AGRADECIMIENTOS

Mi primer agradecimiento es para Juan Llorens, que me animó a incorporarme a su equipo; él me proporcionó motivación, visión y todo tipo de facilidades para iniciar la investigación. Quiero agradecer muy especialmente su acogida en el grupo.

Este trabajo no habría sido posible sin el apoyo de todo el equipo del grupo Knowledge Reuse. De ellos he recibido el soporte técnico siempre que lo he necesitado. Agradezco la ayuda de Roy y Chema, disponibles en los momentos más difíciles. Roy me ha ofrecido todo el soporte técnico que he necesitado para el desarrollo de Twiga. La ayuda de Eugenio ha sido especialmente valiosa al haber puesto a mi disposición los juegos de datos para la experimentación y toda la información necesaria referida a su investigación previa. Eduardo y Borja tuvieron la paciencia de escuchar mis preparaciones y ofrecerme comentarios muy valiosos.

Pero a quien estoy más agradecido es a mis directores de tesis, Anabel y Valentín. Juntos hemos recorrido el camino. Ellos me han apoyado en todo momento, me han ayudado a encontrar soluciones, y me han dado la energía que me faltaba en los momentos de desánimo. Sin el optimismo y visión de Valentín y sin las gestiones, cariño y apoyo de Anabel, este trabajo nunca habría sido finalizado.

A todos, muchas gracias.

# CONTENIDOS PUBLICADOS

| | |
|---|---|
| Título | *Genetic Algorithms: a practical approach to generate Textual Patterns for Requirements Authoring.* |
| Autores | Jesús Poza; Valentín Moreno; Anabel Fraga; José María Álvarez-Rodríguez |
| Revista | *Applied Sciences* |
| Año | 2021 |
| Vol., *Issue* | Vol.: 11, *Issue*: 23 |
| DOI:/URL | https://doi.org/10.3390/app112311378 |
| Referencia | [1] |
| Declaración | Todo el material de esta fuente incluido en la tesis está señalado por medios tipográficos y una referencia explícita |

# RESUMEN

Un factor crítico que influye en el éxito de un proyecto es la disponibilidad un conjunto de requisitos completo y de calidad que reflejen de manera completa y consistente las necesidades a las que el proyecto debe dar respuesta.

Los patrones de texto son artefactos de conocimiento que se utilizan como plantillas para guiar a los ingenieros y analistas en la creación de requisitos y que permiten también validar su calidad; guían en la redacción de requisitos haciendo uso de lenguaje natural con ciertas restricciones sintácticas y de vocabulario que imponen los propios patrones para asegurar su calidad.

Sin embargo, la generación de patrones para un dominio concreto es una actividad que requiere mucho tiempo, es costosa y debe ser realizada por especialistas.

Esta tesis propone un método de generación automática de patrones de texto válidos para la creación de nuevos requisitos a partir de un conjunto inicial de requisitos de calidad en un dominio determinado. Se propone el uso de algoritmos genéticos y una estrategia de separación y conquista. El algoritmo genético encuentra patrones que reconocen a un número determinado de requisitos del conjunto a partir del conjunto inicial de requisitos. Aplicando la estrategia de separación y conquista, si estos patrones candidatos proporcionan un rendimiento adecuado, se incorporan al conjunto de soluciones y la búsqueda evolutiva continúa con un conjunto más pequeño de requisitos que incluye sólo los que aún no han sido conquistados. El resultado final de la doble iteración de algoritmo de conquista y algoritmo genético es un conjunto de patrones válidos, encontrados a partir del conjunto inicial de requisitos, que pueden ser usados para redactar nuevos requisitos en el mismo dominio o para validar la calidad de otros requisitos ya escritos, siempre en el mismo dominio.

Se ha definido una metodología de trabajo completa y un algoritmo basado en el uso de programación genética junto con la estrategia de separación y conquista (*Separate-and-Conquer*) para obtener conjuntos de patrones de texto capaces de reconocer conjuntos de

requisitos. A partir de la metodología se ha desarrollado una herramienta funcional completa, Twiga, con alta capacidad de parametrización para ajustar su funcionamiento a distintos contextos y objetivos de generación de conjuntos de patrones. Esta herramienta ha sido testeada y los rangos de funcionamiento de sus principales parámetros establecidos con un primer bloque de experimentos haciendo uso de requisitos artificiales, requisitos generados a partir de patrones preestablecidos y, por tanto, conocidos. Este entorno de experimentación controlado ha permitido definir los valores más adecuados de los parámetros a utilizar en casos de uso reales.

Una vez ajustada la herramienta, se han realizado una serie de experimentos utilizando un corpus de requisitos reales proporcionado por el Grupo de Trabajo de INCOSE (*International Council on Systems Engineering*) en 2016 y que ya fue utilizado en experimento anteriores por el grupo de investigación.

Los resultados muestran que este método puede generar conjuntos de patrones válidos, mejorando en un 233% los resultados experimentos previos del grupo que utilizaron otras aproximaciones, con valores de ratio de requisitos coincidentes por patrón de 2,87, frente a 1,23 obtenidos es los experimentos anteriores.

**Palabras clave:** Patrones de Texto, Algoritmos Genéticos, Ingeniería de Requisitos Reutilización del Conocimiento.

# ABSTRACT

A critical factor influencing the successful execution of any type of project is the availability of a complete set of requirements with the proper quality.

Text patterns are knowledge artifacts used as templates to guide engineers and analysts in the process of creation of requirements, enabling the use of natural language with certain syntactic and vocabulary constraints imposed by the patterns themselves to ensure their requirement quality. However, generating patterns for a particular domain is a time-consuming, costly activity that must be performed by specialists.

This thesis proposes a method for the automatic generation of text patterns from an initial subset of quality requirements. These patterns will be valid for the creation of new requirements or for validating the quality of another corpus of existing requirements of the same domain. The use of Genetic Algorithms and a Separate-and-Conquer strategy is proposed. The genetic algorithm finds, from the initial subset of requirements, patterns that recognize a given number of requirements in the subset. Applying the Separate-and-Conquer strategy, if these candidate patterns provide adequate performance, defined parametrically, they are incorporated into the final solution and the evolutionary search continues with a smaller subset of requirements that includes only those that have not yet been conquered.

A complete working methodology based on the use of Genetic Algorithms together with the Separate-and-Conquer strategy have been defined to obtain sets of valid text patterns for the writing of new requirements.

A complete functional tool, Twiga, has been developed based on this methodology, with a high capacity of parameterization to adjust its operation to different contexts and objectives. This tool has been tested and the operating ranges of its main parameters were established with a block of experiments using "artificial" requirements i.e., requirements generated automatically from pre-established and, therefore, known patterns. This

controlled experimentation environment has made it possible to define the most appropriate values of the parameters to be used in actual use cases.

With the main parameters adjusted using the previous experiments, a series of experiments have been performed using a corpus of requirements provided by the INCOSE (International Council on Systems Engineering) in 2016 and already used in previous investigations by the research group.

The results show that this method can automatically generate a set of valid patterns suitable for the generation of new requirements, improving previous investigation by 233%, with values of the ratio of matching requirements per pattern of 2.87, compared to 1.23 obtained in previous methods.

**Keywords:** Text Patterns, Genetic Algorithms, Requirements Engineering, Knowledge Reuse.

# CONTENIDOS

## Índice de Tablas



## Índice de Figuras

# 1 INTRODUCCIÓN

En este capítulo se introduce la investigación objeto de la presente tesis que ha sido desarrollada entre el periodo 2018 – 2022 en el Departamento de Informática de la Universidad Carlos III de Madrid.

Explicamos el contexto de la investigación y el problema general de la dificultad que tiene la redacción de requisitos de calidad en el ámbito de la Ingeniería de Sistemas. A partir de este problema general, proponemos una nueva aproximación para la obtención de patrones automáticamente que ayuden a la redacción de estos requisitos y a validar su calidad. Planteamos las hipótesis de partida y los objetivos específicos que queremos conseguir con la investigación.

Finalmente se apuntan las restricciones que han limitado la investigación y que pueden dar lugar a trabajos futuros de mejora.

## 1.1 Contexto de la investigación

Esta tesis se ha realizado dentro del grupo de investigación de la Universidad Carlos III de Madrid *Knowledge Reuse Group* (KR) [2]. El grupo KR investiga en las áreas de interés de representación, recuperación y reutilización del conocimiento aplicados a la Ingeniería de Software.

KR realiza proyectos de investigación para la empresa The Reuse Company (TRC). TRC ha desarrollado el conjunto de productos *Systems Engineering Suite* (en adelante SE *suite*) [3]. Se trata de un conjunto de herramientas para dar soporte a todo el ciclo de vida de la ingeniería de sistemas incluyendo la ingeniería de requisitos: autoría, calidad, verificación y validación de especificaciones, interoperabilidad con sistemas de diseño y herramientas MBSE, gestión y reutilización del conocimiento.

Estas herramientas se basan en la disponibilidad de patrones de texto para todas las actividades relacionadas con la autoría, la determinación de la calidad de los requisitos escritos por los ingenieros y la indexación de conocimiento para su reutilización.

En este contexto surge como necesidad el desarrollo de métodos y herramientas que faciliten o automaticen la creación de patrones de texto. El grupo de investigación *Knowledge Reuse Group* (KR) ha realizado distintas aproximaciones para la generación automática de patrones [4], a partir de subconjuntos determinados de requisitos de calidad de dominios determinados. Esta tesis parte de los resultados previos y, utilizando una aproximación y metodología distinta, persigue la generación automática de patrones que sirvan como fuente de configuración de la herramienta de *Knowledge Manager*, componente base del SE *suite*.

La presente investigación se relaciona con los siguientes dominios de conocimiento:

- **Ingeniería de Sistemas**: Entendida como la aplicación multidisciplinar de principios analíticos, matemáticos y científicos para formular, seleccionar, desarrollar y satisfacer las necesidades operativas de los usuarios, así como para minimizar los costes de desarrollo y del ciclo de vida del producto desarrollado [5] . Esta tesis aporta herramientas para el análisis y definición de especificaciones de calidad.

- **Ingeniería de Software**: Disciplina de la ingeniería de sistemas específica para el desarrollo de software, contemplando toda las etapas del ciclo de vida del software: desde las iniciales de conceptualización y especificación, hasta las finales de mantenimiento, incluyendo desarrollo, explotación, mantenimiento y retirada del mismo [6]. Esta investigación se enfoca a las etapas de especificación.
- **Ingeniería de Requisitos**: Rama de la ingeniería del software encargada de la captación de las necesidades de los usuarios y su transformación en requisitos formales mediante el análisis, redacción, verificación y validación de su calidad. Comprende la traslación de necesidades de usuarios a corpus de requisitos completos con distinto nivel de formalización y de abstracción según los destinatarios de los mismos (de usuario, de diseño, etc.). Nuestra investigación está centrada en la redacción y validación de requisitos.
- **Patrones**: Un patrón de requisitos es una plantilla basada en texto formada por restricciones sintácticas y semánticas. Una gran cantidad de requisitos escritos son comunes a todos los proyectos. Esto significa que al desarrollar un patrón de requisitos se puede ahorrar mucho tiempo y mejorar la calidad de los requisitos. Un patrón de requisitos permite escribir un requisito indicando qué información se necesita para ese tipo de requisito. Esta tesis desarrolla una metodología y una herramienta para la generación automática de patrones de requisitos.
- **Herramientas de Gestión de Requisitos**: Las herramientas de gestión de requisitos se pueden enmarcar dentro del conjunto de herramientas CASE (*Computer Aided Software Engineering*). Existen un amplio conjunto de herramientas de gestión de requisitos. TRC ha desarrollado una conjunto de productos software, SE *suite*, que incluye el componente RAT (*Requirements Authoring Tool*) para ayuda a la redacción de requisitos y validación de su calidad y el componente KM (*Knowledge Manager*) como repositorio y gestor de ontologías y patrones de texto. Twiga, la herramienta desarrollada en esta investigación utiliza estos componentes y sus mismos modelos de datos, para generar patrones compatibles con las mismas y se espera en un futuro integrar la funcionalidad dentro de esta *suite*.

## 1.2 Descripción del problema

Diversos estudios confirman que uno de los problemas principales causante del fracaso en los proyectos son las malas especificaciones y la mala gestión de requisitos. Así, el Project Management Institute (PMI) indica en su informe "*Pulse of the Profession*" del año 2014 [7] que la causa principal del fracaso en los proyectos se debe a una pobre definición y gestión de requisitos, cifrando en un 47% el porcentaje de proyectos fracasados debido a esta causa.

En el origen de esta mala gestión está la no disponibilidad de habilidades ni herramientas de gestión de requisitos como muestra la Tabla 1-1.

TABLA 1-1. NIVEL DE DEFINICIÓN DE MÉTODOS DE GESTIÓN DE REQUISITOS.

| Nivel | Porcentaje |
|---|---|
| Definidos completamente métodos y con herramientas de soporte | 13% |
| Alto nivel de definición de métodos y uso de herramientas | 36% |
| Parcial o mínimamente definidos métodos y herramientas | 49% |
| Sin ningún método o herramienta | 2% |

También lo refleja Ian Sommerville en su libro sobre Ingeniería de requisitos [6]. Shah & Patel [8] mencionan la necesidad de procesos de elicitación, reutilización y revisión de calidad de requisitos, así como la gestión de la volatilidad de los mismos en los desarrollos de software. The Standish Group realiza, desde el año 1994 el denominado "*The Chaos Report*". Sistemáticamente aparece en este informe la falta de calidad de requisitos como una de las causas más importantes de los fallos de los proyecto [9].

F. Brooks [10] indica que el desarrollo conceptual y establecimiento de requisitos es la parte más compleja del desarrollo de software. Además, indica que es la actividad de todas las que constituyen el ciclo de vida del desarrollo de software que tiene más impacto en los resultados finales. Para Brooks "*la tarea más importante que el ingeniero de software hace para el cliente es la extracción y el refinamiento iterativos de los requisitos del producto.*"

E. Braude [11] identifica un mal análisis de requisitos como la causa de la mayor parte de defectos en el software entregado. Estos defectos son, adicionalmente, los más cotosos de arreglar.

Por último, Marrero et al. [12] hacen una mención importante al modo en que se redactan los requisitos y a la propensión a la ambigüedad de dicho modo: "*el problema es que generalmente los requisitos son redactados en lenguaje natural, cuya falta de normalización y ambigüedad hace que sean difíciles de representar, procesar y, por supuesto, recuperar y reutilizar*" .

Identificamos por tanto que, dentro de la ingeniería de requisitos, las actividades de redacción y formalización de los requisitos y la verificación de su calidad son un problema al que se debe atender para buscar soluciones que faciliten la realización de estos procesos.

## 1.3 Motivación

El proceso de ingeniería de requisitos comprende varias actividades para obtener, analizar, especificar, verificar y validar los requisitos a diferentes niveles de abstracción (estrategia de negocio, usuario/partes interesadas, sistema, software, telecomunicaciones, concepto operativo del sistema, etc.). Normalmente, estas actividades se llevan a cabo utilizando diferentes técnicas y apoyándose en diferentes herramientas para generar finalmente la documentación de especificación.

Más concretamente, la redacción de requisitos suele realizarse mediante declaraciones basadas en texto para que sean comprensibles para todas las partes (usuarios especificadores, diseñadores, etc.). Existen otras técnicas: sistemas basados en el modelado (MBSE), sistemas más formales como los modelos lógicos (por ejemplo, la lógica temporal lineal) pero, al final, se suelen verbalizar/escribir para facilitar la comunicación entre los agentes implicados en el proceso de especificación. En este sentido, las técnicas semiformales como los patrones también se utilizan para restringir cómo se pueden escribir las declaraciones de requisitos (sintaxis) y lo que pueden especificar (semántica).

Para facilitar la redacción de requisitos y también para evaluar su calidad existen herramientas de software. SE *suite* de TRC es una de ellas.

La herramienta *Requirements Authoring Tool* (RAT) dentro de la SE *suite* está específicamente orientada a facilitar la redacción de requisitos y a validar su calidad. Esta herramienta ha sido desplegada con éxito en diferentes sistemas de seguridad crítica como el aeroespacial y el de automoción. La solución técnica está basada en la definición de un vocabulario, y de un conjunto de patrones. Los patrones son guías sintácticas y semánticas que ayudan al autor a redactar los requisitos limitando los tipos de frases que se pueden utilizar, el vocabulario, etc.; aseguran la composición sintáctica y disminuyen la ambigüedad o el uso de vocabulario no permitido. El resultado son requisitos de mayor calidad desde su redacción inicial. Los patrones también sirven para evaluar la calidad de requisitos ya redactados [13].

Los patrones que usa esta herramienta deben ser generados por especialistas que necesitan conocimientos en profundidad de la lengua en que se van a escribir los requisitos, el dominio técnico para el que se están generando y los principios de la generación de patrones sintácticos y semánticos de los productos software contenidos en la SE *suite*. Es por tanto difícil formar a estos especialistas y constituyen una barrera de entrada muy importante para la implantación de la herramienta, haciendo que el proceso de puesta en marcha y optimización de la SE *suite* sea costoso y lento.

La principal motivación de esta investigación consiste entonces en facilitar el trabajo de estos especialistas poniendo a su disposición una herramienta para la creación automática de patrones de requisitos. En esta tesis se ha definido una metodología y se ha desarrollado una herramienta, Twiga, basada en dicha metodología que genera automáticamente patrones a partir de un primer subconjunto de requisitos. Los ingenieros disponen de un primer set de patrones para configurar la herramienta, disminuyendo los tiempos y costes de implantación de SE *suite* en entornos reales de uso.

## 1.4 Hipótesis

Se ha partido de las siguientes hipótesis

- **Hipótesis 1**.- Es posible generar patrones de requisitos automáticamente usando algoritmos genéticos a partir de un corpus de requisitos de un mismo dominio previamente calificados por expertos como requisitos de calidad. Estos patrones son equivalentes en calidad a los generados por expertos.
- **Hipótesis 2**.- Si añadimos al algoritmo anterior la técnica de separar y conquistar, entonces se puede además generar un conjunto completo y limitado de patrones válido para reconocer un conjunto homogéneo de requisitos de un dominio específico.
- **Hipótesis 3**.- Es posible identificar e implementar parámetros en los algoritmos genéticos y de conquista que permitan ajustar su comportamiento a distintos escenarios reales de uso, generando poblaciones de patrones más generalistas, capaces de mayor grado de libertad para la escritura de nuevos requisitos, para escenarios en que así se requiera como en los primeros estadios de especificación, requisitos funcionales genéricos o más específicas, con menor grado de libertad para la escritura, útiles por ejemplo en fases avanzadas del diseño, donde se requiere mayor exactitud y especificidad en la redacción.

## 1.5 Objetivos

Distinguimos dos tipos de objetivos dentro de esta investigación: objetivo general y objetivos específicos. Este es el objetivo general de la investigación:

- **OBGEN1.-** Obtención de una herramienta software para la generación automática de patrones de requisitos basada en la implementación de algoritmos genéticos y de la estrategia de separación y conquista.

Adicionalmente, se consideran los siguientes objetivos específicos en la investigación:

- **OBESP1**.- Identificar, dentro de la técnica de los algoritmos genéticos, las variables que permiten alterar el comportamiento del algoritmo y sus valores más eficientes para la obtención de los mejores resultados.

- **OBESP2**.- Dotar a la herramienta de flexibilidad mediante su parametrización para conseguir distintos resultados de patrones en función del uso posterior que se quiera hacer de ellos.
- **OBESP3**.- Conseguir resultados en términos de tamaño de población de patrones mejores que los obtenidos en investigaciones anteriores utilizando aproximaciones metodológicas distintas.
- **OBESP4.-** Comprobar la validez de la herramienta con distintos corpus de requisitos.

No se han considerado objetivos dentro de la investigación, y podrán por tanto ser objeto de futuros trabajos de investigación y desarrollo, los siguientes.

- Optimizar la herramienta en cuanto a tiempos de ejecución. Se considerarán válidos tiempos de consecución de poblaciones de patrones de texto del entorno de horas en función del corpus de partida.
- Generar una interfaz de usuario válida para que la herramienta pueda ser utilizada por usuarios ajenos a esta investigación.

## 1.6 Limitaciones a la investigación

Las principales restricciones y condicionantes de la investigación que han limitado el ámbito de estudio y los resultados han sido:

### *1.6.1 Set de datos: el problema del juego de imitación*

Una restricción importante de esta investigación ha sido que, en la experimentación, todos los conjuntos de requisitos (RS) provienen de la misma fuente (requisitos de INCOSE) y han sido redactados por unos pocos analistas. Adicionalmente, han sido preseleccionados solamente los requisitos considerados como de calidad.

Esta aproximación presenta el problema de que los resultados, los patrones obtenidos, tengan un sesgo y propongan solamente patrones ajustados a la manera de redactar de los autores de los requisitos con los que se ha trabajado, fomentando el juego de imitación y penalizando la creatividad o la obtención de otros patrones igualmente válidos para el dominio pero que no pueden obtenerse a partir de los requisitos con los que se trabaja.

Esto es una limitación es intrínseca al modelo y a la investigación, puesto que los patrones que encuentren los algoritmos siempre están basados en los conjuntos de requisitos RS de entrada.

No obstante, se propone en trabajos futuros, diseñar experimentos específicos combinando conjuntos de requisitos (RS) para analizar esta limitación y encontrar posibles alternativas de mejora.

### *1.6.2 Confidencialidad en el uso de los juegos de datos reales*

En la investigación la experimentación se ha realizado con un conjunto de 10 corpus de requisitos proporcionados por INCOSE (Organización Internacional de Ingeniería de Sistemas) [14] y utilizados en investigaciones previas [4]. Se trata de conjuntos de requisitos reales de proyectos de software.

Los requisitos proporcionados por el INCOSE están protegidos por un acuerdo de confidencialidad. Por ello, en las tablas y figuras de ejemplo que se muestran en esta tesis, hemos utilizado requisitos textuales alternativos.

### *1.6.3 Verificación de validez de los resultados en entornos reales de trabajo*

Los juegos de patrones encontrados no han podido ser validados en los procesos siguientes de autoría de requisitos o procesos de verificación de la calidad de requisitos utilizando la SE *suite*, en entornos reales de trabajo. Esto es debido a que solamente se han generado juegos de patrones con conjuntos de requisitos reales, pero correspondientes a dominios y entornos ajenos al grupo de investigación.

Un trabajo posterior a esta tesis consistirá en realizar esta validación con entornos reales de clientes de la SE *suite* de TRC y conjuntos de requisitos reales.

### *1.6.4 Uso de otros algoritmos bioinspirados*

Existen otros algoritmos inspirados en comportamientos que encontramos en la naturaleza, adicionales a los algoritmos genéticos o evolutivos, que al inicio de la investigación se consideraron como posibles líneas de investigación. Concretamente se

consideraron de manera teórica, sistemas inmunes artificiales (AIS) o algoritmos inmunológicos [15] y algoritmos de optimización de colonia de hormigas [16].

Cuando realizamos las primeras pruebas de concepto de los algoritmos genéticos se obtuvieron buenos resultados, pero encontramos que solo el alcance del desarrollo completo de estos algoritmos genéticos y los de conquista iban a requerir un tiempo y esfuerzo que no permitirían explorar el resto de algoritmos.

Adicionalmente, otras investigaciones similares [17] se habían basado exclusivamente en el uso de algoritmos genéticos y de conquista.

Decidimos entonces, excluir de los objetivos de la investigación la exploración de otros algoritmos, su diseño para la resolución específica del problema de creación de patrones automáticos y la experimentación correspondiente.

## 1.7 Planificación global

Esta investigación se ha realizado siguiendo las prácticas propuestas por PMI para gestión de proyectos. Se ha seguido un modelo de desarrollo híbrido [18], con procesos iniciales de enfoque y estudio en cascada, seguidos de tres iteraciones completas de desarrollo; cada iteración ha incluido los procesos de análisis, desarrollo, experimentación y revisión de resultados e identificación de mejoras. La Figura 1-1 muestra a alto nivel los grandes procesos que se ha seguido y la planificación general

FIGURA 1-1 METODOLOGÍA UTILIZADA PARA EL DESARROLLO DE LA INVESTIGACIÓN


### *1.7.1 Enfoque y revisión bibliográfica*

Este proceso consistió en buscar alternativas de solución técnica al problema de generación automática de patrones para requisitos.

Se analizaron aproximaciones previas, como el trabajo de investigación dentro del grupo realizado para la definición de una metodología orientada a la optimización de la calidad de requisitos [4]. A. Fraga y V. Moreno, directora y tutor de esta tesis, también habían realizado intentos previos con otras aproximaciones [19] [20].

También se inició una búsqueda sistemática bibliográfica de los temas e investigaciones relacionadas con el problema de la generación automática de patrones en general y de patrones de texto para requisitos en particular, así como técnicas de procesamiento de lenguaje natural y uso de algoritmos genéticos y técnicas de separación y conquista como métodos a aplicar en nuestra investigación:

- **Ingeniería de Requisitos**: literatura general, definiciones, procesos, metodologías y herramientas de apoyo a la ingeniería de requisitos.
- **Reutilización del conocimiento**: Conceptos y definiciones; investigaciones relacionadas con el uso de patrones como base para la reutilización de conocimiento, considerando en este caso los requisitos y su procesamiento con

técnicas de NLP y con definición de ontologías, como base de conocimiento; cómo enmarcar esta investigación dentro de este área de conocimiento.

- **Procesamiento de Lenguaje Natural**: Técnicas estándar de procesamiento de lenguaje natural. Uso de componentes (slots) sintáctico – semánticos para la construcción de patrones. Usamos la misma estructura y formato de requisitos y patrones que utiliza la herramienta *SES Engineering Studio,* utilizando sus algoritmos de procesamiento de lenguaje natural para tratar los requisitos y generar los patrones de requisitos.
- **Patrones**: Conceptos generales y aplicación del concepto de patrón como plantilla de texto para la escritura de requisitos. Se ha utilizado en el mismo sentido y con las mismas características estructurales y de comportamiento que se utiliza en las herramientas de la SE *suite.*
- **Algoritmos genéticos**: Definiciones y conceptos básicos; diseño y programación de este tipo de algoritmos. También se recogieron trabajos de investigación que utilizaran estos algoritmos en distintos dominios y también para búsqueda de patrones; los casos de uso similares encontrados sirvieron como base para la definición de un enfoque concreto de investigación.
- **Técnica de separación y conquista**. Partimos del artículo de J. Furnkranz [21], en el que plantea esta técnica general de búsqueda de conjunto de soluciones parar grupos de elementos. Posteriormente encontramos también artículos donde se utiliza esta técnica para la generación de patrones para un corpus de expresiones regulares [17].

De toda la bibliografía analizada, los trabajos realizados por Bartolli et al. [22] para conseguir patrones en expresiones regulares basados en el uso de programación genética y técnicas de separación y conquista sirvieron de base para plantear nuestro nuevo enfoque del problema.

El resultado final de esta fase fue la definición de nuestras hipótesis de trabajo tal y como se describen en el apartado 1.4 de esta tesis.

### *1.7.2 Iteraciones de desarrollo y experimentación*

Una vez realizado el enfoque y planteadas hipótesis iniciamos iteraciones completas de avance con una misma metodología: análisis y diseño técnico de una solución, programación y desarrollo de la herramienta, experimentación con distintos juegos de datos y revisión de resultados para validar el planteamiento y encontrar mejoras para la siguiente iteración.

La primera iteración consistió en el desarrollo de un prototipo general para validar que era posible encontrar patrones en grupos de requisitos. Los resultados fueron satisfactorios y el prototipo permitió identificar variables o parámetros que afectaban al comportamiento del algoritmo.

El objetivo de la segunda iteración fue la incorporación de la técnica de separación y conquista, así como la calibración de los distintos parámetros de comportamiento de los algoritmos. Se utilizaron requisitos artificiales generados a partir de patrones definidos por nosotros con el fin de disponer de juego de datos controlados.

Finalmente, la última iteración permitió generar la versión de la herramienta Twiga 3, válida para la experimentación con conjuntos reales de requisitos. Se utilizaron juegos de requisitos proporcionados por INCOSE.

### *1.7.3 Experimentación final. Análisis y generación de documentación*

Utilizando la herramienta Twiga 3 y los distintos conjuntos de requisitos de INCOSE, se realizaron juegos de experimentos reajustando la calibración de parámetros para mejorar los resultados finales.

A partir de estos resultados se procedió a la generación de documentación: publicación [1] y redacción de memoria final de la tesis.

## 1.8 Estructura de este documento

La presente memoria está organizada por capítulos con el contenido que se resume a continuación:

- **Capítulo 1 – Introducción**: Contexto y objetivos de la investigación. Descripción detallada del problema que se desea resolver y las motivaciones origen de esta tesis. Lista sistemática de hipótesis planteadas y objetivos a cumplir en el desarrollo de la investigación. También se indican las limitaciones que se han encontrado en la investigación.
- **Capítulo 2 – Estado del arte**: Análisis y descripción de los temas relacionados con la investigación. Capítulo descriptivo de la situación actual referida a los principios, métodos, técnicas y productos e investigaciones relacionadas con esta investigación. Incluye una breve descripción de la investigación que se ha usado como referencia realizada dentro del grupo KR de investigación de la Universidad Carlos III para la obtención automática de patrones de requisitos utilizando una metodología distinta.
- **Capítulo 3 – Solución. Metodología**: Detalle de la metodología desarrollada en el marco de investigación. Incorpora un apartado dedicado a definiciones propias del marco de investigación y de la propia metodología. Describe los procesos definidos, entradas, salidas y objetivos. Finalmente, en este capítulo se describe también la herramienta Twiga y cómo se ha desarrollado.
- **Capítulo 4 – Experimentación y Resultados**: Presentación y análisis de los experimentos y resultados. Se detallan los resultados obtenidos para validar la metodología y se presentan otros datos resultados de la investigación de interés.
- **Capítulo 5 – Conclusiones, Aportaciones y trabajos futuros**: Se presentan las conclusiones de la investigación, hasta qué punto se han alcanzado los objetivos, si se han verificado las hipótesis y las principales aportaciones académicas y tecnológicas de la investigación. Finalmente se recogen líneas de investigación y trabajo que no han sido abordadas en esta tesis por exceder el marco temporal y de trabajo pero que consideramos interesantes abordar en trabajos futuros.
- **Capítulo 6 – Bibliografía**: Referencias y fuentes de información utilizadas en esta investigación.

- **Capítulo 7 – Anexos**: Información adicional a la tesis, incluyendo el registro sistemático de los resultados de los experimentos que se han utilizado para la redacción de la tesis y presentación de conclusiones.

# 2 ESTADO DEL ARTE

Este capítulo analiza la situación actual en el ámbito científico y también en el de desarrollo de productos comerciales de apoyo a la Ingeniería de Sistemas e Ingeniería de software y, más concretamente, productos de soporte a la redacción de requisitos; métodos de procesamiento de lenguaje natural y otras técnicas asociadas.

Hacemos también una descripción del estado actual de avance de los algoritmos genéticos, sus características, funciones principales y ámbitos de aplicación. Añadimos a esta técnica la estrategia de conquista para resolución de problemas. Estas dos técnicas serán la base de nuestro algoritmo de generación automática de patrones.

Por último, describimos trabajos previos relacionados y realizados en el mismo grupo de investigación. Estos trabajos permiten contrastar los resultados de nuestra investigación.

## 2.1 Ingeniería de Sistemas

Esta investigación se enmarca en la disciplina de la Ingeniería de requisitos que a su vez se puede considerar como una parte de la Ingeniería de Sistemas, que consiste en la identificación de necesidades, su análisis, diseño e implementación de soluciones considerando todo el ciclo de vida de la solución, y todas las restricciones asociadas (de coste, regulatorias, etc.).

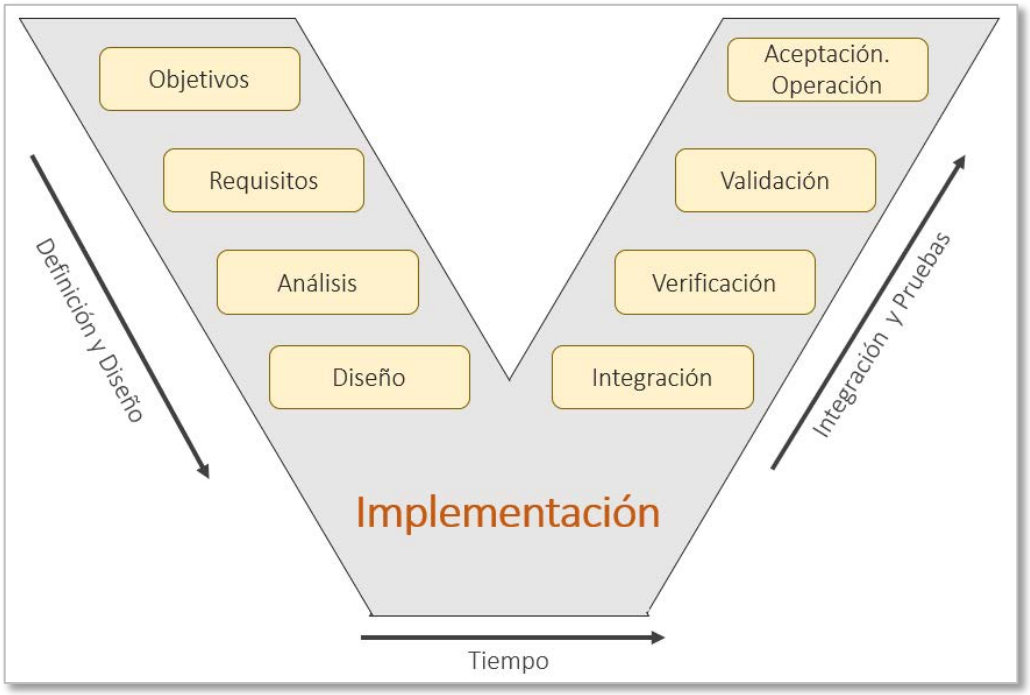
El INCOSE (*International Council on Systems Engineering*) hace la siguiente definición [23]: "*La ingeniería de sistemas es un enfoque transdisciplinario e integrador que permite la realización, el uso y la retirada con éxito de los sistemas de ingeniería, utilizando principios y conceptos de sistemas y métodos científicos, tecnológicos y de gestión*".

El INCOSE utiliza el término ingeniería en un sentido amplio como la acción de trabajar para conseguir algo.

En cuanto a los Sistemas de Ingeniería, pueden estar compuestos por personas, productos, servicios, información y procesos. Los Sistemas pueden tener Subsistemas y éstos a su vez, otros Subsistemas de menor nivel.

Aunque no es un estándar, el modelo en V o *V-diagram* es un modelo de procesos visual y popular para definir las grandes actividades incluidas dentro de la ingeniería de sistemas, el ciclo de vida global. Se muestra en la Figura 2-1 [24] [25][26].

FIGURA 2-1 MODELO V DE DESCRIPCIÓN DE PROCESOS DE IS. ADAPTACIÓN DE [25]




INCOSE utiliza también el modelo V y propone las siguientes actividades dentro de su *Systems Engineering Process*:

- **Definición de objetivos**: Establecimiento y balanceo de objetivos. Integración y equilibrio de los objetivos de los distintos interesados, propósito general del sistema y especificación de los criterios de éxito. También se puede incluir aquí la identificación de las necesidades reales o previstas de los interesados, el concepto operativo y la funcionalidad requerida.
- **Definición del ciclo de vida**: establecer un modelo de ciclo de vida para el sistema a implementar, desde su concepción, pasando por su implementación, soporte y mantenimiento posterior y, en última instancia, su retirada. Se enfoca en procesos y unas estructuras de gobierno adecuados, teniendo en cuenta los niveles de complejidad, incertidumbre, cambio y variedad.
- **Definición de Línea Base de requisitos**: elaboración de una línea de base de requisitos, incluyendo su verificación y su validación con los interesados. y la modelización de los requisitos y la arquitectura de la solución seleccionada para cada fase de la empresa**.**
- **Análisis y diseño de la solución**: generación de conceptos, prototipos, arquitecturas y soluciones alternativas.

- **Implementación y validación del sistema**: Conforme a la especificación; incluye también la verificación y validación del propio sistema, su funcionamiento, el cumplimiento de especificaciones mediante la ejecución de las pruebas que se hayan definido.
- **Operación, Mantenimiento y Retirada del sistema:** Las actividades de explotación del sistema desarrollado, su soporte y mantenimiento son también consideradas como parte a implementar por la Ingeniería de Sistemas, así como la definición de actividades de retirada del Sistema cuando éste ha cumplido su vida útil o los objetivos para los que se definió.

Es importante considerar que tanto en el modelo V como en el *Systems Engineering Process* de INCOSE, todas las actividades de diseño, implementación, validación y operación del sistema se sustentan en la existencia de un corpus de requisitos completo generado en las actividades previas de definición. Todos los elementos del sistema, así como todas las pruebas de validación, deben estar trazados con uno o más de los requisitos definidos. Asimismo, cada requisito debe ser abordado por al menos un elemento del sistema y una prueba de validación. Este rigor garantiza que no se haga nada innecesario por un lado y que todos los requisitos estén reflejados en el producto final, como funcionalidad, restricción, etc. [23].

En este sentido, la investigación objeto de esta tesis adquiere especial importancia por cuanto su objetivo es el soporte a la redacción de requisitos de calidad.

## 2.2 Ingeniería de Requisitos

La Ingeniería de requisitos es un componente de la Ingeniería de Sistemas que tiene como objetivo es comprender y extraer las necesidades de los actores que actúan como especificadores del sistema (clientes, usuarios…), registrarlas en conjuntos formales de requisitos y gestionar su cumplimiento y evolución a lo largo del desarrollo de un sistema.

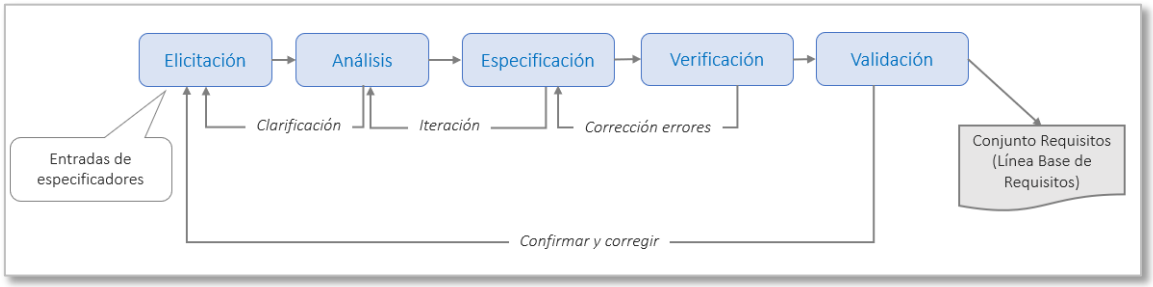
La norma ISO/IEC/IEEE-29148 define la Ingeniería de requisitos como "*Una función interdisciplinar que media entre los dominios del cliente y el proveedor para establecer y mantener los requisitos que deber cumplir un sistema, software o servicio de interés. La Ingeniería de requisitos se ocupa de descubrir, obtener, desarrollar, analizar,*

*determinar métodos de verificación, validez, comunicación, documentación, y gestión de los requisitos".*

P. Loucopoulos [27] hace la siguiente definición referida específicamente a los desarrollos de software: "*La Ingeniería de requisitos se encarga de todas la actividades que intentan comprender las necesidades exactas de los usuarios de un sistema software y traducir tales necesidades a especificaciones precisas y no ambiguas para que posteriormente puedan ser usadas en el desarrollo de un sistema. La Ingeniería de requisitos puede ser definida como el desarrollo sistemático de los requisitos a través de un proceso iterativo y cooperativo en el que se analiza el problema, se documenta el resultado en diversos formatos de representación, y se comprueba la exactitud de la comprensión alcanzada*".

La Figura 2-2 muestra los procesos generales de generación y gestión de requisitos [28]:

FIGURA 2-2 MODELO GENERAL DE PROCESOS DE INGENIERÍA DE REQUISITOS.




- **Elicitación**: Proceso para el descubrimiento y escritura de los requisitos a partir de las partes interesadas, documentación o sistemas previos, etc. Las técnicas más habituales son las entrevistas, el prototipado, análisis de documentación previa y consulta de expertos. Este es el primer paso del desarrollo de requisitos.
- **Análisis**: Consiste en la comprensión precisa de cada necesidad y su clasificación. Se utilizan entrevistas con expertos y reuniones con los interesados para la aclaración de conceptos y verificación de la comprensión correcta de las especificaciones.
- **Registro y Especificación**: Consiste en escribir o representar los requisitos recopilados de una manera bien organizada que facilite la comunicación eficaz y la gestión de los requisitos durante todo el ciclo de vida de desarrollo del sistema. El resultado es un conjunto de requisitos escritos y diagramas entendibles por

todos los implicados en el proyectos. Constituyen en el estándar de PMBoK de PMI [29] parte del *Scope Baseline*, la línea base a partir de la cual se desarrolla el sistema.

- **Verificación**: Este proceso consiste en analizar la validez interna de los requisitos, desde su corrección sintáctica y semántica, hasta la consistencia de todo el conjunto de requisitos (posibles contradicciones, repeticiones, etc.). Esta actividad, que puede ser realizada de forma autónoma por los ingenieros de requisitos, debe llevarse a cabo antes del proceso de validación con las partes interesadas.
- **Validación**: Consiste en confirmar que se ha identificado y especificado el conjunto correcto de requisitos para construir una solución que cumpla con las necesidades y especificaciones. Esta tarea revisa los requisitos documentados con los propios interesados y se asegura junto con ellos que lo escrito se corresponde con sus necesidades; también se concretan los criterios de aceptación de los productos a desarrollar a partir de los requisitos. Los requisitos no validados exigen más elicitación, análisis o especificación.

Esta investigación se enfoca en el proceso de registro y especificación, así como en el de verificación, desarrollando una herramienta que facilite la redacción con calidad del conjunto de requisitos que van a constituir la especificación del sistema.

## 2.3 Herramientas de soporte a la Ingeniería de requisitos

Existe un buen número de productos software de mercado que automatizan total o parcialmente los procesos de ingeniería de requisitos. Estas herramientas están enfocadas principalmente en proporcionar de un repositorio centralizado de requisitos, así como a gestionar la trazabilidad de dichos requisitos contra los diseños, versiones de desarrollo del producto software en cuestión y traza con pruebas de verificación y validación. El término trazabilidad se utiliza en el sentido de indicar que existe algún tipo de relación entre el requisito en y los elementos que van a componer el sistema software (componentes de diseño, clases de modelo de información, etc.).

Las herramientas también proporcionan funcionalidades de agrupamiento y clasificación de requisitos, por tipologías estándares, por bloques funcionales, etc. Las agrupaciones

pueden tener a su vez subagrupaciones y atributos específicos de las mismas. La funcionalidad que subyace a esta característica es la capacidad de búsqueda y ordenación de requisitos y en general al proceso de gestión de estos.

En este grupo de herramientas se encuentran los productos comerciales Visure [30] de *Visure Solutions*, DOORs de IBM [31] o Jama Connect de jama software [32] entre otros muchos.

El siguiente nivel de soporte al proceso de Ingeniería de requisitos lo proporcionan las herramientas, a veces extensiones de los propios productos comerciales anteriores, que apoyan la actividad de redacción (*Authoring*) de requisitos. Requirements Authoring Tool RAT [33] es parte de la SE *suite* TRC e incluye soporte a la redacción y verificación de la calidad de los requisitos escritos. Está basada en el uso de patrones de texto y guía a los autores en la actividad de redacción a partir de diccionarios y patrones para dominios determinados. La investigación objeto de esta tesis se enmarca en la obtención automática de los patrones de requisitos que la herramienta RAT requiere a partir de conjuntos informados de requisitos de calidad.

## 2.4 Procesamiento de lenguaje Natural

El procesamiento del lenguaje natural (NLP) es un área del desarrollo de sistemas de software cuya finalidad es la interpretación de textos escritos en lenguaje natural por parte de los sistemas informáticos. Busca entender el lenguaje natural mediante la sistematización del vocabulario, el análisis sintáctico y gramatical y el análisis semántico de las palabras.

Las técnicas de Procesamiento del Lenguaje Natural se utilizan para resolver multitud de situaciones:

- Extracción de información; minería de textos y de contenido: determinar el conocimiento de los documentos a partir de una colección documental [34]
- Comprensión: representación de texto en lenguaje natural usando los términos que contiene y las relaciones entre los mismos
- Traducción automática y Procesamiento de voz

- Generación de texto: resúmenes, soporte a la autoría y redacción de mails [35]
- Corrección ortográfica, de estilo o incluso soporte al lenguaje [36]

El Procesamiento de Lenguaje Natural comienza con una serie de tareas que se denominan colectivamente normalización del texto [37]. Normalizar el texto significa convertirlo en una forma estructurada, más fácil de tratar por los algoritmos. Una vez normalizado se le aplican técnicas de análisis léxico, sintáctico, semántico y pragmático:

- **Segmentación de oraciones:** el primer paso es dividir el texto a analizar en oraciones. Las frases pueden ser consideradas como ideas diferentes con un significado singular. En la Ingeniería de requisitos un requisito suele estar constituido por una frase.
- **Tokenización**: consiste en dividir la frase en palabras utilizando una lista de separadores. En las lenguas con alfabeto latino se consideran separadores de palabras el espacio, la coma, el punto, el punto y coma y otros signos de puntuación. Cada palabra recibe el nombre de token.
- **Lematización o análisis léxico**: El análisis léxico tiene como objetivo obtener etiquetas estándar para cada palabra o token mediante un estudio que identifique el giro del vocabulario, por ejemplo, el género, el número y las irregularidades verbales de las palabras candidatas. Una forma eficiente de realizar este análisis es mediante el uso de un autómata finito que toma un repositorio de términos y equivalencias entre términos para hacer una conversión de un token a un formato estándar [38].
- **Análisis sintáctico y etiquetado gramatical**: El objetivo del análisis sintáctico es encontrar las relaciones sintácticas entre las palabras. Esto permitirá el entendimiento de la semántica de los tokens en fases posteriores. En este proceso se asignan categorías gramaticales a los términos o tokens de la oración. Las etiquetas se definen en un diccionario de términos estándar vinculados a categorías gramaticales (sustantivos, verbos, adverbios, etc.). Para hacer más eficiente este proceso se realiza con anterioridad los procesos de normalización (análisis léxico). El etiquetado gramatical es un factor clave en la identificación y generación de patrones, en los que los patrones consisten en categorías sintácticas o gramaticales y no en los propios términos. La precisión de esta técnica depende en gran medida de riqueza del diccionario de etiquetas gramaticales.

- **Análisis semántico y pragmático**: el análisis semántico tiene como objetivo interpretar el significado de las expresiones. Se realiza después del análisis léxico y análisis sintáctico. Los problemas de implementación se deben a la ambigüedad léxica del lenguaje, ya que una palabra homónima puede tener múltiples categorías gramaticales; otros tipos de ambigüedad son la ambigüedad referencial que consiste en que el significado de los términos puede variar según sus pronombres y la ambigüedad de alcance que se produce cuando hay palabras adicionales en el contexto que cambian el significado de las palabras relacionadas [20]. El análisis pragmático, por su parte, también depende de las etapas anteriores y tiene como objetivo interpretar las palabras o frases en un contexto definido [39]. Este análisis no sólo contempla la semántica del término analizado, sino que también considera la semántica de los términos contiguos dentro del mismo contexto.

En esta investigación se han utilizado las herramientas y metodologías desarrolladas por del grupo de investigación de la Universidad Carlos III de Madrid *Knowledge Reuse Group* (KR) que se describen en el apartado 2.7 de esta tesis. Estas herramientas no hacen uso del análisis pragmático, pero si del resto de métodos para el análisis de requisitos y para la elaboración y almacenamiento de patrones.

## 2.5 Patrones

### *2.5.1 Concepto de patrón*

Los patrones son una técnica de reúso de la información. Se trata de volver a utilizar soluciones ya implementadas a problemas nuevos que tienen un cierto grado de similitud con el problema anterior resuelto cuya solución se va a utilizar como "patrón". Así la misma solución, adaptada, puede ser usada para resolver otros problemas similares en distintas situaciones. Los patrones son la manera de transmitir formalmente la información y experiencia previamente adquirida.

El concepto de patrón como forma genérica de registrar y comunicar soluciones válidas a problemas similares se ha extendido a muchos ámbitos de la actividad humana en general [40]. Según F. Buschman un patrón describe una solución genérica a problemas de

diseño recurrentes. Un patrón, como modelo de solución, se compone de un conjunto de elementos, relaciones y formas en que dichos componentes interactúan entre sí.

Esta definición de patrón fue aplicada por C. Alexander en el ámbito de la ingeniería civil y describe con precisión el concepto y uso de patrón [41]: "*Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma*".

O. Hurtado [42] identifica las siguientes características o propiedades de los patrones, válidas para su aplicación en cualquier ámbito de uso del concepto de patrón:

- Debe ser una solución que compila conocimientos y experiencias previas de expertos.
- Debe facilitar la comunicación, ser un medio de transmisión de esos conocimientos y experiencias a nuevos expertos o actores, facilitando asimismo el aprendizaje.
- La solución ha debido ser experimentada en más de una ocasión en resolución de problemas similares.
- Debe proveer una plantilla conceptual con nivel de abstracción válido para su aplicación como solución a nuevos problemas que puedan aparecer de carácter similar a los ya resueltos.

### *2.5.2 Patrones aplicados a la ingeniería de software*

El concepto genérico de patrón se traslada al ámbito de la ingeniería de software aplicado a modelos de diseño de software, en el sentido de maneras de resolver retos de desarrollo; son los denominados patrones de diseño definidos por E. Gamma [43]. En los patrones de diseño, cada patrón describe la solución a un problema común en un entorno determinado. Para E. Gamma Un patrón de diseño "*abstrae e identifica los aspectos claves de una estructura de diseño común*"; también lo aplica al concepto de diseño orientado a objetos reutilizables. Cada patrón de diseño se centra en un problema concreto.

Esta primera aplicación del concepto de patrón se ha extendido a todos los ámbitos de la ingeniería de software de manera que se pueden identificar multitud de definiciones de patrones aplicados a distintos momentos del ciclo de vida y métodos dentro de la ingeniería de software:

- **Patrones de análisis**: Estos patrones están orientados a facilitar los procesos de análisis o búsqueda de soluciones a partir de unas especificaciones. Facilitan la elaboración de los conceptos y soluciones que el sistema de software va a implementar. Son modelos y métodos más o menos formales que facilitan a los analistas su labor y van desde plantillas documentales hasta sistemas de representación gráfica [44].
- **Patrones de arquitectura**: Definen un modelo arquitectónico o meta arquitectura para estructurar y organizar un sistema software. Proporcionan módulos predefinidos y relaciones entre los mismos [45] [40].
- **Patrones de diseño**: Describen estructuras comunes de componentes de sistemas de software. Fue la primera aproximación al uso de patrones en ingeniería de software [43] [46].
- **Patrones de programación**: Proponen plantillas de programación para resolución de algoritmos o situaciones concretas recurrentes. Pueden ser genéricos o incluso específicos de lenguajes de programación, explotando las características propias de cada lenguaje [47] [48].

### *2.5.3 Patrones de requisitos*

El concepto de patrón se ha extendido también al ámbito de la ingeniería de requisitos. Se definen como conjuntos de restricciones sintáctico-semánticas que deben cumplir los requisitos para ser considerados como válidos para ese patrón determinado. Si se siguen correctamente, las patrones son una forma sencilla de aumentar la calidad de los requisitos, evitan las estructuras complejas, la ambigüedad y la incoherencia. Además, las plantillas facilitan el análisis automatizado de los requisitos por parte de herramientas de procesamiento del lenguaje natural o por herramientas de gestión de requisitos como la SE *suite*.

Dos patrones genéricos de requisitos bien conocidos son, por un lado, el definido por K. Pohl y C. Rupp conocido como Rupp's "*boilerplate*" [49] que se muestra en la Figura 2-3 y por otro el modelo o plantilla EARS (*Easy Approach to Requirements Syntax*) [50], introducida por A. Mavin et al. y mostrada en la Figura 2-4**.**

FIGURA 2-3 RUPP'S BOILERPLATE

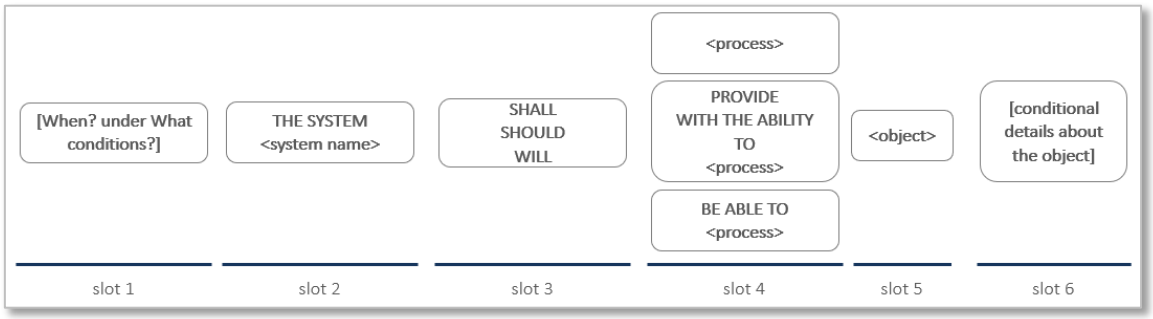
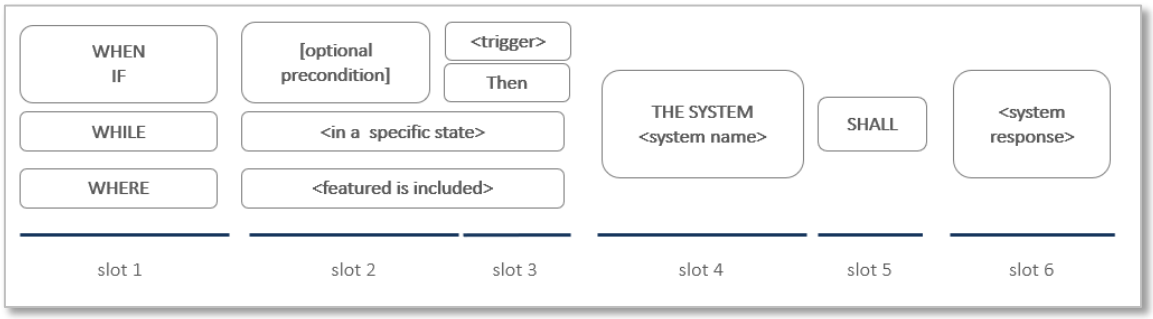



FIGURA 2-4 EARS BOILERPLATE




Se tratan de plantillas de propósito general que proporcionan estructuras gramaticales básicas para escribir requisitos. Ofrecen un mecanismo sencillo para evitar estructuras complejas, ambigüedad, etc.

En estos patrones cada "caja" representa un espacio contenedor donde se pueden insertar palabras que cumplan con la restricción indicada. En esta investigación denominaremos a los espacios contenedores con el término "slot" y a las palabras permitidas en cada slot con el término "token".

En estas definiciones aparece otro concepto que será muy útil para la generación de requisitos: concepto de slot opcional, esto es, un requisito es válido para el patrón tanto si hay contenido en el slot o no (ie.: [optional precondition] en la Figura 2-3 y Figura 2-4)

El patrón de Rupp se compone de los siguientes slots:

- una condición opcional al principio del requisito,
- el nombre del sistema,
- un modal (deberá)que especifica la importancia del requisito,
- la funcionalidad; este slot puede redactarse de tres formas diferentes dependiendo de cómo se vaya a realizar la funcionalidad prevista,
- el objeto para el que se necesita la funcionalidad o al que se aplica la funcionalidad,
- detalles adicionales opcionales sobre el objeto.

El patrón de EARS se compone de los slots:

- una condición opcional al principio,
- el nombre del sistema,
- un modal,
- la respuesta del sistema que representa su comportamiento.

Estos primeros patrones, son patrones muy genéricos, válidos para cualquier dominio. Carecen del concepto de slot semántico, esto es un slot donde sólo se pueden introducir términos de un vocabulario determinado.

O. Daramola [51] introduce una definición más genérica de patrón de requisitos como: "*una plantilla textual de especificación de requisitos, que están basados en patrones predefinidos con el fin de reducir ambigüedad y asegurar la consistencia en la forma de expresar los requisitos*".

E. Hull, J. Dick y K. Jackson [52] en su libro *Requirements Engineering* avanzan en la definición de patrones o *boilerplates*. Proponen la definición de conjuntos de *boilerplates* para escribir, coleccionar y clasificar las diferentes maneras de expresar necesidades. Sus definiciones de patrones son mucho más flexibles por cuanto permiten incorporar cualquier tipo de slot y cualquier tipo de estructura. Además, incorporan el concepto de

slot semántico. El siguiente es un ejemplo de boilerplate tal y como lo definen los autores en el libro referenciado:

```
The <system> shall <function> <object> every <performance> <units>
```

Los términos que aparecen entre los símbolos “<” y “>” representan un espacio contenedor, que denominaremos en adelante slot, donde se pueden insertar palabras que cumplan con la restricción indicada.

De nuevo la definición de patrón está basada en una secuencia de slots, que en este caso pueden representar tanto restricciones sintácticas o gramaticales (la propia estructura del patrón) como restricciones semánticas (ie: system).

Por último, S. Withall también introduce los patrones de requisitos en su libro *Software Requirements Patterns* [53]. Propone un conjunto de 30 modelos de patrones de requisitos. Según S. Withall más de la mitad de los requisitos para sistemas de software comunes pueden ser reusados entre aplicaciones.

## 2.6 Generación automática de patrones de texto

Existen investigaciones previas en el ámbito del procesamiento del lenguaje natural para la generación automática de patrones de texto, con distintas finalidades. E. Riloff [54] generó patrones de texto automáticamente para mejorar la base de conocimiento y mejorar mediante tratamientos estadísticos la indización de partes relevantes. En un trabajo previo [55] la autora realizó un sistema de construcción de diccionarios automáticos mediante la extracción automática de patrones utilizando reglas heurísticas con textos previamente cualificados con tags sintácticos y semánticos.

C. Rehberg [56] dispone de la patente "*Automatic pattern generation in natural language processing*" en la que se mejora los procesos de búsqueda y reutilización de información a partir de la comprensión de las frases de un texto determinado. En esta aproximación se generan patrones de equivalencia de significado de las palabras que componen las frases. A partir de diccionarios que son bases de datos de palabras y sus categorías, así como su

asociación semántica se crean patrones para relacionar las palabras con su significado a partir de las palabras que aparecen en una frase determinada.

A. Fraga y V. Moreno, directora y tutor de esta tesis, han realizado investigaciones previas con otras aproximaciones [19] [20][57]. En estos casos, la metodología está basada en el uso de técnicas de procesamiento de lenguaje natural tratando a los requisitos y convirtiéndolos en conjuntos de tokens. Una vez transformados en tokens sintáctico-semánticos, se procede a contar frecuencias de parejas adyacentes de tokens. Aquellas parejas que aparecen como más frecuentes se consideran primeros patrones básicos (patrones de 2 elementos). Estas parejas se sustituyen allá donde aparecen por un nuevo slot representando al subpatrón y se repite el proceso. Una aproximación similar fue también utilizada por E. Parra [4] en un trabajo similar a esta investigación que se describe en el apartado 2.10 de esta tesis.

## 2.7 Herramientas del grupo de investigación KR utilizadas

El grupo de investigación de la Universidad Carlos III de Madrid Knowledge Reuse Group (KR) ha desarrollado un conjunto de herramientas para el reúso de la información. En esta investigación se han utilizado herramientas y modelos desarrollados por el grupo:

- **Modelo RSHP** [58]: Modelo base de representación de información; usado para el almacenamiento de información de requisitos y patrones.
- **Indexador del Knowledge Manager (KM)**: para el procesamiento de lenguaje natural de los requisitos y su almacenamiento conforme al modelo RSHP

### *2.7.1 Modelo RSHP*

RSHP es un modelo relacional de representación de la información basado en relaciones que maneja todo tipo de artefactos (modelos, textos, códigos, bases de datos, etc.) utilizando un mismo esquema [59].

En el Grupo de Reutilización del Conocimiento de la Universidad Carlos III de Madrid se utiliza el modelo RSHP para proyectos relacionados con el procesamiento del lenguaje

natural [60] [61]. El modelo RSHP permite crear una ontología debido a su poder para representar cualquier tipo de conocimiento.

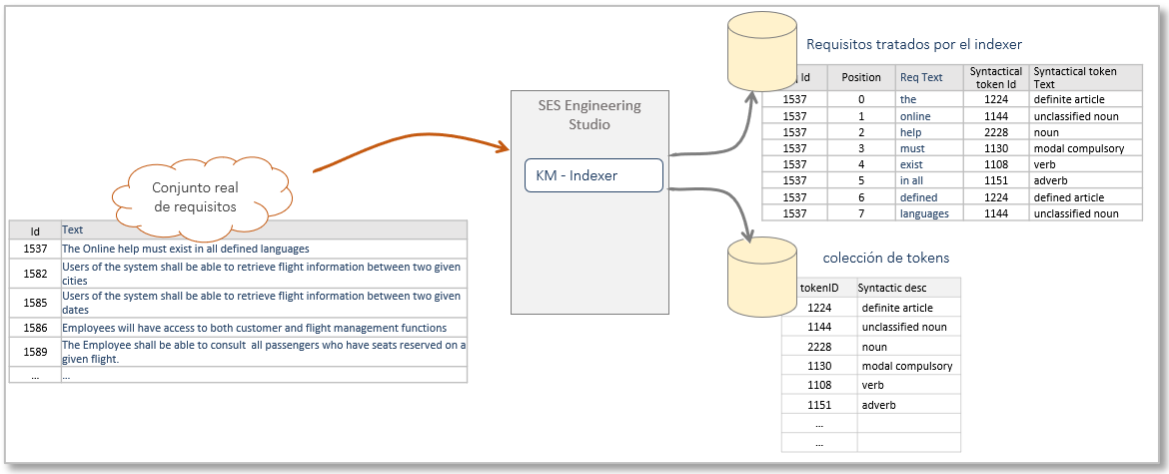
También se utiliza este modelo para almacenar y relacionar los requisitos y los juegos de patrones dentro de la SE *suite*.

Esta investigación hace uso de este modelo para la representación de requisitos, tokens sintáctico-semánticos y patrones de requisitos.

### *2.7.2 Indexador del Knowledge Manager (KM)*

El indexador es una herramienta desarrollada por KR que es utilizada en la SE *suite*. Analiza los requisitos utilizando las etapas estándar de PNL de Tokenización de palabras, Lematización y etiquetado gramatical y semántico. Asigna etiquetas sintácticas a cada palabra o elemento de los requisitos (token), como se muestra en la Figura 2-5**.**

FIGURA 2-5 CÓMO PROCESA KM INDEXER LOS REQUISITOS.




| Id | Text |
|---|---|
| 1537 | The Online help must exist in all defined languages |
| 1582 | Users of the system shall be able to retrieve flight information between two given cities |
| 1585 | Users of the system shall be able to retrieve flight information between two given dates |
| 1586 | Employees will have access to both customer and flight management functions |
| 1589 | The Employee shall be able to consult all passengers who have seats reserved on a given flight. |
| ... | ... |

| Id | Position | Req Text | Syntactical token Id | Syntactical token Text |
|---|---|---|---|---|
| 1537 | 0 | the | 1224 | definite article |
| 1537 | 1 | online | 1144 | unclassified noun |
| 1537 | 2 | help | 2228 | noun |
| 1537 | 3 | must | 1130 | modal compulsory |
| 1537 | 4 | exist | 1108 | verb |
| 1537 | 5 | in all | 1151 | adverb |
| 1537 | 6 | defined | 1224 | defined article |
| 1537 | 7 | languages | 1144 | unclassified noun |

| tokenID | Syntactic desc |
|---|---|
| 1224 | definite article |
| 1144 | unclassified noun |
| 2228 | noun |
| 1130 | modal compulsory |
| 1108 | verb |
| 1151 | adverb |
| ... | |
| ... | |

Las tablas, los atributos y las relaciones se han simplificado para mostrar los atributos relevantes en esta investigación.

## 2.8 Algoritmos genéticos

Los Algoritmos genéticos (AG) son una aproximación a la resolución problemas inspirados en los principios de la selección natural y evolución de las especies de Darwin: diversidad genética inicial derivada de fuentes de variabilidad (mutaciones), selección natural de los individuos mejor adaptados, cruce de padres mejor adaptados, mutación, obtención de nuevas generaciones más adaptadas etc. [62][63]. La base es la generación

de una variedad de posibles soluciones a un problema (población). Utilizando un criterio de selección (función de *fitness*), el algoritmo selecciona los mejores individuos a partir de los cuales genera una descendencia que hereda las características de los padres. Si los padres demuestran buen *fitness*, el proceso de reproducción (intercambio de genes de los padres) generará una mejor descendencia. Se trata de un proceso iterativo que, al final puede encontrar una solución óptima al problema.

Los AG se han utilizado en una gran variedad de escenarios: búsqueda de la ubicación óptima de los aerogeneradores en los parques eólicos [64], priorización de requisitos [65], generación de patrones de expresiones regulares [17]. Un ejemplo característico es la resolución del denominado "problema del viajante de comercio" [66] [67].

Esta es la secuencia de actividades y bucle principal de un AG genérico:

A) **Codificación de la solución**. Definir genes, individuos y población.

B) **Definición de una función de *fitness*** para evaluar la idoneidad de cada individuo como posible solución al problema.

C) **Generación aleatoria de una población inicial**, generación 1.

D) **Bucle principal**. Iteraciones hasta encontrar una solución final a todo el problema o hasta alcanzar un límite paramétrico de número de iteraciones.

I - **Evaluación** la población de la generación "n" utilizando la función de *fitness*.

II - **Selección** de los mejores progenitores, aquéllos con mejor *fitness*, para utilizarlos como padres para la siguiente generación.

III - **Cruce de los padres**. Se desarrolla una función específica de cruce con la que se genera la generación "n+1" de población.

IV - **Mutación** de ciertos individuos de la nueva generación. De nuevo con una función específica y con el fin de introducir aleatoriedad y explorar posibles nuevos "genes" potencialmente positivos para la resolución del problema.

V - **Sustitución** de la generación "n" por la generación "n+1". Esta generación "n+1" tendrá un valor de *fitness* global mayor que la precedente, acercándose el AG a la resolución del problema.

E) **HASTA** que se encuentre la solución o se alcance el límite paramétrico establecido de iteraciones.

### A) Codificación de la solución: Definir Genes, Alelos, Individuos y Población

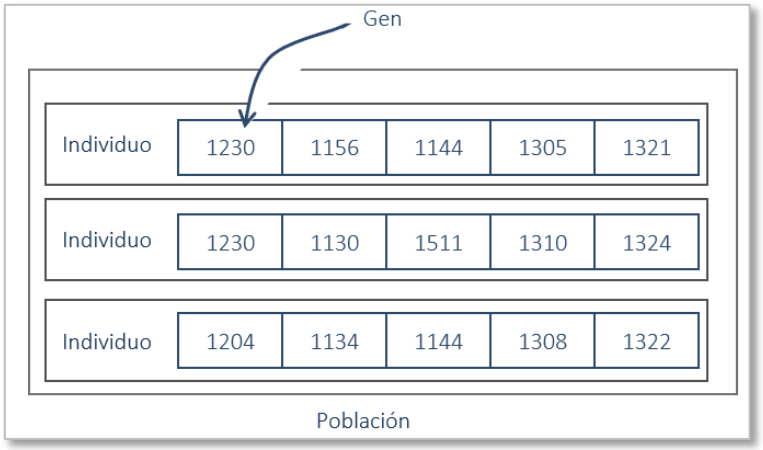
Para poder utilizar la técnica de algoritmos genéticos se requiere modelar las soluciones como un conjunto de parámetros que pueden tomar distintos valores. Estos parámetros constituyen los genes en la terminología del AG. El conjunto de parámetros o genes se denomina cromosoma o individuo [68]. El siguiente nivel de agregación es el individuo, que puede contener uno o más cromosomas.

Así, el Algoritmo genético comienza con la identificación de un conjunto de variables y posibles valores que caracterizan las soluciones de un problema determinado. Estas variables se establecerán como genes, y el conjunto de variables o genes se convierte en el genotipo[1] de un individuo. Un individuo consistirá en una posible solución final del problema.

La Figura 2-6 muestra un ejemplo de Población, Individuos cromosomas y genes. En este ejemplo los individuos tienen un único cromosoma, y los cromosomas tienen 5 genes; los genes toman valores numéricos discretos. Cada valor es un alelo del gen correspondiente. Cada individuo es una posible solución al problema que se desea resolver.

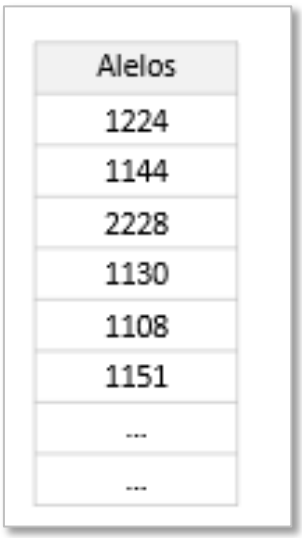
[1] En biología, los genes se agrupan en cromosomas y un individuo puede contener uno o más cromosomas. La mayoría de las bacterias tienen un solo cromosoma. Los eucariotas suelen contener varios cromosomas

FIGURA 2-6 GENES, CROMOSOMAS, INDIVIDUOS Y POBLACIÓN EN AG




Los alelos son todas las posibles variaciones de valores del gen como se muestra en la Figura 2-7. El AG utiliza las colecciones de alelos como fuente para rellenar los genes de los individuos

FIGURA 2-7 COLECCIÓN DE ALELOS




En esta investigación se utiliza el identificador numérico de la colección de etiquetas sintácticas, atributo TokenID, generada por el indexador KM como base de datos de alelos (véase Figura 2-5).

**B) Definición de Función de fitness**

La función de fitness evalúa la idoneidad de los individuos para resolver el problema y les asigna un valor o puntuación de aptitud. Este valor se utilizará para seleccionar los mejores individuos de la población para generar la siguiente generación. La función de *fitness* se utiliza como base del proceso de selección natural. La probabilidad de que un individuo sea seleccionado para la para la reproducción se basa en su puntuación de fitness.

**C) Generación aleatoria de una población inicial**

La población inicial para comenzar las iteraciones del AG se genera aleatoriamente utilizando los alelos disponibles para los distintos genes. El número de individuos de la población inicial es de nuevo un parámetro del algoritmo a ajustar en función del problema a resolver. Poblaciones con muchos individuos a evaluar contendrán mayor variabilidad y podrán explorar un espacio mayor de posibles soluciones, pero requerirán tiempos de cálculo mayores; por otro lado, la dispersión de la población inicial puede ser también un problema para conseguir que las generaciones sucesivas converjan hacia soluciones válidas.

**D.I) Evaluación**

El primer paso del bucle principal de un AG es la evaluación de todos los individuos de la población utilizando la función de *fitness* definida, asignando este valor a cada individuo.

**D.II) Selección**

El proceso de selección se encarga de elegir qué individuos se reproducirán, es decir, serán utilizados para crear la siguiente generación. Utilizando los principios de la selección natural, el proceso dará más oportunidades a los individuos "mejor adaptados", en el caso de AG los individuos con un mejor valor de *fitness*, para ser seleccionados.

El subconjunto de individuos que se utilizará para generar descendencia lo denominaremos pool de apareamiento. Existen distintos métodos descritos para obtener el pool de apareamiento [69]. El método de selección utilizado y el tamaño del subconjunto de individuos (el pool de apareamiento), afecta al comportamiento del AG y deben considerarse como parte de los ajustes del experimento.

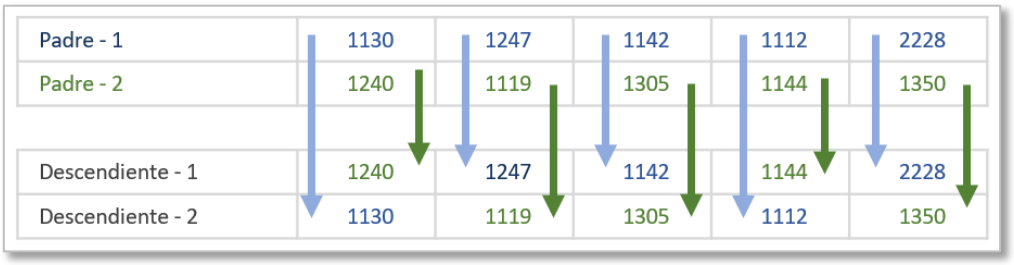
En este experimento, se ha utilizado el método llamado selección por torneos [70]. Se elige un grupo de individuos al azar de la población. Este grupo participa en un torneo en el que se selecciona el mejor individuo con el mayor valor de *fitness* y se incluye en el pool de apareamiento. Este torneo se repite tantas veces como miembros tenga el pool de apareamiento, hasta completarlo. En las repeticiones de los torneos, un individuo que haya sido seleccionado en torneos anteriores puede volver a ser seleccionado para competir en el torneo actual. No se permite sin embargo la selección del mismo individuo más de una vez para formar parte del pool de apareamiento. El número de individuos que participen en el torneo determinará la variedad de individuos en pool de apareamiento (varianza de la aptitud de la población). Dependiendo de la estrategia evolutiva, a veces es más eficiente seleccionar sólo los mejores individuos o, si se pueden encontrar múltiples soluciones posibles y distintas (picos evolutivos), es mejor introducir individuos más diversos en el pool con diferentes puntuaciones de fitness.

### D.III) Cruce de padres

El cruce consiste en el intercambio aleatorio de genes entre los padres para generar nuevos individuos (hijos) que constituirán una nueva generación. La idea principal del cruce es que si se toman dos individuos bien adaptados (en términos AG esto significa con un buen valor de *fitness*) y se produce una descendencia que comparte genes de ambos, esta descendencia tendrá un mejor valor de *fitness*, será por tanto una mejor solución al problema. Esto ocurrirá cuando la descendencia herede genes que promueven una mejor solución al problema de ambos progenitores. Si, por el contrario, algún descendiente hereda genes con valores que no aproximan a la solución del problema, este descendiente tendrá un menor valor de fitness y no se considera para la nueva generación. Existen múltiples algoritmos de cruce basados en el número de puntos de cruce [71]. En este

experimento, se ha utilizado el cruce uniforme, que consiste en repartir aleatoriamente cada gen de los padres entre los genes de los descendientes como se muestra en la Figura 2-8. El cruce uniforme es el que más CPU consume, por lo que tiene un rendimiento significativamente menor.

FIGURA 2-8 INTERCAMBIO DE GENES EN EL PROCESO DE CRUCE DE PADRES




### D.IV) Mutación

La mutación consiste en el cambio aleatorio de algunos genes en algunos individuos de la nueva generación. El nivel de mutación, o probabilidad de introducir una mutación, es un parámetro del AG global.

Cada una de estas mutaciones varía la puntuación de fitness del individuo y se utiliza para introducir variabilidad en la población y aumentar aleatoriamente el espacio de búsqueda de posibles soluciones.

Un nivel de mutación elevado aumenta la variabilidad de la población, pero a cambio tiende a disminuir el fitness global de la población al sustituir el proceso de selección basado en la idoneidad por un principio aleatorio. Los algoritmos de mutación y los de cruce deben utilizarse de forma equilibrada. Si el algoritmo de mutación es el único operador desaparece el concepto de selección y convergencia hacia una solución basada en la evaluación del valor de *fitness* o idoneidad de la solución y el AG global estará explorando el espacio total de soluciones fundamentalmente al azar. Por otro lado, si no se utilizan mutaciones en absoluto, el AG global convergerá sólo en una dirección, y encontrará soluciones o picos "locales" sin la capacidad de saltar a una nueva posible nueva solución mejor en el espacio.

La Figura 2-9 muestra gráficamente los efectos de la selección evolutiva y los efectos de la mutación. Cuando una población se encuentra en la zona A de la figura, el proceso evolutivo de selección natural conducirá a las siguientes generaciones hacia el pico A', que consiste en una solución local al problema. Sin embargo, este proceso nunca permitirá saltar hacia el espació de soluciones de la zona B de la figura. Para que se pueda explorar este espacio, se requiere una mutación de algún individuo que lo lleve desde la zona A hasta la zona B. Este proceso es aleatorio puesto que desconocemos a priori la existencia de la zona B como zona de posible solución.

FIGURA 2-9 SELECCIÓN .VS. MUTACIÓN. EXPLORACIÓN DEL ESPACIO DE SOLUCIONES


El proceso de mutación permite al AG saltar de las soluciones locales (picos locales) en el espacio de soluciones. El proceso de selección conduce a las poblaciones hacia los picos locales.

### D.V) Sustitución

Una vez creada la nueva descendencia, se genera a partir de ella la generación "n+1" sustituyendo en la población parental individuos de la generación "n" por los nuevos individuos. De nuevo existen distintas estrategias de sustitución en función del objetivo del AG [72]:

- **Sustituir todos:** Todos los miembros de la generación parental son reemplazados por la nueva población. Es el mecanismo más sencillo, pero pueden perderse individuos con un alto valor de *fitness* de la generación parental que pueden ser útiles en sucesivos bucles del AG.
- **Sustitución estable:** Esta técnica elimina los n miembros de la generación parental con el peor valor de *fitness* y se sustituyen por los n individuos de la

población obtenida mediante cruce con mejor valor de *fitness*. Esto garantiza que los mejores individuos se mantienen y pueden ser seleccionados de nuevo en las siguientes generaciones. Provoca una convergencia más rápida pero también se pierde en diversidad genética.

- **Sustitución estable sin duplicados:** Similar a la sustitución estable pero no se permiten clones (individuos con el mismo genotipo).

En nuestra metodología, hemos elegido el modelo de sustitución estable: la población tiene un tamaño fijo; a medida que se forman nuevas generaciones, los individuos con peor *fitness* se eliminan, proporcionando espacio para nuevos descendientes. Permitimos los duplicados, ya que el método de Sustitución estable sin duplicados añade una sobrecarga de tiempo sin una mejora significativa de los resultados.

### E) Terminación

La secuencia anterior de actividades, desde la actividad D.I) selección hasta la D.V) sustitución, se repite para producir individuos en cada nueva generación que sean mejores que la anterior. Las condiciones de terminación de un AG determinan el fin de la producción de nuevas generaciones. La terminación puede basarse en [73]:

- Encontrar una solución completa y válida al problema.
- Alcanzar un número máximo de generaciones definidas paramétricamente.
- Encontrar un pico en el que después de un número determinado de generaciones, no aparece una mejora significativa de valor de *fitness* global de la población. De nuevo el número de generaciones sin mejora y la mejora mínima requerida del valor de fitness, son parámetros del AG.

## 2.9 Estrategias de conquista (*Separate-and-Conquer strategy*)

Esta es una estrategia introducida por J. Furnkranz [21]. Se aplica cuando se requieren varias soluciones para distintos componentes de un problema. Hay un bucle general que busca una solución que explique una parte de las instancias o resuelva parcialmente el problema. Cuando encuentra una solución parcial de este tipo, la almacena y marca las

instancias resueltas, esto es, las conquista. Las instancias resueltas se retiran y se continua la búsqueda con el bucle general.

Nuestro enfoque de *separate-and-conquer* es similar a la aproximación utilizada por A. Bartoli en su artículo para la búsqueda de patrones en expresiones regulares [17]. En este experimento combinan algoritmos genéticos y la y la estrategia de separar y conquistar para encontrar patrones para la extracción de información en expresiones regulares. En nuestra investigación buscaremos soluciones parciales con un rendimiento adecuado utilizando algoritmos genéticos; estas soluciones parciales son patrones que reconocen un subconjunto de requisitos; los patrones se almacenan y los requisitos reconocidos son marcados como tal y separados del conjunto de población de requisitos, es decir conquistados. Una vez hecho esto, se inicia un nuevo bucle de búsqueda con la población restante. El algoritmo finaliza si se han reconocido todos los requisitos o si se ha alcanzado un valor paramétrico de número de iteraciones.

## 2.10 Trabajos relacionados

Dentro del grupo de investigación *Knowledge Reuse Group* se han realizado estudios y aproximaciones previas al problema de la obtención de patrones para ser usados en la SE *suite*.

Así A. Fraga y V. Moreno, directora y tutor de esta tesis, han realizado intentos previos con otras aproximaciones [19] [20]. En los dos trabajos se utilizaban las herramientas del grupo de investigación para la tokenización y procesamiento de lenguaje natural, así como el mismo modelo de almacenamiento de información RSHP [59]. En ambos casos, utilizando distintos corpus de textos se usó la misma aproximación: análisis de frecuencia de parejas de tokens adyacentes en los textos. Se ordenaban las parejas de tokens adyacentes por su frecuencia de aparición y los más frecuentes se consideraban patrones de primera generación. Se procedía a su sustitución por un token nuevo representando un subpatrón y se repetía el proceso. Los resultados dependían fundamentalmente de la parametrización de la frecuencia de aparición de tokens adyacentes y de la riqueza del vocabulario considerado y su agrupación semántica previa.

Estos trabajos no contemplan los conceptos de tokens opcionales ni el token comodín como se ha introducido en esta investigación, conforme se explica en los apartados 3.2.4 y 3.2.5 de esta tesis. Estas dos nuevas características han hecho que la presente investigación sea capaz de generar mejores patrones para la autoría de requisitos.

En la tesis de E. Parra [4] se realizó un trabajo de búsqueda automática de patrones utilizando una metodología similar a los trabajos previos. De nuevo se utilizaron las herramientas de la SE *suite*, en concreto la herramienta del indexador de KM para *tokenizar* los requisitos y asignar categorías sintáctico-semánticas a los tokens. El método de búsqueda de patrones fue equivalente: conteo de frecuencia de pares consecutivos de tokens, sustitución de las parejas más frecuentes por un token representando un primer patrón binario o subpatrón; sustitución de las parejas por el subpatrón y nuevo análisis de frecuencia de tokens consecutivos. La condición de parada de estas iteraciones era el valor mínimo de frecuencia de tokens adyacentes a encontrar; si no aparecía al menos una pareja con una frecuencia superior al valor mínimo, el bucle finalizaba con los patrones encontrados.

Esta tesis se realizó utilizando un conjunto de requisitos proporcionados por INCOSE, el Consejo Internacional de Ingeniería de Sistemas. Utilizando un conjunto de 545 requisitos previamente calificados por los expertos como buenos requisitos, se obtuvieron 442 patrones de texto. La relación entre los patrones de texto encontrados y el número de requisitos fue de 0,81.

En esta investigación se ha utilizado el miso conjunto de requisitos proporcionado por INCOSE que se utilizaron en la tesis de E. Parra. Esto ha permitido comparar los resultados contra el trabajo anterior.

# 3 SOLUCIÓN. METODOLOGÍA

En este capítulo describimos el detalle de la solución global que se ha implementado en esta investigación para obtener patrones automáticos para autoría de requisitos.

Hemos introducido un primer apartado de definiciones y vocabulario que usaremos en la solución y que resulta necesario para entender la metodología y la solución técnica implementada.

El apartado de procesos y diseño de algoritmos detalla los procesos generales y los algoritmos genéticos y de conquista aplicados a la obtención de requisitos.

Finalmente se describe la herramienta Twiga, producto de esta investigación, que genera conjuntos de patrones RPS automáticamente a partir de conjuntos de requisitos. Se detallan funcionalidades y cómo se implementó mediante el uso de sucesivas iteraciones de diseño, desarrollo, pruebas y experimentación.

## 3.1 Introducción

En este apartado se describe la metodología general de desarrollo de la investigación. Utilizamos el término metodología en un sentido amplio, siguiendo la definición J. Estefan [74]. Así consideramos parte la metodología:

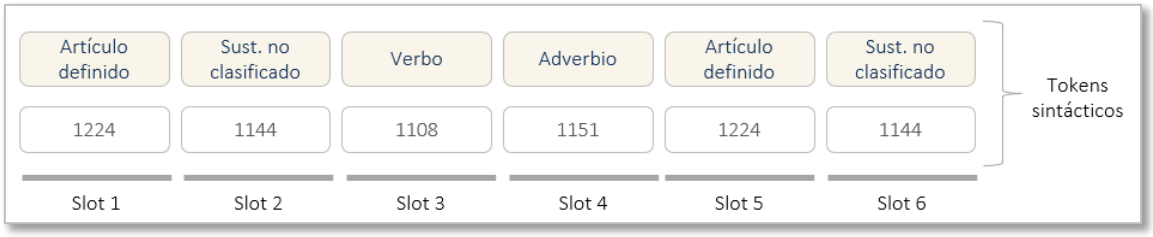
- **Procesos**: Agrupación ordenada de tareas para conseguir un determinado objetivo. Los procesos tienen unas entradas (documentos, insumos, etc.) y generan una salida en la que dichas entradas han sido procesadas por las tareas para generar las salidas.
- **Métodos**: Los métodos son técnicas para realizar las tareas. Definen cómo realizar las tareas identificadas en los procesos (se pueden utilizar términos análogos como procedimiento, función, práctica, técnica).
- **Herramientas:** Instrumentos que facilitan o automatizan la ejecución de las tareas conforme se han definido en sus métodos de ejecución.

## 3.2 Definiciones básicas

### *3.2.1 Patrones, tokens y slots*

Un patrón de texto es una plantilla, una representación formal de un modelo que puede utilizarse para representar varias frases del mundo real. Un slot, dentro de un patrón, es un contenedor de un concepto que puede tratarse de un concepto sintáctico o semántico. Estos conceptos o contenidos de los slots se denominan tokens. La Figura 3-1 resume cómo se representan los patrones, los slots y los tokens. En la herramienta *Knowledge Manager* (KM) de la SE *suite* los tokens se codifican con un número entero como se muestra en la Tabla 3-1. Los tokens pueden ser de carácter sintáctico, definen la categoría sintáctica del token, o semánticos, representan un concepto. La definición de tokens semánticos y el vocabulario asociado a cada token semántico, es realizado por expertos en función del dominio en análisis.

FIGURA 3-1 REPRESENTACIÓN DE UN PATRÓN




Un patrón lo componen grupos de slots que contienen tokens. Los tokens son números enteros que representan términos sintácticos.

TABLA 3-1 TABLA EJEMPLO DE TOKENS DE LA HERRAMIENTA KM

| TokenID | Término sintáctico | TokenID | Término sintáctico |
|---|---|---|---|
| 1134 | Acrónimos (ok) | 2228 | Sustantivo |
| 1103 | Adjetivo | 1123 | Número |
| 1151 | Adverbio | 1117 | Signo de exclamación inicial |
| 1106 | Frase adverbial | 1115 | Signo de interrogación inicial |
| 1130 | Artículo | 1170 | Signo de interrogación final |
| 1210 | Verbo aspectual | 1153 | Adverbio de lugar |
| 1224 | Artículo definido | 1109 | Adverbio de tiempo |
| 1171 | Frase modal | 1241 | Adjetivo no clasificado |
| 1130 | Año | 1144 | Sustantivo no clasificado |
| 1128 | Mes | 1108 | Verbo |

Los patrones de texto permiten el reconocimiento de requisitos similares a lo largo de un conjunto de requisitos de un mismo dominio. La Figura 3-2 muestra cómo dos requisitos similares son reconocidos por un mismo un patrón.

FIGURA 3-2 RECONOCIMIENTO DE REQUISITOS POR UN PATRÓN


### *3.2.2 Requisitos y Conjunto de Requisitos (RS)*

Para nuestro propósito, los requisitos son sólo frases de texto. Son tratados con la herramienta indexador del Knowledge Manager (KM) mediante los procesos descritos de procesamiento de lenguaje natural (segmentación de oraciones, *tokenización*, *lematización* o análisis léxico, análisis sintáctico y etiquetado gramatical, análisis semántico*)* y son transformados en conjunto de tokens sintáctico-semánticos almacenándose en tablas estructuradas que contienen como atributos los propios tokens numéricos y las palabras que constituyen el requisito. La Figura 3-3 muestra la conversión de un requisito en tokens.

FIGURA 3-3 CONVERSIÓN DE UN REQUISITO EN TOKENS USANDO LA HERRAMIENTA KM


Los requisitos y los patrones tienen pues la misma estructura y utilizan los mismos tokens sintáctico-semánticos (a excepción del token comodín exclusivo de patrones, así como la cualificación de opcionalidad de un token, que es también exclusivo de los patrones).

Un conjunto de requisitos (*Requirements Set*, RS) son un grupo de frases que definen las necesidades de los usuarios en un dominio concreto. El término dominio se utiliza como un área de conocimiento con una terminología común [75]. Por ejemplo, se puede definir un RS para especificar todas las necesidades eléctricas de la cabina de un avión. Un RS

comparte un vocabulario de dominio propio del dominio al que pertenece la especificación y las reglas sintácticas utilizadas en la escritura de especificaciones.

En los experimentos realizados se han utilizado distintos conjuntos de requisitos, guardando cada uno de ellos una cierta homogeneidad. Se tratan del mismo tipo de requisitos (ej.: funcionales, de seguridad, etc.) y aplicados al mismo dominio (ej.: ingeniería aeronáutica, desarrollo de software, soporte a clientes…).

### *3.2.3 Reconocimiento de requisitos por un patrón (Pattern matching)*

El reconocimiento de un requisito por un patrón se efectúa cuando todos los tokens del requisito coinciden con los tokens del patrón. En este caso, decimos que el requisito "*matchea*" con el patrón o que el requisito es reconocido por el patrón.

En la Figura 3-2 se muestra, en una visión de texto, cómo un mismo patrón reconoce a dos requisitos similares. En la Figura 3-4 se muestra un ejemplo del proceso de verificación de la coincidencia de varios requisitos con un patrón específico. Para considerar que un requisito es reconocido por un patrón todos los tokens del requisito deben coincidir con los tokens del patrón y en el mismo orden. En la figura, los tokens sombreados en verde son coincidentes, los tokens sombreados en naranja son no coincidentes.

FIGURA 3-4 PROCESO DE VERIFICACIÓN DE COINCIDENCIA DE REQUISITOS CON UN PATRÓN

| | | | | | | | | Requisito *matchea con* el patrón |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1142 | 1112 | 1130 | | | SÍ |
| Requisito 2 | 1130 | 1144 | 1142 | 1350 | | | | NO |
| Patrón | 1130 | 1247 | 1142 | 1112 | 1130 | 1350 | | |

Hay dos escenarios de uso del reconocimiento de requisitos por patrones. En el primer escenario, el usado por la herramienta RQA que verifica la calidad de un requisito, la herramienta busca si el requisito que se está analizando *matchea* con algún patrón del

RPS que tiene configurado. Si se encuentra alguna coincidencia se cualifica el requisito como de buena calidad ya que cumple con los criterios sintácticos y el vocabulario establecidos por el patrón que ha reconocido al requisito.

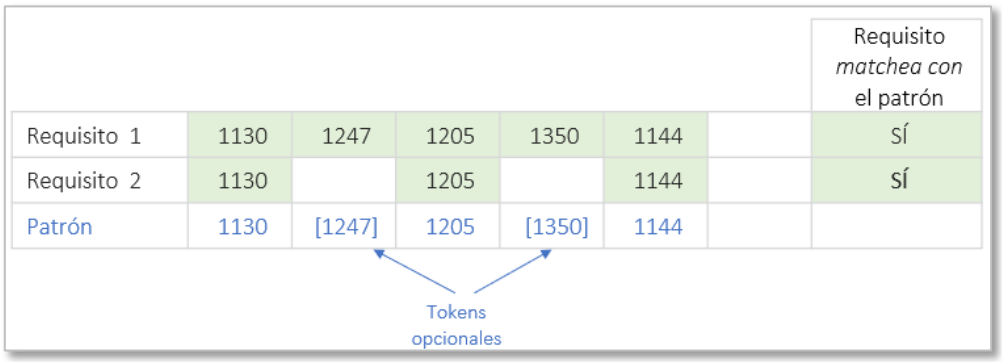
En el segundo escenario, el de autoría que implementa la herramienta RAT, la herramienta analiza conforme se va escribiendo un requisito la parte de requisito ya escrita, busca los patrones que coinciden con el trozo de frase y ofrece tokens válidos para finalizar el proceso de autoría del requisito.

### *3.2.4 Opcionalidad de un token de un patrón*

En la definición de patrones se han implementado dos mecanismos que proporcionan flexibilidad a los patrones para poder reconocer un mayor número de requisitos: la opcionalidad de tokens y el token comodín explicado en el siguiente apartado.

La opcionalidad es un atributo propio de los tokens de los patrones. Si un token de un patrón es opcional significa que los requisitos que no tengan ese token en la posición en que aparece en el patrón pero que tienen el resto de los tokens coincidentes se consideran como reconocidos por el patrón. La Figura 3-5 muestra cómo funciona el mecanismo de opcionalidad: el Requisito 2 se considera reconocido por el patrón, aunque no tenga el token 1247 en el slot 2 y el token 1350 en el slot 4. Los tokens opcionales son representados por Twiga utilizando los corchetes "[]" como muestra en la figura.

FIGURA 3-5 FUNCIONAMIENTO DE TOKENS OPCIONALES EN PATRONES



### *3.2.5 Token Comodín*

El token comodín, exclusivo de patrones, funciona haciendo que cualquier token de un requisito en la posición correspondiente sea marcado como reconocido. Esta característica se extiende para los siguientes tokens del requisito hasta un máximo marcado por el parámetro afinidad explicado en el siguiente apartado. Si en el recorrido de los tokens del requisito aparece un token igual al siguiente token del patrón, el token comodín deja de surtir efecto y el proceso de reconocimiento continua con su funcionamiento normal. La Figura 3-6 muestra cómo funciona un token comodín. En esta figura el Token comodín, representado por Twiga por un asterisco, permite que se consideren como reconocidos los tokens 1205 y 1350 del Requisito 1 así como el token 1350 del Requisito 2. El patrón es por tanto capaz de reconocer ambos requisitos que son similares, pero no iguales.

FIGURA 3-6 FUNCIONAMIENTO DEL TOKEN COMODÍN

| | | | | | | | Requisito *matchea con* el patrón |
|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1205 | 1350 | 1144 | | SÍ |
| Requisito 2 | 1130 | 1247 | 1350 | | | | SÍ |
| Patrón | 1130 | 1247 | * | | 1144 | | |

### *3.2.6 Parámetros que limitan al token comodín*

*Afinidad*

El token comodín permitiría, eventualmente, la generación de un patrón único con un token comodín como se muestra en la Figura 3-7. Este patrón universal reconocería cualquier requisito en cualquier dominio:

FIGURA 3-7 PATRÓN UNIVERSAL REPRESENTADO POR UN COMODÍN

| | |
|---|---|
| Patrón universal | * |

El algoritmo genético encontrará este patrón universal como el mejor posible ya que reconoce a todos los requisitos del conjunto de requisitos RS en estudio.

Para evitar este problema, hemos definido un parámetro en Twiga que limita el número de slots que un token comodín puede reconocer. Se trata del parámetro de *afinidad*. Este parámetro toma un valor entero que determina el número máximo de slots a reconocer por el token comodín. Así, un valor 3 de afinidad significa que los tokens comodines solo pueden reconocer 3 slots de los requisitos. La Figura 3-8 muestra cómo los diferentes valores de los parámetros de afinidad afectan al proceso de reconocimiento de requisitos por un mismo patrón con el mismo requisito.

FIGURA 3-8 IMPACTO DEL PARÁMETRO AFINIDAD EN EL RECONOCIMIENTO DE REQUISITOS

| | | | | | | | | Requisito reconocido por patrón |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1205 | 1350 | 1170 | 1144 | | SÍ |
| Patrón | 1130 | 1247 | | * | | 1144 | | |

*Valor del parámetro afinidad = 3. El Requisito es reconocido por el patrón.*

| | | | | | | | | Requisito reconocido por patrón |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1205 | 1350 | 1170 | 1144 | | NO |
| Patrón | 1130 | 1247 | | * | 1144 | | | |

*Valor del parámetro afinidad = 2. El Requisito NO es reconocido por el patrón.*

*Sensibilidad a tokens semánticos*

Este parámetro de Twiga permite variar el comportamiento del token comodín respecto a los tokens semánticos. Los tokens semánticos son aquéllos que contienen términos que han identificados del dominio a que pertenece el RS. Este parámetro, de carácter booleano, permite o prohíbe que un token comodín de un patrón reconozca tokens semánticos de requisitos. Así si el parámetro está con valor true entonces el token comodín no reconocerá un token semántico si se encuentra durante el proceso de reconocimiento de requisitos. La Figura 3-9 muestra cómo el valor de este parámetro

afecta al proceso de reconocimiento de requisitos por un mismo patrón con el mismo requisito que contiene el token 1170 que es de carácter semántico.

FIGURA 3-9 IMPACTO DEL PARÁMETRO SENSIBILIDAD SEMÁNTICA AL RECONOCIMIENTO DE REQUISITOS

| | | | | | | | | Requisito reconocido por patrón |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1205 | 1350 | 1170 | 1144 | | SÍ |
| Patrón | 1130 | 1247 | | * | | 1144 | | |

*Valor del parámetro afinidad = 3. Sensibilidad semántica = False. El token 1170 es un token semántico. pero el Requisito es reconocido por el patrón.*

| | | | | | | | | Requisito reconocido por patrón |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1205 | 1350 | 1170 | 1144 | | NO |
| Patrón | 1130 | 1247 | | * | 1144 | | | |

*Valor del parámetro afinidad = 3. Sensibilidad semántica = True. El token 1170 es un token semántico. El Requisito NO es reconocido por el patrón.*

Los parámetros afinidad y sensibilidad a tokens semánticos son una potente herramienta para gestionar el comportamiento de Twiga. Según el objetivo que se persiga en la especificidad dentro de la búsqueda de patrones, se puede ampliar o reducir la distancia de palabras específicas deseadas centrándose sólo en las palabras importantes (tokens semánticos). Esta capacidad responde a la hipótesis 3 planteada en el apartado 1.4 de esta tesis en el sentido de que se puede ajustar el comportamiento de la herramienta Twiga para encontrar conjuntos de patrones RPS con mayor o menor grado de especificidad hacia el RS estudiado en función del objetivo del RPS a generar.

### *3.2.7 Conjunto de Patrones (RPS)*

Un conjunto de patrones de requisitos (RPS) es un grupo de patrones que reconocen todos o una parte mayoritaria (definida paramétricamente) de los requisitos de un conjunto determinado de requisitos (RS). Los patrones incorporan las distintas estructuras sintácticas válidas que aparecen en el RS, así como el vocabulario utilizado.

La SE *suite* utiliza conjuntos de patrones (RPS) distintos para distintos dominios en sus herramientas de *Authoring* (RAT) y verificación de la calidad de requisitos (RQA).

Un RPS se construye utilizando una muestra significativa de requisitos escritos previamente, seleccionados y calificados como buenos por expertos. A partir de este conjunto inicial de requisitos, expertos en la definición de patrones para la SE *suite* identifican y generan manualmente un RPS que será posteriormente utilizado para redactar nuevos requisitos del dominio o para evaluar la calidad de nuevos requisitos con la SE *suite*.

Este proceso requiere un alto grado de conocimiento y experiencia, tanto en el dominio al que pertenece el RS como en las estructuras de patrones y de procesado de lenguaje natural. Es un proceso costoso en términos de tiempo y debe actualizarse periódicamente a medida que se incorporan nuevos requisitos al dominio con nuevo vocabulario válido, etc.

La calidad de un RPS se evalúa por su simplicidad, medida por el número de patrones que tiene el RPS, y por su exhaustividad, medida por el número total de requisitos que reconoce. Un buen conjunto de patrones RPS debe tener el menor número posible de patrones y ser capaz de reconocer el máximo número de requisitos.

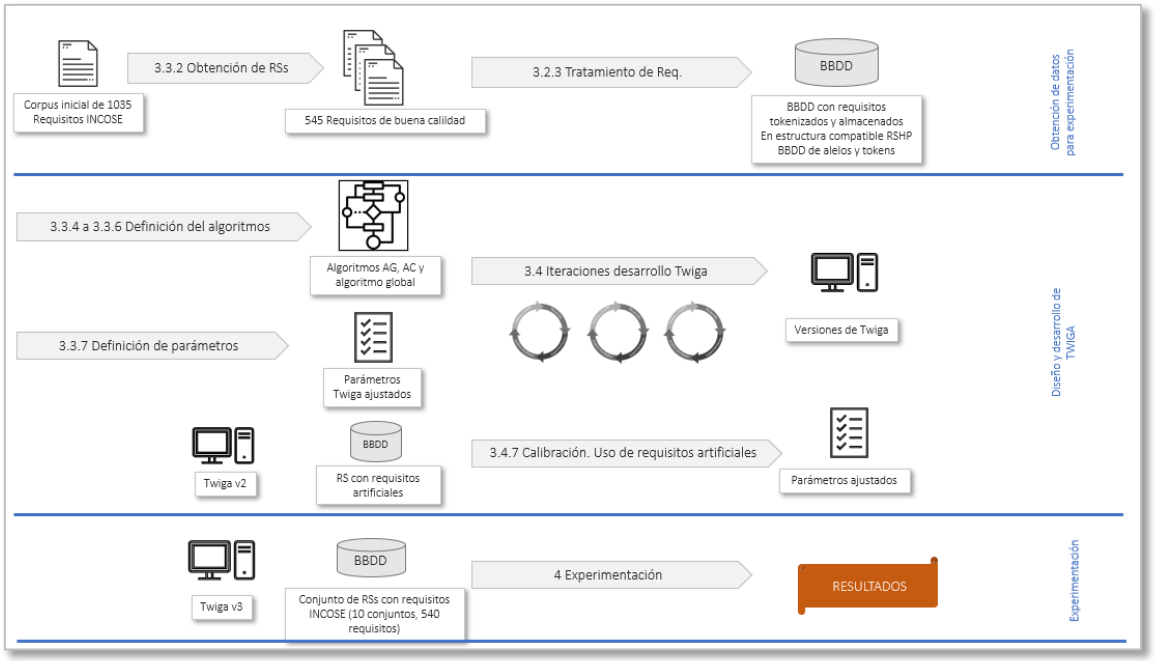
Esta investigación aborda el problema de generar RPSs válidos apoyando o sustituyendo la labor de expertos con la consiguiente reducción de costes derivados. El objetivo de la herramienta Twiga, basada en la implementación de algoritmos genéticos y de la estrategia de separación y conquista, es proporcionar RPSs con el mínimo número de patrones posibles y el máximo número de requisitos coincidentes para un RS concreto.

## 3.3 Procesos y diseño de algoritmos

### *3.3.1 Procesos generales*

La Figura 3-10 describe la metodología a partir de los procesos que se han seguido, incluyendo las 3 iteraciones de mejora de la herramienta Twiga.

FIGURA 3-10 METODOLOGÍA. PROCESOS GENERALES




### *3.3.2 Obtención, clasificación y selección de los Requisitos*

*Corpus de datos*

Esta investigación ha partido de un corpus de 1.035 requisitos reales de distintos dominios facilitado por el grupo de trabajo de requisitos de INCOSE (*International Council on Systems Engineering*) en 2016[2] al grupo de investigación de la Universidad Carlos III de Madrid *Knowledge Reuse Group* (KR).

*Clasificación de los Requisitos. El problema de la clasificación*

Estos requisitos habían sido previamente clasificados por expertos como de buen o mala calidad.

Esta clasificación binaria puede considerarse como una limitación ya que pueden existir requisitos parcialmente bien definidos o se podría plantear n sistema de clasificación cuantitativo que permitiera evaluar cada requisitos.

[2] Los requisitos proporcionados por INCOSE están protegidos por una cláusula de confidencialidad, por lo que en las tablas o figuras de ejemplo que se muestran en este documento, utilizamos requisitos textuales alternativos.

No se ha abordado esta posibilidad porque el objetivo principal de esta tesis es la obtención automática de patrones y la comparación de la nueva metodología con métodos previos. El interés era testear la metodología y poderlo comparar utilizando el mismo juego de datos de investigaciones anteriores.

Se menciona aquí esta restricción por cuanto podrían plantearse escenarios futuros de investigación donde exista una mayor riqueza en la evaluación de requisitos y esto permita también adaptar los algoritmos genéticos para obtener mejores patrones.

*Selección de requisitos. Uso de requisitos de calidad y descarte de requisitos de baja calidad*

Del corpus de requisitos solamente se seleccionaron los requisitos clasificados como de calidad por los expertos, un total de 545 requisitos.

En una fase inicial de la definición de la metodología, se pensó utilizar también los requisitos de baja calidad, incorporarlos a los conjuntos de requisitos (RS) y considerarlos en la función de evaluación del algoritmo genético (véase función de fitness en apartado 3.3.4); se realizaron primeros experimentos en la iteración 1 de Twiga. Los resultados obtenidos no fueron buenos y la complejidad que adquiría la función de evaluación nos hizo descartar esta opción y posponerla para posibles investigaciones futuras (véase apartado 5.3).

### *3.3.3 Tratamiento de los requisitos*

Los requisitos habían sido tratados en experimentos anteriores del grupo y en la tesis de E. Parra [4], convirtiéndolos en conjuntos de tokens sintácticos – semánticos almacenados en una estructura de BBDD compatible con el modelo RSHP [59] y con el almacenamiento de la SE *suite.*

Para realizar este proceso se utiliza el indexador de la herramienta *Knowledge Manager*, parte de la SE *suite* que realiza un el procesado de lenguaje natural similar al indicado en el apartado 2.4 esta tesis:

- Separación en frases de los requisitos

- Tokenización, identificación de palabras
- Lematización, análisis léxico
- Asignación de tokens sintácticos-semánticos
- Almacenamiento en estructuras de base de datos compatibles con el modelo RSHP

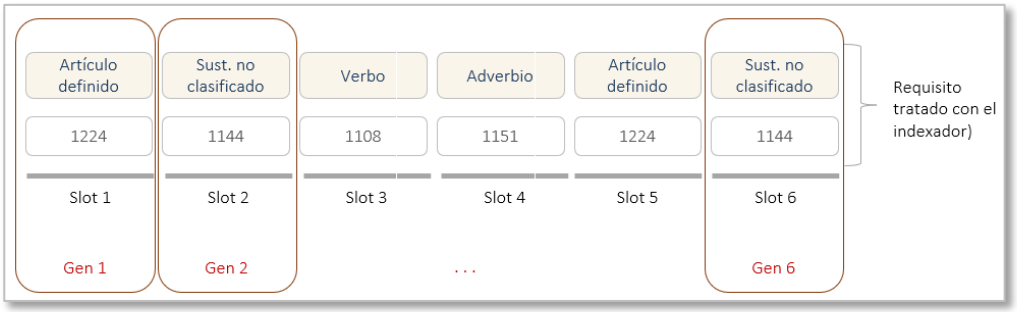
El uso de este procesado del conjunto de requisitos se hizo con la finalidad de utilizar los mismos datos de entrada en nuestros experimentos a los usados por E. Parra y poder así comparar resultados usando los mismos escenarios.

### *3.3.4 Algoritmo genético (AG) para búsqueda de patrones*

*Definiciones del AG*

- **Genes:** En este experimento, los genes son los slots que se rellenan con tokens. El gen es la información mínima que se puede heredar de padres a hijos. Se rellena con tokens que actúan como alelos del gen. Cada gen (slot) puede llenarse con diferentes Tokens (alelos) como se muestra en la Figura 3-11.

FIGURA 3-11 CONCEPTO DE GEN PARA EL AG




- **Individuos**: En la metodología un individuo representa un único requisito o patrón. Un individuo tiene un sólo un cromosoma (conjunto de genes ordenados). Los individuos de pueden tener diferente número de genes. Esta es una diferencia significativa respecto al paradigma de la naturaleza, donde todos los individuos de la misma especie comparten el mismo genotipo, es decir, el mismo número de cromosomas y el mismo número de genes. Los patrones tienen, además, para cada token, un atributo adicional que determina si puede ser opcional. Si el atributo es verdadero, el token puede aparecer o no en los requisitos en el proceso de reconocimiento. También hay un token especial presente sólo en los individuos

de los patrones, es el token comodín, utilizado también en el proceso de emparejamiento de los requisitos.

- **Poblaciones**: Una población es un grupo de individuos. Existen dos tipos de poblaciones: poblaciones de requisitos, también denominadas conjunto de requisitos (RS) y poblaciones de patrones también llamadas conjuntos de patrones (RPS). Como se ha comentado en la definición de individuos, las poblaciones contienen individuos que pueden tener distintos números de genes (tokens) como se muestra en la Figura 3-12.

FIGURA 3-12 POBLACIÓN DE REQUISITOS (RS)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1144 | 1350 | | | |
| Requisito 2 | 1130 | 1247 | 1342 | 1350 | 1144 | 1267 | |
| Requisito 3 | 1130 | 1144 | 1110 | 1248 | 1130 | | |
| Requisito 4 | 1165 | 1144 | 1342 | 1350 | 1144 | 1246 | |
| Requisito 5 | 1165 | 1144 | 1342 | 1350 | 1144 | 1094 | 1237 |

- **Frecuencia de tokens**: En una población de requisitos (RS), se calcula la frecuencia de aparición de cada token conforme a la ecuación 1. Esta frecuencia se utilizará en el AG en el proceso de creación de poblaciones iniciales de patrones, así como en los procesos de mutación cuando se sustituya un token de un individuo por otro. La Tabla 3-2 muestra un ejemplo de biblioteca de tokens con sus frecuencias relativas para el RS con id 41.

$$\text{Frecuencia token} = \frac{\text{Núm. de veces que aparece el token en RS}}{\text{Núm. total de tokens en RS}} \qquad (1)$$

TABLA 3-2 EJEMPLO DE BIBLIOTECA DE TOKENS CON SU FRECUENCIA RELATIVA EN UN RS ESPECÍFICO

| RS ID | Token ID | Apariciones en RS | Frecuencia | Tag sintáctico |
|---|---|---|---|---|
| 41 | 1144 | 3431 | 0,23657 | sustantivo no clasificado |
| 41 | 1119 | 2228 | 0,15362 | sustantivo |
| 41 | 1224 | 1308 | 0,09019 | artículo definido |
| 41 | 1108 | 630 | 0,04344 | verbo |
| 41 | 1110 | 597 | 0,04116 | símbolo |
| 41 | 1230 | 512 | 0,03530 | número |
| 41 | 1229 | 476 | 0,03282 | preposición |
| 41 | 1012 | 372 | 0,02565 | adjetivo |
| 41 | 1213 | 357 | 0,02462 | preposición |

*Procesos del AG*

- **Generación aleatoria de población inicial de patrones**: El primer paso en el AG es la creación de una población aleatoria de patrones. Esta será la primera generación del algoritmo. El número de individuos de la población es un parámetro del algoritmo; los valores más altos permiten una mayor diversidad de genes individuales, pero repercute en el rendimiento. Los tokens (alelos) para rellenar los slots de los individuos de la población de patrones se seleccionan considerando su frecuencia relativa en la RS del experimento. La probabilidad de selección de un token determinado coincide con su frecuencia de aparición en la biblioteca (ver Tabla 3-2). Así, si un token particular tiene una frecuencia de 0,03 en el RS, tendrá una probabilidad de ser elegido de 0,03 al llenar cada slot de cada individuo de la población de patrones. Esta significa que tendremos frecuencias similares de tokens y diversidad tanto en el RS como en la población inicial de patrones RPS.
- **Evaluación, función de *fitness*:** definimos dos métricas para evaluar el rendimiento de un patrón en un conjunto de requisitos particular como se muestra en las ecuaciones (2) y (3).

- ***RM.*** Cuenta el número de requisitos que un patrón reconoce en el RS. El valor de RM se utiliza para ordenar los patrones. Los mejores patrones son los que tienen un valor RM más alto.
- ***SM.*** Si varios patrones tienen el mismo RM, se utiliza la segunda métrica SM, para clasificarlos. El SM cuenta el número total de tokens que reconoce el patrón, considerando todo el RS. Figura 3-13 se muestra un ejemplo de cálculo del *fitness* para un patrón con un conjunto de 2 requisitos.

$$\boldsymbol{RM} = \textit{Recuento de requisitos reconocidos por el patrón} \qquad (2)$$

$$\boldsymbol{SM} = \sum_{req=1}^{req=Núm\ reqs.} \sum_{slot=1}^{slot=Núm\ slots\ del\ Req.} \alpha \qquad (3)$$

$\alpha = 1$ *si token requisito = token patrón*
$\alpha = 0$ *si token es comodín o token requisitos <> token patrón*

FIGURA 3-13 CÁLCULO DEL FITNESS DE UN PATRÓN PARA UNA RS PARTICULAR DE 2 REQUISITOS

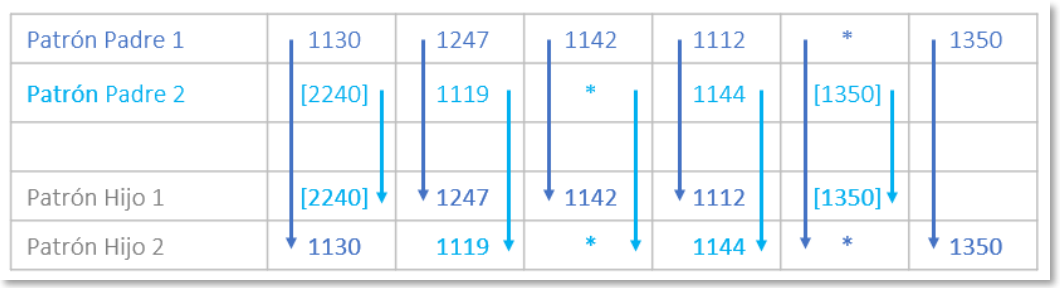
| | | | | | | | RM | SM |
|---|---|---|---|---|---|---|---|---|
| Requisito 1 | 1130 | 1247 | 1142 | 1112 | 1130 | | 1 | 4 |
| Requisito 2 | 1130 | 1144 | 1142 | 1350 | | | 0 | 2 |
| Patrón | 1130 | 1247 | 1142 | 1112 | * | 1350 | | |

- **Selección de individuos para la reproducción (*pool* de reproducción)**: La selección de los patrones padres para el apareamiento se realiza mediante torneo como se explica en apartado 2.8 de esta tesis. El torneo está parametrizado para modular si sólo se seleccionan los mejores progenitores o también se seleccionan individuos con peor Fitness parar aumentar la variabilidad. Esto permite ajustar la variabilidad en los descendientes y gestionar la velocidad de convergencia. Si sólo se eligen los mejores individuos, el algoritmo convergerá rápidamente hacia una posible solución, pero no analizará otras posibles soluciones mejores. El proceso comienza con la selección aleatoria de un padre candidato. Este candidato se

comparará con otros candidatos elegidos al azar. El que posee mejor fitness se selecciona para y pasa a formar parte del pool de reproducción. Este bucle se realiza tantas veces como número de individuos se han definido para el pool de reproducción.

- **Cruce de los padres**. Para obtener los hijos que constituirán la siguiente generación se realiza un intercambio al azar de genes entre los padres. Cada pareja de padres genera una pareja de descendientes como muestra la Figura 3-14.

FIGURA 3-14 EJEMPLO DE PROCESO DE CRUCE




- **Mutación de ciertos individuos**: Una vez generados los hijos se continúa con el siguiente proceso consistente en mutar ciertos individuos hijos para introducir variabilidad. La frecuencia de mutación es un parámetro del AG que permite modular el nivel de variabilidad a introducir. Existen cuatro tipos de mutaciones que se seleccionan al azar.
  - Cambio de token (alelo). En un slot determinado del individuo a mutar se sustituye un token por otro de la biblioteca de alelos disponibles del RS. Para la selección del token de la biblioteca de alelos se tendrá en cuenta de nuevo, al igual que al crear la población inicial, su frecuencia de aparición en la biblioteca. La probabilidad de ser elegido será igual a su frecuencia de aparición.
  - Cambio de la característica de opcionalidad de un token. Se selecciona aleatoriamente un token del individuo a mutar y se le cambia el atributo de opcionalidad.
  - Aumento de tamaño. Se incrementa el número de slots del individuo mutado incorporando nuevos slots al final del patrón. Estos nuevos slots se rellenan con tokens de la biblioteca de alelos disponibles del RS.

- Disminución del tamaño. Se disminuye el número de slots del individuo a mutar quitándolos del final del patrón. El número de slots a reducir es aleatorio.

- **Sustitución de individuos**: Los individuos hijos reemplazan a los miembros de la generación parental con menor fitness. Se crea así la generación n+1.
- **Finalización del AG**: Los procesos del AG de evaluación – selección – cruce – mutación y sustitución se ejecutan en un bucle de manera que se van consiguiendo generaciones hijas cada vez con mejor fitness, esto es, patrones con mayor valor de RM y SM. Las condiciones de finalización del bucle son:
  - Se consigue el reconocimiento de todos los requisitos de la población RS.
  - Se alcanza el número máximo de generaciones especificado por un parámetro específico del AG.
  - Se alcanza un número de generaciones sin incremento significativo del fitness global de las poblaciones hijas. Esta condición se incluye para evitar la ejecución del bucle y consumo de tiempo cuando la población ha alcanzado un pico del que las mutaciones no son capaces de sacar a la población. El incremento de fitness mínimo y el número de generaciones consecutivas sin alcanzar el mínimo de incremento de fitness son dos parámetros del AG.

### *3.3.5 Algoritmo de conquista (AC)*

Cuando se finaliza un AG por alguna de las condiciones de finalización previstas se activa el proceso de conquista de requisitos reconocidos. Denominamos a este proceso Algoritmo de conquista AC y consta de los siguientes procesos:

- **Salvado del mejor patrón encontrado:** De la población final de patrones resultantes de la ejecución del AG se selecciona el patrón que más requisitos reconoce. Este patrón es almacenado en la población de patrones finales a entregar (RPS). Solamente se salvan patrones con $RM > 1$.
- **Conquista de requisitos reconocidos:** Los requisitos reconocidos por el patrón salvado son marcados como conquistados y dejan de ser tenidos en cuenta en sucesivas ejecuciones del AG. De esta manera el número total de requisitos a

reconocer del conjunto inicial de requisitos RS va disminuyendo conforme se ejecutan sucesivas iteraciones del AG y de conquista.

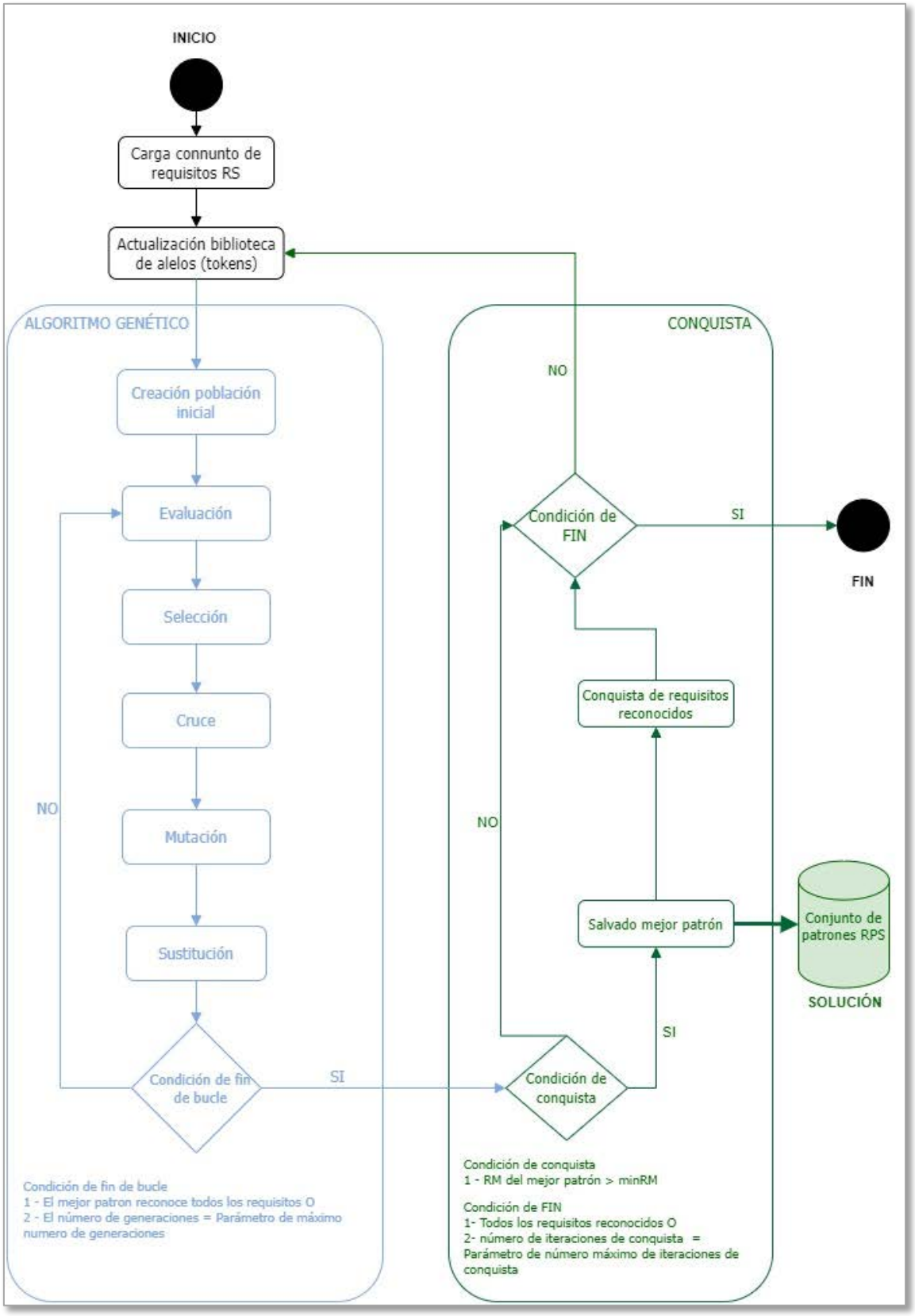
- **Actualización de la biblioteca de alelos**: La tabla de tokens (alelos) se actualiza incluyendo exclusivamente los tokens que quedan en la población de requisitos por conquistar. Los tokens de requisitos ya conquistados y que no aparecen en los requisitos pendientes de reconocer son eliminados.
- **Condición de parada**: Las condiciones de parada de la iteración de conquista son:
  - Todos los requisitos han sido reconocidos.
  - Se alcanza el número máximo de iteraciones especificado por un parámetro concreto.

### *3.3.6 Algoritmo global. Obtención de una solución*

El algoritmo global implementado en la metodología es una doble iteración de los procesos de conquista y de AG como se muestra en la Figura 3-15.

Se considera una solución final el conjunto RPS obtenido por este algoritmo global.

FIGURA 3-15 DIAGRAMA DE FLUJOS GLOBAL



### *3.3.7 Parámetros principales de los algoritmos*

La Tabla 3-3 muestra los principales parámetros definidos para el algoritmo genético AG y el algoritmo de conquista AC. Estos parámetros permiten modificar el comportamiento general de los algoritmos y se han utilizado para optimizar los resultados, así como para adaptar los algoritmos a las características concretas de cada conjunto de requisitos RS.

TABLA 3-3 PRINCIPALES PARÁMETROS DE LOS ALGORITMOS AG Y AC

| Parámetro | Descripción |
|---|---|
| Afinidad | Número de slots de los requisitos que se permiten reconocer con un token comodín de un patrón. |
| Generaciones | Número máximo de generaciones para el AG. |
| Mínimo número de requisitos a reconocer por un patrón (*MinRm*) | Número de requisitos que un patrón encontrado con el AG debe reconocer para ser incluido en el RPS por el algoritmo de conquista AC. Si este mínimo no es alcanzado por ningún patrón generado en una iteración de AG, entonces la iteración de conquista no obtiene ni almacena ningún patrón en RPS. |
| Número máximo de iteraciones de conquista | Número máximo de iteraciones del AC. Es una condición de salida para el AC. |
| Probabilidad de mutación | Probabilidad de introducir mutaciones en individuos de una población de una generación determinada. Una probabilidad = 0,2 provocará mutaciones en el 20% de los miembros de la población en cada generación de AG. |
| Tamaño de la población | Número de patrones (individuos) de la población de patrones. |
| Tamaño de pool de reproducción | Número de miembros de la población de apareamiento. Debe ser un número menor que los miembros de la Población Patrón. |
| Tamaño máximo del patrón | Número máximo de slots de los patrones generados. |
| Torneo | Número de ejecuciones del torneo de selección de población. Un número mayor hará que el proceso de selección sea más efectivo en la elección de padres con mejor *fitness*. A cambio se perderá diversidad genética para la reproducción. |

También se han definido para permitir su uso en distintos escenarios con distintos objetivos, así como para permitir la evolución de la herramienta Twiga en futuros trabajos de investigación (véase apartado 5.3).

*Escenarios en función del objetivo*

La afinidad define lo específico que es un patrón determinado. Si queremos patrones más generalistas, aumentaremos este valor, permitiendo a los autores incluir más conceptos o términos en el proceso de creación de requisitos. Esto puede ser útil en los primeros pasos de la definición de requisitos (requisitos de usuario, requisitos globales). Para requisitos más específicos (por ejemplo, requisitos de software, arquitectura o diseño) los buenos patrones deben ser más específicos. Para conseguirlo, el parámetro de afinidad debe ser menor.

El parámetro *MinRm* determinará la precisión de los patrones encontrados. Valores bajos del parámetro (por ejemplo 2) generarán un conjunto de patrones RPS mayor y muy preciso. Esto puede ser interesante en los pasos finales de la construcción.

*Escenarios en función del rendimiento*

El resto de los parámetros afectan fundamentalmente al rendimiento de los algoritmos, es decir, al tiempo que se tarda en encontrar las posibles soluciones. Valores altos de los parámetros Generaciones, Tamaño de la población, Tamaño de pool de reproducción y Tamaño máximo del patrón disminuirán el rendimiento. Sin embargo, si se disminuye su valor significativamente, es probable que el AG no encuentre soluciones válidas.

La probabilidad de mutación y el tamaño del pool de reproducción también afectan a la velocidad de convergencia, es decir, al número de generaciones que se requieren para obtener mejores resultados y a la capacidad de explorar más soluciones del espectro de soluciones.

### *3.3.8 Métricas para medir la calidad de una solución*

Para medir la calidad de una solución se han definido las métricas indicadas en la Tabla 3-4.

TABLA 3-4 MÉTRICAS DE CALIDAD

| Métrica | Descripción | Fórmula | Buena calidad representada por |
|---|---|---|---|
| ***AvRm*** | Promedio de requisitos reconocidos por patrón | $\boldsymbol{AvRm} = \frac{\sum_{p=1}^{Pat.en\ RPS} NumReqR_p}{Pat\ en\ RPS}$ | Alto Valor |
| ***%Rm*** | Porcentaje de requisitos del RS reconocidos por RPS | $\%\boldsymbol{Rm} = \frac{\sum_{p=1}^{Pat\ \text{en}\ RPS} \text{NumReqR}_p}{Req\ \text{en}\ RS} * 100$ | Alto Valor |
| ***BPRm*** | Número de requisitos reconocidos por el mejor patrón del RPS | $\boldsymbol{BPRm} = MAX(NumReqR_p)$ | Alto valor |

Siendo:

- *NumReqR* es el número de requisitos reconocidos por un patrón.
- *Pat en RPS* es el número de patrones total del RPS.
- *Req en RS* es el número de requisitos en RS.

El significado de cada métrica es el siguiente:

- ***AvRm*** es la métrica principal utilizada para evaluar la calidad de una solución. La peor solución tendrá un valor AvRm igual a 1; eso significa que hemos encontrado un patrón para cada requisito, por lo que no se ha producido ninguna mejora respecto a la situación inicial. Valores altos significan que hemos obtenido patrones generalistas capaces de reconocer a varios requisitos. Esta es una buena situación para la autoría, al menos en las fases iniciales, donde se necesita flexibilidad.
- ***%Rm*** es el porcentaje total de requisitos del RS reconocidos por el RPS encontrado. Intentaremos maximizar esta métrica. Sin embargo, es difícil obtener un valor del 100% ya que en cada RS encontramos cierta cantidad de requisitos "únicos", es decir, requisitos que no se ajustan ningún tipo de plantilla. Si este

valor es demasiado bajo (por debajo del 70%), es posible que el conjunto de requisitos RS incluya un vocabulario demasiado amplio o que haya más de un dominio o tipo de requisitos en la RS.

- *BPRm* se refiere al número de requisitos que reconoce el mejor patrón del RPS; se trata de una buena métrica del rendimiento del AG ya que la función de
- del AG se basa en este valor para el proceso de evaluación y selección de patrones

## 3.4 Herramienta Twiga

### *3.4.1 Introducción*

Para la realización de los experimentos de esta investigación, se ha diseñado y desarrollado una herramienta propia que hemos denominado Twiga. Twiga es una palabra suajili que significa jirafa y se ha elegido por ser una especie muy representativa de la evolución de las especies.

Esta herramienta se ha desarrollado siguiendo los estándares tecnológicos y de almacenamiento de información de la SE *suite* con el fin de que en un futuro pueda ser incorporada bien como un componente o funcionalidad adicional a alguna de las herramientas existentes en la *suite*, o como una herramienta propia.

Twiga es por tanto un producto final resultante de esta investigación, si bien no está desarrollada con funcionalidades e interfaces de usuario que permita ser utilizada por usuarios finales, ajenos a la investigación. La versión final es un prototipo que requiere de conocimientos de diseño y de la programación realizada.

### *3.4.2 Iteraciones de Desarrollo de la herramienta Twiga*

Como se menciona en el apartado 1.7.2, el método de desarrollo de la herramienta Twiga ha sido mediante iteraciones completas de análisis, desarrollo, experimentación e identificación de mejoras para la siguiente iteración.

Cada iteración tenía un objetivo general definido a priori e iba incorporando mejoras en las funciones, parametrización y utilizando distintos juegos de datos (véase Figura 1-1)

*Iteración 1 - Desarrollo de Prototipo*

La primera iteración consistió en el desarrollo de un prototipo general para validar que era posible encontrar patrones en grupos de requisitos con un algoritmo genético (véase Hipótesis 1 en el apartado 1.4).

Se utilizaron juegos de datos no reales de requisitos.

Se generó una versión prototipo de la aplicación Twiga. Los principales hitos asociados a esta iteración fueron:

- Definiciones básicas del algoritmo genético: genes, individuos y poblaciones aplicadas a requisitos y patrones, token comodín y concepto de opcionalidad de un token de un patrón para dotar a los patrones de flexibilidad.
- Preparación de un modelo de datos y una BBDD para requisitos y patrones compatible con el modelo RSHP (véase apartado 3.3.3 de este documento).
- Selección de herramienta para el desarrollo de Twiga, así como el SGBDD. Diseño de la arquitectura técnica de Twiga (clases, componentes, entradas y salidas de datos y resultados, *front-end* de usuario, etc.).
- Diseño e implementación de los procesos del algoritmo genético. Se diseñaron e implementaron primeras versiones de las funciones de creación de población inicial, función de *fitness* o evaluación, selección de mejores patrones, cruce de patrones padres, mutación y sustitución.
- Verificación de funcionamiento del algoritmo, búsqueda y reconocimiento de requisitos por los patrones generados.

En esta primera iteración solamente se implementó el algoritmo genético, no la técnica de conquista y separación.

Los resultados fueron satisfactorios y el prototipo permitió identificar los primeros parámetros que afectaban al comportamiento del algoritmo genético.

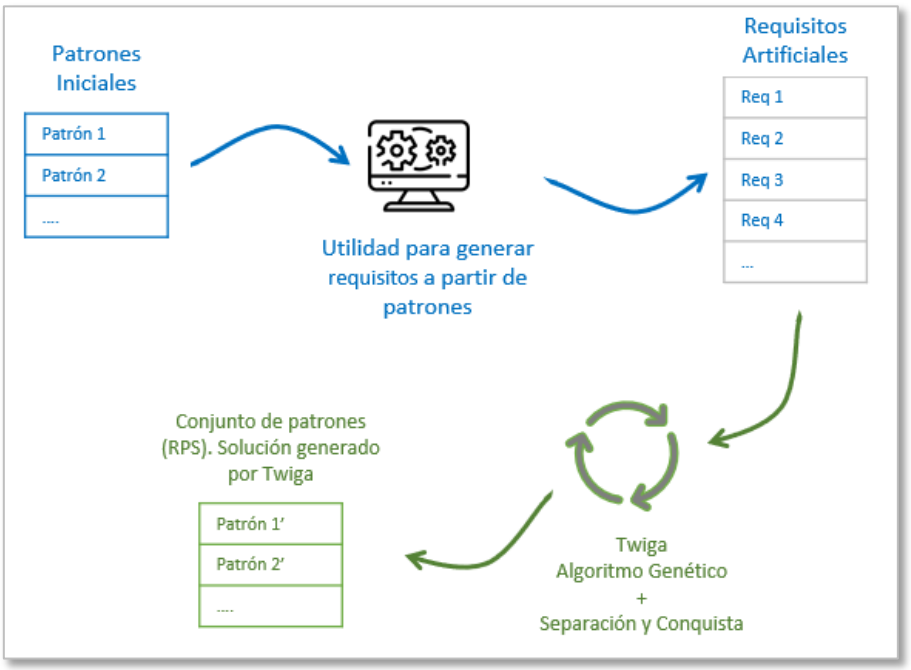
*Iteración 2 –Experimento con Requisitos Artificiales*

El objetivo de la segunda iteración fue la incorporación de la técnica de separación y conquista para resolver el problema de encontrar un conjunto completo de patrones (RPS)

capaz de reconocer a un conjunto de requisitos (RS) (véase Hipótesis 2). También se abordaron en esta iteración los objetivos específicos OBESP1 y OBESP2).

Se utilizaron requisitos que hemos denominado artificiales, generados a partir de patrones iniciales definidos por nosotros tal y como se ilustra en la Figura 3-16. Se trataba de confirmar con juegos de datos controlados la validez de la solución completa (uso de AG y de técnicas de separación y conquista) y qué parámetros y valores de los mismos eran los más adecuados en función de los distintos juegos de datos.

FIGURA 3-16 EXPERIMENTO CON REQUISITOS ARTIFICIALES




Se generó una nueva versión Twiga v2. Los principales hitos asociados a esta iteración fueron:

- Diseño técnico y desarrollo de la técnica de separación y conquista.
- Definición de formato y salida de una solución completa de conjunto de patrones RPS, así como la información paramétrica asociada a la solución.
- Parametrización y calibración: Identificación de nuevos parámetros que afectan al comportamiento de los algoritmos (objetivos OBESP1 y OBESP2) y calibración

principal de los mismos utilizando juegos controlados de datos (requisitos artificiales).

Para validar las distintas calibraciones y parametrizaciones de los algoritmos se comparaban el conjunto de patrones resultado obtenidos (RPS) con Twiga frente los patrones origen utilizados para generar los requisitos artificiales.

*Iteración 3 –Twiga v3 – Experimentos con Requisitos INCOSE*

El objetivo de la última iteración fue la experimentación con conjuntos reales de requisitos. El corpus de requisitos seleccionado fue el mismo que el usado en experimentos previos del grupo de investigación [4] con el fin de poder comparar resultados y cumplir así con el objetivo específico OBESP3. Este corpus fue agrupado en conjuntos homogéneos de requisitos RS (véase apartado 3.3.1) con el fin de probar la herramienta en distintos escenarios (objetivo específico OBESP4).

En esta iteración se desarrolló Twiga 3, mejorando el modelo de datos; se optimizaron funciones del algoritmo genético a partir de los resultados de la iteración anterior. También se incorporó la capacidad de registrar volcar los resultados en base de datos y ficheros estructurados para disponer de evidencias y poder analizar en detalle los resultados.

Finalmente, en esta iteración, se realizó toda la experimentación final con los distintos RS y se analizaron los resultados.

### *3.4.3 Diseño Técnico*

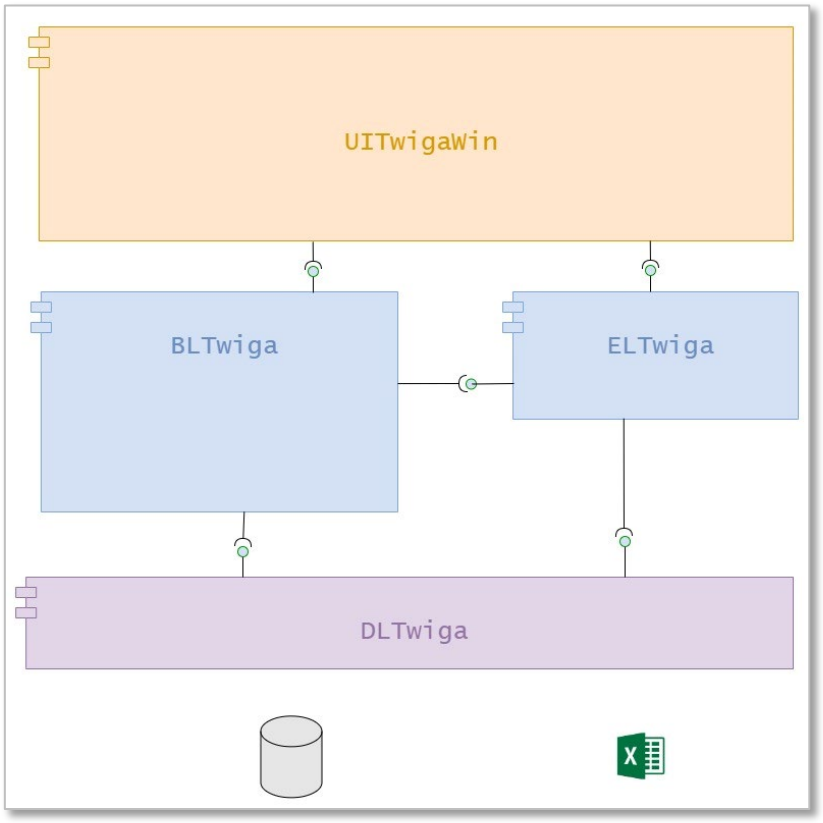
*Arquitectura*

El modelo arquitectónico seguido para el desarrollo de la herramienta ha sido el modelo de multi capas [76]. Este modelo está basado en la separación y aislamiento de funciones en distintas capas de desarrollo, siendo cada capa responsable de un conjunto de capacidades.

Las capas están superpuestas y solamente hay comunicación entre capas adyacentes o con capas que contienen funcionalidades de propósito general.

En nuestro caso hemos definido las capas que se muestran en la Figura 3-17.

FIGURA 3-17 CAPAS DE ARQUITECTURA EN TWIGA




La Tabla 3-5 resume la función principal de cada capa:

TABLA 3-5 CAPAS DE ARQUITECTURA EN TWIGA Y FUNCIÓN

| Capa | Función |
|---|---|
| UITwigaWin | Capa de Presentación e interacción con el usuario |
| BLTwiga | Capa de lógica y de algoritmos para experimentación |
| ELTwiga | Capa para funciones auxiliares |
| DLTwiga | Capa de gestión de entrada y salida de datos |

*Diagrama de componentes*

La Figura 3-18 muestra los componentes definidos y las capas en los que se ubican. En la Tabla 3-6 se describe la funcionalidad básica de cada componente.

FIGURA 3-18 DIAGRAMA DE COMPONENTES DE TWIGA

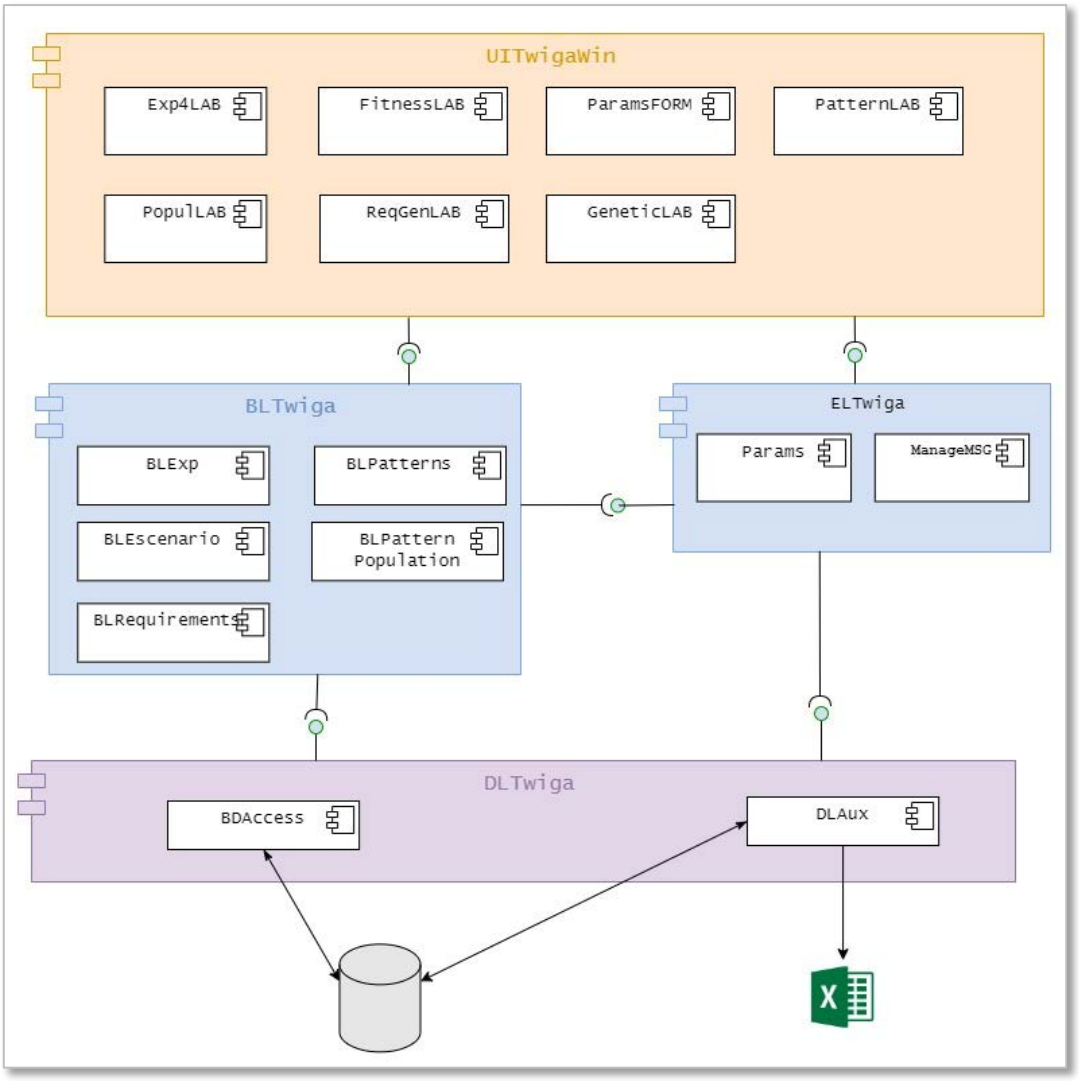


TABLA 3-6 FUNCIONALIDAD BÁSICA DE CADA COMPONENTE DE DISEÑO DE TWIGA

| Capa | Componente | Función |
|---|---|---|
| UITwigaWin | Exp4LAB | Formulario desde donde se parametrizan y ejecutan los experimentos finales de Twiga 3. |
| | FitnessLAB | Formulario para ejecutar experimentaos relacionados con la función de *Fitness*. Permite variar ciertos parámetros de la función de *fitness* y muestra métricas y resultados en distintos escenarios. |
| | ParamsFORM | Formulario que permite modificar ciertos parámetros principales de los experimentos sin necesidad de modificar códigos o tablas de BBDD. |
| | PatternLAB | Formulario para ejecutar experimentos relacionados con la generación de patrones, y verificar de frecuencia de aparición de alelos según parametrización. |
| | PopulLAB | Formulario para realizar pruebas relacionadas con la generación de poblaciones de patrones. |
| | ReqGenLAB | Formulario para la generación de requisitos artificiales a partir de patrones predefinidos. Permite definir patrones y, a partir de los mismos, generar poblaciones de requisitos basados en dichos patrones. El número de requisitos artificiales a generar es definido por el usuario. Estas poblaciones pueden ser identificadas y almacenadas en BBDD a petición del usuario para posteriores experimentos. |
| | GeneticLAB | Formulario para realizar pruebas relacionadas con el funcionamiento del algoritmo genético. Permite ejecutar paso a paso el algoritmo genético y cambiar parámetros principales del algoritmo. El formulario va mostrando, en cada generación, los valores de *fitness* de la población, los requisitos que han sido reconocidos y otras métricas. También permite con funcionalidad como funcionalidad de usuario la opción de conquistar requisitos con los patrones encontrados. Es un Laboratorio básico para analizar el y ajustar el comportamiento de los algoritmos genéticos y de conquista. |
| BLTwiga | BLExp | Componente que contiene los métodos y algoritmos para la experimentación. Este componente es utilizado por los distintos formularios de Laboratorio (GeneticLAB, etc.) y también por el formulario Exp4LAB |

| Capa | Componente | Función |
|---|---|---|
| | BLPatterns | Métodos de gestión de la clase Pattern. Contiene todas las lógicas para la creación de patrones y funciones genéticas de patrones (*fitness*, selección, cruce, mutación, reemplazo) |
| | BLEscenario | Componente para la carga y actualización de escenarios: poblaciones de requisitos a tratar en los experimentos. |
| | BLPattern Population | Métodos para la gestión de poblaciones de patrones. |
| | BLRequirements | Métodos para la gestión de requisitos. Contiene también las lógicas del algoritmo de conquista de requisitos. |
| ELTwiga | Params | Componente donde se centralizan todos los parámetros del sistema y sus valores por defecto. Este componente es utilizado por el resto de las componentes para leer los valores por defecto de cada parámetro. |
| | ManageMGS | Componente auxiliar para gestionar los mensajes informativos y de error del sistema. Permite cambiar a distintos modos de *debugging* ofreciendo distinto tipo de información, así como dónde se muestra dicha información (pantalla, fichero de log, etc.) |
| DLTwiga | BDAccess | Componente para la gestión de accesos a Base de Datos. Lee y escribe en Base de Datos. Es utilizado por el resto de componentes de las capas de lógica (BLTwiga y ELTwiga). |
| | DLAux | Componente de gestión de otras entradas y salidas (lectura y escritura en ficheros estructurados). Es utilizado por el resto de componentes de las capas de lógica (BLTwiga y ELTwiga). |

*Tecnologías usadas*

Para el desarrollo de Twiga se han utilizado las mismas herramientas de diseño y desarrollo que utiliza la SE *suite* para que sea compatible con dicha *suite*. Se han utilizado las siguientes tecnologías y herramientas:

- **SQL Server v19 Express Edition**: Motor de Base de Datos para tablas de la SE *suite* y tablas propias de Twiga.
- **SQL Sever Management Studio v18**: Sistema para la gestión de la Base de Datos.

- **Visual Studio Enterprise 2019**: Entorno de desarrollo integrado (IDE) y pruebas.
- **Microsoft Azure**: Repositorio Git y copias de seguridad.
- **MS-Excel**: Almacenamiento y gestión de resultados de experimentos.
- Como herramientas de desarrollo se han utilizado:
- **.NET**
- **C#**: Lenguaje de programación.
- **Visual Basic for Excel**: Macros y utilidades en MS-Excel para tratar los resultados de los experimentos.

### *3.4.4 Interfaz de Usuario*

Twiga es una aplicación MS-Windows con una interfaz de ventanas Windows. Se ha desarrollado para ser utilizada exclusivamente en el ámbito de esta investigación, como un prototipo. No posee interfaces de usuario pensadas para ser utilizadas por personas ajenas a esta investigación. No obstante, sí que se han desarrollado formularios que facilitan la ejecución de ciertos experimentos, los que más número de repeticiones han requerido y con más variaciones paramétricas, así como otros formularios auxiliares. La Figura 3-19, Figura 3-20 y Figura 3-21 muestran algunos de estos formularios y el aspecto general de Twiga.

FIGURA 3-19 FORMULARIO PARA EJECUTAR EXPERIMENTOS FINALES

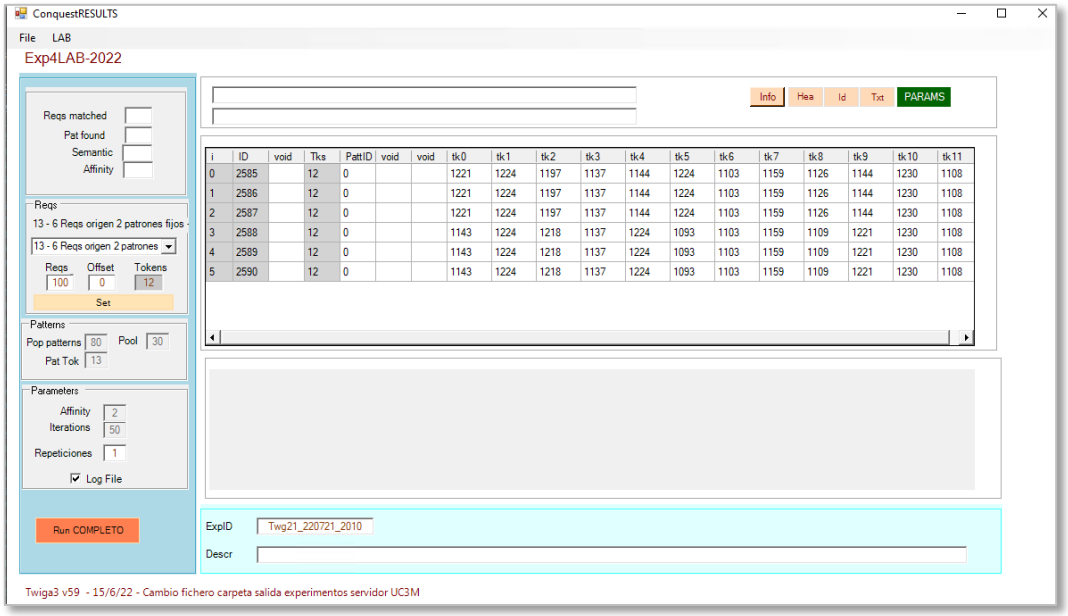



FIGURA 3-20 FORMULARIO PARA HACER PRUEBAS DEL ALGORITMO GENÉTICO

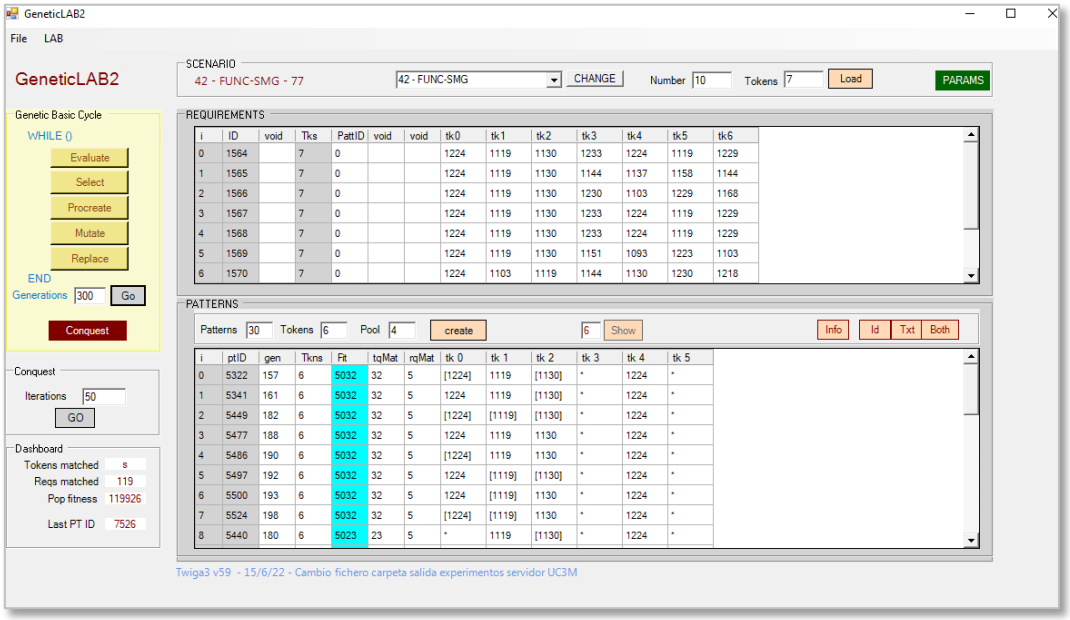
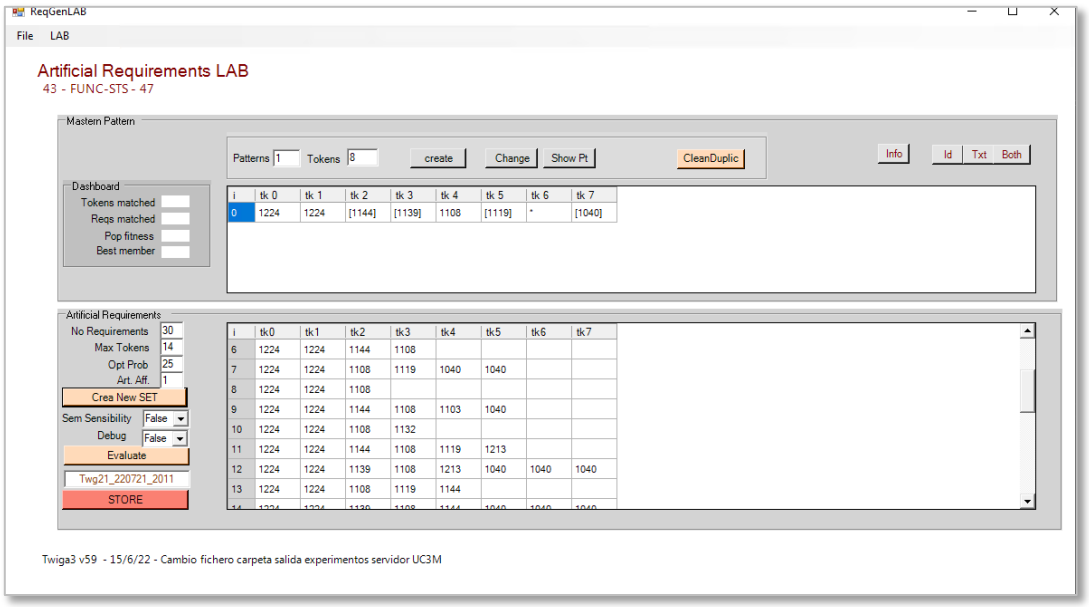


FIGURA 3-21 FORMULARIO PARA GENERAR REQUISITOS ARTIFICIALES A PARTIR DE UN PATRÓN




### *3.4.5 Modelo de Datos*

Para el almacenamiento y gestión de datos de requisitos y patrones se ha utilizado un modelo de datos compatible con el modelo RSHP y con la SE *suite*. Se han añadido nuevas clases requeridas para la gestión de la propia herramienta Twiga. En la Figura 3-22 se muestra el diagrama Entidad Relación de Twiga, así como sus principales atributos.

FIGURA 3-22 DIAGRAMA ENTIDAD RELACIÓN DE TWIGA

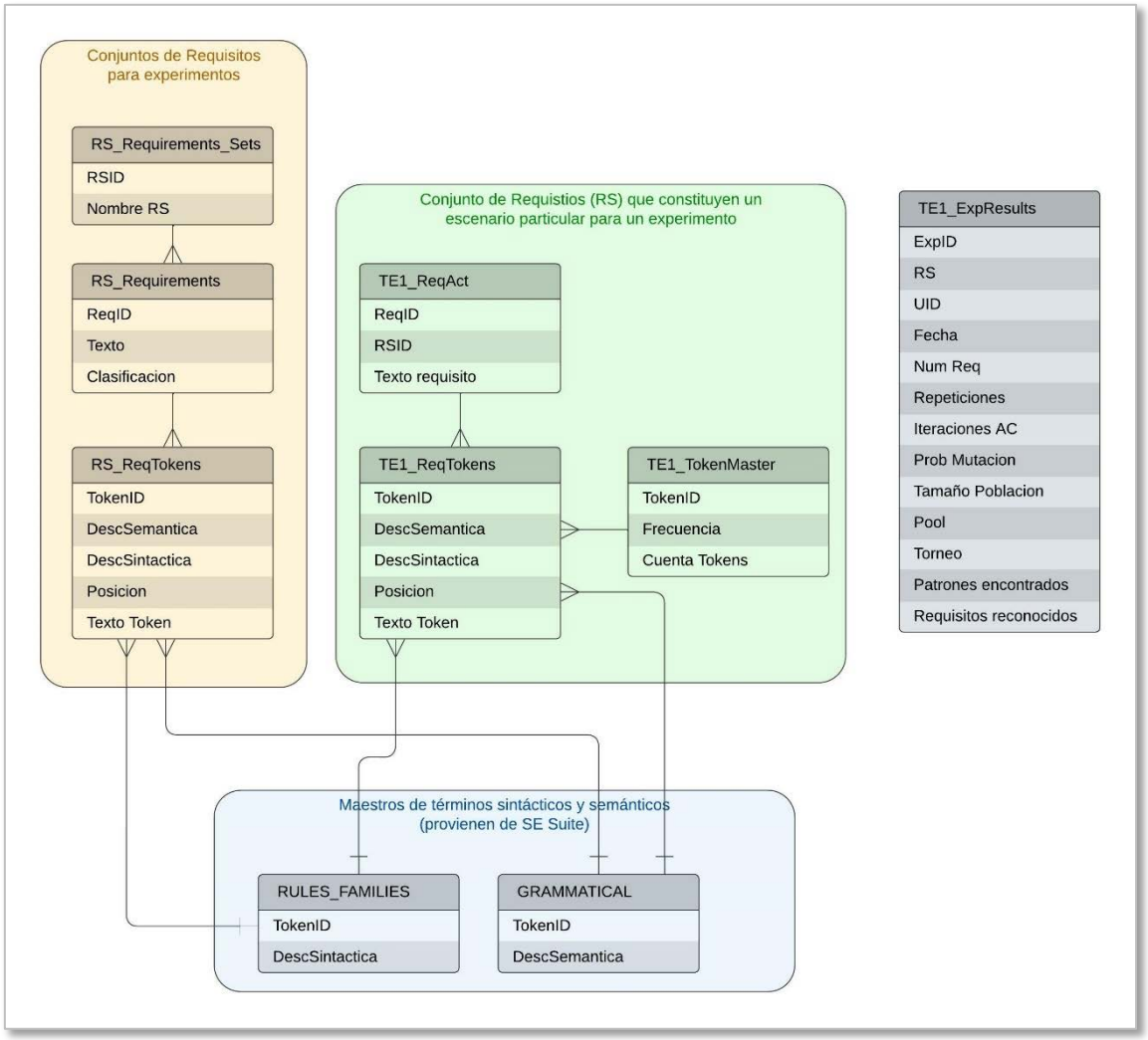



TABLA 3-7 PRINCIPALES ENTIDADES DE TWIGA Y CONTENIDO

| Entidad | Contenido / Función |
|---|---|
| RS_Requirements_Sets | Maestro de los conjuntos de requisitos (RS) utilizados en la experimentación. |
| RS_Requirements | Requisitos de todos los conjuntos de requisitos. Contiene el texto original de cada requisito. |
| RS_ReqTokens | Tokens de los requisitos. Contiene información sobre si los tokens tienen contenido solo sintáctico o sintáctico y semántico. |

| Entidad | Contenido / Función |
|---|---|
| RULES_FAMILIES | Esta entidad proviene de la SES *Suite*. Se han cargado un subconjunto de atributos que informan de los términos sintácticos a que se corresponden los tokens. |
| GRAMMATICAL | Similar a RULES_FAMILIES, pero con información de términos semánticos que se corresponden con los tokens; para tokens que están caracterizados con contenido semántico. |
| TE1_ReqAct | Contiene el conjunto de requisitos (RS) que se van a utilizar para un experimento genético concreto. |
| TE1_ReqTokens | Tokens y sus atributos del conjunto de requisito (RS) de un experimento concreto. |
| TE1_TokenMaster | Biblioteca de alelos con los datos de frecuencia del conjunto de requisitos (RS) cargado para experimentar. |
| TE1_ExpResults | Almacena información general de un experimento, así como información de los valores de los parámetros utilizados. Se trata de una tabla resumen de experimentos realizados. La información detallada de cada experimento, incluyendo todos los parámetros utilizados, información de rendimiento, patrones encontrados y requisitos reconocidos por cada patrón no se almacenan en BBDD, sino que son exportados en formato estructurado como se muestra en el apartado 3.4.6 |

### *3.4.6 Almacenamiento final de resultados*

En Base de Datos solamente se almacena información resumida de resultados de experimentos, en la tabla TE1_AxpResults. El detalle de toda la información que se genera en un experimento es volcado por Twiga en ficheros de texto estructurados para su posterior procesamiento. Se generan 4 ficheros por cada experimento:

- **Fichero de log:** con información sobre rendimiento, información de variación de valores de *fitness* y otros valores del algoritmo genético, motivo de parada del algoritmo genético y del algoritmo de conquista.
- **ExpResults**: Información general del experimento y valores de parámetros utilizados. Similar a lo almacenado en BBDD.
- **PatternsFound**: Lista de patrones encontrados con información sobre momento en que han aparecido, número de requisitos que reconocen, etc.

- **ReqsRecognized**: Lista de requisitos reconocidos por cada patrón encontrado.

Para facilitar el tratamiento de estos ficheros se ha utilizado una plantilla de MS-Excel y un conjunto de macros para ordenar y tratar la información. En el Anexo 7.3 de esta tesis se muestran los ficheros finales de resultados.

### *3.4.7 Calibración de los parámetros; experimentos con requisitos artificiales*

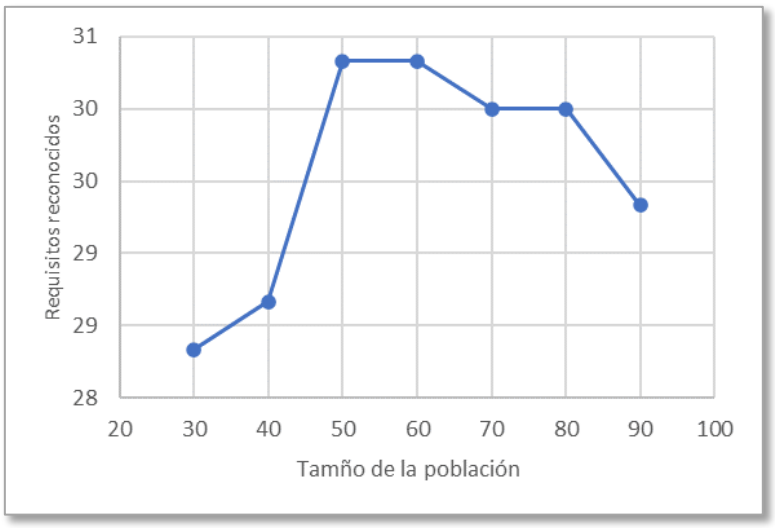
Los parámetros fueron calibrados con experimentos específicos utilizando requisitos artificiales para disponer de un entorno controlado durante la iteración 2 del proceso de desarrollo de la herramienta (ver apartado 3.4.2 en este mismo capítulo y Figura 3-16).

Se realizaron experimentos variando un parámetro en cada conjunto de experimentos y manteniendo el resto invariables. La Tabla 3-8 y la Figura 3-23 muestra como ejemplo los resultados de calibración del parámetro Tamaño de la población de patrones del AG. En este caso los valores que mejor rendimiento proporcionan en términos de resultados de las métricas de calidad Porcentaje de requisitos reconocidos *(%Rm)* y Promedio de requisitos reconocidos por patrón *(AvRm)* son valores de tamaño de la población de entre 50 y 80 individuos. Estos son los valores que se utilizaron en la experimentación con los conjuntos de requisitos reales.

TABLA 3-8 CALIBRACIÓN DEL TAMAÑO DE LA POBLACIÓN

| Tamaño de la población | Núm. Requisito (RS) | Requisitos reconocidos | Patrones encontrados RPS | *%Rm* | *AvRm* |
|---|---|---|---|---|---|
| 30 | 35 | 28,33 | 5,67 | 81,0% | 5,00 |
| 40 | 35 | 28,67 | 5,00 | 81,9% | 5,73 |
| 50 | 35 | 30,33 | 5,83 | 86,7% | 5,20 |
| 60 | 35 | 30,33 | 5,67 | 86,7% | 5,35 |
| 70 | 35 | 30,00 | 5,33 | 85,7% | 5,63 |
| 80 | 35 | 30,00 | 4,33 | 85,7% | 6,92 |
| 90 | 35 | 29,33 | 4,33 | 83,8% | 6,77 |

FIGURA 3-23 CALIBRACIÓN DEL TAMAÑO DE LA POBLACIÓN




La misma aproximación se realizó para los parámetros principales de los algoritmos, obteniéndose los valores que se usaron posteriormente en los experimentos con conjuntos de requisitos reales y que se muestran en la Tabla 3-9.

TABLA 3-9 CALIBRACIÓN DE LOS PARÁMETROS DEL AG Y DEL AQ UTILIZADOS EN LOS EXPERIMENTOS

| Parámetro | Valor |
|---|---|
| Tamaño de la población | 50 a 80 |
| Tamaño del pool de reproducción | 20 |
| Afinidad | 3 |
| Generaciones | 300 |
| Número máximo de iteraciones de conquista | 50 |
| Mínimo número de requisitos a reconocer por un patrón (*MinRm*) | 2 |
| Nivel de mutación | 25% |
| Tamaño máximo del patrón | 13 |

# 4 EXPERIMENTOS Y RESULTADOS

En este capítulo describe cómo se ha planteado la experimentación para dar cobertura a las hipótesis planteadas y también para la verificación de cumplimiento de los objetivos.

Se describen en detalle los conjuntos de datos utilizados pertenecientes todos al mismo corpus de requisitos proporcionados por INCOSE.

La experimentación sistemática se ha realizado con la herramienta Twiga en la versión correspondiente a la tercera iteración de desarrollo (Twiga 3). Con el fin de entender mejor cómo se obtienen resultados con la herramienta se incluye un apartado con ejemplos de los resultados, que muestra la herramienta cuando busca patrones en conjuntos de requisitos.

Finalmente se muestran y analizan los resultados de la experimentación y se hace un análisis comparativo de estos resultados frente a los obtenidos en experimentos anteriores del grupo.

## 4.1 Planteamiento de la experimentación

La experimentación se ha diseñado para confirmar las hipótesis de partida: es posible el uso de los algoritmos genéticos para encontrar automáticamente patrones en conjuntos de requisitos (**Hipótesis 1**). Adicionalmente el uso de la técnica de separación u conquista, permite encontrar un conjunto de patrones RPS que reconozcan conjuntos completos de requisitos RS de un ismo dominio (**Hipótesis 2**). La parametrización de la herramienta permite optimizar y cambiar el tamaño de RPS en función de los objetivos (**Hipótesis 3**); en este caso se ha optimizado la parametrización para obtener conjunto de patrones RPS válidos para ser usados para la autoría de nuevos requisitos del mismo dominio.

La calidad de las soluciones se ha establecido utilizando las métricas definidas en el apartado 3.3.8 de esta tesis. *AvRm*, el promedio de requisitos reconocidos por un patrón y el número total de patrones requeridos para reconocer los conjuntos de requisitos RS se utilizarán para comparar el rendimiento del método propuesto en esta tesis con experimentos anteriores realizados en el grupo de investigación.

## 4.2 Juegos de datos para la experimentación

Del corpus de 1035 requisitos totales proporcionados por INCOSE se han utilizado exclusivamente los 545 requisitos clasificados como de buena calidad (véase apartado 3.3.2 de esta tesis).

Se ha realizado un trabajo adicional de clasificación por dominios funcionales o por tipos de requisitos para generar conjuntos de requisitos (RS) homogéneos partiendo de una primera clasificación proporcionada también por INCOSE. Se han generado 10 RS tal y como aparecen en la Tabla 4-1.

TABLA 4-1 CONJUNTOS DE REQUISITOS RS UTILIZADOS EN LOS EXPERIMENTOS

| RS Id | Nombre RS | Número de requisitos | Contenido |
|---|---|---|---|
| 3 | Hwsw_1 | 45 | Requisitos de hardware y software de base |
| 41 | Funct_Gen1 | 34 | Requisitos funcionales |
| 42 | Funct_SMG | 77 | Requisitos funcionales |
| 43 | Funct_STS | 47 | Requisitos funcionales |
| 44 | Funct_Gen2 | 101 | Requisitos funcionales |
| 45 | Funct_QAR | 60 | Requisitos funcionales y de calidad |
| 46 | Funct_QM | 65 | Requisitos funcionales y de calidad |
| 47 | Funct_SMG | 75 | Requisitos funcionales |
| 48 | Hwsw_2 | 23 | Requisitos de hardware y software de base |
| 49 | SEC | 18 | Requisitos de seguridad |
| | TOTAL | 545 | |

Se ha seleccionado este estos conjuntos de requisitos porque ha sido utilizados en otros trabajos de investigación del grupo KR; en particular en la investigación de la tesis de E. Parra [4]. Esto nos ha permitido contrastar los resultados de nuestra investigación con aproximaciones y experimentos previos (véase apartado 4.5 en este mismo Capítulo).

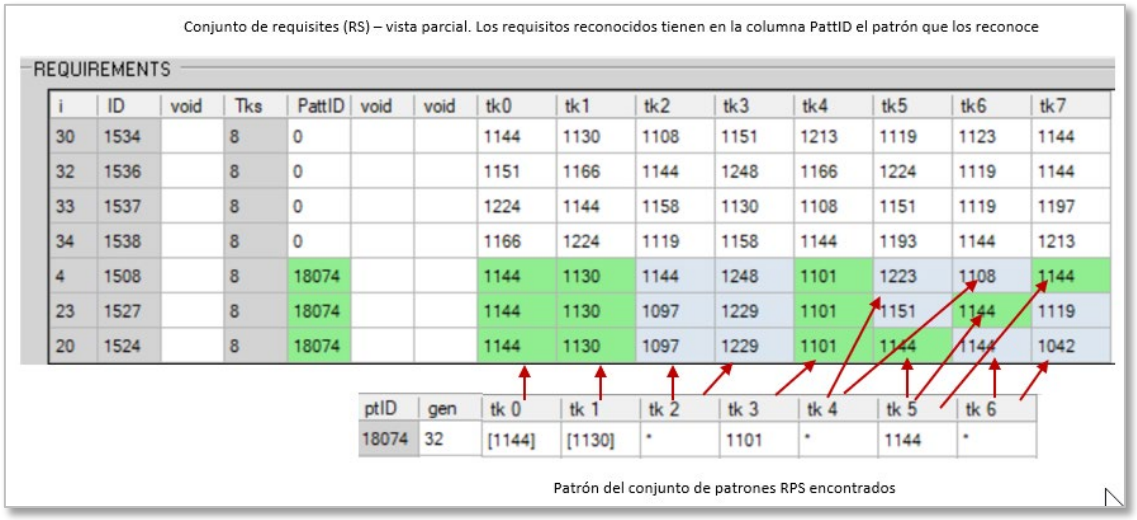
## 4.3 Ejemplo de resultados y salidas de la herramienta

### Pantallas de visualización en Twiga

La Figura 4-1 muestra una instantánea real de cómo en Twiga se visualizan los requisitos y los patrones que encuentra el AG. En esta pantalla se puede ver cómo un patrón reconoce varios requisitos de un RS en particular.

Tanto en las pantallas de Twiga como en los registros de salida, los tokens se representan por su ID numérico tal y como se almacena en la Base de Datos. Estos IDs representan términos sintácticos o semánticos (véase Figura 3-3 y Tabla 3-1).

FIGURA 4-1 VISUALIZACIÓN EN TWIGA DE PATRONES Y REQUISITOS.




| i | ID | void | Tks | PattID | void | void | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 1534 | | 8 | 0 | | | 1144 | 1130 | 1108 | 1151 | 1213 | 1119 | 1123 | 1144 |
| 32 | 1536 | | 8 | 0 | | | 1151 | 1166 | 1144 | 1248 | 1166 | 1224 | 1119 | 1144 |
| 33 | 1537 | | 8 | 0 | | | 1224 | 1144 | 1158 | 1130 | 1108 | 1151 | 1119 | 1197 |
| 34 | 1538 | | 8 | 0 | | | 1166 | 1224 | 1119 | 1158 | 1144 | 1193 | 1144 | 1213 |
| 4 | 1508 | | 8 | 18074 | | | 1144 | 1130 | 1144 | 1248 | 1101 | 1223 | 1108 | 1144 |
| 23 | 1527 | | 8 | 18074 | | | 1144 | 1130 | 1097 | 1229 | 1101 | 1151 | 1144 | 1119 |
| 20 | 1524 | | 8 | 18074 | | | 1144 | 1130 | 1097 | 1229 | 1101 | 1144 | 1144 | 1042 |

| ptID | gen | tk 0 | tk 1 | tk 2 | tk 3 | tk 4 | tk 5 | tk 6 |
|---|---|---|---|---|---|---|---|---|
| 18074 | 32 | [1144] | [1130] | * | 1101 | * | 1144 | * |

En este ejemplo, el patrón con ID 18074 se ha encontrado en la generación 32 (columna gen de la tabla), del algoritmo genético y reconoce y conquista 3 requisitos: los requisitos con ID 1508, 1527 y 1524. En estos requisitos los slots con fondo verde contienen tokens que coinciden con los tokens del patrón en el mismo slot y los slots con fondo azul son slots reconocidos por tokens comodín del patrón. En el patrón los asteriscos representan el token comodín como se describe en el apartado 3.2.5 y los corchetes indican tokens opcionales como se describe en el apartado 3.2.4. Los requisitos con el atributo PattID con valor 0 son requisitos todavía no conquistados.

Esto es sólo un ejemplo de una iteración de conquista. En los experimentos reales, se ejecutan varias iteraciones de conquista para obtener conjuntos de patrones.

### Registros de salida

Twiga, de manera adicional a las pantallas / formularios donde se parametrizan, gestionan y visualizan los experimentos, genera registros de salida con toda la información relativa al experimento. Los resultados registrados son:

- Resumen de información de cada experimento y resultado. Un ejemplo se puede ver en la Tabla 4-2.
- Patrones encontrados en un experimento. Un ejemplo se puede ver en la Tabla 4-3.

- Lista de requisitos reconocidos por cada patrón encontrado en un experimento. Un ejemplo se puede ver en la Tabla 4-4.

TABLA 4-2 TABLA RESUMEN DE RESULTADO DE UN EXPERIMENTO REGISTRADA POR TWIGA

| | | | | Resultados | |
|---|---|---|---|---|---|
| ID Exp. | RS | Núm. requisitos | Otros atributos … | Patrones encontrados RPS | Requisitos reconocidos |
| 1 | Hwsw_1 | 45 | … | 9 | 41 |
| 2 | Funct_Gen1 | 34 | … | 7 | 25 |

TABLA 4-3 EJEMPLO DE LISTA DE PATRONES ENCONTRADOS EN UN EXPERIMENTO

| ID Exp. | RS | Patt ID | Otros atributos | Token 0 | … | Token n |
|---|---|---|---|---|---|---|
| 1 | Hwsw_1 | 759 | … | * | … | [1144] |
| 1 | Hwsw_1 | 1985 | … | [1144] | … | 1119 |

TABLA 4-4 EJEMPLO DE LISTA DE REQUISITOS RECONOCIDOS EN UN EXPERIMENTO

| ID Exp. | RS | Patt ID | Otros atributos | Id de requisito reconocido |
|---|---|---|---|---|
| 1 | Hwsw_1 | 759 | … | 1504 |
| 1 | Hwsw_1 | 759 | … | 1507 |
| 1 | Hwsw_1 | 759 | …. | 1508 |

En el Anexo 7.3 se recogen los registros completos de resultados de los experimentos utilizados en esta tesis.

## 4.4 Resultados

### *4.4.1 Resultados de los experimentos*

La Tabla 4-5 muestra los resultados de los experimentos realizados con cada set de datos. En todos los experimentos se mantuvo la misma parametrización indicada en la Tabla 3-9.

TABLA 4-5 RESULTADOS DE LOS EXPERIMENTOS

| ID Exp. | Nombre RS | Núm. requisitos | Requisitos reconocidos | Patrones encontrados RPS | *%Rm* | *AvRm* | *BPRm* |
|---|---|---|---|---|---|---|---|
| 1 | Hwsw_1 | 45 | 41 | 9 | 91% | 4,56 | 11 |
| 2 | Funct_Gen1 | 34 | 25 | 7 | 74% | 3,57 | 6 |
| 3 | Funct_SMG | 77 | 73 | 20 | 95% | 3,65 | 15 |
| 4 | Funct_STS | 47 | 47 | 14 | 100% | 3,36 | 14 |
| 5 | Funct_Gen2 | 101 | 93 | 22 | 92% | 4,23 | 17 |
| 6 | Funct_QAR | 60 | 52 | 10 | 87% | 5,20 | 15 |
| 7 | Funct_QM | 65 | 58 | 14 | 89% | 4,14 | 9 |
| 8 | Funct_SMG | 75 | 54 | 12 | 72% | 4,50 | 16 |
| 9 | Hwsw_2 | 23 | 17 | 5 | 74% | 3,40 | 6 |
| 10 | SEC | 18 | 12 | 4 | 67% | 3,00 | 5 |
| | TOTAL | 545 | 472 | 117 | 87% | 4,03 | |

Los siguientes son los aspectos más significativos de los resultados obtenidos

- Las medias de reconocimientos de requisitos oscilan entre un mínimo de *%Rm* = 67% y un máximo de reconocimiento de todos los requisitos (*%Rm* = 100%) según el RS con el que se experimenta.
- La media de requisitos reconocidos por patrón, *AvRm*, está entre 3,00 y 5,2. Este resultado supera ampliamente los resultados de experimentos previos (véase apartado 4.5). Esto significa que el AG ha encontrado requisitos lo suficientemente flexibles como para reconocer varios requisitos similares.

- Como el parámetro *MinRm* que fija el número mínimo de requisitos a reconocer por un patrón está fijado en 2, los requisitos no reconocidos son, de alguna manera, requisitos únicos en el RS.
- La métrica de calidad de Número de requisitos reconocidos por el mejor patrón (*BPRm*) toma valores de 5 a 15 requisitos reconocidos por un mismo patrón en distintos RS. La variación está relacionada fundamentalmente con la variabilidad de requisitos en cada RS; los RS con menor vocabulario y tamaño menor de requisitos muestran mejores valores de *BPRm* y mayores valores de *AvRm* lo que significa que se pueden encontrar conjunto de patrones RPS menores para reconocer todo el RS.

### *4.4.2 Tiempos de ejecución del algoritmo*

La Tabla 4-6 muestra información sobre el tiempo medio de ejecución de los experimentos con los distintos RS y el tiempo medio necesario para obtener un nuevo patrón válido.

La optimización de la herramienta no ha sido objetivo de esta experimentación y podrá ser objeto de futuros trabajos.

TABLA 4-6 TIEMPO DE EJECUCIÓN DE LOS EXPERIMENTOS.

| ID Exp. | Nombre RS | Núm. requisitos | Requisitos reconocidos | Duración (*minutos*) | Tiempo medio para encontrar cada patrón (minutos) |
|---|---|---|---|---|---|
| 1 | Hwsw_1 | 45 | 41 | 20,92 | 2,32 |
| 2 | Funct_Gen1 | 34 | 25 | 16,04 | 2,29 |
| 3 | Funct_SMG | 77 | 73 | 47,23 | 2,36 |
| 4 | Funct_STS | 47 | 47 | 20,97 | 1,50 |
| 5 | Funct_Gen2 | 101 | 93 | 48,56 | 2,21 |
| 6 | Funct_QAR | 60 | 52 | 21,43 | 2,14 |
| 7 | Funct_QM | 65 | 58 | 25,02 | 1,79 |
| 8 | Funct_SMG | 75 | 54 | 24,67 | 2,06 |
| 9 | Hwsw_2 | 23 | 17 | 7,59 | 1,52 |
| 10 | SEC | 18 | 12 | 10,83 | 2,71 |

Las consideraciones más significativas de estos resultados son:

- Existe una correlación entre el tamaño del RS y la duración del experimento como se muestra en la Figura 4-2. Esto tiene sentido por cuanto todos los bucles del AG se ejecutan tantas veces como número de requisitos tiene el RS.
- El promedio de tiempo requerido para encontrar un patrón es relativamente constante en cada RS como muestra la Figura 4-3. En este caso no se aprecia relación con el tamaño del RS. Es probable que existan otros parámetros que estén relacionados con este tiempo y puedan ser optimizados en trabajos futuros como el tamaño de la población de patrones del AG.

FIGURA 4-2 CORRELACIÓN ENTRE TAMAÑO DEL RS Y TIEMPOS DE EJECUCIÓN

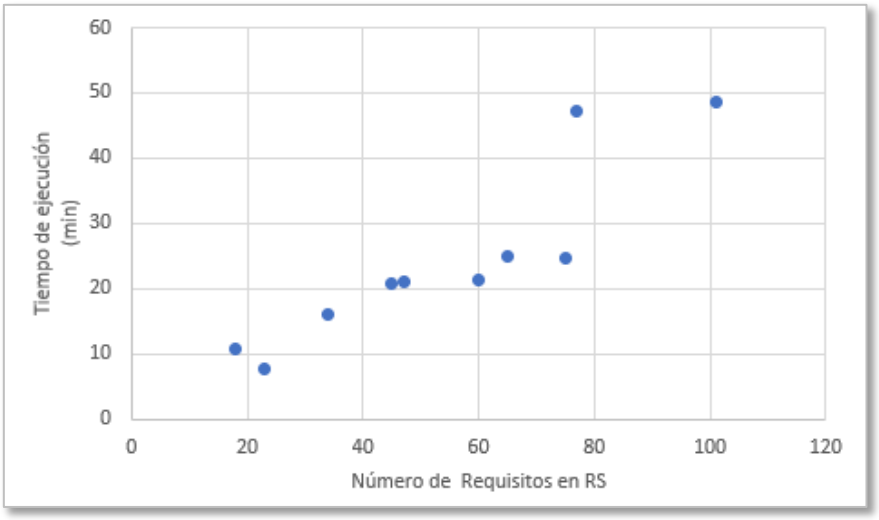
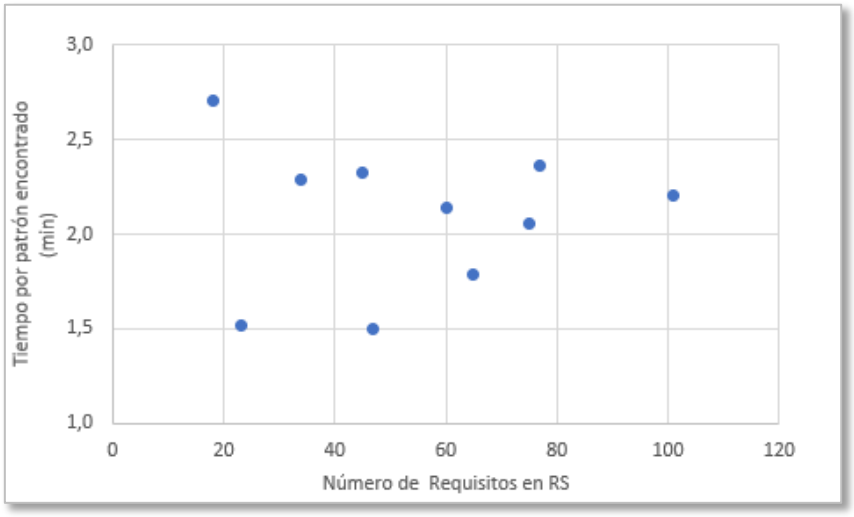



FIGURA 4-3 TIEMPO MEDIO REQUERIDO PARA ENCONTRAR UN PATRÓN.




### *4.4.3 Impacto del tamaño de los requisitos y del parámetro Afinidad*

El tamaño de los requisitos, es decir, el número de palabras de un requisito, ha sido identificado en estudios anteriores [77] como la principal métrica para determinar la calidad de los requisitos. Los requisitos de menor tamaño son, en general, los mejor definidos.

Este parámetro también tiene un gran impacto en la búsqueda de patrones de texto utilizando nuestra metodología. La Tabla 4-7 muestra cómo el número máximo de palabras de los requisitos influye en el número de patrones encontrados. También se

analiza el impacto de cambio en el parámetro afinidad, directamente relacionado con el tamaño de los requisitos.

TABLA 4-7 IMPACTO DEL NÚMERO DE PALABRAS DE LOS REQUISITOS Y DEL PARÁMETRO AFINIDAD

| | Afinidad = 3 | | Afinidad = 4 | |
|---|---|---|---|---|
| Núm. Máximo de palabras en requisito | Requisitos reconocidos | RPS | Requisitos reconocidos | RPS |
| 12 | 40 | 5 | 45 | 5 |
| 13 | 40 | 5 | 45 | 4 |
| 14 | 21 | 6 | 41 | 4 |
| 15 | 19 | 4 | 38 | 5 |
| 16 | 12 | 3 | 35 | 4 |

Las consideraciones más relevantes respecto a los resultados mostrados en la Tabla 4-7 son:

- En general cuando los requisitos tienen más de 13 palabras cae el rendimiento del algoritmo que no es capaz de encontrar patrones que reconozcan a los requisitos.
- El parámetro afinidad, número de slots de requisitos que reconoce un token comodín de un patrón, tiene un alto impacto en los requisitos reconocidos y en la capacidad de reconocer requisitos más extensos.
- Valores mayores de este parámetro permiten reconocer más requisitos con el mismo número de patrones. Los patrones así obtenidos con son más generalistas, menos específicos.

## 4.5 Comparación con experimentos anteriores

Uno de los objetivos de esta tesis ha sido verificar si la aproximación metodológica de uso de algoritmos genéticos (AG) y técnicas de conquista (AC) mejoran otras aproximaciones previas realizadas por el grupo de investigación. En concreto hemos querido validar los resultados contra los obtenidos en la tesis de E. Parra [4] en la que se

utilizó como metodología básica la frecuencia de aparición de patrones binarios consecutivos y su sustitución por subpatrones (véase apartado 2.10 de esta tesis).

Con el fin de poder comparar los resultados de ambas metodologías, se han utilizado los mismos corpus y número de requisitos, así como el mismo proceso de tratamiento de lenguaje natural.

No obstante, para poder comparar los resultados se requiere una corrección en los resultados obtenidos en nuestros experimentos. Esta corrección consiste en añadir en nuestros experimentos un patrón por cada requisito que no ha sido reconocido, ya que el experimento de E. Parra se diseñó para reconocer todos los requisitos, utilizando si era necesario, patrones para un único requisito. Con esta corrección se dispone del mismo escenario en cada método y los resultados son comparables. La Tabla 4-8 muestra esta lo datos originales de nuestros experimentos en la fila 1, los mismos datos corregidos para hacerlos comparables con el experimento de E. Parra en la fila 2 y los resultados obtenidos por E. Parra en la fila 3.

TABLA 4-8 COMPARACIÓN DE RESULTADOS CON EXPERIMENTO DE E. PARRA

| Fila | Experimento | Núm. Requisitos | Requisitos reconocidos | RPS | *AvRm* |
|---|---|---|---|---|---|
| 1 | Experimento propio (uso de AG + AQ) | 545 | 472 | 117 | 4,03 |
| 2 | Experimento propio corregido (adición de patrones para requisitos no reconocidos) | 545 | 545 | 190 [1] | 2,87 |
| 3 | Experimento E. Parra [3] | 545 | 545 | 442 | 1,23 |

1- Añadimos un nuevo patrón para cada requisito no reconocido por el RPS obtenido en los experimentos. Esto permite comparar los resultados con el experimento de E. Parra.

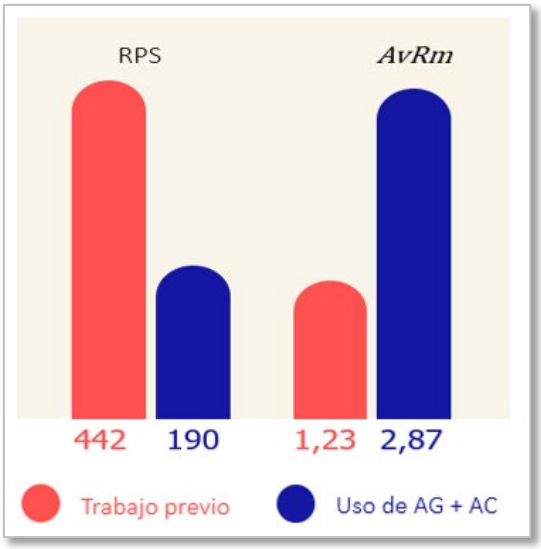
En el experimento realizado por E. Parra, se necesitaron 442 patrones para hacer reconocer los 545 requisitos del RS, con un valor de la métrica de calidad Media de requisitos reconocidos por patrón *AvRm* de 1,23. En nuestro experimento, una vez corregidos los resultados, hemos obtenido un total de 190 patrones (véase fila 2 de la Tabla 4-8). El valor de la métrica *AvRm* es en nuestro experimento de 2,87.

La relación de mejora de $AvRm$ en nuestra investigación respecto al experimento realizado por E. Parra es de un 2,33 lo que representa una mejora del 233% conforme a la ecuación (4).

$$\text{Mejora} = \frac{AvRm \text{ experimentos de esta tesis}}{AvRm \text{ experimento E. Parra}} \times 100 \qquad (4)$$

La Figura 4-4 muestra cómo la metodología y desarrollos realizados en esta tesis mejoran los resultados de la investigación previa.

FIGURA 4-4 COMPARACIÓN DE RESULTADOS C ON LOS OBTENIDOS POR E. PARRA



# 5 CONCLUSIONES, APORTACIONES Y TRABAJOS FUTUROS

En este capítulo presentamos las conclusiones obtenidas tras la experimentación y su posterior análisis; hacemos también una verificación de hasta qué punto nuestras hipótesis de partida eran válidas y hasta qué punto se han cumplido los objetivos generales y específicos planteados al inicio del a investigación.

Por último, proponemos posibles líneas y trabajos futuros de investigación partiendo de los resultados de la presente tesis doctoral que no han sido abordados por limitaciones del ámbito de investigación o por exceder los objetivos de la presente tesis.

## 5.1 Conclusiones

### *5.1.1 Conclusiones científicas*

Las conclusiones científicas que derivamos de la investigación llevada a cabo en esta tesis son:

- **Viabilidad del uso de algoritmos genéticos**: Los algoritmos genéticos son una aproximación válida para la generación automática de patrones de requisitos, si se dispone de un conjunto inicial de requisitos de calidad homogéneo en un dominio determinado.
- **Importancia de la función de evaluación**: De las múltiples funciones que componen los algoritmos implementados, la función de *fitness* del algoritmo genético es la función clave ya que es la que permite determinar cómo de buenos son los individuos de una población y clasificarlos para continuar con el resto de funciones del AG. Por tanto, la resolución de otros tipos de problemas con Algoritmos Genéticos requiere en primer lugar diseñar una buena función de evaluación de los individuos-solución que se definan. Sin este prerrequisito no es posible el uso de AG como aproximación a un problema.
- **Posibilidad de uso de estrategia de conquista junto con el algoritmo genético:** Si se añaden iteraciones de conquista, es posible generar conjuntos de patrones para reconocer conjuntos de requisitos. Esto permite ofrecer una solución completa al problema de preparar un conjunto de patrones para un dominio determinado a partir de un subconjunto de requisitos.

### *5.1.2 Conclusiones y resultados tecnológicos*

Las conclusiones y resultados tecnológicos derivados de esta tesis son:

- **Implementación de un prototipo funcional**: Se ha desarrollado una herramienta completa, Twiga que implementa la metodología y algoritmos de esta tesis. Esta herramienta ha sido desarrollada como prototipo de un posible producto o componente a integrar en la SE *suite*.
- **Arquitectura compatible con SE *suite*:** La arquitectura implementada permitirá en un futuro incorporar como un componente adicional o como un producto independiente la funcionalidad de generación automática de patrones dentro de la

SE *suite*. También sería viable el desarrollo de servicios a consumir por otros sistemas en una arquitectura de servicios web.

- **Prototipo con un alto nivel de parametrización:** Se ha implementado un componente específico de parametrización, con un número total de 34 variables parametrizables. Este componente unifica y abstrae del resto de clases y métodos de Twiga toda la funcionalidad relacionada con el cambio de comportamiento de los algoritmos.
- **Capacidad de reúso del algoritmo genético AG:** Se ofrece un conjunto de métodos del algoritmo genético para el problema de la generación automática de patrones que, por su diseño, permiten generar, a partir de los mismo y con un nivel de abstracción mayor, una máquina general de ejecución de algoritmos genéticos en la que se podrían definir funciones específicas (de generación de población, evaluación de la población, reproducción, etc.) distintas que se invocarían desde el algoritmo genético genérico para resolver otros posibles problemas.

### *5.1.3 Validación de las hipótesis de partida*

Abordamos aquí hasta qué punto nuestras hipótesis de partida han sido validadas mediante la experimentación y análisis de resultados y hasta qué punto por tanto se pueden considerar como confirmadas o refutadas desde el punto de vista científico.

- **Hipótesis 1**. – Es posible generar patrones de requisitos automáticamente usando algoritmos genéticos (AG):
    - Los resultados muestran que el uso de algoritmos genéticos permite encontrar patrones de requisitos automáticamente sin intervención de expertos y con el tratamiento previo adecuado de un cuerpo de requisitos de calidad. En los distintos experimentos efectuados se han encontrado conjuntos de patrones que reconocían entre un 67% y un 100% de requisitos según el conjunto de requisitos RS de partida.
- **Hipótesis 2**. – Añadiendo a los AG técnicas de separación y conquista se puede generar un conjunto completo de patrones válido para un dominio específico:
    - Se han obtenido, mediante la implementación de las funciones de separación y conquista en Twiga, conjuntos de patrones con capacidad de reconocimiento de conjuntos de requisitos RS. La Tabla 4-5 muestra el

número de patrones encontrados en cada conjunto de experimentación. Esto permite ofrecer una solución completa al problema de preparar un conjunto de patrones para un dominio determinado a partir de un subconjunto de requisitos de calidad.

- **Hipótesis 3**. – Es posible identificar e implementar parámetros en los algoritmos que permitan modular su comportamiento en función del objetivo que se persiga para los patrones en función de escenarios de uso de los patrones:
    - Es posible identificar parámetros que afectan significativamente el comportamiento de los algoritmos. La variación de valores de estos parámetros hace que el comportamiento del algoritmo varíe y se encuentren distintos conjuntos de soluciones, que pueden ser válidas en función del objetivo o caso de uso concreto al que se esté aplicando.
    - Se pueden definir por tanto escenarios de uso de la metodología y solución tecnológica implementada en esta tesis, que respondan a distintas necesidades. Un escenario de uso es entonces, a los efectos de esta investigación, un conjunto de valores concreto de los parámetros relevantes implementados en el AG y en el AQ que provocan un comportamiento global determinado de la herramienta.
    - Se han implementado 34 variables parametrizables principales (véase apartado 3.3.7**¡Error! No se encuentra el origen de la referencia.** y apartado 7.2).

### *5.1.4 Cumplimiento de Objetivos*

Consideramos cumplidos los objetivos planteados al inicio de esta tesis a excepción del objetivo específico **OBESP4** "comprobar la validez de la herramienta con distintos corpus de requisitos", que consideramos solo parcialmente cumplido.

Se detallan a continuación los objetivos y cómo se consideran cumplidos con los resultados obtenidos en esta tesis:

- **OBGEN1:** Obtención de una herramienta software para la generación automática de patrones. Lo consideramos cumplido con la implementación de la herramienta Twiga y con los resultados obtenidos en la experimentación con esta herramienta.

Esta herramienta, por su arquitectura, podrá además ser incorporada a la SE *suite* si así se decide.

- **OBESP1** y **OBESP2**: Identificar variables y parametrizar la herramienta para dotarla de flexibilidad y poder adaptarla a distintos escenarios de uso. Se han identificado hasta 34 variables o parámetros tal y como se ha comentado en el apartado anterior de validación de hipótesis.
- **OBESP3:** Conseguir resultados de población de patrones mejores a los obtenidos en investigaciones previas del grupo. Los resultados que se han obtenido con Twiga utilizando distintos los mismos conjuntos de requisitos que en experimentaciones previas mejoran los resultados en un 233% como se muestra en la Tabla 4-8.
- **OBESP4:** Comprobar la validez de la herramienta con distintos corpus de requisitos. Lo consideramos parcialmente cumplido. Hemos realizado experimentos con conjuntos de distintos tipos de requisitos (análisis, diseño, software, técnicos) y de distintos ámbitos (seguridad, calidad…) como se muestra en la Tabla 4-5, pero todos los RS proceden de un mismo origen: colección de requisitos proporcionada por el INCOSE. La dificultad de obtener conjuntos de requisitos reales por problemas de acceso, confidencialidad, etc., así como la necesidad de ser tratados y evaluados por expertos para identificar los requisitos de calidad, ha hecho inviable la disponibilidad de otros conjuntos de requisitos RS y la realización de experimentos con ellos. No obstante, consideramos que los experimentos realizados son suficientes para confirmar la validez de la aproximación metodológica y también la validez de la implementación en la herramienta Twiga.

Durante el desarrollo de la investigación, se han identificado otros objetivos que no han sido abordados como parte de la tesis por exceder en tiempo y esfuerzo el ámbito de la misma. Estos nuevos objetivos o posibles escenarios de investigación y avance se han recogido y documentado en el apartado 5.3 de esta tesis.

## 5.2 Principales aportaciones

Esta tesis doctoral ofrece una nueva aproximación metodológica y desarrolla un prototipo plenamente funcional para la generación automática de patrones de texto aplicada al ámbito de la ingeniería de requisitos. En esta investigación hemos comprobado que la unión del uso de algoritmos genéticos y algoritmos de conquista permiten generar patrones automáticamente. Esta es una aproximación novedosa, que no había sido utilizada con anterioridad en este ámbito.

Consideramos esta la principal aportación a la Comunidad Académica y a la Comunidad Industrial.

La identificación de múltiples parámetros que afectan al comportamiento de los algoritmos es también una aportación importante por cuanto permite evolucionar la investigación con nuevos experimentos ante nuevos escenarios utilizando los mismos algoritmos y desarrollos y alterando el comportamiento global mediante el ajuste de estos parámetros.

La propia herramienta Twiga, plenamente funcional, aunque se trate de un prototipo, y a disposición de la Comunidad, es una aportación que puede facilitar el desarrollo de nuevos componentes o productos a partir de ella.

## 5.3 Futuros trabajos

Durante el planteamiento y ejecución de esta investigación, hemos identificado distintas vías de mejora y de experimentación que podrían completar los resultados finales, tanto a nivel de la metodología (cambios que podrían mejorar o completar la metodología) como mejoras a nivel de la propia herramienta Twiga desarrollada. Estas mejoras no se han abordado por exceder el ámbito, alcance y tiempo de esta investigación. Mencionamos aquí posibles líneas de trabajo futuro que pueden ser abordadas para continuar con la metodología propuesta de obtención automática de patrones de requisitos y desarrollo de una herramienta plenamente funcional, así como su posible conversión en un producto final susceptible de ser comercializado o incorporado a la SE *suite*.

- **Uso de requisitos de distintas fuentes:** En relación con la limitación del modelo y de la investigación relacionada con los juegos de requisitos de entrada y el problema de juego de imitación (véase apartado **1.6.1**), creemos que es posible diseñar nuevos experimentos combinando requisitos del mismo dominio pero que provengan de distintos autores. El objetivo de estos experimentos sería estudiar en detalle la aparición de posibles sesgos a la hora de encontrar patrones en función de los orígenes (autores) de los requisitos, analizar hasta qué punto es un problema y si se puede corregir el comportamiento de los algoritmos.
- **Uso de requisitos de baja calidad para mejora del AG:** En una fase inicial de la definición de la metodología, se pensó utilizar también los requisitos de baja calidad del corpus de requisitos de INCOSE y considerarlos en la función de evaluación del algoritmo genético (véase función de fitness en apartado 3.3.4). La manera de considerarlos sería penalizando a patrones generados por el algoritmo genético que reconocieran también requisitos de baja calidad. Pensamos que podría ser una mejora en el sentido de que se podrían encontrar mejores patrones que no fueran capaces de reconocer requisitos "malos".
- **Definición y concreción de escenarios de uso:** Creemos que es conveniente también la realización de experimentos adicionales que puedan identificar la parametrización adecuada para distintos escenarios o casos de uso reales y que respondan a problemas típicos relacionados con la ingeniería de requisitos. La Tabla 5-1 es una primera lista de posibles escenarios, objetivos y parámetros que deben ser analizados para obtener la adecuación al escenario concreto de Twiga.

TABLA 5-1 ESCENARIOS DE UTILIZACIÓN DE LA METODOLOGÍA EN FUTUROS TRABAJOS.

| Escenario | Objetivo | Valor del parámetro |
|---|---|---|
| 1-Introducción a las herramientas de autoría. | Disponer de un RPS generalista que permita crear requisitos con relativa libertad en un dominio concreto | Afinidad – Valor alto<br>*MinRm* – Valor alto |
| 2-Utilización de herramientas de creación en un entorno muy restringido en el que se requiere precisión | Disponer de un RPS muy específico que dirija la autoría de forma muy limitada (por ejemplo, la fase de diseño) | Afinidad – Valor bajo<br>*MinRm* – Valor bajo |

- **Mejora del comportamiento del Token Comodín:** Los tokens comodines (apartado 3.2.5) permiten reconocer un número limitado paramétricamente de tokens de requisitos en el algoritmo de reconocimiento de requisitos. Una mejora importante en función del caso de uso real al que se quiera aplicar Twiga, puede ser el que los tokens de los requisitos que sean de tipo semántico no sean reconocibles por el token comodín. En el modelo de datos de la SE *suite* el carácter semántico de cada token es un atributo que está informado y por tanto podemos saber, para cada token, si se trata de un término cualificado como con significado semántico en el dominio en el que se está trabajando. Esta modificación de Twiga se podría incorporar fácilmente mediante la adición de un parámetro que indique si los tokens comodines pueden reconocer o no tokens semánticos y la actualizando el algoritmo de reconocimiento.
- **Optimización del rendimiento de Twiga:** Considerando que el rendimiento no ha sido un objetivo de esta investigación, si se desea evolucionar Twiga para convertirlo en un producto para uso comercial o como parte de la SE *suite*, serán necesarios trabajos de optimización de algoritmos. Creemos que los tiempos de ejecución mostrados en la Tabla 4-6 pueden ser mejorados de manera significativa si se analizan y optimizan los algoritmos.
- **Uso de otros algoritmos bioinspirados:** Debido a la complejidad de la incorporación de otros tipos de algoritmos bioinspirados como posibles fuentes de solución, se descartaron al inicio de la investigación el uso de algoritmos inmunológicos y de algoritmos de enjambre (colonias de hormigas, etc.), que inicialmente y de forma teórica, durante los primeros estudios, fueron considerados para la investigación. Los algoritmos inmunológicos, al permitir mayor variabilidad en la generación de posibles soluciones, podrían añadir un mayor espacio para la búsqueda de mejores soluciones, pero sería necesario hacer un diseño concreto de algoritmo aplicable a la generación automática de patrones, que no hemos realizado en esta investigación. Esta puede ser una línea de investigación que creemos que merece la pena explorar.

# 6 BIBLIOGRAFIA


[1] J. Poza, V. Moreno, A. Fraga, and J. M. Álvarez-Rodriguez, "Genetic Algorithms: A Practical Approach to Generate Textual Patterns for Requirements Authoring," *Appl. Sci.*, vol. 11, no. 23, p. 11378, 2021, [Online]. Available: https://doi.org/10.3390/app112311378

[2] "Knowledge Reuse Group: Grupo de reutilización de conocimiento." http://www.kr.inf.uc3m.es/ (accessed Aug. 11, 2022).

[3] Reuse Company, "Systems Engineering Suite." https://www.reusecompany.com/systems-engineering-suite (accessed Oct. 10, 2020).

[4] E. Parra, "Metodología orientada a la optimización automática de la calidad de los requisitos," *Tesis doctoral*, 2016. https://e-archivo.uc3m.es/handle/10016/23936 (accessed Aug. 01, 2019).

[5] C.S. Wasson, *System Engineering Analysis, Design, and Development: Concepts, Principles*. John Wiley & Sons., 2015.

[6] I. Sommerville, *Ingeniería de Software*, 10 Ed. Pearson Education, 2016.

[7] Project Management Institute, "Requirements Management: A Core Competency for Project and Program Success," *PMI's Pulse Prof.*, pp. 1–20, 2014, [Online]. Available: http://www.pmi.org/learning/thought-leadership/pulse

[8] T. Shah and S. V Patel, "A Review of Requirement Engineering Issues and Challenges in Various Software Development Methods," *Artic. Int. J. Comput. Appl.*, vol. 99, no. 15, pp. 975–8887, 2014, doi: 10.5120/17451-8370.

[9] T. S. Group International, "Chaos report 2015," 2015. https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf (accessed Sep. 21, 2021).

[10] F. P. Brooks, *The Mythical Man-Month, Anniversary edition with 4 new chapters*. Boston: Addison-Wesley, 1995.

[11] M. B. J Braude, *Software Engineering: Modern Approaches*. Waveland Press, 2016.

[12] M. Marrero, S. Sánchez-Cuadrado, A. Fraga, and J. Llorens, "Applying Ontologies and Intelligent Text Pr ocessing in Requirements Reuse," *First Work. Knowl. reuse*, pp. 25–29, 2008.

[13] G. Génova, J. M. Fuentes, J. L. Morillo, O. Hurtado, and V. Moreno, "A framework to measure and improve the quality of textual requirements," *Requir. Eng.*, vol. 18, no. 1, pp. 25–41, 2013, doi: 10.1007/s00766-011-0134-z.

[14] "International Council on Systems Engineering Website." https://www.incose.org/ (accessed Feb. 16, 2020).

[15] J.O. Kephart, “A biologically inspired immune system for computers,” in *Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994, pp. 130–139.

[16] A. K. Kar, “Bio inspired computing – A review of algorithms and scope of applications,” *Expert Syst. Appl.*, vol. 59, pp. 20–32, Oct. 2016, doi: 10.1016/J.ESWA.2016.04.018.

[17] A. Bartoli and A. De Lorenzo, “Learning text patterns using separate-and-conquer genetic programming,” *Eur. Conf. Genet. Program.*, pp. 16–27, 2015, doi: 10.1007/978-3-319-16501-1_2.

[18] PMI, *A Guide to the Project Management Body of Knowledge*, Sixth. Pennsyvania USA: PMI, 2017.

[19] A. Fraga, J. Garcia, E. Parra, and V. Moreno, “Extraction of Patterns Using NLP: Genetic Deafness,” *SEKE*, pp. 428–431, 2017, doi: 10.18293/SEKE2017-204.

[20] P. Suarez, V. Moreno, A. Fraga, and J. Llorens, “Automatic Generation of Semantic Patterns using Techniques of Natural Language Processing,” in *Proceedings of the 4th International Workshop on Software Knowledge - SKY*, 2013, pp. 34–44. doi: 10.5220/0004641500340044.

[21] J. Furnkranz, “Separate-and-Conquer Rule Learning,” *Artificial Intelligence Review*, vol. 13, no. 3, pp. 3–54, 1999.

[22] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao, “Learning text patterns using separate-and-conquer genetic programming,” 2015. doi: 10.1007/978-3-319-16501-1_2.

[23] H. Cotterman, K. Forsberg, and H. Mooz, *Visualizing project management: models and frameworks for mastering complex systems*. New York: John Wiley & Sons., 2005.

[24] D. Walden, G. Roedler, K. Forsberg, and D. Hamelin, *Systems Engineering Handbook. A guide for system life cycle processes and activities*, 4th ed. San Diego: INCOSE, 2015.

[25] J. O. Clark, “System of systems engineering and family of systems engineering from a standards,v-model, and dual-v model perspective,” *2009 IEEE Int. Syst. Conf. Proc.*, pp. 381–387, 2009, doi: 10.1109/SYSTEMS.2009.4815831.

[26] “V-Modell ® XT-MODELL,” 2004.

[27] P. Loucopoulos and V. Karakostas, “System Requirements Engineering.” McGraw-Hill, 1995.

[28] D. Pandey, U. Suman, and A. K. Ramani, “An effective requirement engineering process model for software development and requirements management,” *Proc. - 2nd Int. Conf. Adv. Recent Technol. Commun. Comput. ARTCom 2010*, pp. 287–291, 2010, doi: 10.1109/ARTCOM.2010.24.

[29] PMI, *The Standard for Project Management (PMBOK GUIDE)*, Seventh Ed. Pennsylvania: Project Management Institute, 2021.

[30] “Herramienta de gestión de requisitos - Visure Solutions.” https://visuresolutions.com/es/ (accessed Mar. 23, 2022).

[31] “Descripción general de DOORS - Documentación de IBM.” https://www.ibm.com/docs/es/ermd/9.7.2?topic=engineering-requirements-management-doors-overview (accessed Mar. 23, 2022).

[32] “Jama Connect | Collaboration Tool | SaaS Requirements Management.” https://www.jamasoftware.com/platform/jama-connect/ (accessed Mar. 23, 2022).

[33] “RAT – AUTHORING Tools.” https://www.reusecompany.com/rat-authoring-tools (accessed Mar. 23, 2022).

[34] J. Cowie and W. Lehnert, “Information extraction,” *Commun. ACM*, vol. 39, no. 1, pp. 80–91, 1996.

[35] E. Sneiders, J. Sjöbergh, and A. Alfalahi, “Automated email answering by text-pattern matching: Performance and error analysis,” *Expert Syst.*, vol. 35, no. 1, Feb. 2018, doi: 10.1111/EXSY.12251.

[36] “Software Themis - Corrector de lenguaje inclusivo.” https://themis.es/ (accessed Mar. 03, 2022).

[37] D. Jurafsky and J. H. Martin, *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft*. 2017.

[38] J. E. Hopcroft, R. Motwani, and J. D. Ullman, “Introduction to Automata Theory, Languages, and Computation, 2nd Edition,” *SIGACT News*, vol. 32, no. 1, pp. 60–65, Mar. 2001, doi: 10.1145/568438.568455.

[39] D. Mirman, “Mechanisms of Semantic Ambiguity Resolution: Insights from Speech Perception,” *Res. Lang. Comput.*, vol. 6, no. 3–4, pp. 293–309, 2008.

[40] F. Buschmann, K. Henney, and D. Schmidt, *Pattern oriented software architecture. On patterns and pattern languages (Vol 5)*, 1st editio. Chichester [England]: Wiley, 2007.

[41] C. Alexander, *A pattern language: towns, buildings, construction*. Oxford University Press, 1977.

[42] O. Hurtado, G. Génova, and A. Fraga, “Modelo de requisitos orientado al reúso efectivo (MORORE),” Universidad Carlos III, 2017. Accessed: Mar. 02, 2022. [Online]. Available: https://e-archivo.uc3m.es/handle/10016/25285

[43] E. Gamma *et al.*, “Design Patterns Elements of Reusable Object-Oriented Software,” 1995.

[44] A. Schweiger, “Applying software patterns to requirements engineering for avionics systems,” *Systems Conference (SysCon)*. IEEE International, 2013. doi: 10.1109/SysCon.2013.6549853.

[45] M. Kircher and P. Jain, *Pattern-oriented software architecture.: (Patterns for resource management)*, vol. 3. San Francisco, Calif: Wiley-Blackwell, 2005.

[46] C. Larman, *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*, 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR, c2005, 2005.

[47] K. Mens, I. Michiels, and R. Wuyts, "Supporting software development through declaratively codified programming patterns," *Expert Syst. Appl.*, vol. 23, no. 4, pp. 405–413, Nov. 2002, doi: 10.1016/S0957-4174(02)00076-3.

[48] R. Nystrom, *Game programming patterns*. Genever Benning, 2014.

[49] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals*, 2nd ed. Rocky Nook, 2015.

[50] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "EARS (Easy Approach to Requirements Syntax)," in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2009, pp. 317–322. doi: 10.1109/RE.2009.9.

[51] O. Daramola, G. Sindre, and T. Stalhane, "Pattern-based security requirements specification using ontologies and boilerplates," in *2012 2nd IEEE International Workshop on Requirements Patterns, RePa 2012 - Proceedings*, 2012, pp. 54–59. doi: 10.1109/REPA.2012.6359973.

[52] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-61073-3.

[53] S. Withall, *Software Requirement Patterns*. Redmond: Microsoft Press, 2010.

[54] E. Riloff, "Automatically generating extraction patterns from untagged text," in *Proceedings of the national conference on artificial intelligence*, 1996, pp. 1044–1049.

[55] E. Riloff, "An empirical study of automated dictionary construction for information extraction in three domains," *Artif. Intell.*, vol. 85, no. 1–2, pp. 101–134, 1996.

[56] C. P. Rehberg, "Automatic pattern generation in natural language processing." Google Patents, 2012.

[57] V. Moreno, P. M. Suárez, A. Fraga, J. Llorens, and E. Parra, "Método de generación de patrones semánticos," *Pct/es2013/070638, Issued*, 2013.

[58] J. Llorens, J. Morato, and G. Genova, "RSHP: an information representation model based on relationships," in *Soft computing in software engineering*, Springer, 2004, pp. 221–253.

[59] J. Llorens, J. Morato, and G. Genova, "RSHP: an information representation model based on relationships," pp. 221–253, 2004, doi: 10.1007/978-3-540-44405-3_8.

[60] A. Fraga, "A methodology for reusing any kind of knowledge: Universal Knowledge Reuse," Universidad Carlos III de Maderid, 2010.

[61] P. Suárez, V. Moreno, A. Fraga, and J. Llorens, "Automatic Generation of Semantic Patterns using Techniques of Natural Language Processing," *SKY 2013*, pp. 34–44, 2013.

[62] Melanie Mitchell, *An introduction to genetic algorithms*. MIT Press, 1998.

[63] V. Mallawaarachi, "Introduction to Genetic Algorithms — Including Example Code," *Toward Data Science Medium*, 2017. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3 (accessed Jul. 23, 2018).

[64] M. Khanali, S. Ahmadzadegan, M. Omid, F. Keyhani Nasab, and K. W. Chau, "Optimizing layout of wind farm turbines using genetic algorithms in Tehran province, Iran," *Int. J. Energy Environ. Eng.*, vol. 9, no. 4, pp. 399–411, Dec. 2018, doi: 10.1007/s40095-018-0280-x.

[65] P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 173–187, Jan. 2013, doi: 10.1016/J.INFSOF.2012.07.003.

[66] M. N. Karova, J. Petkova, and S. P. Penev, "Web Application of Traveling Salesman Problem using Genetic Algorithms," in *Proceedings of Papers, volume2, XLII International Scientific Conference on Information, Communication and energy Systems and Technologies ICEST* , 2007, pp. 24–27.

[67] E. Stoltz, "Evolution of a salesman: A complete genetic algorithm tutorial for Python," *Toward Data Science Medium*, 2018. https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35 (accessed Feb. 18, 2021).

[68] M. Gestal, "Introducción a los algoritmos genéticos." https://cursa.ihmc.us/rid=1KNKMJ4LN-11XXFSG-1KV5 (accessed Jul. 15, 2020).

[69] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evol. Comput.*, vol. 4, no. 4, pp. 361–394, 1996, doi: 10.1162/EVCO.1996.4.4.361.

[70] T. Blickle and L. Thiele, "A Mathematical Analysis of Tournament Selection," *ICGA*, vol. 95, pp. 9–15, 1995.

[71] S. Picek, M. Golub, and D. Jakobovic, "Evaluation of crossover operator performance in genetic algorithms with binary representation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6840 LNBI, pp. 223–230, 2011, doi: 10.1007/978-3-642-24553-4_31.

[72] E. K. Burke and K. Graham, "Genetic algorithms," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Second Edition., Springer, Ed. 2014, pp. 93–118. doi: 10.1007/978-1-4614-6940-7.

[73] J. R. Koza and R. Poli, "Genetic Programming," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques,* Second Edition., Springer, Ed. 2014, pp. 143–185. doi: 10.1007/978-1-4614-6940-7.

[74] J. Estefan, "Survey of model-based systems engineering (MBSE) methodologies. Differentiating Methodologies from Processes, Methods and lifecycle Models,"

*Citeseer*, 2008, doi: 10.1109/35.295942.

[75] M. Harsu, "A survey on domain engineering." https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.197.7897&rep=rep1&type=pdf (accessed Sep. 12, 2020).

[76] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[77] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Inf. Softw. Technol.*, vol. 67, pp. 180–195, Nov. 2015, doi: 10.1016/j.infsof.2015.07.006.

# 7 ANEXOS

## 7.1 Acrónimos

| Término | Significado |
|---|---|
| AC | Algoritmo de conquista – Algoritmos basados en la estrategia Separate-and-Conquer aplicable cuando se requieren varias soluciones para distintos componentes de un problema. Estos algoritmos se basan en bucles de búsqueda de soluciones parciales, marcado y separación de los componentes para los que se ha encontrado una solución, y continuación del bucle en la búsqueda de otras soluciones parciales. En esta investigación se han utilizado para marcar requisitos dentro de un RS para los que el algoritmo genético ha encontrado un patrón que los reconoce, almacenar dicho patrón en el RPS y continuación del bucle de búsqueda automática de patrones para los requisitos del RS para los que todavía no se ha encontrado un patrón. |
| AG | Algoritmo genético – Algoritmos inspirados en los principios de la selección natural y evolución de las especies de Darwin. Utilizados en esta investigación como mecanismo básico para la búsqueda automática de patrones que reconozcan a requisitos en un RS. |
| AvRm | *Average Requirements matched* – Métrica de calidad utilizada para evaluar la calidad de un RPS o comparar distintos RPS encontrados para un mismo RS. Es el promedio de requisitos reconocidos por los patrones de un RPS. |
| BPRm | *Best Pattern Requirements matched* – Métrica de calidad utilizada para evaluar la calidad de un RPS o comparar distintos RPS encontrados para un mismo RS. Es el número de requisitos reconocidos por el mejor patrón del RPS. |
| EARS | *Easy Approach to Requirements Syntax* – Uno de los primeros modelos o patrones para escribir requisitos basados en plantillas de propósito general que proporcionan estructuras sintácticas básicas para escribir los requisitos. |
| INCOSE | *International Council on Systems Engineering* – Organización sin ánimo de lucro, fundada en 1990, que tiene como objetivo el desarrollo de la Ingeniería de Sistemas. Tiene una red profesional de Ingenieros de Sistemas, realiza publicaciones, emite estándares y tiene |

| | |
|---|---|
| | certificaciones en relación a la Ingeniería de Sistemas. En esta investigación INCOSE ha colaborado proporcionando juegos de requisitos que se han utilizado como datos de entrada en la experimentación. |
| KM | *Knowledge Manager* – Componente de la SE *suite* que permite almacenar y gestionar el conocimiento: requisitos , vocabularios de dominios y grupos semánticos, etc. en una Base de Conocimiento común basada en el modelo de representación RSHP. |
| KR | *Knowledge Reuse Group* – Grupo de investigación de la Universidad Carlos III dentro del cual se ha realizado esta tesis. |
| MBSE | *Model Based System Engineering* – Metodología de ingeniería de sistemas basada en el uso formalizado de diagramas predefinidos para documentar todas las actividades de ingeniería de sistemas: diseño, análisis, verificación y validación, procesos, etc. La documentación clásica basada en documentos es sustituida o completada con la realización de modelos gráficos. Derivado de UML define 9 tipos de diagramas para la representación de sistemas. |
| MinRm | *Minimum number of Requirements Matched* – Parámetro de Twiga que define el número mínimo de requisitos que un patrón encontrado por el AG debe reconocer para ser considerado como solución parcial válida por el AC y ser por tanto utilizado para conquistar los requisitos reconocidos y para almacenarse dentro del RPS. Afecta al comportamiento general de Twiga y a la solución final encontrada: valores bajos de este parámetro generarán soluciones menos genéricas y RPSs de mayor volumen con patrones más específicos. |
| NLP | *Natural Language Processing* – Disciplina enmarcada dentro de las técnicas de inteligencia artificial en informática que estudia y desarrolla métodos para procesar, entender y analizar el lenguaje natural humano por parte de sistemas informáticos. |
| PMBoK | *Project Management Body of Knolwedge* – Conjunto de estándares que incluyen vocabulario, guías de actuación, definición de procesos, entradas y salidas de los mismos y métodos de trabajo para la gestión profesional de proyectos. Emitido por el Project Management Institute es un estándar de facto para la gestión de proyectos. Los profesionales de dirección de proyectos pueden obtener la certificación PMP proporcionada por el PMI que certifica conocimientos y experiencia en gestión de proyectos basados en PMBoK. |
| PMI | *Project Management Institute* – Asociación americana fundada en 1969 que promueve la profesión de gestión de proyectos. PMI ofrece |

| | |
|---|---|
| | estándares reconocidos globalmente para le gestión de porfolios y proyectos, como el PMBoK, así como formación, investigación y certificaciones relacionadas con la experiencia y conocimientos en gestión de proyectos. |
| RAT | *Requirements Authoring Tool* – Componente de la SE Suite de TRC que ayuda a la redacción de requisitos. A partir de patrones de requisitos y de un vocabulario de dominio, esta herramienta va analizando los requisitos conforme son escritos por los ingenieros validando si son acordes a algún patrón y si utilizan el vocabulario adecuado. Esta investigación tiene como finalidad la creación automática de patrones utilizados por RAT y por RQA para facilitar su implantación y para obtener mejores conjuntos de patrones. |
| RM | *Requirements matched* – Métrica de evaluación del rendimiento de un patrón determinado frente a un RS. Es la número de requisitos que el patrón reconoce del RS. |
| RPS | *Requirements Pattern Set* – Conjunto de patrones que reconocen todos o una parte mayoritaria (definida paramétricamente) de los requisitos de un RS. En esta investigación un RPS es una solución final generada por Twiga para un RS determinado. |
| RQA | *Requirements Quality Analysis – Quality Studio* – Componente de la SE Suite de TRC. Su función es analizar la calidad de un conjunto de requisitos, informando a los ingenieros sobre errores en la redacción y propuestas de mejora. Al igual que la herramienta RAT, utiliza patrones de requisitos y vocabularios de dominio para realizar el análisis. Esta investigación tiene como objeto la generación automática de patrones válidos para RAT y para RQA. |
| RS | *Requirements Set* – Conjunto homogéneo de requisitos de un dominio concreto. En esta investigación un RS es la unidad de experimentación, sobre la que se aplica la metodología para encontrar patrones de requisitos. |
| RSHP | Modelo de representación de información basado en relaciones. El modelo permite manejar todo tipo de componentes (artefactos): textuales, modelos de diseño, código, etc., utilizando el mismo modelo de representación y de datos. Es el modelo base para el almacenamiento de información de SE suite y ha sido seguido como modelo de datos en esta investigación, añadiendo exclusivamente entidades particulares para el funcionamiento de los algoritmos genéticos y de conquista propios de Twiga. |

| | |
|---|---|
| SE *suite* | *Systems Engineering Suite* – Conjunto de productos de software desarrollados por la empresa The Reuse Company. Estos productos implementan funcionalidades para la gestión de todo el ciclo de vida de la Ingeniería de Sistemas. Una parte de este ciclo de vida es la correspondiente a la Ingeniería de requisitos (autoría, verificación de la calidad, validación, trazabilidad, etc.). La suite utiliza patrones de requisitos para estas funcionalidades. Esta investigación se enmarca en este ámbito, proponiendo un método y desarrollando la herramienta Twiga de generación automática de estos patrones. |
| SM | *Tokens matched* – Métrica de evaluación del rendimiento de un patrón determinado frente a un RS. Es la número de total de tokens que reconoce un patrón en el RS. |
| TRC | *The Reuse Company* – Compañía de capital español, cuya misión es la generación de soluciones tecnológicas para el soporte y digitalización del ciclo de vida de la Ingeniería de Sistemas. Cuenta, entre sus productos comerciales, con el conjunto de herramientas *System Engineering Suite* que, entre otras capacidades, ofrece las de apoyo a la autoría y validación de calidad de requisitos de un sistema. Esta investigación propone una metodología y desarrolla una solución, Twiga, no comercial, que podrá incorporarse a la suite para automatizar la generación de patrones de requisitos, necesarios para el funcionamiento de estos componentes de autoría y de verificación de calidad de requisitos de SE Suite. |

## 7.2 Otros parámetros de Twiga

La herramienta Twiga ha sido parametrizada al máximo con el fin de poder modular su comportamiento para adaptarlo a distintos objetivos de experimentación. Aunque esta tesis se ha centrado exclusivamente en la obtención de patrones de requisitos válidos para ser incorporados en la SE *suite* para su uso en autoría de requisitos. La variación en los parámetros permitirá en futuras investigaciones diseñar otros escenarios de utilización (véase apartado □ de esta tesis). Se listan aquí otros parámetros de Twiga distintos a los mostrados en el apartado 3.3.7 de esta tesis) que han sido los que se han utilizado para ajustar los experimentos de esta investigación.

```
#region REQUIREMENTS AND REQUERIMENTS TOKENS

public static int NOOFREQS = 100;                    //   Número de requisitos para un experimento
public static int NOOFTOTALREQS = 0;                 //   Número total de requisitos del escenario
public static int OFFSET = 0;                        //   Twiga 3 - para cargar subconjuntos de
                                                          requisitos del escenario
public static int NOMAXTOKENS_OF_REQS = 12;          //   Max number of tokens to load for a given
                                                          requirement
                                                     //   JUN20 - only if LIMIT_NOTOKENS_OF_REQS
                                                          is set to true
public static bool LIMIT_NOTOKENS_OF_REQS = true;    //   S = true limita el número máximo de
                                                          tokens de los requisitos
#endregion

#region ARTIFICIALREQUIREMENTS

public static int ARTIFICIAL_AFFINITY = 1;           //  Max number of tokens to load for a given
                                                         requirement
public static int ARTIFICIAL_OPTIONAL = 25;          //    % de aplicación de opcional (100 -> no
                                                         se considera opcional)
public static int NOMINTOKENS_OF_REQS = 4;           //   SEP21 número mínimo de tokens cuando se
                                                         crean requisitos artificiales


#endregion

#region PATTERNS POPULATION

public static int NOPATTERNPOPMEMBERS = 80;          //   Número de individuos de las poblaciones
                                                         de patrones
public static int MAXTOKENS_OF_PATTERNS = 13;        //   Número máximo de tokens de los patrones
public static bool PATTERNFIXEDNOSLOTS = true;       //   Si = true - todos los patrones tienen
                                                         núm. fijo slots.
                                                     //   Si = false - Número variable de slots
                                                         para cada patrón
public static bool USEFREQ = true;                   //   Si = true los tokens de la BBDD de alelos
                                                         se seleccionan conforme a su frecuencia
                                                         (ver Pattern.getRandomTag)
public static int LASTPTID = 1;                      //   Initial Pattern ID

#endregion

#region ALGORITMO GENETICO AG

public static int POOL = 30;                         //   Tamaño del Pool de reproducción
```

```
public static int TOURNEYS = 4;                    //  Torneos a realizar en el proceso de
                                                       selección
public static int MUTLEVEL = 25;                   //  Frecuencia de mutación usada en el
                                                       proceso de mutación
public static int GENERATIONS = 300;               //  Número máximo de generaciones
public static int MAXGENFAILED = 10;               //  SEP 21 se limita el Número de
                                                       generaciones sin mejorar el fitness de
                                                       la población se alcanza esta valor sin
                                                       haber mejorado el fitness el algoritmo
                                                       genético para las generaciones

public static decimal MINFITNESSVAR = 10;          //  Si en la generación analizada la
                                                       variación de fitness es menor se
                                                       considera fallida.
                                                   //
public static bool AVOIDSELECTREPEATED = true;     //  En el proceso de selección. Si = true,
                                                       un mismo individuo no se puede volver a
                                                       seleccionar para el Pool

public static bool AVOIDCLONES = true;             //  Si = true, no se admiten individuos
                                                       clones en una población
#endregion

#region ALGORITMO DE CONQUISTA AC

public static int CONQUER_ITERATIONS = 50;         //  Número máximo de iteraciones de
                                                       conquista.
public static int MAXITERFAILED = 5;               //  SEP21 limita el Número de iteraciones.
                                                       Si se alcanza el valor del parámetro sin
                                                       haber conquistado nada AC se detiene
public static int MINREQSMATCHED = 2;              //  número mínimo de requisitos que machea
                                                       un patrón para seleccionarlo como patrón
                                                       bueno en el proceso de conquista


#endregion

#region FITNESS ALGORITHM PARAMS

public static decimal FACTOR_TK_MATCHED_GOODREQ =  //  Factores multiplicadores para cálculo de
(decimal)1.0;                                          fitness
public static decimal FACTOR_RQ_MATCHED_GOODREQ =
(decimal)1000.0;
public static decimal FACTOR_TK_MATCHED_BADREQ =
(decimal)0;
```

```
public static decimal FACTOR_RQ_MATCHED_BADREQ =
(decimal)0;

#endregion

#region WILDCARD MANAGEMENT

public static int WILDCARDSPROB = 20;              //   Probabilidad de que un token sea un
                                                        token comodín (0-nada)
public static int AFFINITY = 3;                    //   Número de tokens de requisitos
                                                        reconocidos por un token comodín de un
                                                        patrón
public static double AFF_PORCENTAJE = 0.2;         //   Para adecuar AFFINITY a la longitud
                                                        total de los requisitos
public static bool SEMANTIC_SENSIBILITY = false;   //   Si = true, los tokens comodín no pueden
                                                        reconocer tokens semánticos de
                                                        requisitos

#endregion
```

## 7.3 Registros de los experimentos

### *7.3.1 Tabla resumen de resultados de los Experimentos*

| | | | Parámetros principales | | | | | | | | | **Resultados** | | Métricas Calidad | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ExpID | RS | Núm. Req. | Afinidad | Generaciones AG | MinRM | Iteraciones de conquista AC | Probabilidad mutación | Tamaño de la población de patrones | Pool | Tamaño máximo de patrón | Torneo | Patrones encontrados RPS | Requisitos reconocidos | AvRm Promedio de requisitos | %Rm |
| 1 | Hwsw_1 | 45 | 4 | 300 | 2 | 30 | 0,25 | 80 | 20 | 13 | 3 | 9 | 41 | 4,6 | 91% |
| 2 | Funct_Gen1 | 34 | 3 | 300 | 2 | 100 | 0,25 | 80 | 30 | 13 | 3 | 7 | 25 | 3,6 | 74% |
| 3 | Funct_SMG | 77 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 20 | 73 | 3,7 | 95% |
| 4 | Funct_STS | 47 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 14 | 47 | 3,4 | 100% |
| 5 | Funct_Gen2 | 101 | 3 | 300 | 2 | 100 | 0,25 | 80 | 30 | 13 | 3 | 22 | 93 | 4,2 | 92% |
| 6 | Funct_QAR | 60 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 10 | 52 | 5,2 | 87% |
| 7 | Funct_QM | 65 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 14 | 58 | 4,1 | 89% |
| 8 | Funct_SMG | 75 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 12 | 54 | 4,5 | 72% |
| 9 | Hwsw_2 | 23 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 5 | 17 | 3,4 | 74% |

| | | | Parámetros principales | | | | | | | | | Resultados | | Métricas Calidad | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ExpID | RS | Núm. Req. | Afinidad | Generaciones AG | MinRM | Iteraciones de conquista AC | Probabilidad mutación | Tamaño de la población de patrones | Pool | Tamaño máximo de patrón | Torneo | Patrones encontrados RPS | Requisitos reconocidos | AvRm Promedio de requisitos | %Rm |
| 10 | SEC | 18 | 3 | 300 | 2 | 50 | 0,25 | 80 | 30 | 13 | 3 | 4 | 12 | 3,0 | 67% |

### *7.3.2 Tabla de patrones encontrados*

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Hwsw_1 | 759 | 27 | 1 | 18 | [1144] | * | 1144 | * | 1144 | [1144] | [1144] | * | 1144 | * | [1144] | * | * |
| 1 | Hwsw_1 | 1985 | 38 | 2 | 7 | [1144] | * | [1119] | [1224] | [1144] | 1108 | [1151] | * | 1144 | * | 1119 | | |
| 1 | Hwsw_1 | 2803 | 26 | 3 | 3 | * | [1144] | [1131] | 1130 | * | 1144 | * | [1130] | 1144 | [1119] | * | 1108 | |
| 1 | Hwsw_1 | 3339 | 11 | 4 | 2 | * | 1144 | 1110 | 1144 | [1151] | 1144 | [1193] | [1119] | * | 1144 | * | 1144 | 1144 |
| 1 | Hwsw_1 | 5686 | 25 | 6 | 3 | [1144] | [1224] | [1130] | * | 1151 | [1119] | * | 1144 | [1203] | * | 1119 | * | [1144] |
| 1 | Hwsw_1 | 6759 | 35 | 7 | 2 | * | * | 1144 | * | 1144 | * | [1151] | [1110] | [1110] | 1119 | * | [1144] | 1230 |
| 1 | Hwsw_1 | 9814 | 38 | 10 | 2 | [1224] | 1144 | * | 1228 | * | [1076] | * | 1144 | [1144] | * | * | [1144] | 1108 |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Hwsw_1 | 11243 | 34 | 12 | 2 | [1165] | [1144] | * | 1110 | 1144 | [1144] | [1108] | 1130 | * | [1152] | * | [1119] | 1144 |
| 1 | Hwsw_1 | 12565 | 41 | 13 | 2 | [1144] | * | [1151] | [1119] | 1130 | [1224] | [1144] | [1119] | * | 1119 | * | 1103 | [1144] |
| 2 | Funct_Gen1 | 1868 | 48 | 1 | 5 | [1119] | * | 1119 | [1119] | [1144] | * | 1119 | [1130] | [1230] | * | [1213] | 1144 | * |
| 2 | Funct_Gen1 | 4676 | 71 | 2 | 4 | [1224] | [1103] | * | 1144 | [1224] | * | 1144 | * | [1144] | 1119 | * | 1224 | * |
| 2 | Funct_Gen1 | 7535 | 61 | 3 | 4 | [1224] | [1119] | [1144] | [1130] | * | [1108] | 1224 | * | 1144 | [1119] | * | 1144 | * |
| 2 | Funct_Gen1 | 11347 | 92 | 4 | 4 | 1224 | [1119] | [1132] | * | 1229 | * | 1119 | * | [1224] | * | 1144 | [1119] | * |
| 2 | Funct_Gen1 | 13491 | 45 | 5 | 2 | [1224] | [1119] | [1144] | * | 1130 | * | [1229] | 1144 | * | [1119] | 1144 | [1224] | * |
| 2 | Funct_Gen1 | 16158 | 41 | 6 | 4 | [1224] | * | 1130 | * | 1103 | * | 1144 | [1119] | [1108] | * | [1126] | 1144 | * |
| 2 | Funct_Gen1 | 25998 | 35 | 9 | 2 | 1224 | [1144] | [1119] | * | 1119 | [1144] | * | 1144 | * | [1108] | 1119 | * | 1224 |
| 3 | Funct_SMG | 2117 | 55 | 1 | 9 | * | [1224] | 1119 | 1130 | [1130] | [1144] | [1158] | * | 1144 | * | 1144 | * | 1119 |
| 3 | Funct_SMG | 4172 | 49 | 2 | 6 | [1224] | [1224] | 1119 | [1119] | 1132 | * | [1119] | 1144 | [1119] | [1237] | * | 1119 | * |
| 3 | Funct_SMG | 5495 | 29 | 3 | 12 | [1224] | * | [1103] | 1229 | * | [1144] | [1108] | * | 1144 | * | [1108] | 1144 | * |
| 3 | Funct_SMG | 8694 | 47 | 4 | 6 | [1224] | [1119] | * | 1119 | * | 1144 | [1119] | * | 1144 | * | [1144] | 1144 | * |
| 3 | Funct_SMG | 10835 | 45 | 5 | 4 | 1224 | * | 1130 | * | 1144 | * | * | * | 1144 | * | 1144 | [1144] | * |
| 3 | Funct_SMG | 14635 | 45 | 7 | 2 | 1224 | [1119] | [1130] | * | 1144 | [1130] | * | [1144] | 1119 | * | [1224] | [1224] | 1119 |
| 3 | Funct_SMG | 17814 | 57 | 8 | 2 | 1224 | [1224] | 1119 | [1130] | [1248] | * | 1119 | * | 1119 | [1224] | [1130] | * | 1119 |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | Funct_SMG | 21101 | 17 | 11 | 3 | * | [1119] | 1224 | 1119 | [1119] | * | 1229 | * | 1119 | * | 1119 | 1119 | 1144 |
| 3 | Funct_SMG | 24536 | 35 | 13 | 3 | 1228 | * | 1119 | * | 1130 | * | * | [1119] | 1144 | * | 1144 | [1119] | 1144 |
| 3 | Funct_SMG | 29655 | 53 | 17 | 2 | 1224 | 1119 | [1108] | [1144] | * | [1103] | 1224 | [1144] | * | [1224] | 1108 | * | 1108 |
| 3 | Funct_SMG | 36718 | 50 | 21 | 2 | 1224 | 1119 | [1130] | * | 1144 | * | [1144] | 1103 | [1108] | * | 1119 | * | * |
| 3 | Funct_SMG | 39426 | 55 | 22 | 4 | 1224 | [1119] | * | 1224 | * | * | [1103] | [1223] | [1228] | 1158 | * | 1119 | * |
| 3 | Funct_SMG | 42131 | 68 | 23 | 2 | 1224 | 1119 | [1130] | * | 1103 | * | [1119] | 1144 | * | [1144] | [1224] | 1119 | [1094] |
| 3 | Funct_SMG | 49422 | 54 | 28 | 2 | 1224 | 1119 | * | [1130] | * | * | 1229 | [1119] | * | [1119] | 1144 | * | 1144 |
| 3 | Funct_SMG | 52008 | 64 | 29 | 2 | [1224] | [1223] | 1119 | [1108] | [1130] | * | 1119 | * | * | 1119 | * | 1144 | 1144 |
| 3 | Funct_SMG | 55256 | 83 | 30 | 3 | [1224] | * | 1119 | [1108] | * | 1224 | * | 1144 | [1144] | * | [1119] | [1224] | 1103 |
| 3 | Funct_SMG | 62450 | 41 | 34 | 2 | * | [1224] | 1119 | * | [1144] | 1119 | * | * | [1119] | 1108 | * | [1108] | 1224 |
| 3 | Funct_SMG | 66057 | 34 | 36 | 2 | [1224] | * | 1158 | * | [1229] | 1223 | [1144] | * | [1224] | 1119 | [1110] | * | * |
| 3 | Funct_SMG | 83792 | 27 | 45 | 2 | * | [1224] | [1144] | [1224] | * | 1108 | [1144] | * | 1119 | * | 1213 | * | 1224 |
| 3 | Funct_SMG | 87305 | 46 | 47 | 3 | [1224] | 1119 | [1144] | * | * | [1103] | 1229 | * | 1108 | [1228] | [1108] | * | 1144 |
| 4 | Funct_STS | 1317 | 33 | 1 | 14 | 1224 | 1119 | * | 1229 | [1119] | * | 1144 | * | [1119] | * | 1108 | [1229] | * |
| 4 | Funct_STS | 3166 | 37 | 2 | 4 | * | 1119 | [1130] | [1144] | * | 1224 | * | 1144 | * | [1119] | 1144 | * | 1144 |
| 4 | Funct_STS | 7025 | 47 | 5 | 4 | 1224 | 1119 | 1130 | * | 1229 | * | 1108 | [1119] | [1158] | * | 1144 | [1223] | * |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | Funct_STS | 8613 | 39 | 6 | 2 | * | [1224] | 1119 | 1130 | * | 1144 | * | [1144] | 1119 | * | [1119] | 1119 | 1144 |
| 4 | Funct_STS | 11233 | 53 | 7 | 3 | [1224] | [1158] | [1119] | * | [1119] | [1132] | 1144 | * | 1224 | * | 1224 | * | 1119 |
| 4 | Funct_STS | 12941 | 39 | 8 | 2 | 1224 | * | 1119 | * | [1159] | 1224 | 1144 | * | 1151 | * | 1224 | 1144 | [1224] |
| 4 | Funct_STS | 13892 | 6 | 9 | 3 | * | 1119 | * | [1224] | 1119 | * | [1226] | 1119 | * | [1224] | * | [1132] | * |
| 4 | Funct_STS | 18130 | 60 | 10 | 4 | * | 1119 | * | 1097 | * | 1144 | * | 1144 | * | [1144] | [1108] | * | 1144 |
| 4 | Funct_STS | 22450 | 35 | 14 | 2 | 1224 | [1119] | [1130] | * | [1103] | 1229 | * | 1119 | * | 1119 | * | [1110] | 1144 |
| 4 | Funct_STS | 26903 | 34 | 16 | 2 | [1224] | 1119 | [1119] | [1144] | [1130] | * | 1144 | 1144 | [1144] | * | [1119] | 1144 | * |
| 4 | Funct_STS | 31233 | 42 | 18 | 2 | [1224] | * | [1119] | * | 1230 | [1224] | * | 1224 | * | 1119 | [1224] | [1144] | 1132 |
| 4 | Funct_STS | 32598 | 23 | 19 | 2 | * | 1119 | [1119] | [1229] | * | 1040 | * | [1229] | 1224 | 1119 | * | 1119 | [1139] |
| 4 | Funct_STS | 36396 | 30 | 22 | 2 | [1224] | [1119] | [1130] | [1158] | * | [1103] | 1229 | * | 1139 | * | 1119 | * | 1108 |
| 4 | Funct_STS | 38368 | 36 | 23 | 1 | 1224 | 1119 | * | 1139 | * | [1144] | * | [1130] | [1108] | 1110 | * | 1139 | 1119 |
| 5 | Funct_Gen2 | 1924 | 50 | 1 | 17 | [1224] | [1119] | [1132] | * | 1224 | * | [1144] | [1119] | * | 1144 | * | 1224 | * |
| 5 | Funct_Gen2 | 4555 | 57 | 2 | 12 | [1224] | [1119] | [1224] | * | 1119 | * | 1119 | * | 1224 | * | 1108 | [1144] | [1119] |
| 5 | Funct_Gen2 | 6750 | 51 | 3 | 4 | 1224 | [1144] | * | 1119 | [1224] | * | [1119] | 1144 | * | [1224] | [1144] | 1119 | * |
| 5 | Funct_Gen2 | 8862 | 41 | 4 | 8 | * | 1119 | [1132] | * | * | 1144 | * | [1119] | 1144 | * | 1119 | * | |
| 5 | Funct_Gen2 | 11560 | 40 | 6 | 3 | [1223] | [1224] | * | [1224] | 1119 | * | 1230 | * | 1119 | * | [1224] | [1119] | 1108 |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | Funct_Gen2 | 14767 | 80 | 7 | 5 | [1224] | [1224] | * | 1119 | * | [1144] | 1119 | [1224] | * | 1144 | * | 1144 | * |
| 5 | Funct_Gen2 | 16781 | 40 | 8 | 3 | [1224] | [1139] | [1144] | * | 1119 | * | [1224] | * | 1224 | * | 1108 | * | 1119 |
| 5 | Funct_Gen2 | 19969 | 59 | 9 | 3 | * | 1119 | [1119] | * | 1119 | [1144] | * | 1108 | [1119] | [1132] | * | [1224] | 1144 |
| 5 | Funct_Gen2 | 23370 | 65 | 10 | 3 | [1224] | [1119] | [1130] | * | 1224 | * | [1224] | 1119 | * | 1119 | * | 1144 | * |
| 5 | Funct_Gen2 | 27476 | 93 | 11 | 5 | [1224] | * | [1144] | 1119 | [1224] | * | [1119] | [1144] | 1229 | * | 1144 | [1144] | * |
| 5 | Funct_Gen2 | 32395 | 61 | 13 | 3 | 1224 | [1144] | 1119 | [1132] | * | 1108 | [1144] | * | 1144 | [1108] | [1119] | * | 1108 |
| 5 | Funct_Gen2 | 34789 | 58 | 14 | 3 | [1224] | [1119] | [1132] | * | [1132] | 1229 | * | [1224] | 1119 | [1236] | * | 1224 | 1144 |
| 5 | Funct_Gen2 | 37697 | 59 | 15 | 4 | [1224] | [1119] | * | 1119 | * | 1224 | * | [1144] | [1119] | * | [1110] | 1229 | * |
| 5 | Funct_Gen2 | 41582 | 50 | 18 | 3 | [1156] | * | [1224] | * | 1144 | * | [1110] | 1224 | * | 1144 | * | 1158 | * |
| 5 | Funct_Gen2 | 55413 | 63 | 23 | 2 | 1224 | [1224] | [1230] | 1119 | * | 1119 | * | [1144] | 1248 | [1119] | * | [1144] | 1108 |
| 5 | Funct_Gen2 | 63659 | 39 | 28 | 2 | * | * | [1228] | [1119] | 1230 | * | 1224 | 1119 | * | [1119] | [1144] | 1108 | [1119] |
| 5 | Funct_Gen2 | 72053 | 72 | 32 | 2 | [1224] | * | 1119 | * | [1144] | 1230 | * | 1144 | [1108] | * | 1144 | * | 1144 |
| 5 | Funct_Gen2 | 88358 | 51 | 40 | 2 | 1224 | 1119 | [1132] | * | [1108] | 1119 | [1237] | * | 1158 | * | [1144] | 1237 | * |
| 5 | Funct_Gen2 | 97810 | 63 | 45 | 2 | 1224 | [1119] | 1130 | * | 1144 | * | [1230] | [1119] | [1119] | * | [1119] | 1144 | * |
| 5 | Funct_Gen2 | 99966 | 48 | 46 | 3 | * | 1144 | * | [1103] | 1119 | * | 1144 | [1144] | * | 1108 | 1119 | [1095] | * |
| 5 | Funct_Gen2 | 107254 | 44 | 50 | 2 | [1224] | [1119] | * | 1132 | * | [1224] | 1103 | * | [1108] | [1119] | 1166 | * | 1103 |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | Funct_Gen2 | 110756 | 68 | 51 | 2 | [1224] | 1119 | * | [1130] | [1230] | 1144 | * | [1108] | 1158 | * | [1119] | 1248 | * |
| 6 | Funct_QAR | 1604 | 42 | 1 | 9 | 1224 | [1248] | [1119] | * | 1119 | * | 1144 | [1108] | [1224] | * | 1119 | [1144] | * |
| 6 | Funct_QAR | 3734 | 50 | 2 | 11 | [1224] | 1119 | * | 1103 | * | [1138] | [1144] | [1119] | 1144 | * | 1119 | * | * |
| 6 | Funct_QAR | 5683 | 48 | 3 | 7 | 1224 | [1119] | 1130 | * | 1224 | [1229] | [1144] | [1229] | * | 1119 | * | 1119 | * |
| 6 | Funct_QAR | 8072 | 52 | 4 | 7 | [1224] | * | 1119 | * | [1119] | 1144 | * | 1144 | * | 1144 | [1144] | * | 1119 |
| 6 | Funct_QAR | 10185 | 53 | 5 | 2 | [1224] | [1119] | * | [1119] | [1229] | 1144 | [1119] | * | [1119] | 1229 | * | [1103] | 1144 |
| 6 | Funct_QAR | 12507 | 34 | 6 | 7 | 1224 | [1119] | * | 1229 | * | 1119 | * | [1119] | * | [1119] | 1144 | [1144] | * |
| 6 | Funct_QAR | 15435 | 44 | 8 | 3 | 1224 | 1119 | [1144] | [1119] | * | [1144] | 1229 | * | 1224 | [1144] | * | * | 1144 |
| 6 | Funct_QAR | 24014 | 44 | 13 | 2 | [1224] | [1119] | * | 1144 | 1144 | * | [1119] | 1144 | 1144 | * | [1103] | 1144 | * |
| 6 | Funct_QAR | 28909 | 45 | 16 | 2 | * | [1224] | * | [1130] | 1144 | [1110] | * | 1230 | [1103] | * | * | [1224] | 1119 |
| 6 | Funct_QAR | 31924 | 36 | 18 | 2 | [1132] | [1224] | [1119] | * | 1144 | * | [1224] | 1119 | * | 1119 | * | 1119 | * |
| 7 | Funct_QM | 1778 | 46 | 1 | 9 | [1224] | 1119 | [1131] | * | [1119] | 1144 | * | 1119 | * | 1165 | [1224] | * | 1144 |
| 7 | Funct_QM | 3441 | 41 | 2 | 6 | * | [1224] | [1224] | 1119 | * | 1119 | * | 1144 | [1119] | * | 1144 | 1144 | 1144 |
| 7 | Funct_QM | 6143 | 67 | 3 | 9 | * | 1230 | * | * | [1119] | 1224 | [1119] | [1223] | [1144] | [1144] | * | 1119 | * |
| 7 | Funct_QM | 8357 | 45 | 4 | 6 | 1224 | * | 1119 | * | 1144 | * | [1119] | 1144 | [1130] | * | 1144 | * | * |
| 7 | Funct_QM | 11127 | 34 | 6 | 4 | [1224] | [1119] | 1130 | * | [1229] | 1144 | * | 1144 | * | [1144] | 1144 | [1108] | * |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | Funct_QM | 13356 | 43 | 7 | 3 | 1224 | 1119 | [1130] | * | [1119] | 1224 | * | 1144 | * | 1119 | * | [1224] | 1229 |
| 7 | Funct_QM | 15346 | 49 | 8 | 4 | [1224] | [1119] | * | [1224] | * | [1103] | [1144] | 1119 | [1223] | [1119] | * | 1108 | * |
| 7 | Funct_QM | 19016 | 51 | 11 | 3 | [1165] | [1224] | * | * | [1130] | 1230 | [1144] | * | 1144 | * | 1119 | * | 1119 |
| 7 | Funct_QM | 24304 | 38 | 15 | 4 | [1224] | * | 1119 | [1131] | [1094] | * | [1144] | 1229 | [1144] | * | [1224] | 1119 | * |
| 7 | Funct_QM | 42612 | 79 | 23 | 2 | [1224] | 1119 | [1130] | [1230] | * | [1144] | 1223 | * | [1144] | 1119 | * | [1224] | 1144 |
| 7 | Funct_QM | 44448 | 45 | 24 | 2 | * | 1224 | 1119 | [1131] | * | [1224] | 1144 | * | * | [1144] | [1119] | 1193 | * |
| 7 | Funct_QM | 47338 | 29 | 27 | 2 | [1144] | * | 1144 | * | 1119 | * | [1230] | 1119 | [1151] | * | 1110 | * | 1222 |
| 7 | Funct_QM | 50967 | 58 | 29 | 2 | * | [1224] | 1119 | * | 1108 | * | [1119] | 1224 | * | [1224] | [1144] | [1119] | * |
| 7 | Funct_QM | 57325 | 43 | 34 | 2 | [1224] | [1119] | * | 1103 | [1229] | * | [1119] | [1224] | 1144 | [1144] | * | [1223] | 1119 |
| 8 | Funct_SMG | 2742 | 71 | 1 | 11 | [1224] | [1119] | * | 1130 | [1144] | * | 1229 | * | [1119] | * | 1144 | * | 1151 |
| 8 | Funct_SMG | 5273 | 43 | 3 | 12 | * | 1130 | [1230] | * | 1144 | * | [1229] | 1144 | * | 1144 | * | [1144] | 1119 |
| 8 | Funct_SMG | 6995 | 37 | 4 | 3 | 1224 | [1224] | 1119 | * | 1119 | [1223] | * | 1119 | * | 1144 | * | 1144 | * |
| 8 | Funct_SMG | 9184 | 46 | 5 | 3 | [1224] | [1144] | [1119] | [1119] | * | 1230 | 1103 | * | 1119 | * | [1144] | 1119 | 1119 |
| 8 | Funct_SMG | 10852 | 37 | 6 | 6 | [1224] | * | 1119 | [1130] | * | 1108 | * | 1119 | [1248] | * | [1119] | * | 1144 |
| 8 | Funct_SMG | 13091 | 48 | 7 | 5 | * | 1119 | * | [1119] | 1144 | * | 1119 | [1094] | * | [1119] | 1197 | * | 1144 |
| 8 | Funct_SMG | 15318 | 53 | 8 | 2 | [1230] | [1119] | [1224] | * | [1119] | * | [1144] | 1229 | * | 1229 | * | 1144 | 1119 |

| ExpID | RS | Patt ID | Generación AG | Iteración AC | Núm. Req. Reconocidos | tk0 | tk1 | tk2 | tk3 | tk4 | tk5 | tk6 | tk7 | tk8 | tk9 | tk10 | tk11 | tk12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Funct_SMG | 16969 | 40 | 9 | 2 | [1224] | * | 1119 | 1130 | * | [1119] | [1224] | [1144] | 1119 | * | 1119 | [1110] | * |
| 8 | Funct_SMG | 21201 | 63 | 11 | 3 | [1224] | [1119] | * | 1119 | * | * | [1119] | 1144 | * | 1144 | * | 1119 | * |
| 8 | Funct_SMG | 23087 | 48 | 12 | 2 | [1042] | [1224] | * | 1119 | * | [1233] | 1230 | * | 1144 | * | 1119 | * | 1144 |
| 8 | Funct_SMG | 25386 | 50 | 13 | 3 | [1224] | [1144] | [1119] | * | 1144 | * | * | 1144 | * | * | 1119 | * | 1131 |
| 8 | Funct_SMG | 28187 | 48 | 15 | 2 | [1224] | * | [1224] | 1119 | * | 1144 | [1213] | [1119] | * | [1224] | [1144] | [1086] | 1119 |
| 9 | Hwsw_2 | 1690 | 44 | 1 | 5 | [1119] | * | [1224] | * | 1144 | * | [1144] | * | 1144 | 1144 | * | 1110 | * |
| 9 | Hwsw_2 | 3405 | 42 | 2 | 5 | [1119] | [1224] | [1144] | [1119] | [1144] | * | 1224 | [1144] | * | [1224] | 1144 | * | 1119 |
| 9 | Hwsw_2 | 5144 | 30 | 3 | 3 | 1224 | * | [1130] | 1230 | * | 1213 | * | 1119 | * | 1119 | * | [1144] | 1119 |
| 9 | Hwsw_2 | 7488 | 48 | 4 | 2 | * | [1224] | [1119] | * | 1119 | * | [1230] | [1108] | * | 1144 | [1144] | * | 1119 |
| 9 | Hwsw_2 | 11211 | 43 | 6 | 2 | [1144] | [1224] | * | [1119] | 1230 | * | 1119 | [1230] | * | [1119] | [1144] | * | 1108 |
| 10 | SEC | 1047 | 26 | 1 | 5 | [1224] | [1119] | * | 1144 | * | [1229] | 1144 | * | [1144] | 1119 | * | 1144 | * |
| 10 | SEC | 4044 | 58 | 2 | 2 | * | 1119 | * | 1144 | * | [1042] | [1130] | [1144] | 1119 | * | 1130 | * | 1144 |
| 10 | SEC | 14874 | 70 | 9 | 3 | [1224] | * | [1144] | 1130 | [1230] | * | 1229 | * | [1144] | 1119 | * | 1144 | |
| 10 | SEC | 21001 | 64 | 12 | 2 | [1224] | [1119] | [1144] | [1130] | 1230 | * | 1108 | [1213] | * | 1144 | * | 1144 | * |

### *7.3.3 Tabla de requisitos reconocidos*

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 1 | Hwsw_1 | 759 | 1504 |
| 1 | Hwsw_1 | 759 | 1507 |
| 1 | Hwsw_1 | 759 | 1508 |
| 1 | Hwsw_1 | 759 | 1511 |
| 1 | Hwsw_1 | 759 | 1512 |
| 1 | Hwsw_1 | 759 | 1515 |
| 1 | Hwsw_1 | 759 | 1516 |
| 1 | Hwsw_1 | 759 | 1518 |
| 1 | Hwsw_1 | 759 | 1521 |
| 1 | Hwsw_1 | 759 | 1522 |
| 1 | Hwsw_1 | 759 | 1523 |
| 1 | Hwsw_1 | 759 | 1524 |
| 1 | Hwsw_1 | 759 | 1525 |
| 1 | Hwsw_1 | 759 | 1526 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 1 | Hwsw_1 | 759 | 1536 |
| 1 | Hwsw_1 | 759 | 1538 |
| 1 | Hwsw_1 | 759 | 1543 |
| 1 | Hwsw_1 | 759 | 1547 |
| 1 | Hwsw_1 | 1985 | 1505 |
| 1 | Hwsw_1 | 1985 | 1530 |
| 1 | Hwsw_1 | 1985 | 1531 |
| 1 | Hwsw_1 | 1985 | 1532 |
| 1 | Hwsw_1 | 1985 | 1533 |
| 1 | Hwsw_1 | 1985 | 1534 |
| 1 | Hwsw_1 | 1985 | 1535 |
| 1 | Hwsw_1 | 2803 | 1510 |
| 1 | Hwsw_1 | 2803 | 1527 |
| 1 | Hwsw_1 | 2803 | 1539 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 1 | Hwsw_1 | 3339 | 1517 |
| 1 | Hwsw_1 | 3339 | 1520 |
| 1 | Hwsw_1 | 5686 | 1528 |
| 1 | Hwsw_1 | 5686 | 1529 |
| 1 | Hwsw_1 | 5686 | 1537 |
| 1 | Hwsw_1 | 6759 | 1506 |
| 1 | Hwsw_1 | 6759 | 1542 |
| 1 | Hwsw_1 | 9814 | 1540 |
| 1 | Hwsw_1 | 9814 | 1541 |
| 1 | Hwsw_1 | 11243 | 1513 |
| 1 | Hwsw_1 | 11243 | 1514 |
| 1 | Hwsw_1 | 12565 | 1544 |
| 1 | Hwsw_1 | 12565 | 1546 |
| 2 | Funct_Gen1 | 1868 | 1559 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 2 | Funct_Gen1 | 1868 | 2035 |
| 2 | Funct_Gen1 | 1868 | 2041 |
| 2 | Funct_Gen1 | 1868 | 2042 |
| 2 | Funct_Gen1 | 1868 | 2043 |
| 2 | Funct_Gen1 | 4676 | 1560 |
| 2 | Funct_Gen1 | 4676 | 2038 |
| 2 | Funct_Gen1 | 4676 | 2039 |
| 2 | Funct_Gen1 | 4676 | 2040 |
| 2 | Funct_Gen1 | 7535 | 1555 |
| 2 | Funct_Gen1 | 7535 | 1556 |
| 2 | Funct_Gen1 | 7535 | 1563 |
| 2 | Funct_Gen1 | 7535 | 2037 |
| 2 | Funct_Gen1 | 11347 | 2044 |
| 2 | Funct_Gen1 | 11347 | 2046 |
| 2 | Funct_Gen1 | 11347 | 2047 |
| 2 | Funct_Gen1 | 11347 | 2048 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 2 | Funct_Gen1 | 13491 | 2030 |
| 2 | Funct_Gen1 | 13491 | 2045 |
| 2 | Funct_Gen1 | 16158 | 1549 |
| 2 | Funct_Gen1 | 16158 | 2033 |
| 2 | Funct_Gen1 | 16158 | 2034 |
| 2 | Funct_Gen1 | 16158 | 2036 |
| 2 | Funct_Gen1 | 25998 | 1553 |
| 2 | Funct_Gen1 | 25998 | 1558 |
| 3 | Funct_SMG | 2117 | 1565 |
| 3 | Funct_SMG | 2117 | 1597 |
| 3 | Funct_SMG | 2117 | 1609 |
| 3 | Funct_SMG | 2117 | 1610 |
| 3 | Funct_SMG | 2117 | 1619 |
| 3 | Funct_SMG | 2117 | 1620 |
| 3 | Funct_SMG | 2117 | 1621 |
| 3 | Funct_SMG | 2117 | 1635 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 3 | Funct_SMG | 2117 | 1636 |
| 3 | Funct_SMG | 4172 | 1571 |
| 3 | Funct_SMG | 4172 | 1575 |
| 3 | Funct_SMG | 4172 | 1581 |
| 3 | Funct_SMG | 4172 | 1588 |
| 3 | Funct_SMG | 4172 | 1594 |
| 3 | Funct_SMG | 4172 | 1595 |
| 3 | Funct_SMG | 5495 | 1577 |
| 3 | Funct_SMG | 5495 | 1591 |
| 3 | Funct_SMG | 5495 | 1602 |
| 3 | Funct_SMG | 5495 | 1607 |
| 3 | Funct_SMG | 5495 | 1612 |
| 3 | Funct_SMG | 5495 | 1613 |
| 3 | Funct_SMG | 5495 | 1615 |
| 3 | Funct_SMG | 5495 | 1616 |
| 3 | Funct_SMG | 5495 | 1617 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 3 | Funct_SMG | 5495 | 1627 |
| 3 | Funct_SMG | 5495 | 1634 |
| 3 | Funct_SMG | 5495 | 1638 |
| 3 | Funct_SMG | 8694 | 1564 |
| 3 | Funct_SMG | 8694 | 1572 |
| 3 | Funct_SMG | 8694 | 1574 |
| 3 | Funct_SMG | 8694 | 1587 |
| 3 | Funct_SMG | 8694 | 1599 |
| 3 | Funct_SMG | 8694 | 1605 |
| 3 | Funct_SMG | 10835 | 1570 |
| 3 | Funct_SMG | 10835 | 1606 |
| 3 | Funct_SMG | 10835 | 1622 |
| 3 | Funct_SMG | 10835 | 1628 |
| 3 | Funct_SMG | 14635 | 1608 |
| 3 | Funct_SMG | 14635 | 1632 |
| 3 | Funct_SMG | 17814 | 1567 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 3 | Funct_SMG | 17814 | 1573 |
| 3 | Funct_SMG | 21101 | 1585 |
| 3 | Funct_SMG | 21101 | 1624 |
| 3 | Funct_SMG | 21101 | 1639 |
| 3 | Funct_SMG | 24536 | 1578 |
| 3 | Funct_SMG | 24536 | 1579 |
| 3 | Funct_SMG | 24536 | 1580 |
| 3 | Funct_SMG | 29655 | 1576 |
| 3 | Funct_SMG | 29655 | 1611 |
| 3 | Funct_SMG | 36718 | 1604 |
| 3 | Funct_SMG | 36718 | 1626 |
| 3 | Funct_SMG | 39426 | 1589 |
| 3 | Funct_SMG | 39426 | 1592 |
| 3 | Funct_SMG | 39426 | 1598 |
| 3 | Funct_SMG | 39426 | 1601 |
| 3 | Funct_SMG | 42131 | 1582 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 3 | Funct_SMG | 42131 | 1584 |
| 3 | Funct_SMG | 49422 | 1568 |
| 3 | Funct_SMG | 49422 | 1623 |
| 3 | Funct_SMG | 52008 | 1629 |
| 3 | Funct_SMG | 52008 | 1630 |
| 3 | Funct_SMG | 55256 | 1586 |
| 3 | Funct_SMG | 55256 | 1593 |
| 3 | Funct_SMG | 55256 | 1633 |
| 3 | Funct_SMG | 62450 | 1603 |
| 3 | Funct_SMG | 62450 | 1637 |
| 3 | Funct_SMG | 66057 | 1614 |
| 3 | Funct_SMG | 66057 | 1618 |
| 3 | Funct_SMG | 83792 | 1590 |
| 3 | Funct_SMG | 83792 | 1596 |
| 3 | Funct_SMG | 87305 | 1566 |
| 3 | Funct_SMG | 87305 | 1631 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 3 | Funct_SMG | 87305 | 1640 |
| 4 | Funct_STS | 1317 | 1648 |
| 4 | Funct_STS | 1317 | 1656 |
| 4 | Funct_STS | 1317 | 1658 |
| 4 | Funct_STS | 1317 | 1661 |
| 4 | Funct_STS | 1317 | 1663 |
| 4 | Funct_STS | 1317 | 1667 |
| 4 | Funct_STS | 1317 | 1668 |
| 4 | Funct_STS | 1317 | 1669 |
| 4 | Funct_STS | 1317 | 1670 |
| 4 | Funct_STS | 1317 | 1677 |
| 4 | Funct_STS | 1317 | 1681 |
| 4 | Funct_STS | 1317 | 1683 |
| 4 | Funct_STS | 1317 | 1685 |
| 4 | Funct_STS | 1317 | 1687 |
| 4 | Funct_STS | 3166 | 1650 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 4 | Funct_STS | 3166 | 1651 |
| 4 | Funct_STS | 3166 | 1660 |
| 4 | Funct_STS | 3166 | 1665 |
| 4 | Funct_STS | 7025 | 1641 |
| 4 | Funct_STS | 7025 | 1653 |
| 4 | Funct_STS | 7025 | 1662 |
| 4 | Funct_STS | 7025 | 1664 |
| 4 | Funct_STS | 8613 | 1643 |
| 4 | Funct_STS | 8613 | 1684 |
| 4 | Funct_STS | 11233 | 1655 |
| 4 | Funct_STS | 11233 | 1679 |
| 4 | Funct_STS | 11233 | 1686 |
| 4 | Funct_STS | 12941 | 1671 |
| 4 | Funct_STS | 12941 | 1676 |
| 4 | Funct_STS | 13892 | 1642 |
| 4 | Funct_STS | 13892 | 1649 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 4 | Funct_STS | 13892 | 1678 |
| 4 | Funct_STS | 18130 | 1652 |
| 4 | Funct_STS | 18130 | 1674 |
| 4 | Funct_STS | 18130 | 1675 |
| 4 | Funct_STS | 18130 | 1680 |
| 4 | Funct_STS | 22450 | 1657 |
| 4 | Funct_STS | 22450 | 1682 |
| 4 | Funct_STS | 26903 | 1646 |
| 4 | Funct_STS | 26903 | 1647 |
| 4 | Funct_STS | 31233 | 1644 |
| 4 | Funct_STS | 31233 | 1666 |
| 4 | Funct_STS | 32598 | 1672 |
| 4 | Funct_STS | 32598 | 1673 |
| 4 | Funct_STS | 36396 | 1654 |
| 4 | Funct_STS | 36396 | 1659 |
| 4 | Funct_STS | 38368 | 1645 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 1924 | 1700 |
| 5 | Funct_Gen2 | 1924 | 1701 |
| 5 | Funct_Gen2 | 1924 | 1702 |
| 5 | Funct_Gen2 | 1924 | 1706 |
| 5 | Funct_Gen2 | 1924 | 1707 |
| 5 | Funct_Gen2 | 1924 | 1712 |
| 5 | Funct_Gen2 | 1924 | 1714 |
| 5 | Funct_Gen2 | 1924 | 1720 |
| 5 | Funct_Gen2 | 1924 | 1736 |
| 5 | Funct_Gen2 | 1924 | 1745 |
| 5 | Funct_Gen2 | 1924 | 1753 |
| 5 | Funct_Gen2 | 1924 | 1754 |
| 5 | Funct_Gen2 | 1924 | 1775 |
| 5 | Funct_Gen2 | 1924 | 1776 |
| 5 | Funct_Gen2 | 1924 | 1780 |
| 5 | Funct_Gen2 | 1924 | 1784 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 1924 | 1785 |
| 5 | Funct_Gen2 | 4555 | 1689 |
| 5 | Funct_Gen2 | 4555 | 1693 |
| 5 | Funct_Gen2 | 4555 | 1710 |
| 5 | Funct_Gen2 | 4555 | 1713 |
| 5 | Funct_Gen2 | 4555 | 1718 |
| 5 | Funct_Gen2 | 4555 | 1731 |
| 5 | Funct_Gen2 | 4555 | 1732 |
| 5 | Funct_Gen2 | 4555 | 1735 |
| 5 | Funct_Gen2 | 4555 | 1747 |
| 5 | Funct_Gen2 | 4555 | 1750 |
| 5 | Funct_Gen2 | 4555 | 1768 |
| 5 | Funct_Gen2 | 4555 | 1774 |
| 5 | Funct_Gen2 | 6750 | 1694 |
| 5 | Funct_Gen2 | 6750 | 1699 |
| 5 | Funct_Gen2 | 6750 | 1708 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 6750 | 1730 |
| 5 | Funct_Gen2 | 8862 | 1717 |
| 5 | Funct_Gen2 | 8862 | 1722 |
| 5 | Funct_Gen2 | 8862 | 1729 |
| 5 | Funct_Gen2 | 8862 | 1734 |
| 5 | Funct_Gen2 | 8862 | 1738 |
| 5 | Funct_Gen2 | 8862 | 1751 |
| 5 | Funct_Gen2 | 8862 | 1762 |
| 5 | Funct_Gen2 | 8862 | 1787 |
| 5 | Funct_Gen2 | 11560 | 1760 |
| 5 | Funct_Gen2 | 11560 | 1764 |
| 5 | Funct_Gen2 | 11560 | 1773 |
| 5 | Funct_Gen2 | 14767 | 1690 |
| 5 | Funct_Gen2 | 14767 | 1695 |
| 5 | Funct_Gen2 | 14767 | 1715 |
| 5 | Funct_Gen2 | 14767 | 1733 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 14767 | 1743 |
| 5 | Funct_Gen2 | 16781 | 1725 |
| 5 | Funct_Gen2 | 16781 | 1755 |
| 5 | Funct_Gen2 | 16781 | 1786 |
| 5 | Funct_Gen2 | 19969 | 1744 |
| 5 | Funct_Gen2 | 19969 | 1746 |
| 5 | Funct_Gen2 | 19969 | 1771 |
| 5 | Funct_Gen2 | 23370 | 1757 |
| 5 | Funct_Gen2 | 23370 | 1766 |
| 5 | Funct_Gen2 | 23370 | 1769 |
| 5 | Funct_Gen2 | 27476 | 1704 |
| 5 | Funct_Gen2 | 27476 | 1724 |
| 5 | Funct_Gen2 | 27476 | 1763 |
| 5 | Funct_Gen2 | 27476 | 1777 |
| 5 | Funct_Gen2 | 27476 | 1788 |
| 5 | Funct_Gen2 | 32395 | 1691 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 32395 | 1723 |
| 5 | Funct_Gen2 | 32395 | 1758 |
| 5 | Funct_Gen2 | 34789 | 1719 |
| 5 | Funct_Gen2 | 34789 | 1726 |
| 5 | Funct_Gen2 | 34789 | 1752 |
| 5 | Funct_Gen2 | 37697 | 1696 |
| 5 | Funct_Gen2 | 37697 | 1709 |
| 5 | Funct_Gen2 | 37697 | 1737 |
| 5 | Funct_Gen2 | 37697 | 1759 |
| 5 | Funct_Gen2 | 41582 | 1756 |
| 5 | Funct_Gen2 | 41582 | 1761 |
| 5 | Funct_Gen2 | 41582 | 1783 |
| 5 | Funct_Gen2 | 55413 | 1692 |
| 5 | Funct_Gen2 | 55413 | 1727 |
| 5 | Funct_Gen2 | 63659 | 1778 |
| 5 | Funct_Gen2 | 63659 | 1782 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 5 | Funct_Gen2 | 72053 | 1749 |
| 5 | Funct_Gen2 | 72053 | 1767 |
| 5 | Funct_Gen2 | 88358 | 1716 |
| 5 | Funct_Gen2 | 88358 | 1739 |
| 5 | Funct_Gen2 | 97810 | 1728 |
| 5 | Funct_Gen2 | 97810 | 1770 |
| 5 | Funct_Gen2 | 99966 | 1721 |
| 5 | Funct_Gen2 | 99966 | 1772 |
| 5 | Funct_Gen2 | 99966 | 1781 |
| 5 | Funct_Gen2 | 107254 | 1703 |
| 5 | Funct_Gen2 | 107254 | 1741 |
| 5 | Funct_Gen2 | 110756 | 1688 |
| 5 | Funct_Gen2 | 110756 | 1740 |
| 6 | Funct_QAR | 1604 | 1798 |
| 6 | Funct_QAR | 1604 | 1825 |
| 6 | Funct_QAR | 1604 | 1828 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 6 | Funct_QAR | 1604 | 1830 |
| 6 | Funct_QAR | 1604 | 1831 |
| 6 | Funct_QAR | 1604 | 1833 |
| 6 | Funct_QAR | 1604 | 1834 |
| 6 | Funct_QAR | 1604 | 1838 |
| 6 | Funct_QAR | 1604 | 1839 |
| 6 | Funct_QAR | 3734 | 1797 |
| 6 | Funct_QAR | 3734 | 1806 |
| 6 | Funct_QAR | 3734 | 1808 |
| 6 | Funct_QAR | 3734 | 1810 |
| 6 | Funct_QAR | 3734 | 1813 |
| 6 | Funct_QAR | 3734 | 1814 |
| 6 | Funct_QAR | 3734 | 1817 |
| 6 | Funct_QAR | 3734 | 1818 |
| 6 | Funct_QAR | 3734 | 1822 |
| 6 | Funct_QAR | 3734 | 1823 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 6 | Funct_QAR | 3734 | 1846 |
| 6 | Funct_QAR | 5683 | 1789 |
| 6 | Funct_QAR | 5683 | 1793 |
| 6 | Funct_QAR | 5683 | 1802 |
| 6 | Funct_QAR | 5683 | 1819 |
| 6 | Funct_QAR | 5683 | 1820 |
| 6 | Funct_QAR | 5683 | 1821 |
| 6 | Funct_QAR | 5683 | 1829 |
| 6 | Funct_QAR | 8072 | 1796 |
| 6 | Funct_QAR | 8072 | 1799 |
| 6 | Funct_QAR | 8072 | 1801 |
| 6 | Funct_QAR | 8072 | 1824 |
| 6 | Funct_QAR | 8072 | 1827 |
| 6 | Funct_QAR | 8072 | 1837 |
| 6 | Funct_QAR | 8072 | 1845 |
| 6 | Funct_QAR | 10185 | 1792 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 6 | Funct_QAR | 10185 | 1848 |
| 6 | Funct_QAR | 12507 | 1794 |
| 6 | Funct_QAR | 12507 | 1812 |
| 6 | Funct_QAR | 12507 | 1815 |
| 6 | Funct_QAR | 12507 | 1835 |
| 6 | Funct_QAR | 12507 | 1840 |
| 6 | Funct_QAR | 12507 | 1841 |
| 6 | Funct_QAR | 12507 | 1843 |
| 6 | Funct_QAR | 15435 | 1791 |
| 6 | Funct_QAR | 15435 | 1809 |
| 6 | Funct_QAR | 15435 | 1844 |
| 6 | Funct_QAR | 24014 | 1803 |
| 6 | Funct_QAR | 24014 | 1847 |
| 6 | Funct_QAR | 28909 | 1800 |
| 6 | Funct_QAR | 28909 | 1805 |
| 6 | Funct_QAR | 31924 | 1807 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 6 | Funct_QAR | 31924 | 1811 |
| 7 | Funct_QM | 1778 | 1851 |
| 7 | Funct_QM | 1778 | 1852 |
| 7 | Funct_QM | 1778 | 1855 |
| 7 | Funct_QM | 1778 | 1861 |
| 7 | Funct_QM | 1778 | 1864 |
| 7 | Funct_QM | 1778 | 1886 |
| 7 | Funct_QM | 1778 | 1891 |
| 7 | Funct_QM | 1778 | 1903 |
| 7 | Funct_QM | 1778 | 1911 |
| 7 | Funct_QM | 3441 | 1867 |
| 7 | Funct_QM | 3441 | 1877 |
| 7 | Funct_QM | 3441 | 1879 |
| 7 | Funct_QM | 3441 | 1881 |
| 7 | Funct_QM | 3441 | 1897 |
| 7 | Funct_QM | 3441 | 1913 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 7 | Funct_QM | 6143 | 1859 |
| 7 | Funct_QM | 6143 | 1870 |
| 7 | Funct_QM | 6143 | 1875 |
| 7 | Funct_QM | 6143 | 1900 |
| 7 | Funct_QM | 6143 | 1904 |
| 7 | Funct_QM | 6143 | 1905 |
| 7 | Funct_QM | 6143 | 1906 |
| 7 | Funct_QM | 6143 | 1907 |
| 7 | Funct_QM | 6143 | 1908 |
| 7 | Funct_QM | 8357 | 1866 |
| 7 | Funct_QM | 8357 | 1868 |
| 7 | Funct_QM | 8357 | 1872 |
| 7 | Funct_QM | 8357 | 1884 |
| 7 | Funct_QM | 8357 | 1887 |
| 7 | Funct_QM | 8357 | 1902 |
| 7 | Funct_QM | 11127 | 1860 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 7 | Funct_QM | 11127 | 1865 |
| 7 | Funct_QM | 11127 | 1878 |
| 7 | Funct_QM | 11127 | 1883 |
| 7 | Funct_QM | 13356 | 1874 |
| 7 | Funct_QM | 13356 | 1895 |
| 7 | Funct_QM | 13356 | 1899 |
| 7 | Funct_QM | 15346 | 1850 |
| 7 | Funct_QM | 15346 | 1885 |
| 7 | Funct_QM | 15346 | 1888 |
| 7 | Funct_QM | 15346 | 1893 |
| 7 | Funct_QM | 19016 | 1857 |
| 7 | Funct_QM | 19016 | 1901 |
| 7 | Funct_QM | 19016 | 1909 |
| 7 | Funct_QM | 24304 | 1854 |
| 7 | Funct_QM | 24304 | 1880 |
| 7 | Funct_QM | 24304 | 1890 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 7 | Funct_QM | 24304 | 1896 |
| 7 | Funct_QM | 42612 | 1849 |
| 7 | Funct_QM | 42612 | 1892 |
| 7 | Funct_QM | 44448 | 1856 |
| 7 | Funct_QM | 44448 | 1910 |
| 7 | Funct_QM | 47338 | 1889 |
| 7 | Funct_QM | 47338 | 1894 |
| 7 | Funct_QM | 50967 | 1862 |
| 7 | Funct_QM | 50967 | 1876 |
| 7 | Funct_QM | 57325 | 1882 |
| 7 | Funct_QM | 57325 | 1912 |
| 8 | Funct_SMG | 2742 | 1919 |
| 8 | Funct_SMG | 2742 | 1922 |
| 8 | Funct_SMG | 2742 | 1934 |
| 8 | Funct_SMG | 2742 | 1939 |
| 8 | Funct_SMG | 2742 | 1940 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 8 | Funct_SMG | 2742 | 1942 |
| 8 | Funct_SMG | 2742 | 1947 |
| 8 | Funct_SMG | 2742 | 1950 |
| 8 | Funct_SMG | 2742 | 1956 |
| 8 | Funct_SMG | 2742 | 1957 |
| 8 | Funct_SMG | 2742 | 1965 |
| 8 | Funct_SMG | 5273 | 1914 |
| 8 | Funct_SMG | 5273 | 1927 |
| 8 | Funct_SMG | 5273 | 1931 |
| 8 | Funct_SMG | 5273 | 1932 |
| 8 | Funct_SMG | 5273 | 1941 |
| 8 | Funct_SMG | 5273 | 1944 |
| 8 | Funct_SMG | 5273 | 1949 |
| 8 | Funct_SMG | 5273 | 1951 |
| 8 | Funct_SMG | 5273 | 1961 |
| 8 | Funct_SMG | 5273 | 1966 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 8 | Funct_SMG | 5273 | 1971 |
| 8 | Funct_SMG | 5273 | 1973 |
| 8 | Funct_SMG | 6995 | 1935 |
| 8 | Funct_SMG | 6995 | 1937 |
| 8 | Funct_SMG | 6995 | 1983 |
| 8 | Funct_SMG | 9184 | 1916 |
| 8 | Funct_SMG | 9184 | 1943 |
| 8 | Funct_SMG | 9184 | 1953 |
| 8 | Funct_SMG | 10852 | 1948 |
| 8 | Funct_SMG | 10852 | 1959 |
| 8 | Funct_SMG | 10852 | 1963 |
| 8 | Funct_SMG | 10852 | 1969 |
| 8 | Funct_SMG | 10852 | 1975 |
| 8 | Funct_SMG | 10852 | 1988 |
| 8 | Funct_SMG | 13091 | 1929 |
| 8 | Funct_SMG | 13091 | 1936 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 8 | Funct_SMG | 13091 | 1962 |
| 8 | Funct_SMG | 13091 | 1976 |
| 8 | Funct_SMG | 13091 | 1978 |
| 8 | Funct_SMG | 15318 | 1945 |
| 8 | Funct_SMG | 15318 | 1968 |
| 8 | Funct_SMG | 16969 | 1958 |
| 8 | Funct_SMG | 16969 | 1986 |
| 8 | Funct_SMG | 21201 | 1915 |
| 8 | Funct_SMG | 21201 | 1917 |
| 8 | Funct_SMG | 21201 | 1923 |
| 8 | Funct_SMG | 23087 | 1918 |
| 8 | Funct_SMG | 23087 | 1928 |
| 8 | Funct_SMG | 25386 | 1967 |
| 8 | Funct_SMG | 25386 | 1982 |
| 8 | Funct_SMG | 25386 | 1984 |
| 8 | Funct_SMG | 28187 | 1972 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 8 | Funct_SMG | 28187 | 1979 |
| 9 | Hwsw_2 | 1690 | 1994 |
| 9 | Hwsw_2 | 1690 | 1995 |
| 9 | Hwsw_2 | 1690 | 1999 |
| 9 | Hwsw_2 | 1690 | 2008 |
| 9 | Hwsw_2 | 1690 | 2011 |
| 9 | Hwsw_2 | 3405 | 1990 |
| 9 | Hwsw_2 | 3405 | 1997 |
| 9 | Hwsw_2 | 3405 | 2003 |
| 9 | Hwsw_2 | 3405 | 2005 |
| 9 | Hwsw_2 | 3405 | 2006 |
| 9 | Hwsw_2 | 5144 | 2000 |
| 9 | Hwsw_2 | 5144 | 2001 |
| 9 | Hwsw_2 | 5144 | 2009 |
| 9 | Hwsw_2 | 7488 | 1996 |
| 9 | Hwsw_2 | 7488 | 2010 |

| ID Exp. | RS | Patt ID | Req ID |
|---|---|---|---|
| 9 | Hwsw_2 | 11211 | 1991 |
| 9 | Hwsw_2 | 11211 | 1998 |
| 10 | SEC | 1047 | 2015 |
| 10 | SEC | 1047 | 2019 |
| 10 | SEC | 1047 | 2022 |
| 10 | SEC | 1047 | 2024 |
| 10 | SEC | 1047 | 2029 |
| 10 | SEC | 4044 | 2027 |
| 10 | SEC | 4044 | 2028 |
| 10 | SEC | 14874 | 2012 |
| 10 | SEC | 14874 | 2014 |
| 10 | SEC | 14874 | 2018 |
| 10 | SEC | 21001 | 2016 |
| 10 | SEC | 21001 | 2026 |

# TESIS DOCTORAL

Uso de algoritmos genéticos para la creación automática de patrones de requisitos