Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Análisis de impacto en un marco de ingeniería de requisitos basada en pruebas de aceptación

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Santiago Pérez García

Tutor: Patricio Letelier Torres

2016-2017

Resumen

El presente trabajo aborda el desafío que supone realizar análisis de impacto en un producto software. Se centra en el análisis en el nivel de especificación de requisitos, expresados en forma de pruebas de aceptación. Estas pruebas están formuladas en lenguaje natural y un sistema puede tener miles de ellas.

Se proponen e implementan diversos mecanismos para la realización del análisis de impacto: mediante búsqueda de términos y búsqueda a partir de términos de un artefacto. Para ello se usa la tecnología Full-Text Search, que permite realizar búsquedas lingüísticas en textos y documentos.

Palabras clave: análisis de impacto, pruebas de aceptación, ingeniería de requisitos.

Abstract

This paper addresses the challenge of performing change impact analysis on a software product. It focuses at specification of requirements level, which are expressed using acceptance tests. These tests are draw up in natural language and a system can contain thousands of them.

Some mechanisms are proposed and developed to carry out change impact analysis: by searching for terms and searching from terms of an artifact. To do this we use Full-Text Search, a technology that allows us to perform linguistic searches against text and documents.

Keywords: change impact analysis, acceptance test, requirement engineering.

Tabla de contenidos

Tabla de figuras

1. Introducción

1.1 Motivación

La intención del análisis de impacto en los requisitos de un software es pronosticar los efectos que puede tener un cambio, ya sea la introducción de una nueva funcionalidad o la modificación de alguna existente, sobre otras partes del programa, que deben ser identificadas [1].

El análisis de impacto se puede hacer a distintos niveles en un proyecto software. Por ejemplo, a nivel de código fuente, los entornos de desarrollo modernos ya proporcionan herramientas para conocer qué partes están relacionadas entre sí o cuáles son las dependencias entre ellas.

A un nivel más alto, el de los requisitos, el análisis se realiza sobre los artefactos que especifican el programa. Las matrices de trazabilidad son una técnica que se usa para el análisis de impacto que relaciona distintos artefactos del programa: requisitos con casos de uso, casos de uso con interfaces gráficas, casos de uso con clases, etc. Aunque esta información siempre requiere un esfuerzo previo de especificación, pues se deben crear y mantener las matrices.

Sin embargo, la información que tienen estos artefactos, aunque concreta, es limitada, por lo que no se puede profundizar y se necesita trabajo adicional para obtener más precisión del impacto del cambio introducido.

Las pruebas de aceptación son una herramienta muy potente para expresar el detalle de los requisitos. Aunque están expresadas de forma textual, son muy concretas y expresan perfectamente las necesidades del cliente. Estas pruebas tienen un nombre y una descripción textual.

Un producto software puede tener miles de pruebas de aceptación y estas deben estar claramente organizadas según la estructura del producto, de la que sí podemos obtener matrices de trazabilidad (por ejemplo nodos relacionados entre sí o siguiendo una jerarquía). Pero en el contexto de Test-Driven Requirement Engineering (TDRE) [9], donde el desarrollo está basado en las pruebas de aceptación, realmente los requisitos están expresados en las pruebas y no hay forma de llevar a cabo un análisis de impacto sin conocer detalladamente cada prueba.

El presente trabajo se ha realizado en el marco de una práctica de empresa en la cual se desarrolla y mantiene un software ERP para el sector socio-sanitario. Se trata de un producto de gran envergadura con muchas funcionalidades relacionadas entre sí. Se ha trabajado estrechamente junto con el grupo de analistas del equipo de desarrollo para el desarrollo del mismo.

1.2 Objetivos

El objetivo de este trabajo es crear diversos mecanismos que, en el contexto de un ERP de una empresa que utiliza el enfoque TDRE para el desarrollo de software, ayuden a los analistas a saber el impacto puede tener añadir o modificar una funcionalidad de un nodo, modificar una funcionalidad existente o eliminarla.

Uno de estos mecanismos consiste en un buscador que será capaz de encontrar nodos y pruebas de aceptación basándose en uno existente pero añadiendo nuevas funcionalidades, como buscar en más campos o permitiendo una "búsqueda inteligente" que no se limite a encontrar la palabra exacta.

El otro mecanismo consiste en seleccionar una parte del programa y obtener una lista de resultados, ordenados según la criticidad, en la que se mostrarán las partes del programa que pueden ser afectadas y por qué motivos.

1.3 Estructura del trabajo

La organización de la memoria es la siguiente:

En el capítulo 1 se incluye una introducción a la motivación del trabajo y presenta los objetivos.

En el capítulo 2 se describe qué es el análisis de impacto, su utilidad y cómo se realiza actualmente.

En el capítulo 3 se comenta TDRE, el método sobre el cual se trabaja y la herramienta que la soporta.

El capítulo 4 trata de la tecnología que se ha utilizado para la implementación de las herramientas de análisis de impacto.

En el capítulo 5 se describen las herramientas desarrolladas.

En el capítulo 6 se escenifican diversos ejemplos reales de uso de las herramientas.

En el capítulo 7 se tratan las conclusiones y el trabajo futuro que surge a partir de este trabajo.

2. Análisis de impacto

2.1 Definición

La definición más común para el análisis de impacto en el campo de la ingeniería del software es "el proceso de identificar consecuencias potenciales de un cambio o el proceso de estimar qué se necesita modificar para llevar a cabo un cambio" [1]. En otras palabras, se trata de identificar qué partes de un producto software pueden verse afectadas por la introducción de un cambio, ya sea una mejora, un nuevo requisito o la eliminación de una característica. Cabe destacar que este análisis se puede realizar en los distintos artefactos de un producto: el código del programa, el diseño, el modelado, etc.

La importancia de realizar un análisis de impacto con cada cambio reside en [2]:

- Identificar a qué partes del producto es necesario aplicar pruebas de regresión.
- Poder estimar y planificar los recursos dedicados al cambio.
- Reducir la cantidad de fallos introducidos a causa de los efectos que puede introducir un cambio.
- Mejorar en general la eficacia del mantenimiento del software.

En definitiva, realizar un buen análisis de impacto es determinante para asegurar la calidad de cualquier producto software. Desde una pequeña aplicación a grandes proyectos, donde un cambio puede afectar a tantas partes que es muy difícil de controlar sin el uso de una metodología y herramientas adecuadas.

2.2 Proceso

Hablamos de *change set* (conjunto de cambios) cuando nos referimos a las partes del sistema que van a ser cambiadas, es decir, las partes a las que afecta directamente el cambio introducido y de *impact set* (conjunto de impacto) a las partes del producto que pueden verse afectadas por los cambios ocurridos en el *change set*.

Generalmente el proceso que se sigue para realizar el análisis de impacto es el siguiente [2][3]:

1) Se identifica el *change set* en los artefactos a analizar (especificación, código, etc.).
2) Siempre que se encuentre una dependencia se debe trazar una relación entre los artefactos cuyo origen será un elemento del *change set*.
3) Si un cambio afecta cualquiera de estos artefactos se deben buscar recursivamente todos los artefactos afectados por estos.

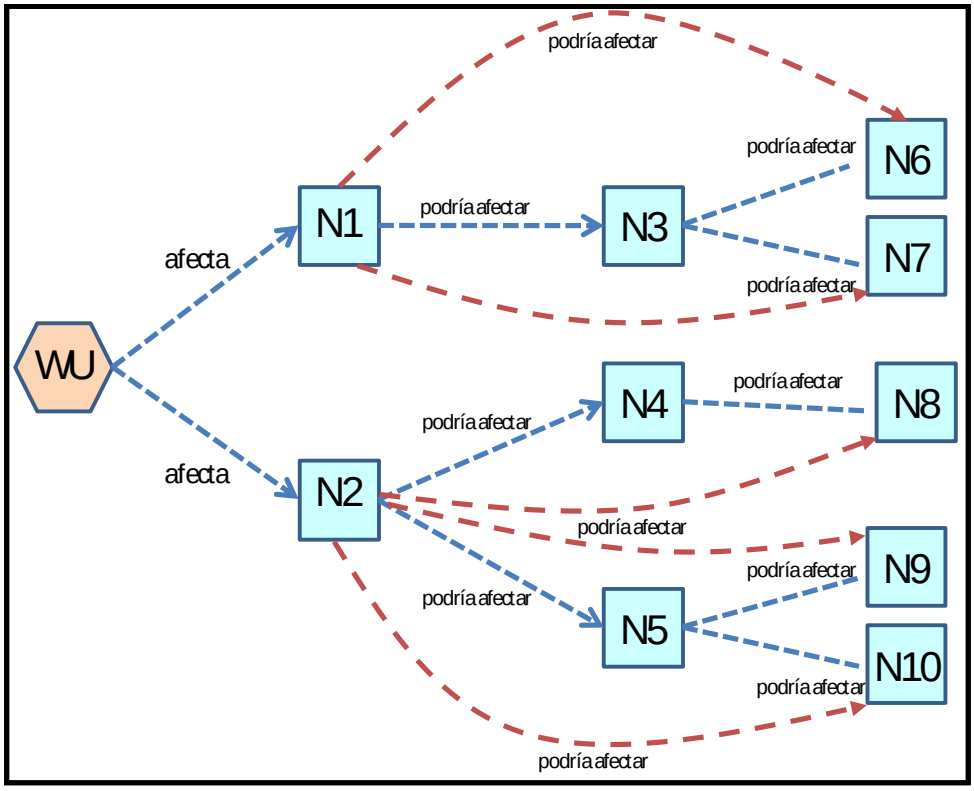


Figura 1: Grafo de dependencias

El proceso se repite hasta que se obtiene un grafo (Figura 1) donde están representados todos los artefactos que pueden ser afectados por un cambio. En una unidad de trabajo (WU) en la que se van a realizar modificaciones en los nodos 1 y 2 (*change set*) se han calculado dependencias hacia los nodos 3, 4 y 5, dependencias que a su vez pueden afectar a los nodos 6, 7, 8, 9 y 10. Todos los nodos a los que podría afectar la modificación de los nodos del *change set* se incluyen en el *impact set*.

El mayor desafío de realizar este análisis surge debido a que dentro de un producto van a estar todos los artefactos relacionados entre sí. Si no controlamos el alcance del análisis de impacto, este nos puede devolver un *impact set* que sea todo el producto, por lo que el análisis de impacto resultaría inútil.

Dos maneras de mitigar este problema son [2]:

- Limitar el número de iteraciones que se realizan para definir la profundidad del análisis. Un número razonable de iteraciones nos devolverá un *impact set* que realmente sea cercano al *change set*.
- Definir restricciones en las relaciones entre artefactos para solo analizar las que nos interesen en ese momento.

2.3 Tipos de análisis de impacto

A nivel de implementación existen diversas técnicas de análisis de impacto que se pueden clasificar en: análisis de impacto dinámico y estático [4].

En el estático se analiza la información estática del producto, que pueden ser diversos artefactos: código, requisitos, diseño, etc. y sus resultados pueden incluir todo el sistema en el *impact set* (comentado en la sección 2.2).

La mayoría de técnicas de análisis de impacto se centran en el código, analizan la sintaxis y la semántica o la evolución de las dependencias del programa [5].

Este análisis puede dividirse en:

- Análisis estructural: se centra en las dependencias del programa y construye un grafo de dependencias.
- Análisis textual: extrae dependencias contextuales a partir de los comentarios en el código.
- Análisis histórico: extrae la información a partir de diversas versiones del producto.

A nivel de requisitos existen técnicas como las matrices de trazabilidad, que sirven, por ejemplo, para saber qué requisitos están cubiertos por qué casos de uso, por lo que un cambio en un requisito seguramente afectará al caso de uso.

Por otra parte, el análisis de impacto dinámico obtiene el *impact set* analizando la información dinámica del programa: ejecutando el código, analizando los resultados de la ejecución, obteniendo el camino que sigue un caso de prueba, etc. Los resultados conseguidos son más precisos (el *impact set* no alcanza todo el producto). Sin embargo, este análisis solo se aplica a la parte del código y no es aplicable a los requisitos.

2.4 Herramientas

Podemos clasificar las herramientas para hacer análisis de impacto en dos categorías: las que trabajan sobre el código y las que trabajan a nivel de requisitos/especificación.

La mayoría de entornos de desarrollo, como Eclipse, NetBeans o VisualStudio, ofrecen en mayor o menor medida herramientas para realizar análisis de impacto. Con ayuda de estas podemos encontrar dependencias entre clases, métodos o arquitectura. También existen herramientas específicas para el análisis como JRipples [6] o analizadores de código que nos pueden ayudar a encontrar las dependencias, como FindBugs [7].

A nivel de requisitos existen diversas herramientas (Figura 2) cuya única diferencia entre como realizan el análisis de impacto y trazabilidad son las diferentes maneras que tienen para generar y mantener las matrices de trazabilidad [8]. Sin embargo, las matrices de trazabilidad (Figura 3) siempre requieren de un mantenimiento manual por parte de los analistas y el nivel de granularidad es bastante alto, de lo contrario las matrices generadas serían de gran tamaño e insostenibles.

Uno de los objetivos de este trabajo es desarrollar herramientas que, al contrario de las abajo mencionadas, automaticen el proceso del análisis de impacto, evitando el mantenimiento manual de las relaciones de trazabilidad.

IRqA	Rational Requisite Pro	DOORS	CaliberRM
Permite la creación de relación de trazabilidad entre requisitos y: -otros requisitos -elementos del dominio del problema (conceptos, entidades) -elementos de la especificación de la solución -escenarios -IRqA test cases y TestDirectos tests -clases de implementación -código fuente a través de la asociación con archivos externos VISUALIZACION -vistas de elementos relacionados -matriz de trazabilidad	Permite la creación de relaciones de trazabilidad entre los tipos de requisitos gestionados por RequisitePro VISUALIZACION -propiedades de los requisitos -matriz de trazabilidad	Permite la creación de relación de trazabilidad entre cualquier par de objetos contenidos en cualquier módulo dentro del repositorio de DOOR. La relación entre objetos está definida por el usuario en el link de módulos. Son establecidos diferentes tipos de relaciones entre un par de módulos (entre sus objetos) . Permite la definición de atributos por cada tipo de relación. La relación puede ser establecida desde: -los módulos que contienen los objetos que el usuario quiere relacionar -desde linkset of link module. VISUALIZACION -matriz de trazabilidad	Permite el establecimiento de relación de trazabilidad a través: -Trazabilidad tabs:es posible establecer trazabilidad entre requisitos y requisitos del proyecto VISUALIZACION -matriz de trazabilidad

Figura 2: Comparativa herramientas gestión de requisitos [8]

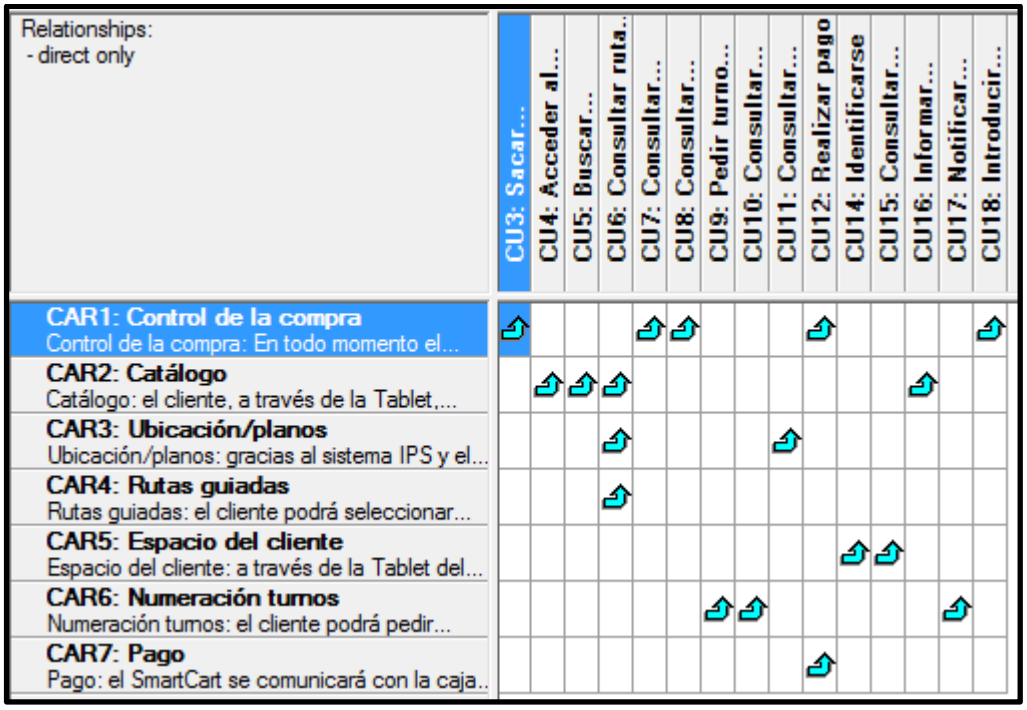


Figura 3: Matriz de trazabilidad entre características y casos de uso

3. TDRE – Test-Driven Requirement Engineering

El contexto de trabajo sobre el que se quieren desarrollar las herramientas para el análisis de impacto se basa en el enfoque Test-Driven Requirement Engineering, que a su vez se basa en Test-Driven Development (TDD) [8]. En TDRE se establece que las pruebas de aceptación (PA) deben dirigir el desarrollo del producto software, es decir, se trabaja con las PA como especificación detallada de los requisitos.

En el modelo en V [8] se consideran las pruebas como un proceso paralelo al análisis y el desarrollo, en lugar de constituir una fase aislada al final del proceso. Partiendo de los requisitos se crean las pruebas de aceptación, del análisis las pruebas de sistema, del diseño de la arquitectura las pruebas de integración y del diseño de los módulos las pruebas unitarias.

Generalmente, tanto las metodologías tradicionales como en las ágiles se utilizan las pruebas de sistema y las pruebas de aceptación, sin embargo en muy pocas estas últimas son las que dirigen en proceso de desarrollo.

En cambio, en TDRE se usa el planteamiento *"no incorporar un requisito a una iteración (o al plan general) sin haber antes establecido sus Pruebas de Aceptación"* [9]. Es decir, se integra todo lo relacionado con la especificación del producto, la validación y las pruebas de aceptación.

Una prueba de aceptación sirve para especificar los requisitos del producto en lenguaje natural y es capaz de precisar una parte del comportamiento del sistema gracias a su granularidad y verificabilidad.

Una PA describe un escenario, compuesto por una situación del sistema, una secuencia de pasos de uso y el resultado alcanzado (Figura 4), todo ello desde la perspectiva del usuario. En el escenario se establecen las condiciones previas que se tiene que satisfacer antes de dar paso a la prueba (estado de la base de datos, navegación entre menús, configuraciones, etc.), los pasos son las acciones de la prueba en sí y el resultado esperado es el efecto esperado de las acciones anteriores.

Está asociada a requisitos funcionales o no funcionales, que pueden tener más de una PA. Por su granularidad cubren todas las situaciones, desde las más típicas hasta las más excepcionales [9]. Es importante que las PA estén ordenadas por este criterio (de más típico a más excepcional), ya que sirve

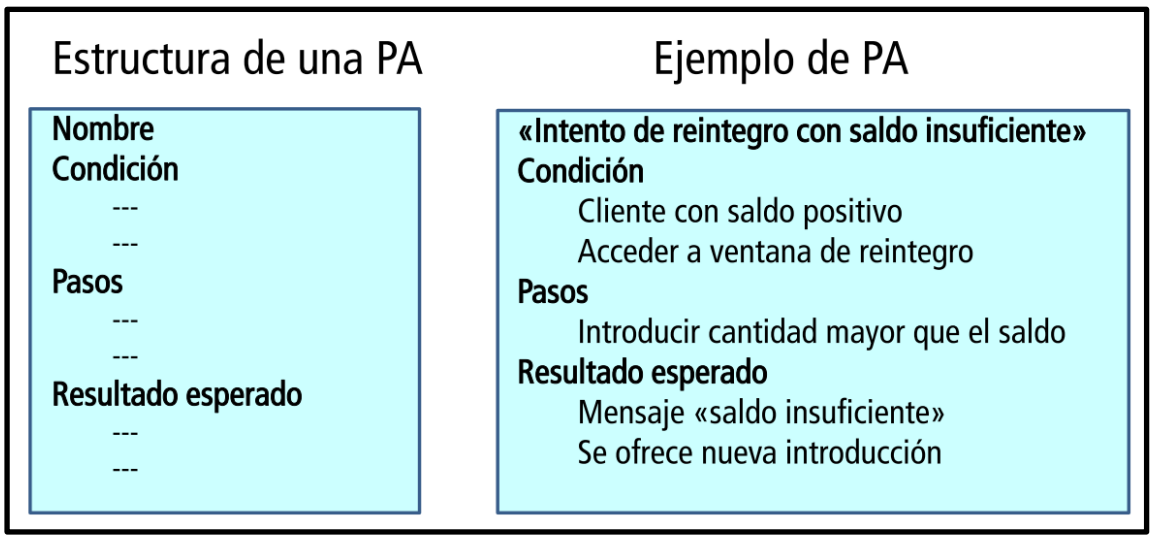


Figura 4: Estructura de una PA

como guía para que los desarrolladores implementen el código y para que los testers puedan hacer su trabajo.

TDRE está pensado para que el desarrollo del código sea de forma incremental con la idea de ir satisfaciendo PA hasta completar todas las funcionalidades.

Con el tiempo el esfuerzo invertido en crear las PA se rentabiliza porque permiten [8]:

- Especificar y validar los requisitos del producto.
- Valorar el esfuerzo de la incorporación de un requisito.
- Negociar el alcance de cada versión del producto con el cliente.
- Guiar la implementación respecto a los requisitos.
- Reutilizar PA que describen un comportamiento que se repite en distintas partes del producto.

3.1 SAPI – Sistema de Ayuda al Proceso de Incidencias

SAPI es una herramienta interna de la empresa en la que se ha desarrollado el presente trabajo que permite la gestión y mantenimiento de un producto software, desde el *backlog*, las iteraciones y las incidencias (unidades mínimas de trabajo) siguiendo *workflows* ágiles que surgió como necesidad de coordinar y controlar el trabajo entre los distintos agentes [10].

Dispone de diversos módulos, entre los que destacan:

- **Planificador de versiones:** permite gestionar toda la información sobre los programas, sus versiones, los agentes asignados, los flujos de trabajo…

- **Planificador personal:** el agente puede ver un listado de todas las incidencias que tienen una actividad asignada a él y su estado (Figura 5).

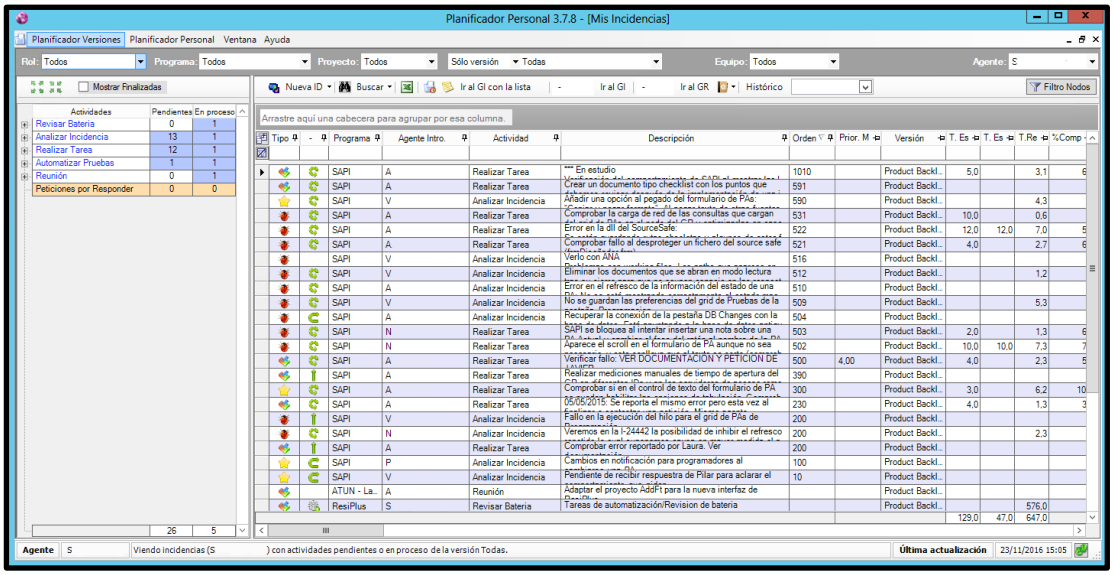


Figura 5: SAPI - Planificador personal

- **Gestor de incidencias:** proporciona todas las funcionalidades asociadas a una incidencia: gestión de documentos, planificación, tipo de incidencia, tiempos registrados…
- **Gestor de requisitos:** es donde se establece toda la estructura del producto y donde se especifican y mantienen los requisitos del mismo (Figura 6).

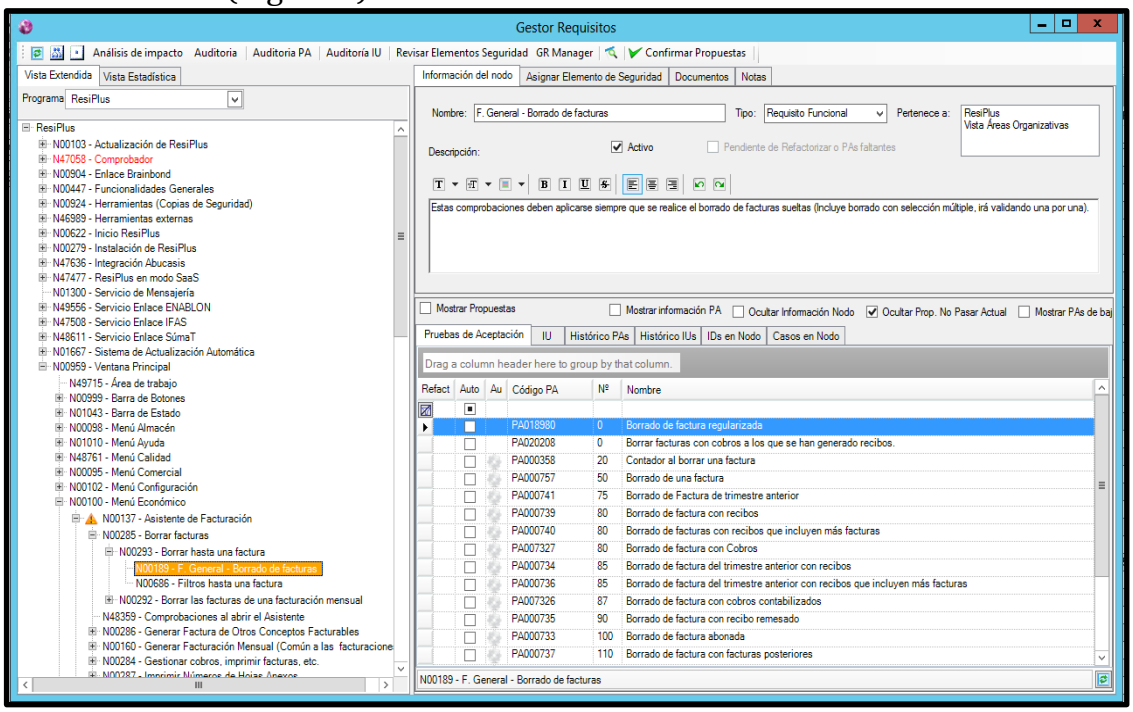


Figura 6: SAPI - Gestor de Requisitos

Este último módulo es el que resulta interesante, pues se pretende realizar análisis de impacto sobre las PA (Figura 6) definidas en el gestor de requisitos (GR).

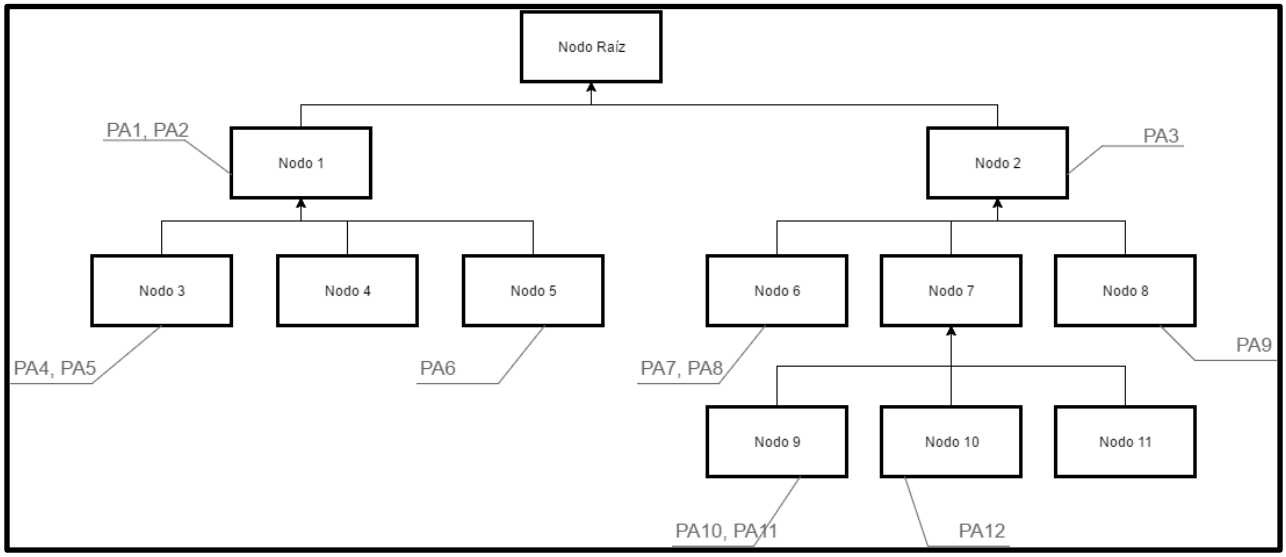


Figura 7: Estructura del producto en SAPI

En el GR se especifican todos los requisitos del programa. En él se organizan los programas mediante un árbol de nodos, donde se agrupan las pruebas de aceptación dependiendo de la funcionalidad a la que correspondan. También para la gestión de requisitos permite incluir interfaces de usuario asociadas a la funcionalidad.

Las pruebas de aceptación permiten garantizar la calidad del programa en relación a los requisitos, funcionales o no, que se han definido para él. Siempre desde la perspectiva del usuario final.

En TDRE se diseñan las pruebas de aceptación junto con los requisitos y no se pasa a la implementación de los requisitos hasta que no están todas las PA diseñadas. Una vez implementado el requisito se ha de comprobar que se satisfacen las pruebas de aceptación.

En el GR se organiza el producto software en nodos (Figura 7), que generalmente están asociados a un requisito (funcional o no), a una interfaz de usuario o a un área organizativa. Un nodo puede a su vez tener nodos hijos, que detallan la especificación del padre. Así se forma un árbol de nodos con relaciones padre-hijo que especifica el producto. Los nodos a su vez pueden contener pruebas de aceptación (en la Figura 6 se observa como el nodo *Borrado de facturas* contiene numerosas PA), que como ya se ha dicho, especifican los requisitos del producto.

4. Full-Text Search

Para la implementación de las herramientas para la ayuda al análisis de impacto se ha optado por usar Full Text Search (FTS) en Microsoft SQL Server 2012 ya que es un entorno en el que ya se trabaja y no se requería ninguna instalación adicional de ningún tipo.

4.1 Definición

FTS es una técnica de búsqueda que permite realizar búsquedas lingüísticas en textos y documentos almacenados en una base de datos [11][12].

Una búsqueda lingüística es una búsqueda basada en una palabra o frase con una determinada configuración de lenguaje (idioma, conjugación de verbos, sinónimos, polisemia, etc.). Esto convierte a FTS en una herramienta mucho más potente que el predicado LIKE clásico de SQL.

Por ejemplo, también es capaz de ignorar palabras raras o no importantes para la búsqueda, ya sea porque vienen predefinidas en el propio lenguaje (preposiciones, conjunciones, etc.) o porque el usuario ha creado su propia lista de palabras a evitar (*stopwords*).

Otras ventajas de usar FTS son que permite hacer búsquedas con peso, es decir, indicar que términos de la búsqueda son más importantes, permite clasificar los resultados según el criterio del usuario y es compatible con multitud de formatos de documentos.

Usar FTS en Microsoft SQL Server es muy sencillo y aunque no es el objetivo de este trabajo se van a comentar los dos pasos principales:

- Crear un catálogo. Un catálogo es donde se almacena toda la información necesaria sobre las tablas para poder realizar consultas con FTS sobre ellas.
- Crear índices. Los índices de full-text sirven para localizar las filas relevantes en una consulta, sin ellos FTS no funciona. Simplemente basta con indicar qué columnas de qué tablas queremos indexar y podemos indicar el tipo de dato y el idioma en el que están.

4.1.1 Consultas

Disponemos de dos predicados de búsqueda, basado en SearchSQL, una extensión de SQL:

a) CONTAINS

Proporciona búsqueda precisa, aproximada, de proximidad o por peso.

Es un predicado que se pone en la cláusula WHERE de la cláusula SELECT de Transact-SQL como una condición más y puede ir junto a otras condiciones.

La sintaxis que sigue es la siguiente [13]:

```
CONTAINS (
      {
         column_name | ( column_list )
       | *
       | PROPERTY ( { column_name }, 'property_name' )

      }
      , '<contains_search_condition>'
      [ , LANGUAGE language_term ]
    )
```

Las columnas deben estar indexadas para poder usarlas en el predicado Contains.

Permite añadir condiciones como el peso de las palabras o búsqueda de proximidad de unas palabras con otras e indicar el idioma sobre el que se trabaja.

En el caso que nos ocupa se ha usado para sustituir el predicado LIKE y realizar así búsquedas de palabras exactas aprovechando la rapidez que ofrecen los índices de FTS.

b) FREETEXT

Proporciona una búsqueda *fuzzy*, es decir, no busca solamente palabras exactas, sino palabras que coincidan con el significado. Es capaz de obtener palabras que son variaciones de la palabra que se busca (plural, conjugación, género…).

Su sintaxis es la siguiente [14]:

```
FREETEXT ( { column_name | (column_list) | * }
          , 'freetext_string' [ , LANGUAGE language_term ] )
```

Es el predicado que se ha usado para crear las funcionalidades de las herramientas. También permite indicar el idioma sobre el que se trabaja.

c) CONTAINSTABLE y FREETEXTTABLE

Los predicados anteriores no ofrecen ninguna posibilidad de limitar el número de resultados en forma de filas que nos devuelven las consultas. Esto supone que se pueden obtener más resultados de los que se necesitan,

ello afecta directamente al rendimiento del sistema. Por otra parte los resultados que se obtienen no están clasificados ni ordenados bajo ningún criterio.

CONTAINSTABLE y FREETEXTTABLE son funciones que ofrecen las mismas prestaciones que los anteriores pero permiten:

- Obtener una columna adicional llamada *rank*, en la que se asigna un valor entre 0 y 1000 e indica cómo de relevante es ese resultado.
- A partir de la columna *rank* se pueden ordenar los datos con la cláusula ORDER BY o seleccionar solo una cantidad determinada de resultados.

La sintaxis de estas funciones se presenta a continuación:

CONTAINSTABLE[16]:

```
CONTAINSTABLE
( table , { column_name | ( column_list ) | * } , '
<contains_search_condition> '
     [ , LANGUAGE language_term]
  [ , top_n_by_rank ]
)
```

FREETEXTTABLE[17]:

```
FREETEXTTABLE (table , { column_name | (column_list) | * }
          , 'freetext_string'
     [ , LANGUAGE language_term ]
     [ , top_n_by_rank ] )
```

Ambas funciones se usan en la cláusula FROM de la consulta y se usan haciendo join con la tabla sobre la que se va a realizar la búsqueda.

Para CONTAINSTABLE la fórmula que calcula el *rank* es la siguiente [15]:

```
StatisticalWeight = Log2( ( 2 + IndexedRowCount ) / KeyRowCount )

Rank = min( MaxQueryRank, HitCount * 16 * StatisticalWeight /
MaxOccurrence )
```

Respecto a FREETEXTTAVLE está basada en la fórmula de clasificación OKAPI BM25 [15], cuyo destino es clasificar documentos en función de su relevancia respecto a una búsqueda determinada. Esta es su sintaxis:

```
Rank = Σ[Terms in Query] w ( ( ( k1 + 1 ) tf ) / ( K + tf ) ) * ( ( k3 + 1 ) qtf / ( k3 + qtf ) ) )
Where:
w is the Robertson-Sparck Jones weight.
In simplified form, w is defined as:
w = log10 ( ( ( r + 0.5 ) * ( N - R + r + 0.5 ) ) / ( ( R - r + 0.5 ) * ( n - r + 0.5 ) ) )
N is the number of indexed rows for the property being queried.
n is the number of rows containing the word.
K is ( k1 * ( ( 1 - b ) + ( b * dl / avdl ) ) ).
dl is the property length, in word occurrences.
avdl is the average length of the property being queried, in word occurrences.
k1, b, and k3 are the constants 1.2, 0.75, and 8.0, respectively.
tf is the frequency of the word in the queried property in a specific row.
qtf is the frequency of the term in the query.
```

4.2 Ejemplos de consultas

A continuación se muestran ejemplos reales de consultas utilizadas en este trabajo.

CONTAINS

```
SELECT DISTINCT n.[IDNodo],n.[nombre]
FROM [Incidencias].[dbo].[GRNodo] n
	INNER JOIN
[Incidencias].[dbo].[GRNodoPadreHijo] nh ON
	n.[IDNodo] = nh.[IDNodoHijo]
WHERE nh.[IDPrograma] = 1
	AND CONTAINS(n.[nombre], N'factura');
```

Esta consulta busca la palabra 'factura' en la columna *nombre* de la tabla *GRNodo* con el predicado CONTAINS, que busca exactamente esa palabra. La ejecución de esta consulta en la base de datos donde se pretende usar esta tecnología devuelve 32 resultados.

FREETEXT

```
SELECT DISTINCT n.[IDNodo],n.[nombre]
FROM [Incidencias].[dbo].[GRNodo] n
	INNER JOIN
[Incidencias].[dbo].[GRNodoPadreHijo] nh ON
	n.[IDNodo] = nh.[IDNodoHijo]
WHERE nh.[IDPrograma] = 1
	AND FREETEXT(n.[nombre], N'factura');
```

La misma consulta pero con el predicado FREETEXT devuelve 114 resultados.

Ambas son consultas fáciles de escribir, pues la búsqueda de texto completo se realiza como una condición más de la cláusula WHERE.

CONTAINSTABLE

```
SELECT DISTINCT n.[IDNodo], n.[nombre], k.rank
FROM [Incidencias].[dbo].[GRNodo] n
	INNER JOIN [Incidencias].[dbo].[GRNodoPadreHijo] nh ON n.[IDNodo] =
nh.[IDNodoHijo]
	INNER JOIN CONTAINSTABLE([Incidencias].[dbo].[GRNodo], (nombre), 'Factura')
k ON n.[IDNodo] = k.[Key]
WHERE (nh.[IDPrograma] = 1);
```

Sin embargo, el uso de las funciones para poder calcular el *rank* complica la escritura de las consultas. Esta consulta busca exactamente la misma información que la realizada con el predicado CONTAINS y obtiene el mismo número de resultados, solo que ahora disponemos de la columna *k.rank*.

FREETEXTTABLE

Lo mismo ocurre para esta función, devuelve los mismos resultados pero con la ventaja de que tenemos la columna con el rango, con la cual podemos hacer operaciones de ordenación o de truncado de resultados.

```
SELECT DISTINCT n.[IDNodo], n.[nombre], k.rank
FROM [Incidencias].[dbo].[GRNodo] n
	INNER JOIN [Incidencias].[dbo].[GRNodoPadreHijo] nh ON n.[IDNodo] =
nh.[IDNodoHijo]
	INNER JOIN FREETEXTTABLE([Incidencias].[dbo].[GRNodo], (nombre), 'Factura')
k ON n.[IDNodo] = k.[Key]
WHERE (nh.[IDPrograma] = 1);
```

5. Aplicación de FTS

Antes de realizar este TFG en el SAPI existían dos buscadores que ofrecían un apoyo muy limitado para el análisis de impacto.

El buscador de nodos permite introducir una palabra y busca todas las ocurrencias de esa palabra en los nombres de los nodos y resalta los nodos que incluyen esa cadena de texto. También permite introducir un código de nodo y resaltar los nodos con ese código. El buscador de PA trabaja de forma similar, solo que busca en los nombres de las PA y además de resaltar el nodo donde se encuentra, resalta esa PA para diferenciarla del resto de las PA del nodo.

La idea era que estos dos buscadores fueran sustituidos por un buscador más avanzado gracias a la tecnología FTS y una nueva herramienta que pueda advertir qué nodos se pueden ver afectados por el cambio en otro nodo.

5.1 Buscador de análisis de impacto

La primera herramienta desarrollada ha sido un nuevo buscador incluido en el GR como herramienta de ayuda para el análisis de impacto. Este buscador sustituye a los antiguos buscadores de nodos y de pruebas de aceptación. Estos buscadores solo permitían buscar en el nombre del nodo y en el nombre de la prueba y en los respectivos códigos de identificación. Esto suponía que era necesario saber el código o alguna palabra exacta de los nombres, reduciendo la eficacia de la búsqueda.

El nuevo buscador permite, usando FTS, buscar en más campos: nombre de nodo, descripción de nodo, nombre de prueba de aceptación y descripción de prueba de aceptación, pudiendo obtener unos resultados más precisos.

Como ya se ha explicado, la búsqueda se hace más completa y al buscar cierto término no solamente busca ese término, sino también variaciones del mismo (conjugaciones de verbos, plural, género, etc.) por lo que obtenemos unos resultados más exactos y fieles a lo que buscamos, por lo que en el contexto del análisis de impacto nos ayuda a encontrar aquellos nodos relacionados con nuestra búsqueda de forma más flexible.

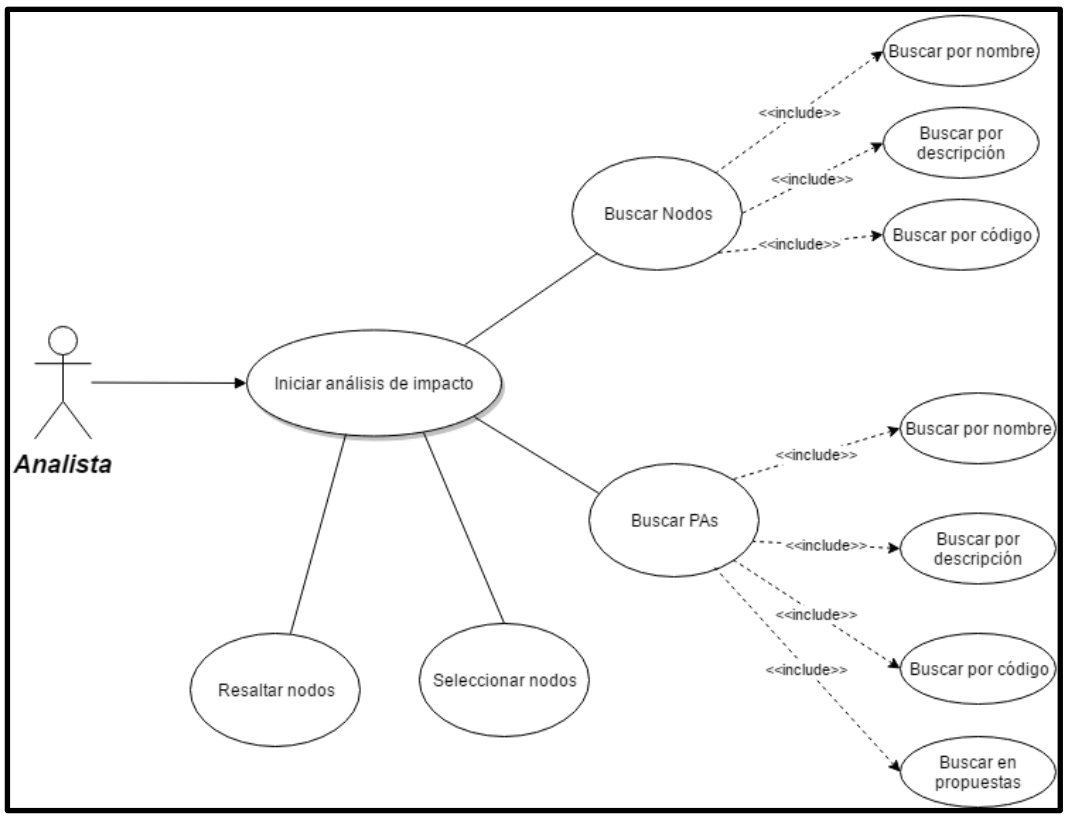


Figura 8: Diagrama casos de uso del buscador

En el diagrama de casos de uso (Figura 8) se detalla la funcionalidad del nuevo al buscador:

- Buscar en los nombres de nodos
- Buscar en las descripciones de nodos
- Buscar por código de nodo
- Buscar por nombre de PA
- Buscar en la descripción de PA
- Buscar en las PA propuestas (nuevas PA, modificaciones o propuestas para eliminarse)
- Buscar por código de PA
- Buscar palabras exactas

El resultado de una búsqueda devuelve una lista de nodos en los que la cadena buscada aparece (ya sea de forma exacta o flexible aprovechando las características que proporciona FTS). Para cada nodo resultante se muestra su código, su nombre y un índice de relevancia calculado gracias al *rank* visto anteriormente.

5.2 Posibles nodos afectados

La segunda herramienta desarrollada surge a partir del primer buscador de análisis de impacto. La funcionalidad que ofrece, aunque es bastante completa, está limitada por la precisión con la que el usuario introduzca los términos de búsqueda. Así, para saber que nodos del programa pueden

ser afectados por la modificación o introducción de una funcionalidad en otro nodo, habría que hacer diversas búsquedas para obtener todos los nodos relacionados. Esto arrojaría miles de resultados en un programa relativamente complejo y grande, así que se ha desarrollado una herramienta que permite encontrar otros nodos a partir de unos términos predefinidos. Como ya se ha visto en la sección 3.1 :

- Los programas están organizados por nodos, que pueden tener nodos hijos. Cada nodo tiene un nombre y una descripción.
- Los nodos contienen PA, que también tienen un nombre y una descripción.
- La única relación que existe es jerárquica (padres e hijos).
- Una PA solo puede estar en un nodo.

Se introduce el concepto de *término*: palabra o conjunto de palabras que describen una parte o una funcionalidad del programa.

Se establece una relación término-nodo si el término aparece en:

- Nombre del nodo
- Descripción del nodo
- Nombre de las PA contenidas en el nodo
- Descripción de las PA contenidas en el nodo

Gracias a estos términos podemos hallar otros nodos que compartan esta relación con los mismos.

Si tenemos un conjunto de nodos N:{N1, N2, N3, N4, N5} y una lista de términos del programa: T: {T1, T2, T3} podemos identificar qué términos están relacionados con un nodo (Figura 9).

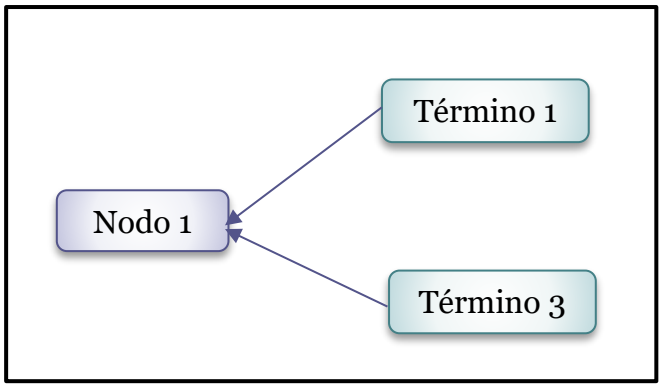


Figura 9: Términos que afectan a un nodo

Con los términos T1 y T3 se hace una búsqueda por todos los nodos y PA del programa, obteniendo los que tienen algún tipo de relación con ellos (Figura 10).

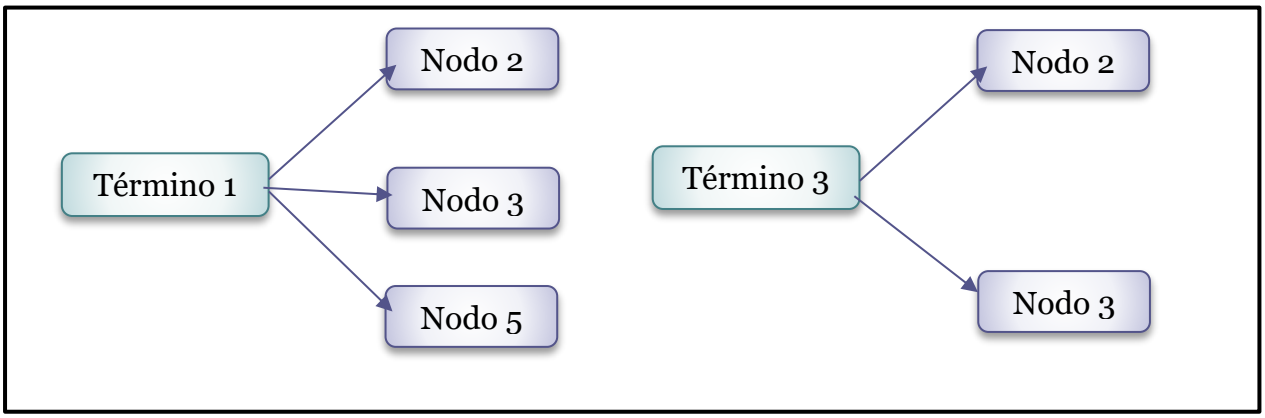


Figura 10: Nodos afectados por los términos

Por tanto, la modificación del N1 puede afectar a los nodos N2, N3 y N5 (Figura 11).

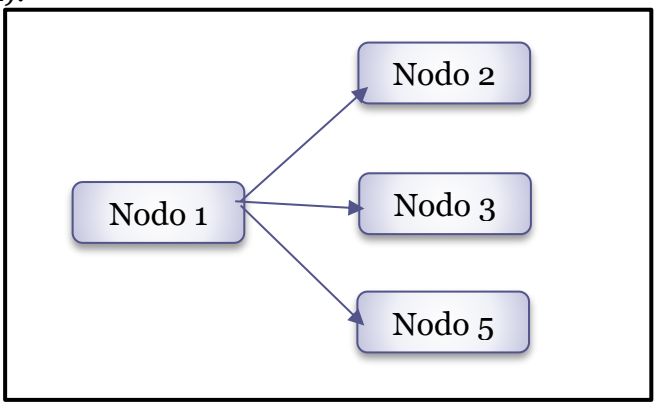


Figura 11: Dependencia directa entre change set e impact set

Pero, ¿en qué medida?

Para esto se usa un índice de relevancia, calculado de nuevo gracias al *rank*, que:

- En primer lugar permite distinguir los nodos más relevantes, en general o por sección.
- Para cada tupla término-nodo permite saber en qué medida un término en concreto afecta a un nodo y en qué apartado.

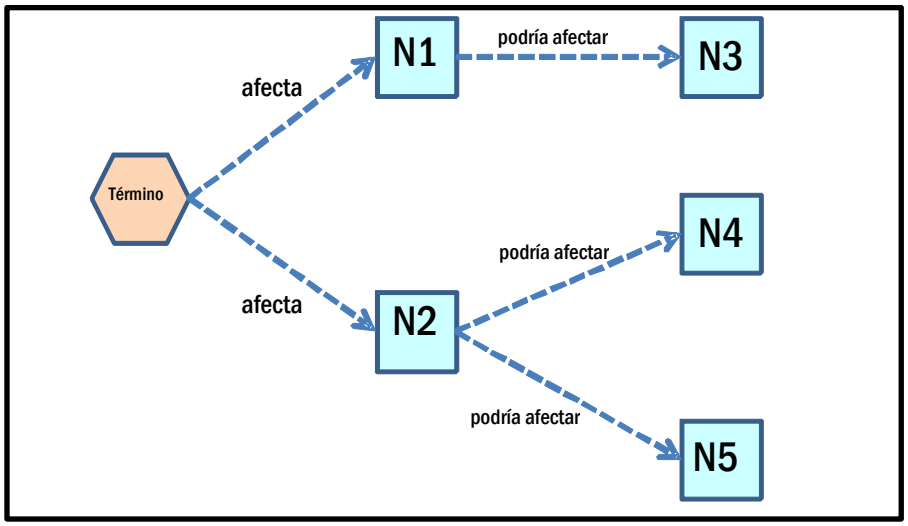


Figura 12: Grafo de dependencias directas

Este proceso es totalmente automático, es decir, no necesita que se hayan establecido previamente relaciones nodo-término. Se aprovecha la tecnología FTS para poder hacer estas búsquedas sobre el lenguaje natural.

Esto presenta la ventaja de ahorrar el trabajo de mantenimiento de las relaciones nodo-término. La única gestión manual debe ser la de crear, eliminar y mantener al día los términos.

Es importante incidir en que solo se pretende hacer un análisis de impacto directo, es decir, solo se calculan los nodos directamente afectados por el *change set*. Los nodos obtenidos al convertir el *impact set* en *change set* no son considerados. Como se observa en la Figura 12, a partir de un término que afecta a un nodo podemos hallar otros nodos a los que directamente podrían afectar a causa de ese término. Sin embargo, como es un análisis de impacto directo no se calculan nodos a los que indirectamente podría afectar como se puede observar en la Figura 1.

La gestión de los términos se hará mediante un gestor de términos que permitirá relacionar términos y una breve descripción con un programa.

5.3 Extensión implementada

En esta sección se va a explicar el funcionamiento detallado de cada herramienta desarrollada a modo de guía de usuario.

5.3.1 Buscador de análisis de impacto

Para usar el nuevo buscador, simplemente se accede a él desde el GR.

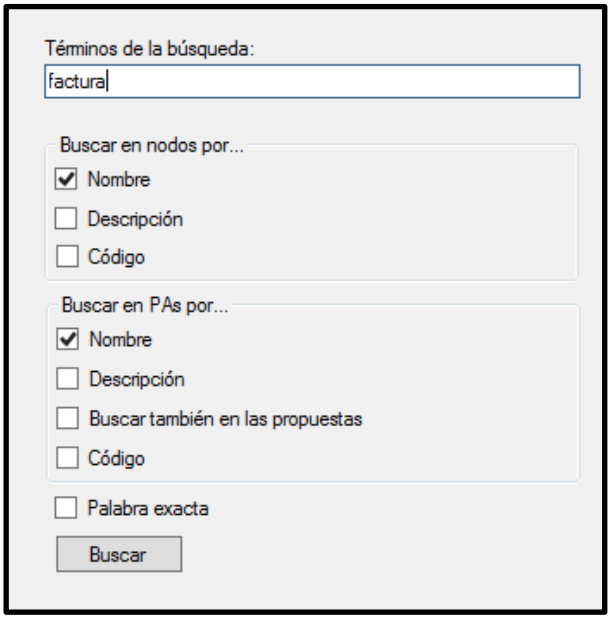


Figura 13: Opciones de búsqueda

Resultado nodos

Selección	IDNodo	Nombre	Índice de relevancia
☐	136	Pestaña Facturas y Cobros (Funcionalidad Común)	559
☐	183	Cálculos de la Facturación de OCFs	559
☐	663	Realizar facturación de OCF	559
☐	917	Creación manual de Facturas de Organizaciones	559
☐	918	Ventana Nueva Factura (Facturas residentes)	559
☐	947	Facturar Concierto Parcial	559
☐	47957	Archivo de Asientos	559
☐	49376	Características comunes de las facturas de una facturación	559
☐	49385	Número por defecto en nueva factura de proveedor	559
☐	49748	Funcionalidad común de la Ventana de Nueva factura	559
☐	49401	Facturar Concierto Centro de Día C. Madrid	487
☐	49555	Facturas de Regularización	487
☐	48849	Controles al crear facturas	487
☐	49052	Asignación Tipo Factura	487
☐	49172	Cálculo y Reparto de Recibos de Facturas	487
☐	47978	Listado Facturas Emitidas agrupado por Grupos de Facturas	487
☐	948	Facturar Concierto Total	487
☐	945	Opciones	487
☐	187	Otros Conceptos Facturables Fljos	487
☐	189	F. General - Borrado de facturas	487
☐	147	Facturación	487
☐	153	Creación manual de facturas	487
☐	271	Facturar SISPAP	487

Figura 14: Resultado de una búsqueda

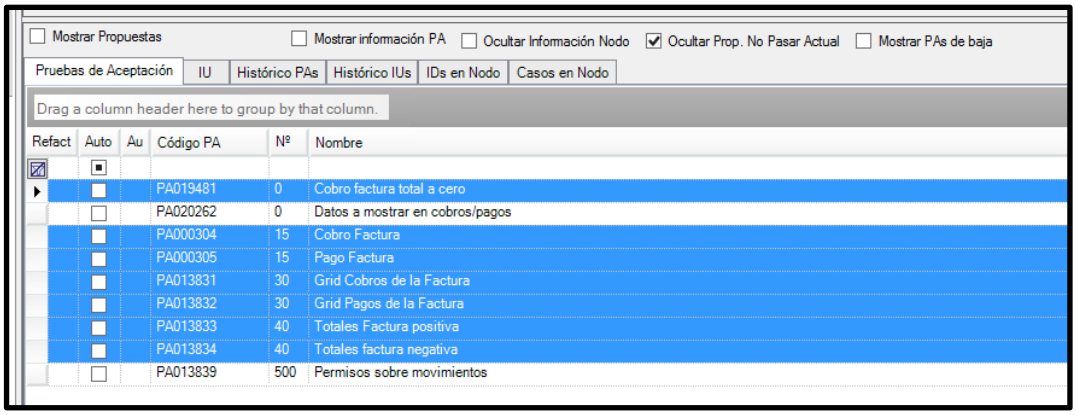


Figura 15: PAs resaltadas dentro de un nodo

La interfaz creada se divide en dos zonas principales. En el panel de la izquierda (Figura 13) están situados el cuadro de búsqueda y todas las opciones que ofrece el buscador para configurar la búsqueda:

- Buscar en los nombres de nodos
- Buscar en las descripciones de nodos
- Buscar por código de nodo
- Buscar por nombre de PA
- Buscar en la descripción de PA
- Buscar en las PA propuestas
- Buscar por código de PA
- Buscar palabras exactas (determina si usar CONTAINS o FREETEXTTABLE)

En el panel de la derecha (Figura 14) hay una tabla donde aparecen los nodos obtenidos en la búsqueda, mostrando su identificador, su nombre y la relevancia. Se pueden seleccionar uno o más nodos o incluso todos con el botón Seleccionar todos de la barra superior. Una vez seleccionados se puede pulsar el botón Resaltar en el árbol, volviendo automáticamente al GR y pudiendo observar en el árbol de nodos del programa los nodos seleccionados resaltados. En caso de haber buscado también pruebas de aceptación, dentro de cada nodo estarán resaltadas las pruebas de aceptación encontradas, diferenciándolas del resto (Figura 15).

5.3.2 Posibles nodos afectados

Primero se debe elegir el nodo sobre el cual queremos analizar el análisis. Para ello seleccionamos el nodo en el árbol de nodos y lanzamos el análisis (Figura 17).

Automáticamente se buscan todos los términos del programa en cuestión que están relacionados con dicho nodo. Esta búsqueda se realiza teniendo en cuenta el nombre de nodo, su descripción, el nombre de todas sus PA y las descripciones de PA.

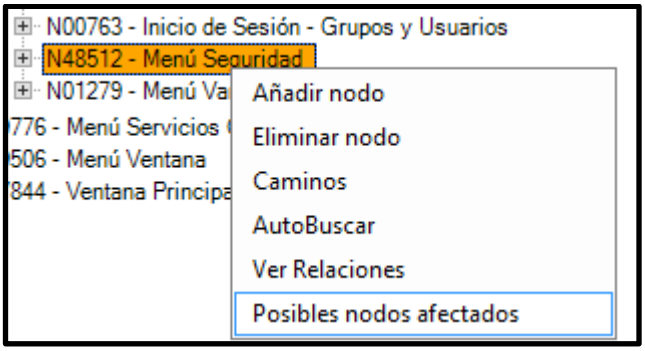


Figura 17: Opción de Posibles Nodos Afectados

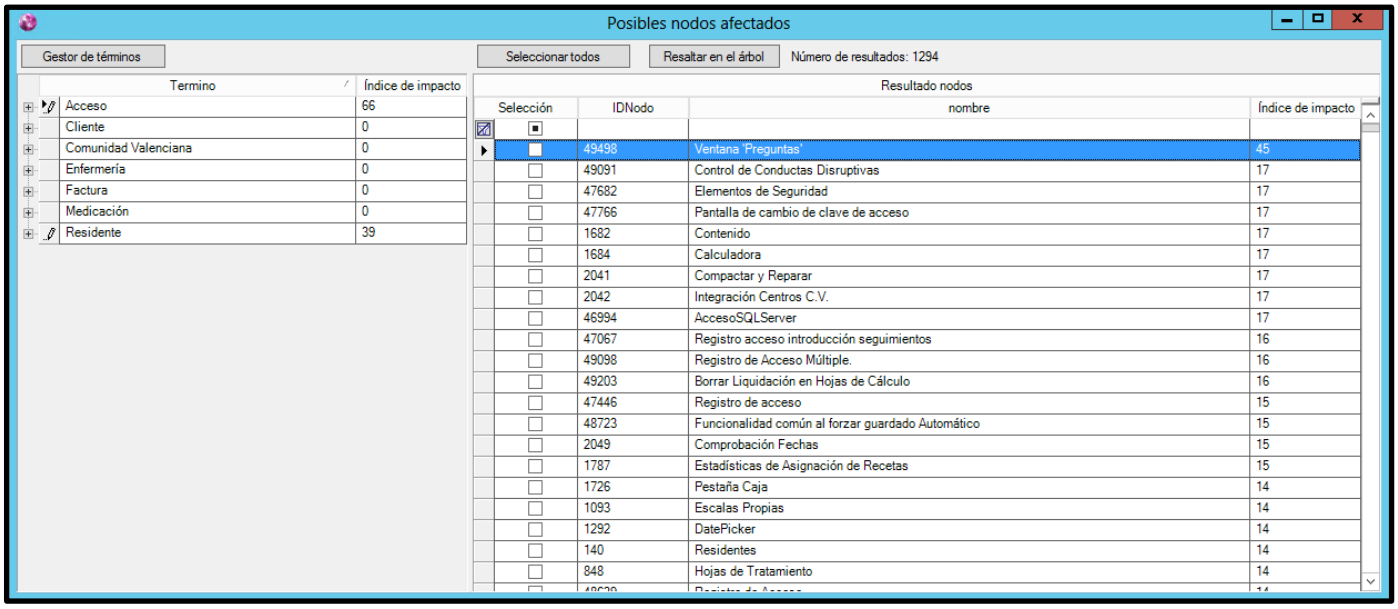


Figura 16: Interfaz principal de nodos afectados

Una vez obtenida la lista de términos que afectan al nodo se lanza una búsqueda por todos los nodos y pruebas de aceptación. Esta búsqueda nos devuelve una lista ordenada (*impact set*), según un índice, de nodos que pueden verse afectados por el nodo a consultar (*change set*).

Los resultados se muestran como en la Figura 16. En la parte izquierda de la ventana se muestra una lista de todos los términos que afectan al *change set* y en la parte derecha se muestra el *impact set* con el índice de relevancia asociado. Este índice se calcula a partir del *rank* proporcionado por FTS.

Los términos de la izquierda también muestran un índice de relevancia para cada nodo (Figura 18). Es decir, se calcula la relevancia que tiene dicho término para el nodo seleccionado a la derecha. Si se despliega la información del término se muestra en qué secciones en concreto aparece el término. Cuanto más grande sea este número, más relevancia tendrá ese término en el nodo. Si es cero, ese término no afecta a ese nodo o a esa sección.

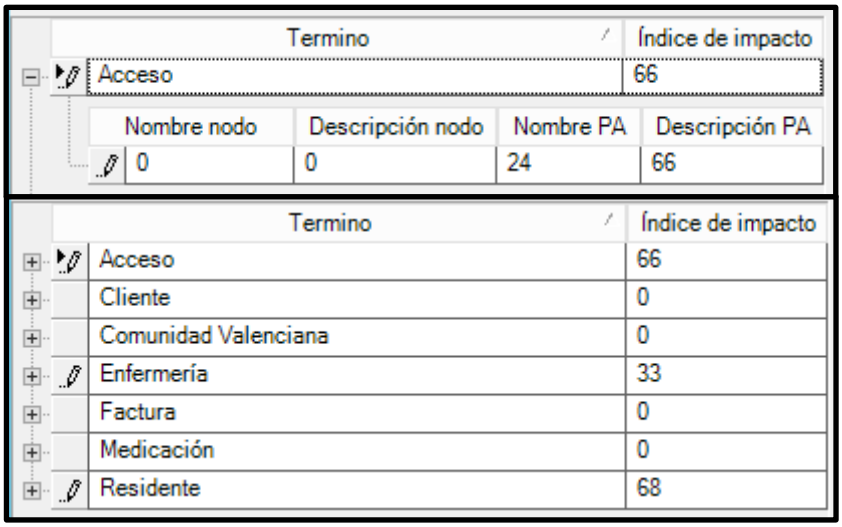
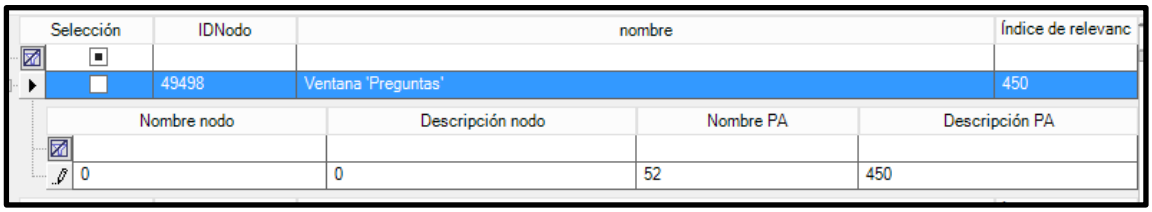


Figura 18: Detalle de los términos que afectan al change set

En el caso de los nodos (

Figura 19, desplegando la fila podemos ver en qué partes y en qué medida



afectan todos los términos a ese nodo.

Figura 19: Detalle de un nodo del impact set

La gestión de los términos se hace mediante un grid que permite su edición, creación y eliminación (Figura 20).

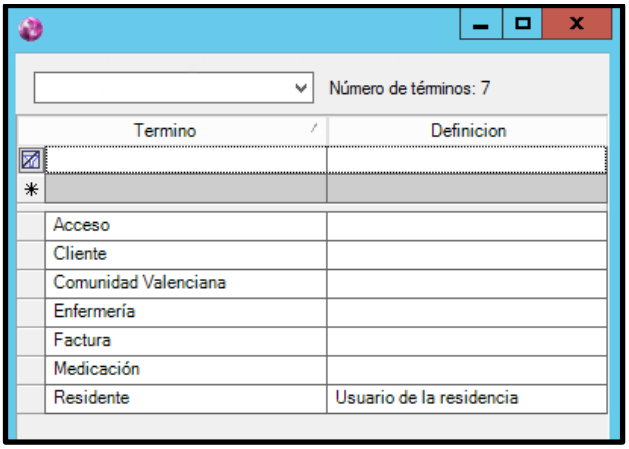


Figura 20: Gestión de términos

6. Uso de las nuevas herramientas

Se ha seleccionado un subconjunto de nodos y sus pruebas de aceptación de la estructura de un programa real sobre el que usar las herramientas de forma ajustada y controlada. Su estructura de nodos es la siguiente:

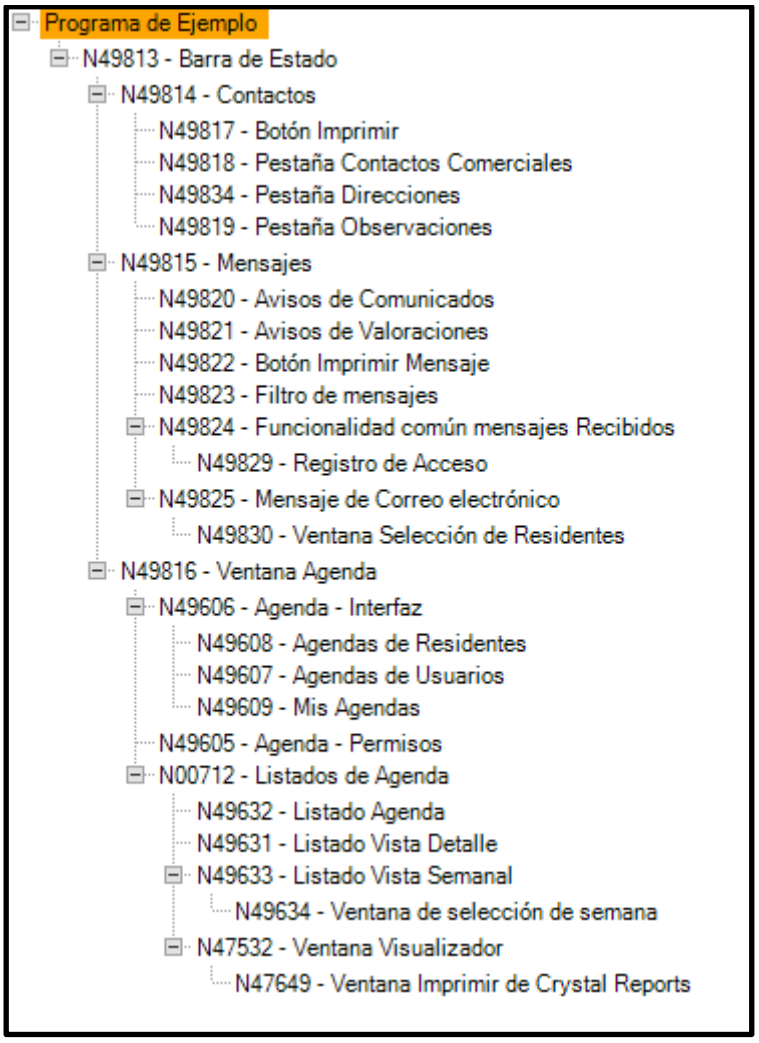


Figura 21: Estructura del programa de ejemplo

Caso 1 (buscador)

En el primer caso se ha utilizado el buscador con la configuración más simple, buscar solo en nombres de nodo para que los resultados sean comparables con la Figura 21. La cadena elegida es “mensaje” que aparece en el nombre de 5 nodos.

Cadena	Búsqueda en:	Resultado esperado		Resultado obtenido	
		Por defecto	Palabra exacta	Por defecto	Palabra exacta
mensaje	Nombre de nodo	Como se puede observar en la Figura 21: 5 nodos (N49815, N49822, N49823, N49824 y N49825)	Como se puede observar en la Figura 21: 2 nodos (N49822, y N49825)	5 nodos (N49815, N49822, N49823, N49824 y N49825) Ver fig.	2 nodos (N49822, y N49825) Ver fig.

Los resultados obtenidos con la configuración por defecto (FREETEXT de FTS) devuelven más nodos que buscando solo por palabra exacta porque tiene en cuenta la cadena "mensaje" y sus variaciones como "mensajes". En cambio buscando la cadena "mensaje" exacta solo encuentra los 2 nodos que tienen esa palabra exacta.

Resultados por defecto obtenidos:

Resultado nodos

Selección	IDNodo	Nombre	Índice de relevancia
☐	49815	Mensajes	143
☐	49822	Botón Imprimir Mensaje	143
☐	49823	Filtro de mensajes	143
☐	49824	Funcionalidad común mensajes Recibidos	143
☐	49825	Mensaje de Correo electrónico	143

Resultados palabra exacta obtenidos:

Resultado nodos

Selección	IDNodo	Nombre	Índice de relevancia
☐	49822	Botón Imprimir Mensaje	160
☐	49825	Mensaje de Correo electrónico	160

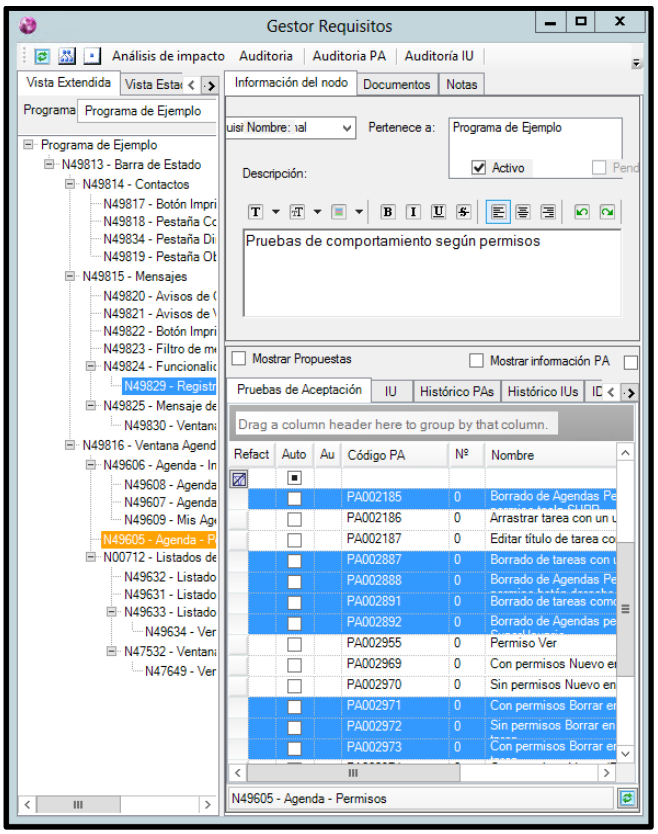
En este caso todos los nodos obtenidos tienen el mismo índice de relevancia porque el impacto de la cadena buscada en los nombres de nodos es el mismo.

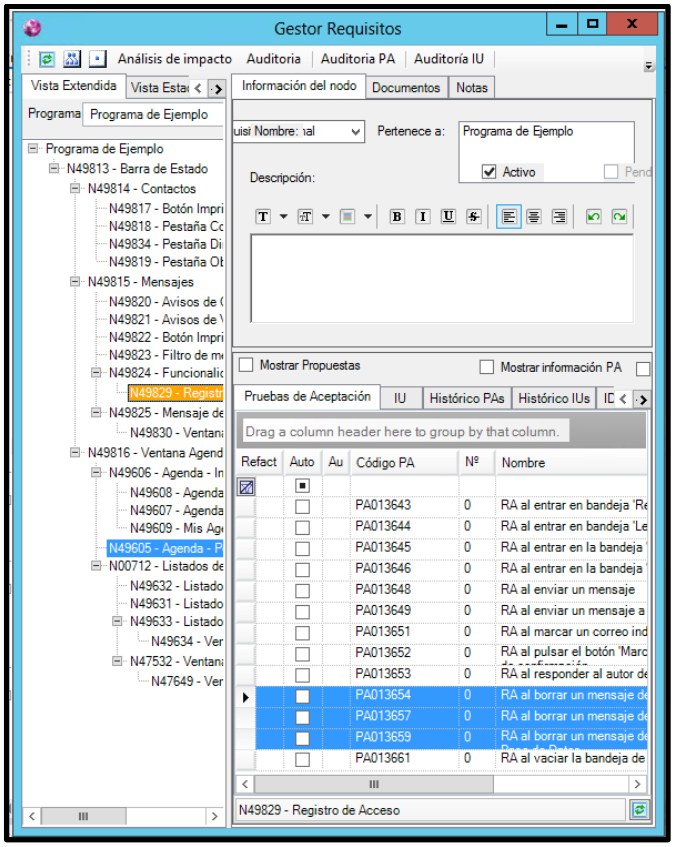
Caso 2 (buscador)

En el segundo caso se ha elegido la cadena “borrado” para comprobar la potencia de FTS respecto a la búsqueda de palabras similares.

Cadena	Búsqueda en:	Resultado esperado		Resultado obtenido	
		Por defecto	Palabra exacta	Por defecto	Palabra exacta
borrado	Nombre PA	Los nodos N49829 y N49605 contienen PAs en cuyos nombres aparece la cadena *borrado* y *borrar*	El nodo N49605 contiene PAs que contienen en el nombre la cadena *borrado*	N49829 y N49605	N49605

En los resultados aparecen nodos cuyas PA contienen en el nombre la palabra “borrado” o alguna similar. En este caso FTS es capaz de encontrar también el verbo “borrar” cuando usamos la búsqueda por defecto como se puede ver en las capturas siguientes.





Caso 3 (nodos afectados)

En el tercer caso se usa la herramienta de obtener los posibles nodos afectados por la modificación de otro nodo. Se introduce únicamente un término en el sistema y nos aseguramos que el nodo que se va a consultar es afectado por dicho término.

Términos en el sistema	Agenda
Nodo (*change set*)	N49609 - Mis agendas
Téminos que afectan al nodo	Agenda
Resultados (*impact set*)	10 nodos

Como resultado se obtienen 10 nodos, en los que el término “agenda” aparece en el nombre del nodo, la descripción del nodo, el nombre de PA o la descripción de la PA. Según el índice de relevancia el nodo más relevante del *impact set* es “Agendas de Usuarios” y el menos relevante “Ventana Agenda”. Este resultado es

Selección	IDNodo	nombre	Índice de relevanc
	49607	Agendas de Usuarios	753
	49608	Agendas de Residentes	724
	49606	Agenda - Interfaz	709
	49632	Listado Agenda	681
	49633	Listado Vista Semanal	649
	49605	Agenda - Permisos	625
	49631	Listado Vista Detalle	423
	49634	Ventana de selección de semana	396
	712	Listados de Agenda	343
	49816	Ventana Agenda	143

debido, entre otros factores al número de ocurrencias del término en cada nodo y la importancia de la palabra.

Los resultados obtenidos pueden considerarse no significativos, pues solo hay un término en el sistema. Para obtener un mejor resultado se debe realizar la búsqueda con más términos, pues un cambio seguramente implique más de un término.

Caso 4 (nodos afectados)

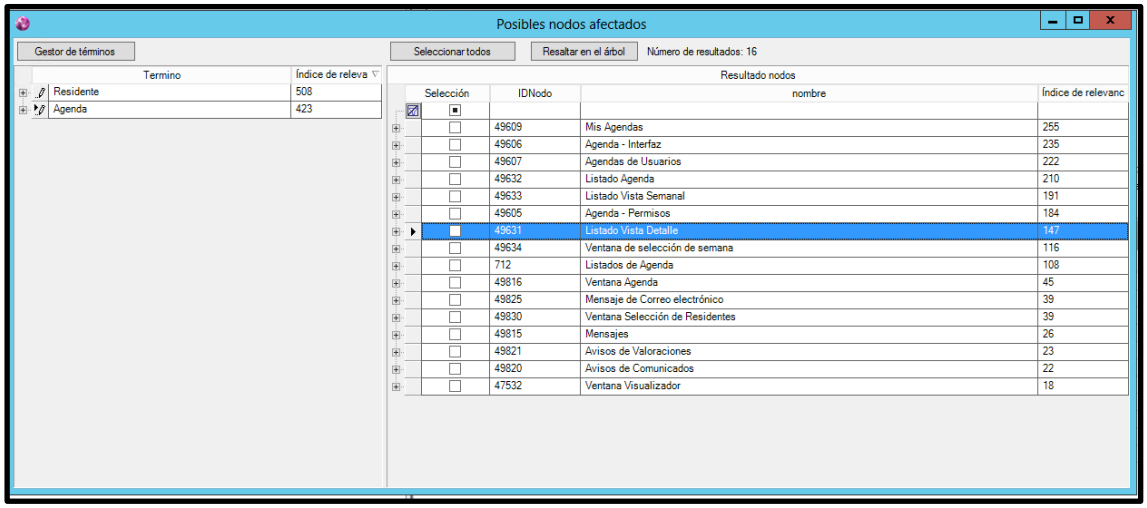
Por último, tenemos un sistema con 3 términos y un nodo al que solo afectan dos de ellos.

Términos en el sistema	Agenda, Enfermería, Residente
Nodo (*change set*)	N49608 - Agendas de Residentes
Téminos que afectan al nodo	Agenda, Residente
Resultados (*impact set*)	16 nodos

En este caso, como hay más términos obtenemos más nodos como resultado y el índice de relevancia adquiere la función de poder hacer una criba de resultados, centrándonos en los que tienen un índice mayor.

Como se ve en la captura el nodo "Listado Vista Detalle" aparece con un índice de 147. Puede resultar interesante saber en qué medida es afectado ese nodo por cada término. Para ello se consulta la tabla de la izquierda y se puede observar que este nodo tiene más relevancia para el término "residente" que para el término "agenda.

Cabe destacar que aunque un nodo sea afectado por más términos que otro, este otro puede tener un índice de relevancia mayor si el impacto que producen los términos sobre él es mayor.



7. Conclusiones

Realizar un buen trabajo en el análisis de impacto previene que la introducción o modificación de un requisito provoque un comportamiento no deseado al resto del producto software. Además, permite dimensionar el esfuerzo de implementación de un cambio.

A nivel de código existen múltiples herramientas que permiten hacer análisis de impacto. Sin embargo, a nivel de requisitos los mecanismos son muy limitados y se centran en mantenimiento manual de matrices de trazabilidad.

En este trabajo se han propuesto mecanismos que usan Full Text Search, que permite hacer búsquedas inteligentes. Esto se ha aplicado sobre los requisitos especificados como Pruebas de Aceptación bajo el enfoque TDRE.

Consideramos que el objetivo de este trabajo se cumplido. Se han desarrollado dos herramientas de apoyo al análisis de impacto integradas en SAPI, la herramienta de gestión de proyectos de la empresa. Las herramientas desarrolladas ya están siendo utilizadas por el grupo de analistas.

Para comprobar la utilidad de las extensiones, hemos tenido reuniones con los analistas, de las cuales se obtuvieron los siguientes comentarios y mejoras:

- El nuevo buscador es más eficiente y aporta mejores resultados que el antiguo.
- Se debería aumentar la granularidad del análisis y no quedarse únicamente a nivel de nodo, si no a nivel de PA, que es donde realmente están definidos los requisitos.
- Aunque FTS ofrece muchas posibilidades a nivel sintáctico, una cadena de texto o término puede que no signifique lo mismo en distintas PA, por lo que se debería buscar la forma de generar términos con significado.
- Poder clasificar las PA mediante etiquetas, que engloben conceptos más generales que los términos.

Además de todas las propuestas de los analistas se va a comenzar a trabajar en los siguientes aspectos:

- Como se comenta en [8], las PA pueden tener distintos tipos de dependencias, interdependencia simple si los pasos de una prueba pueden afectar a las condiciones de otra o dependencia causa-efecto si los resultados de una PA afectan a las condiciones de otra. Esto es

un problema al que el análisis de impacto debe prestarle atención. Otorgando significado a los términos, trabajando sobre PA y partiendo de las herramientas ya creadas, se deberán ampliar para clasificar los distintos tipos de dependencias que nos devuelve el análisis de impacto e incluso poder priorizarlas con más información que únicamente el índice de relevancia.

- Otro campo en el que se desea trabajar es en la visualización y explotación de los datos obtenidos al hacer el análisis de impacto. En el presente trabajo se ha decidido mostrar la información de una manera clásica, en forma de tablas, pero puede ser interesante buscar otro tipo de representación de la información gracias a técnicas de *linked data*.

Finalmente, cabe destacar que este trabajo me ha permitido tener una experiencia profesional importante: vivir de primera mano cómo se aplican las metodologías ágiles, en concreto TDRE, en el entorno empresarial, trabajar en equipo, aprender técnicas como el análisis de impacto y usar tecnologías, como FTS y lenguajes de programación que complementan la formación recibida en el grado.

8. Referencias

[1] R. S. Arnold and S. A. Bohner, “Impact analysis-Towards a framework for comparison,” *Conf. Softw. Maint. Proc.*, pp. 292–301, 1993.

[2] M. Ward, “Change Impact Analysis.” [Online]. Available: http://www.cse.dmu.ac.uk/~mward/msc-se-2015/03-change-impact-analysis.pdf. [Accessed: 21-Nov-2016].

[3] N. Kama, “Change Impact Analysis for the Software Development Phase : State-of-the-art,” *Int. J. Softw. Eng. Its Appl.*, vol. 7, no. 2, pp. 235–244, 2013.

[4] A. Jashki, R. Zafarani, and E. Bagheri, “Towards a More Efficient Static Software Change Impact Analysis Method.”

[5] X. Sun, B. Li, H. Leung, B. Li, and J. Zhu, “Static change impact analysis techniques: A comparative study,” *J. Syst. Softw.*, vol. 109, no. August, pp. 137–149, 2015.

[6] “JRipples tool for Incremental Change.” [Online]. Available: http://jripples.sourceforge.net/.

[7] “FindBugs™ - Find Bugs in Java Programs.” [Online]. Available: http://findbugs.sourceforge.net/.

[8] M. Company Bría, “Análisis de impacto en un proceso de desarrollo centrado en pruebas de aceptación,” Universitat Politècnica de València, 2012.

[9] Marante M., Company M., Letelier P., Suárez F. Gestión de requisitos basada en pruebas de aceptación: Test-Driven en su máxima expresión. XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010), Valencia, Spain, September 7-10, 2010.

[10] F. J. Sanchis Milla, “Definición e implantación de un proceso QA para desarrollo de software,” Universitat Politècnica de València, 2016.

[11] H. Cotter and M. Coles, *Pro Full-Text Search in SQL Server 2008*. Apress, 2009.

[12] J. R. Hamilton and T. K. Nayak, “Microsoft SQL server full-text search,” *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 7–10, 2001.

[13] “CONTAINS (Transact-SQL).” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms187787.aspx.

[14] “FREETEXT (Transact-SQL).” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms176078.aspx.

[15] “How Search Query Results Are Ranked (Full-Text Search).” [Online]. Available: https://technet.microsoft.com/en-us/library/ms142524(v=sql.105).aspx.

[16] “CONTAINSTABLE (Transact-SQL).” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms189760.aspx.

[17] “FREETEXTTABLE (Transact-SQL).” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms177652.aspx.