UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Derivación de requisitos y construcción de trazabilidad entre artefactos del proceso de desarrollo

Autores: Carllinni, Yanela y Datko, Cecilia
Director: Rossi, Gustavo
Codirector: Antonelli, Leandro
Carrera: Licenciatura en Sistemas (Plan 2003)

Resumen

La trazabilidad es una actividad importante en el desarrollo de software. Permite analizar el impacto de cambios de requisitos originados por el cliente como así también estimar el esfuerzo al realizar el mantenimiento. También constituye un elemento importante en el proceso de mejora continua de la organización puesto que es la que permite tener una visión integradora de todos los artefactos que forman parte de cada etapa del proyecto.
El presente trabajo propone un modelo de trazabilidad en el marco de una estrategia de derivación de requerimientos a partir de un conjunto de conceptos del dominio e implementa una herramienta para dar soporte automatizado a tal modelo.
La propuesta abarca el ciclo completo de desarrollo de software a partir del análisis y especificación de requerimientos, cubriendo codificación y testing.

Palabras Claves

Trazabilidad.
Léxico Extendido del Lenguaje (LEL).
Casos de Uso (UC).

Conclusiones

El fundamento de esta tesis se centró en la creación de un modelo de trazabilidad entre artefactos de software que abarque todas las etapas de un proyecto y la elaboración de una aplicación que aplica al modelo propuesto. Así surgió la necesidad de investigar y enriquecer la técnica de derivación de LEL a UC propuesto por [Antonelli 2012], como también de seleccionar una herramienta de tracking que permita integración con un gestor de código fuente y que pueda ser extendida según las necesidades del modelo propuesto.

Trabajos Realizados

Investigación y análisis de estrategia de derivación de requisitos.
Identificación de artefactos del desarrollo de software para considerar incluir en un modelo de traza.
Propuesta de un modelo de trazabilidad entre los productos identificados.
Investigación y selección de herramientas para dar soporte al modelo propuesto.
Creación de plugin en herramienta de tracking para lograr trazabilidad entre artefactos de desarrollo.
Evaluaciones de usabilidad y accesibilidad de la herramienta.

Trabajos Futuros

Obtención de métricas de manera automática en base a la información recolectada por la herramienta.
Generación automática de modelos de requerimientos (diagramas de casos de uso).
Incorporación de otros artefactos de análisis (User Stories).
Restauración a un punto anterior de derivación entre artefactos de análisis.
Enriquecimiento de la estrategia de derivación con el conocimiento suministrado por el usuario
Extensión de MantisBT para que sea posible identificar requerimientos en notas de issue.

Fecha de la presentación: MARZO 2014

Índice General

Índice de figuras 3
Índice de tablas 5
1 Introducción 7
1.1 Motivación 7
1.1.1 Trazabilidad en CMMI 7
1.1.2. Métricas 9
1.2 Background 11
1.2.1 Léxico Extendido del Lenguaje 11
1.2.2 Casos de Uso 15
1.2.3 Herramientas 17
1.3 Conclusión 20
2 Objetivo de la tesis 21
3 Modelo de traza 22
3.1 Esquema de integración 23
3.2 Análisis: Relación LEL - UC 25
3.3 Implementación: Relación UC - Código 35
3.4 Verificación: Relación Incidencia – UC – Código 35
3.5 Conclusión 36
4 Herramienta 37
4.1 Descripción general 37
4.2 Proceso para extender MantisBT 37
4.3 Modelo de traza detallado 39
4.3.1 Análisis: Relación LEL - UC 41
4.3.2 Implementación: Relación UC - Código 45
4.3.3 Verificación: Relación Incidencia – UC – Código 48
4.4 Instalación e integración 51
Paso 1. Configurar MantisBT para integrar con Subversion. 51
Paso 2. Instalar plugin Honey en MantisBT 52
Paso 3. Configurar Subversion 53
4.5 Manual de usuario de Honey 55
4.6 Conclusión 66
5 Usabilidad y accesibilidad del plugin Honey 69

5.1 Usabilidad 69
5.1.1 Atributos de la usabilidad 69
5.1.2 Importancia de la usabilidad 70
5.1.3 Evaluación de la usabilidad 71
5.1.4 Evaluación de la usabilidad en Honey 73
5.2 Accesibilidad 83
5.2.1 La importancia de la accesibilidad 83
5.2.2 Pautas de accesibilidad 83
5.2.3 Accesibilidad aplicada a Honey 86
5.3 Conclusión 86
6 Conclusiones 88
6.1 Resumen 88
6.2 Aspectos positivos y negativos 90
6.3 Futuros trabajos 91
7 Referencias Bibliográficas 92
Anexo A (Herramientas necesarias) 95

Índice de figuras

Figura 1. Relación entre área de proceso “medición y análisis” y el resto de las áreas. 10
Figura 2. Modelo conceptual de traza 22
Figura 3. Trazabilidad que espera lograrse en el trabajo de tesis 23
Figura 4. Integración de artefactos de análisis: LEL-UCs 24
Figura 5. Utilización de servidor de versionado en modelo de trazabilidad 24
Figura 6. Incidencias en herramienta de tracking 24
Figura 7. Derivación LEL - Casos de Usos [Antonelli 2012]. 26
Figura 8. Proceso de derivación de LEL a UC definido por [Antonelli 2012]. 27
Figura 9. Modelo conceptual de traza 36
Figura 10. Modelo de traza detallado 41
Figura 11. Módulo HONEY 42
Figura 12. Definición de pre y post condiciones en Honey 43
Figura 13. Administración de Casos de Uso, reglas y actores no derivados 43
Figura 14. Relación UC manual y Actores derivados 44
Figura 15. Borrado de Actor “Oficina Administrativa” y derivación en LEL 45
Figura 16. Trazabilidad entre artefactos de análisis e implementación 46
Figura 17. Comentario en operación de commit referenciando a una Incidencia 47
Figura 18. Comentario en operación de commit referenciando a un Caso de Uso 48
Figura 19. Path de archivos en las notas de incidencias y Casos de Uso 48
Figura 20. Integración entre artefactos desarrollo y verificación 49
Figura 21. Relación Incidencia – UC 49
Figura 22. Relación Incidencia – UC en Honey 50
Figura 23. Relación Incidencia – Código 50
Figura 24. Plugins instalados en MantisBT 51
Figura 25. Ruta de carpetas donde debe ir el nuevo plugin 52
Figura 26. Administrador de MantisBT donde se visualiza nuevo plugin 52
Figura 27. Visualización de plugin instalado y su correspondiente menú en MantisBT 53
Figura 28. Código del archivo post-commit del servidor SVN para Linux 53
Figura 29. Código del archivo post-commit del servidor SVN para Windows 54
Figura 30. Crear un símbolo en Honey 56
Figura 31. Listado de símbolos de un diccionario LEL en Honey 56
Figura 32. Modificar datos de un símbolo en Honey 57
Figura 33. Eliminar símbolo del LEL de un proyecto 57
Figura 34. Derivar diccionario LEL a Casos de Uso en Honey 58

Figura 35. Crear un caso e uso en Honey 59
Figura 36. Listado de Casos de Uso en Honey 60
Figura 37. Modificar datos de un Caso de Uso en Honey 61
Figura 38. Eliminar símbolo del LEL de un proyecto 62
Figura 39. Crear un actor de Casos de Uso en Honey 63
Figura 40. Búsqueda de actores en un proyecto Honey 63
Figura 41. Modificar un actor en Honey 64
Figura 42. Eliminar un actor en Honey 64
Figura 43. Crear una nueva regla para Casos de Uso en Honey 64
Figura 44. Búsqueda de reglas en un proyecto Honey 65
Figura 45. Modificar una regla en Honey 65
Figura 46. Eliminar una regla en Honey 66
Figura 47. Aporte de Honey al modelo de trazabilidad deseado 67
Figura 48. Características de usabilidad que cumple MantisBT 74
Figura 49. Accesos rápidos en Honey 75
Figura 50. Honey ajustado a principio de heurística de usabilidad 1.1 79
Figura 51. Honey ajustado a principio de heurística de usabilidad 1.4 79
Figura 52. Honey ajustado a principio de heurística de usabilidad 2.4 80
Figura 53. Honey ajustado a principio de heurística de usabilidad 3.2 80
Figura 54. Honey ajustado a principio de heurística de usabilidad 3.3 81
Figura 55. Honey ajustado a principio de heurística de usabilidad 3.5 81
Figura 56. Honey ajustado a principio de heurística de usabilidad 4.1 82
Figura 57. Pantalla de instalación de MantisBT 96

Índice de tablas

Tabla 1. Matriz de trazabilidad 8
Tabla 2. Símbolo de LEL 12
Tabla 3. Símbolo Expediente 28
Tabla 4. Símbolo Expediente Agregado 29
Tabla 5. Símbolo Expediente Principal 29
Tabla 6. Símbolo Expediente Normal 29
Tabla 7. Símbolo Expediente Archivado 29
Tabla 8. Símbolo Agregar 30
Tabla 9. Símbolo Desglosar 30
Tabla 10. Símbolo Caratular 30
Tabla 11. Símbolo Archivar 30
Tabla 12. Símbolo Administrador de mesa de entrada 30
Tabla 13. Símbolo Interesado 31
Tabla 14. Símbolo Administrador de archivo 31
Tabla 15. Símbolo Oficina Administrativa 31
Tabla 16. Símbolo Recibir 31
Tabla 17. Símbolo Consultar 31
Tabla 18. Símbolo Remitir 32
Tabla 19. Caso de Uso derivado del símbolo Archivar 33
Tabla 20. Caso de Uso derivado del símbolo Remitir 33
Tabla 21. Caso de Uso derivado del símbolo Recibir 33
Tabla 22. Caso de Uso derivado del símbolo Agregar 34
Tabla 23. Caso de Uso derivado del símbolo Consultar 34
Tabla 24. Caso de Uso derivado del símbolo Caratular 34
Tabla 25. Caso de Uso derivado del símbolo Desglosar 35
Tabla 26. Contraejemplo de postcondición 42
Tabla 27. Evaluación de usabilidad heurística sobre plugin Honey – Navegación 76
Tabla 28. Evaluación de usabilidad heurística sobre plugin Honey – Herramienta 77
Tabla 29. Evaluación de usabilidad heurística sobre plugin Honey – Consistencia 77
Tabla 30. Evaluación de usabilidad heurística sobre plugin Honey – Reconocer 78
Tabla 31. Evaluación de usabilidad heurística sobre plugin Honey – Errores y Feedback 78
Tabla 32. Evaluación de usabilidad heurística sobre plugin Honey – Ayuda y Documentación 78

Agradecimientos

A mis viejos, los mejores, Juan y Griselda, que me apoyaron en todos mis emprendimientos en la vida, me enseñaron a vivir, a crecer, a valorar el saber y la familia.

A mi sol, mi mejor amiga, mi confidente, Soli, que me acompañó y acompaña en todas. Mi otra mitad y con quien voy de la mano en cada paso de la vida. A mi cuña, Luchin, por cuidarla tanto.

A Manuli, la alegría de casa, y mi cuña, Tefiti, por preguntar, alegrarse y abrazarme en cada paso exitoso de mi carrera.

A Cecy, mi Honey, por ser parte de mi familia, por acompañarme en todo, por creer en mí siempre, y culminar juntas lo que juntas empezamos, la facu. Por darme a mi ahijado, Simón, que me ilumina el alma.

A “La Chula”, mi amiga del alma. A quien debo gran parte de mi hoy en la facultad que termina felizmente y que hizo mi carrera universitaria llevadera y única. Mates, estudio, noches, charlar, risas y amores.

A mi amor, Fer, por su paciencia y comprensión. A quien admiro muchísimo, me inspira a ser mejor persona y con quien quiero compartir el resto de mi vida.

A mis amigos y compañeros de trabajo por todo el apoyo que me brindaron.

Por último agradezco a nuestro co-director, Leandro Antonelli, por toda su paciencia, dedicación y aporte en este trabajo.

Yanela Carllinni

A mis padres, quienes supieron acompañarme y me brindaron todo su apoyo durante el difícil camino de comenzar y terminar una carrera universitaria.

A mi hijo Simón, quien nació al mismo tiempo que la idea de esta tesis y creció junto a ella, compartiendo tardes de mates, noches de desvelo, nervios y ansiedades.

A Yane, mi Honey, compañera y amiga del alma, por luchar a mi lado para lograr este gran objetivo que nos propusimos allá por el 2001 cuando ingresamos a la facultad.

A mi marido y a mi familia política por el aguante.

A mis hermanos, especialmente a Pao quien me brindó su ayuda y siguió paso a paso cada avance.

A mis cuñadas, cuñados y amigos por estar siempre presentes.

A Leandro, por estar al pié del cañón cada vez que lo necesitamos y por ayudarnos en cada paso que dimos.

A mis compañeros de trabajo por ayudarme y alentarme.

A la facu y a todos aquellos que de una u otra manera contribuyeron con su granito de arena para que hoy pueda alcanzar este nuevo logro en mi vida.

Cecilia Datko

1 Introducción

Este capítulo presenta una descripción de los temas de estudio que motivaron a desarrollar el presente trabajo. Se define el concepto de trazabilidad entre artefactos que surgen de las etapas de desarrollo de software y los beneficios de su utilización. Luego se presenta el concepto de medición, el cual se ve enriquecido por la recolección de datos brindada por la traza anteriormente mencionada.

En la sección de Background se describen los artefactos de análisis seleccionados para llevar a cabo la práctica de gestión de requisitos. Luego se detallan las herramientas seleccionadas para relacionar y trazar los artefactos de las distintas etapas del proyecto.

1.1 Motivación

1.1.1 Trazabilidad en CMMI

El Modelo de Madurez de Capacidad Integrado (en inglés Capability Maturity Model Integration, CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Muchas organizaciones valoran medir su progreso llevando a cabo una evaluación (appraisal) y ganando una clasificación del nivel de madurez o de un nivel de capacidad de logro. Este tipo de evaluaciones son realizadas normalmente por las siguientes razones:

- Para determinar qué mejoras se pueden hacer en los procesos de la organización basándose en las mejores prácticas de CMMI.
- Para informar a los clientes externos y proveedores acerca de qué tan bien los procesos de la organización se comparan con las mejores prácticas CMMI.
- Para cumplir los requisitos contractuales de uno o más clientes.

Todos los modelos CMMI contienen áreas específicas de proceso base. Estas áreas de proceso cubren los conceptos básicos que son fundamentales para la mejora de procesos en cualquier área de interés.

Un área de proceso es un grupo de prácticas relacionadas que, cuando se implementan conjuntamente, satisface un conjunto de metas consideradas importantes para mejorar esa área.

El área de proceso de “Gestión de Requisitos”, en particular, describe las actividades para obtener y controlar los cambios a los requisitos, y para asegurar que los planes de proyecto se mantienen actualizados.

A objetivos del presente trabajo, se destaca que una de las prácticas específicas del Área de Gestión de Requisitos consiste en mantener la trazabilidad de los requisitos, esto es: “Una asociación discernible entre dos o más entidades lógicas, tales como requisitos, elementos del sistema, verificaciones o tareas” [CMMI 2010].

Cuando se gestionan adecuadamente los requisitos, se puede establecer la trazabilidad desde un requisito fuente hasta los requisitos de más bajo nivel y también se cubren las relaciones a otras entidades, tales como productos de trabajo intermedios y finales, cambios en la documentación de diseño y planes de pruebas. La trazabilidad se puede mantener tanto vertical como horizontalmente. Por trazabilidad vertical se refiere a la trazabilidad entre objetos diferentes, por ejemplo requerimientos y Casos de Uso. Mientras que por trazabilidad horizontal se refiere a la trazabilidad entre objetos de un mismo tipo, por ejemplo la trazabilidad de los requerimientos entre sí, o de los casos de uso entre sí.

La trazabilidad es particularmente necesaria al evaluar el impacto de los cambios de los requisitos sobre las actividades del proyecto y los productos de trabajo.

Por ejemplo, [Boehm 1997] afirma que si se produce un error en la descripción de los requisitos y se corrige en mantenimiento, el costo de la corrección podría llegar a ser 200 veces mayor comparado con el costo de la misma durante la etapa de requisitos.

La trazabilidad no siempre está automatizada y requiere ser tratada manualmente utilizando hojas de cálculo y bases de datos, entre otras herramientas. Encontramos ejemplos de productos de trabajo en matriz de trazabilidad de los requisitos o sistemas de seguimiento de los requisitos.

Una matriz de trazabilidad, por ejemplo, se utiliza cuando el proyecto es muy grande o complejo y se torna difícil poder saber qué tests ejecutados o diseñados cubren cada una de las especificaciones o requerimientos del proyecto. (Ver Tabla 1)

Imaginemos que tenemos un proyecto con 5 requerimientos (R1 a R5) y hemos diseñado tres casos de prueba (T1, T2 y T3).

- El caso de prueba T1 cubre los requerimientos R1 y R4
- El caso de prueba T2 cubre los requerimientos R3 y R5
- El caso de prueba T3 cubre el requerimiento R3

En este caso la matriz resultante será:

Tabla 1. Matriz de trazabilidad

	T1	T2	T3
R1	X		
R2			
R3		X	X
R4	X		
R5		X	

Viendo la matriz podemos ver claramente dos cosas:

- El requerimiento R3 está probado en 2 casos de prueba.
- El requerimiento R2 no está cubierto.

Gracias a estos datos podemos ver qué partes o módulos del software no están cubiertos y deberían comprobarse por otras pruebas o identificar los requerimientos más críticos para saber si están suficientemente cubiertos (más de un caso de prueba es diseñado y ejecutado para ese requerimiento).

Utilidades de la traza:

- Permite verificar que la funcionalidad esperada ha sido incluida y que no existe funcionalidad innecesaria. Garantiza que todos los requerimientos sean diseñados, y que todos los diseños se codifiquen y se prueben.
- Permite realizar análisis de impacto y mejorar la gestión de cambios. Cuando ocurren cambios en el software, la trazabilidad hace que sea relativamente más fácil evaluar el impacto que los cambios podrían tener en otras partes del proceso de desarrollo.
- Mejora la comunicación y cooperación. Cada integrante del equipo almacena la información relacionada a su área dentro del proyecto de desarrollo y esta información puede ser utilizada por miembros y artefactos de otras áreas del proyecto, asegurando también la contribución de cada individuo.
- Incrementa la recolección de información acerca del proyecto, lo que permite realizar mayores mediciones para mejorar la calidad del producto y evitar desviaciones en costes y plazos, o al menos detectarlas cuanto antes.

Las herramientas actuales están orientadas al tratamiento textual de requisitos, no proveen mecanismos adecuados para configurar la trazabilidad y además presentan problemas de integración entre herramientas para gestión de requisitos y para construcción de software.

1.1.2. Métricas

El área de proceso Medición y Análisis (MA) de CMMI da soporte a todas las áreas, proporcionando prácticas específicas que guían a los proyectos y a las organizaciones para desarrollar y apoyar la capacidad de medición. Los resultados se pueden usar para la toma de decisiones fundamentadas y para la toma de las acciones correctivas apropiadas.

La Figura 1 proporciona una visión general de las interacciones entre el área de proceso de Medición y Análisis y todas las demás.

Figura 1. Relación entre área de proceso "medición y análisis" y el resto de las áreas.

El área de proceso de Medición y Análisis abarca lo siguiente:

- Especificar los objetivos de medición y análisis de tal manera que estén alineados con las necesidades y objetivos de información identificados.
- Especificar las medidas, técnicas de análisis y mecanismos para la recolección de datos, almacenamiento de datos, reportes y retroalimentación.
- Llevar a cabo la recolección, almacenamiento, análisis y reporte de los datos.
- Proporcionar resultados objetivos que puedan usarse para toma de decisiones y para aplicar acciones correctivas apropiadas.

El seguimiento y control del proyecto tiene como objetivo fundamental la vigilancia y recolección de datos de todas las actividades del sistema. Es una de las labores más importantes en todo desarrollo de sistemas, ya que un control adecuado hace posible evitar desviaciones en costes y plazos, o al menos permite una detección temprana.

El seguimiento de un requerimiento, desde que nace hasta su mantenimiento, registrando cada estado o elemento que se produce a partir del mismo, es lo que se conoce como trazabilidad. Una correcta gestión de la trazabilidad permitiría conocer cómo evolucionan los elementos del sistema a lo largo del proceso de desarrollo y cómo se relacionan los diferentes elementos entre sí. Este tipo de información puede utilizarse en diferentes actividades, como el análisis del impacto, la toma de decisiones de diseño y la obtención de métricas que ayuden a mejorar la calidad del producto. Así, sería deseable que cualquier metodología de desarrollo de software proporcionara un soporte adecuado para la gestión de la trazabilidad.

La trazabilidad proporciona métricas fundamentales del proceso tales como cantidad de errores encontrados por requerimiento, artefactos de desarrollo que tienen en común más de un requerimiento, esfuerzo insumido en un error, etc. , que contribuyen a tener una visión más objetiva sobre el estado del proyecto.

La información del proyecto, proceso y producto, que se almacena con el objetivo de obtener mediciones, abarca todas las áreas del proyecto. Cuanta más información se almacene respecto al ciclo de vida del proyecto, mayores y mejores métricas obtendremos. Las mismas ayudan a valorar la calidad de los productos y permiten descubrir y corregir problemas potenciales tales como: no llegar a entregar un requerimiento a tiempo (lo cual puede evitarse si existe histórico del proyecto donde se haya registrado el tiempo insumido en requerimientos de complejidad similar), o conocer la cantidad de código fuente reusado (para conocer el criterio de calidad de reusabilidad al que aplica el producto), o bien, conocer la cantidad de errores encontrados en las sucesivas iteraciones de un producto (permitiendo detectar el área del producto que más atención requiere).

Para medir la calidad de un producto debe ser posible comparar el software (documentos, programas, datos) con una referencia y llegar a una conclusión sobre la calidad.

Mantener la traza permite poder desarrollar métricas para mejorar los proyectos y por ende, facilitar la labor de los analistas del proceso y reducir futuros riesgos.
La generación automática de métricas permitirá a los desarrolladores de software obtener rápidamente un diagnóstico del estado del proyecto en términos de avance, trabajo pendiente, pérdida o cambio de requisitos y prioridades. Con esto se mejorará tanto la calidad de los productos desarrollados como la visibilidad del proceso completo de desarrollo, permitiendo que los riesgos sean identificados en forma temprana. Así se logra tomar decisiones a tiempo y corregir situaciones no deseadas durante el proyecto. Algunas métricas pueden ser: matrices de trazado, vista rápida de requisitos, estadísticas, un árbol de relaciones.

Algunos ejemplos de medidas de uso común son:

- Medidas estimadas y reales del tamaño del producto de trabajo (p. ej. número de páginas).
- Medidas estimadas y reales de esfuerzo y de coste (p. ej. número de horas persona).
- Medidas de calidad (p.ej. número de defectos por grado de severidad).
- Medidas de seguridad de la información (p. ej. número de vulnerabilidades identificadas en el sistema).
- Índice de rendimiento de plazos.
- Densidad de defectos.
- Cobertura de pruebas o de verificación.
- Medidas de fiabilidad (p. ej. tiempo medio entre fallos).

Todas las métricas mencionadas han sido pensadas para facilitar la tarea de los involucrados en el proyecto de desarrollo y ayudarlos a descubrir potenciales problemas en las especificaciones. Si es necesario hacer un cambio en algún requisito, con las herramientas de visualización es más fácil tener una noción de lo que involucra esta modificación dentro del proyecto. La automatización de estos mecanismos de trazabilidad y mediciones pueden ayudar a reducir los costos y tiempos de desarrollo y a disminuir los riesgos asociados al incumplimiento de requisitos. Esto se traduciría en productos confiables y menos costosos.

1.2 Background

1.2.1 Léxico Extendido del Lenguaje

La utilización de glosarios en el proceso de relevamiento de requisitos se ha difundido tanto en el campo del desarrollo como de la investigación dentro de la Ingeniería de Requisitos. Ayuda a comprender mejor el dominio del problema y las verdaderas necesidades del usuario.

El Léxico Extendido del Lenguaje (LEL, en ingles Language Extended Lexicon) es un glosario cuyo objetivo es registrar la definición de los términos que se manejan en el dominio del problema.

Se basa en una simple idea: "entender el lenguaje del problema sin preocuparnos por el problema". [Leite 1989].

Los términos, llamados símbolos en esta estrategia, se definen a través de dos atributos: noción e impactos. La noción describe la denotación del símbolo mientras que los impactos, la connotación.

Cada símbolo pertenece a una de las siguientes cuatro categorías: sujeto, objeto, verbo o estado. (Ver Tabla 2).

Tabla 2. Símbolo de LEL

Nombres	Identifica todos los nombres con los cuales el símbolo es identificado dentro del dominio	**Tipo**	Verbo/Estado/ Objeto/Sujeto
Sinónimos	Identifica con que otros nombres puede identificarse al símbolo		
Noción	Describe qué es el símbolo		
Impacto	Describe qué hace el símbolo / que le hacen		

Mientras se describen los símbolos deben seguirse dos principios básicos:

- Principio de circularidad
- Principio del vocabulario mínimo

El primer principio se basa en describir al máximo un nuevo símbolo, mientras que el segundo se basa en la utilización de la mínima cantidad de símbolos externos al LEL.

La construcción del LEL comienza por obtener información del dominio. A partir de esta información se elabora una lista de símbolos que se deben conocer para entender el lenguaje del dominio. Estos símbolos se deben clasificar para poder definirlos en forma consistente. Luego de clasificarlos, se los define y como producto de la definición se pueden descubrir sinónimos, por lo cual se deben reorganizar los símbolos. La información debe ser validada por los expertos del dominio y controlada por el ingeniero de requerimientos. Si algún nuevo símbolo debe ser definido, se repite el proceso. [Leonardi 2001] [Antonelli 1999].

Los pasos del proceso son los siguientes:

- **Identificación de las fuentes de información:** las fuentes de información para la construcción del LEL son dos: las personas y los documentos. Cada uno tiene ventajas sobre el otro. Los documentos ofrecen un medio concreto, autocontenido e invariable.
  El ingeniero de requerimientos puede leer y volver a leer los documentos para una selección minuciosa de los términos. Los textos se pueden analizar en detalle. Por otra parte, las personas pueden aportar a través de entrevistas mayor información. Una característica propia de la expresión oral de los seres humanos es que en forma

inconsciente utilizan el principio de circularidad. En cambio en la descripción escrita, para una narrativa más entretenida, se introducen muchos sinónimos, lo que puede dificultar el proceso de identificación de símbolos. Es aconsejable utilizar ambas fuentes de información.

Las personas que es aconsejable entrevistar son:

- Las más mencionadas dentro del entorno del sistema.
- Las que toman las decisiones.
- Los responsables del proyecto.
- Los supervisores del proceso.

Por su parte, los documentos que más información brindan son:

- Descripciones textuales del sistema.
- Manuales que definen los procedimientos operacionales.

- **Generación de la lista de símbolos:** una estrategia aconsejable para generar la lista de símbolos consiste en comenzar con una lectura de la documentación para hacer un análisis preliminar del lenguaje del dominio. Con este análisis se obtiene una lista inicial y se adquiere información para poder llevar a cabo entrevistas. Es aconsejable realizar entrevistas semiestructuradas y estructuradas para acotar el lenguaje. Ya que si se comenzara con entrevistas libres, el volumen de información sería muy difícil de manejar.

  Los símbolos que se deben tomar para la lista inicial están dados por palabras o frases que se utilizan con mucha frecuencia, o aquellas que parezcan estar fuera de contexto. El motivo de elegir las palabras o frases que se utilizan con mucha frecuencia es claro: el objetivo del LEL es capturar el lenguaje de un dominio. Los términos más utilizados son significativos en el dominio, por lo tanto deben estar definidos. En cambio los términos que parezcan estar fuera de contexto se deben a que tienen un significado propio en el dominio, distinto del tradicional. Estos términos, con más razón deben estar definidos.

  Cabe destacar que los símbolos no tienen por qué ser palabras individuales. Pueden ser palabras o frases. La razón es que en un lenguaje se puede encontrar un grupo de palabras, en donde cada una de ellas pueda tener cierto significado, pero si se las combina en una frase tienen otro significado.
  Si bien en la etapa de definición es cuando se encuentran los sinónimos, la tarea puede comenzar en esta etapa.

- **Clasificación de la lista de símbolos:** el objetivo de categorizar los símbolos es poder administrar mejor al conjunto. Además, cada categoría determina la forma en que se debe definir cada símbolo. Así se logra una definición consistente y uniforme.

  Para clasificar los símbolos se parte de una clasificación general:

  - Sujeto: Elemento activo dentro del dominio que realiza acciones utilizando objetos. El sujeto (persona, máquina, dispositivo) puede llegar a pasar por distintos estados.

- Verbo: Acción que realiza un sujeto, servicio que brinda un objeto, desencadenante para pasar de un estado a otro.
- Objeto: Elemento pasivo con los cuales se realizan acciones que puede pasar por distintos estados.
- Estado: Situación en la que se encuentra un sujeto o un objeto.

Esta es una clasificación inicial que en función del dominio se puede especializar. La especialización puede darse por ejemplo en los sujetos. Un sistema de administración de empleados de una empresa tiene distintos tipos de empleados. Aquellos en actividad, en condición de retiro y retirados. Y estos a su vez pueden contener subcategorías, por lo cual se necesita explotar la categoría inicial.
Esta categorización sirve para agrupar los símbolos relacionados, lo que permite encontrar sinónimos.
Además, la clasificación permite construir un LEL uniforme. Para cada categoría se debe definir su forma de describir la noción e impacto de los símbolos. Así se asegura que los símbolos son descriptos consistentemente y que es posible contrastarlos, para un mejor entendimiento del lenguaje.

- **Descripción de símbolos:** los símbolos se describen a partir del conocimiento obtenido de la lectura de la documentación y de las entrevistas. Cada símbolo se describe siguiendo las pautas establecidas en la clasificación de los términos. Esta descripción inicial luego será validada y controlada.

  Algunas pautas generales para la descripción de los símbolos, independientemente de la clasificación a la que corresponden, son las siguientes:

  - Cada noción e impacto debe ser descripto con oraciones breves y simples.
  - Las oraciones deben responder a los principios de circularidad y de vocabulario mínimo.
  - Las nociones e impacto de un símbolo pueden representar diferentes puntos de vista o pueden ser complementarios.

  Con la descripción de símbolos se facilita la tarea de encontrar sinónimos. Sin embargo, en la validación posterior, se despeja cualquier duda respecto de los sinónimos.

- **Validación:** consiste en validar la correctitud del LEL contra el usuario. Debido a la gran cantidad de símbolos que pueden estar definidos en el LEL, es impracticable realizar una validación completa y exhaustiva. Sin embargo, es posible mantener sesiones de entrevistas estructuradas para aclarar dudas.

- **Control:** el proceso de control es el que realiza el ingeniero de requerimientos por sus propios medios. El control no tiene que ver con la correctitud de la información del LEL, por el contrario está relacionado con la estructura:

  - Todos los símbolos deben estar definidos. Aquellos de la lista inicial y también los que aparecen en la descripción de los símbolos de la lista inicial.
  - Todos los símbolos deben estar dentro de la clasificación correspondiente. Debido a que la clasificación se realiza antes de la descripción, en esta última se puede

descubrir que un símbolo se colocó en la categoría incorrecta, por lo cual se lo debe reubicar y verificar su definición.
- La descripción de los símbolos debe corresponderse con la de la categoría a la que pertenecen.
- El punto de vista para la descripción de los símbolos debe ser uniforme.
- No deben dejarse sinónimos definidos como símbolos independientes.

En resumen, el proceso de construcción del LEL es el siguiente:

- Entrevistas con los stakeholders (cualquier involucrado en el sistema). En las primeras se deja hablar libremente al stakeholder.
- Luego de la primera entrevista se genera una lista de símbolos candidatos, con las palabras que el stakeholder utiliza más frecuentemente. Esta lista también puede construirse mediante la utilización de documentos.
- Clasificación y descripción de los símbolos.
- Validación y control del LEL.

Antonelli en [Antonelli 2012] ofrece técnicas para obtener especificaciones de requerimientos a partir de conceptos de dominio, símbolos definidos en el diccionario LEL. Estas técnicas consisten en la derivación de LEL a User Stories y de LEL a Casos de Usos.
En este trabajo de tesis se crea un modelo de trazabilidad centrado en ciclos de vida iterativos e incrementales, por tal motivo se utiliza la obtención de Casos de Uso a partir de símbolos del diccionario LEL.

1.2.2 Casos de Uso

Un Caso de Uso (en inglés Use Case, UC) es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un Caso de Uso se denominan actores. En el contexto de ingeniería del software, un UC es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.
Un Caso de Uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada UC proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Normalmente para su redacción, se evita el empleo de jergas técnicas, prefiriendo en su lugar un lenguaje más cercano al usuario final. En ocasiones, se utiliza a usuarios sin experiencia junto a los analistas para el desarrollo de Casos de Uso.
Los diagramas de Casos de Uso, por ejemplo, sirven para visualizar de una manera gráfica y sencilla, la comunicación y el comportamiento de un sistema mediante su interacción con usuarios y/o sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los Casos de Uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de Casos de Uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.
Cada Caso de Uso se centra en describir cómo alcanzar una única meta o tarea de negocio. Desde una perspectiva tradicional de la ingeniería de software, un UC describe una característica del sistema. Para la mayoría de los proyectos de software, esto significa que quizás a veces es

necesario especificar diez o centenares de Casos de Uso para definir completamente el sistema. El grado de formalidad de un proyecto particular de software y la etapa del proyecto influenciarán en el nivel del detalle requerido en cada Caso de Uso.
Los UCs pretenden ser herramientas simples para describir el comportamiento del software o de los sistemas. Cada uno contiene una descripción textual de todas las maneras en que los actores previstos podrían trabajar con el software o el sistema. Los Casos de Uso no describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican cómo se implementará; simplemente muestran los pasos que el actor sigue para realizar una tarea.

Un Caso de Uso debe:

- Describir una tarea del negocio que sirva a una meta de negocio.
- Tener un nivel apropiado del detalle.
- Ser bastante sencillo como para que un desarrollador lo elabore en un único intento.

Situaciones que pueden darse:

- Un actor se comunica con un Caso de Uso (si se trata de un actor primario[1] la comunicación la iniciará el actor, en cambio si es secundario[2], el sistema será el que inicie la comunicación).
- Un Caso de Uso extiende otro Caso de Uso.
- Un Caso de Uso utiliza otro Caso de Uso.

Ventajas**:**

- La técnica de Caso de Uso tiene éxito en sistemas interactivos, ya que expresa la intención que tiene el actor (su usuario) al hacer uso del sistema.
- Como técnica de extracción de requerimiento permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema, evitando que la gente especializada en informática dirija la funcionalidad del nuevo sistema basándose solamente en criterios tecnológicos.

Limitaciones:

- Los Casos de Uso pueden ser útiles para establecer requisitos de comportamiento, pero no establecen completamente los requisitos funcionales ni permiten determinar los requisitos no funcionales.

[1] Un actor primario es quien tiene la meta que cumple el caso de uso.

[2] Un actor secundario es quien realiza algo para cumplir esa meta a pedido del caso de uso.

- Los Casos de Uso deben complementarse con información adicional como reglas de negocio, requisitos no funcionales, diccionario de datos que complementen los requerimientos del sistema. Sin embargo la ingeniería del funcionamiento especifica que cada caso crítico del uso debe tener un requisito no funcional centrado en el funcionamiento asociado.

Los Casos de Uso pueden ser especificados con diferentes niveles de abstracción:

- **Caso de Uso breve**: consiste en algunas sentencias que resumen el objetivo del caso de uso.
- **Caso de Uso casual**: consiste en algunos párrafos que describen la secuencia de las acciones principales del Caso de Uso.
- **Caso de Uso completo**: consiste en un documento formal basado en una plantilla detallada con varias secciones. Este es el más conocido como Caso de Uso. Incluye una descripción del escenario principal como también alternativas o variantes. Luego hay una descripción de cómo implementar la funcionalidad. Siguen descripciones del estado del dominio antes y después de la ejecución de este Caso de Uso, que definen las condiciones que deben ser validadas previas a la ejecución y la situación que debe ser alcanzada luego de la ejecución del Caso de Uso. Estos atributos son las pre y postcondiciones. Finalmente hay una descripción del rol que el usuario debe cumplir mientras interactúa con la funcionalidad, tal como la descripción de acciones que el sistema y el usuario realizan durante la ejecución de la funcionalidad.

Los Casos de Uso utilizan una estructura mucho más completa y compleja que el Léxico Extendido del Lenguaje. La estrategia que se plantea en el presente trabajo transforma la información que el LEL captura, en diferentes Casos de Uso para luego enriquecerlos con actores, reglas, relaciones e interfaces.

1.2.3 Herramientas

Esta sección describe las herramientas seleccionadas que se utilizaron para dar soporte automatizado al modelo conceptual creado en el presente trabajo.

1.2.3.1 Herramienta de tracking: MantisBT

MantisBT es una aplicación web para gestionar las tareas en un equipo de trabajo. Siendo utilizada para registrar incidencias, realizar seguimiento y mejoras continuas y constantes.

Compatible con Chrome, Firefox, Safari, Opera e IE 7 o posteriores. Publicada bajo los términos de la Licencia Pública General de GNU. Está desarrollada en PHP pudiéndose utilizar en cualquier plataforma que lo corra (Windows, Linux, Mac, Solaris, AS400/i5, etc). Funciona con las bases de datos MySQL (Oracle en experimento), MS SQL y PostgreSQL.

Se destaca por su facilidad y flexibilidad de instalar y configurar, permitiendo:

- Configurar las transiciones de estados.
- Introducir diferentes perfiles.
- Definir los filtros de búsqueda.
- Definir las páginas de incidencias.

Permite manejar múltiples proyectos en una misma instancia de la herramienta, así como definir subproyectos y categorías dentro de los mismos.

Permite la definición de *roadmaps* para los proyectos, reportes y gráficos.

Respecto a las incidencias, estas pueden ser de diferentes tipos (Bugs, Task, etc). Se registra el historial de cambios de las mismas y posee un módulo SVN que permite realizar la traza entre las incidencias el código fuente.

Posee un módulo de correo electrónico, permitiendo configurar el envío de notificaciones en transiciones o cambios de estado en los workflows.

Lo completa y potente que es esta herramienta, comparada con las que se encuentran hoy en el mercado, tanto comerciales como libres, más su amplia documentación y plugins para conectarse con otras herramientas (repositorio, entornos de desarrollo, wiki, etc.), sumado a la facilidad de instalación, configuración, administración y extensión, son los motivos que nos llevan a elegir esta herramienta.

1.2.3.2 Repositorio: TortoiseSVN (Cliente)

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Apache Subversion. Esto significa que TortoiseSVN administra archivos y directorios a lo largo del tiempo.

Los archivos se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de archivos ordinario, con la excepción de que registra todos los cambios que se hayan hecho a sus archivos y directorios. Esto le permite al usuario recuperar versiones antiguas de sus archivos y examinar la historia de cómo y cuándo cambiaron sus datos, y quién hizo el cambio.

TortoiseSVN Corre bajo licencia GNU (General Public License); fue diseñado para brindar simplicidad y facilidad en su utilización. Es una herramienta compleja diseñada principalmente para desarrolladores y se caracteriza por tener una curva de aprendizaje casi nula.

Características:

- **Integración con la consola de Windows:** TortoiseSVN se integra con la consola de Windows (por ejemplo, el explorador). Esto significa que el usuario puede seguir trabajando con las herramientas que ya conoce y no tiene que cambiar a una aplicación diferente cada vez que necesite las funciones del control de versiones.

- **Íconos superpuestos:** El estado de cada carpeta y archivo versionado se indica por pequeños íconos superpuestos. De esta forma, el usuario puede ver fácilmente el estado en el que se encuentra su copia de trabajo.

- **Interfaz Gráfica de Usuario:** Cuando se listan los cambios realizados a un archivo o carpeta, puede acceder a una revisión para ver los comentarios de ese commit. Se puede ver también una lista de archivos modificados.

- **Fácil acceso a los comandos de Subversion:** Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.

  Dado que TortoiseSVN es un cliente de Subversion, se listan a continuación algunas de las características propias de Subversion:

- **Versionado de carpetas:** CVS sólo controla la historia de archivos individuales, pero Subversion implementa un sistema "virtual" de archivos versionados que toma en cuenta los cambios en todos los árboles de directorios en el tiempo. Los archivos y los directorios están versionados. Como resultado, hay comandos reales en el lado del cliente, como mover y copiar, que operan en archivos y directorios.

- **Commits atómicos**: Un commit puede entrar al repositorio completamente, o no entrar en absoluto. Esto permite a los desarrolladores generar y confirmar cambios como unidades lógicas.

- **Metadatos versionados:** Cada archivo y carpeta tiene un conjunto invisible de "propiedades" adjunto. Se puede crear y almacenar cualquier par de clave/valor que se desee. Las propiedades se versionan en el tiempo, igual que el contenido de los archivos.

- **Elección de capas de red:** Subversion tiene una noción abstracta del acceso al repositorio, haciendo que se puedan implementar nuevos mecanismos de red fácilmente.

1.2.3.3 Repositorio: VisualSVN (Servidor)

El Servidor VisualSVN permite instalar y administrar un servidor de versionado. Es accesible para pequeñas empresas y usuarios corporativos y se basa en estándares abiertos, ofreciendo una sólida estabilidad, seguridad y rendimiento.

Sus principales características son:

- Trabajos out-of-the-box.
- Active Directory Single Sign –On.
- Delegación de la Gestión del repositorio.
- Consola de gestión de gran alcance.
- Administración remota del servidor.

- Acceso y registro operacional.

Sus principales ventajas son:

- Fácil de instalar, configurar y mantener. Se distribuye como un único paquete de instalación con las últimas versiones de todos los componentes necesarios. El proceso de instalación y actualización es muy simple.

- Proporciona una consola de gestión muy útil llamada Administrador de servidor VisualSVN y permite administrar su servidor de Subversion sin tener que lidiar con los archivos de configuración y las herramientas de línea de comandos.

- Se basa en estándares abiertos y no trata de introducir un sistema de control de versión propietario. Los repositorios de Subversion están almacenados en el formato estándar y el servidor es accesible por los clientes estándar de Subversion como TortoiseSVN.

1.3 Conclusión

Lograr trazabilidad entre artefactos que surgen del proceso de desarrollo y las métricas que se pueden obtener a partir de la información recolectada por la traza, son los motivos que nos llevaron a desarrollar el presente trabajo.

Se seleccionaron herramientas específicas que permiten mantener y acrecentar la traza durante las distintas etapas del ciclo de vida del proyecto:

- Para la etapa de análisis se investigó una técnica llamada Léxico Extendido del Lenguaje, utilizada para elicitar requerimientos. Conjuntamente, se analizó una estrategia de derivación de LEL a Casos de Uso para su automatización. Luego se estudió el concepto de Casos de Uso y su utilización para modelar los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso.

- Para la etapa de implementación se indagó sobre las herramientas TortoiseSVN y VisualSVN utilizadas para el control de versionado.

- Por último, para la etapa de verificación, se evaluó la herramienta MantisBT utilizada para registrar y realizar seguimiento de incidencias.

2 Objetivo de la tesis

Independientemente del modelo de ciclo de vida seleccionado para desarrollar software, todos tienen en común las actividades de Análisis, Implementación y Verificación.

Mantener actualizada la traza representa un reto importante desde el punto de vista de la calidad ya que debe existir alguna manera de hacer un seguimiento de cómo se ha elaborado lo requerido durante todo el proceso de desarrollo.

La trazabilidad entre artefactos de software ayuda a obtener mayor información durante las etapas del ciclo de vida, facilitando la obtención de mediciones para analizar impactos ante posibles cambios y analizar, mejorar y evitar riesgos durante iteraciones actuales y futuras del proyecto. Cuanta más información se almacene respecto al ciclo de vida del proyecto, mayores y mejores métricas obtendremos.

Existen distintas herramientas para lograr trazabilidad, las cuales mantienen registro y relación de artefactos desde el relevamiento hasta la construcción del código ejecutable.

El objetivo de esta tesis es brindar un modelo de trazabilidad así como también una herramienta para soportarlo. Con este fin, se realiza la investigación, implementación y posterior incorporación del Léxico Extendido del Lenguaje (utilizado para capturar el lenguaje de la aplicación con el que se escribirán los requerimientos), Casos de Uso y la derivación del LEL a Casos de Uso, a una herramienta de tracking previamente seleccionada (MantisBT). Existe un módulo que integra la herramienta de tracking con un servidor de versionado que permite lograr la relación unidireccional entre las incidencias y el código fuente. El presente trabajo extiende ese mecanismo de integración para conseguir la trazabilidad bidireccional.

Siguiendo la misma lógica utilizada para crear la traza entre incidencias y código fuente, se incorpora la posibilidad de relacionar el código ejecutable con el Caso de Uso al cual alude. Esto se logra extendiendo la conexión con del repositorio donde se encuentra el código fuente.

3 Modelo de traza

El presente capítulo describe el proceso para lograr un modelo de trazabilidad entre distintos artefactos de un proyecto de software. Dicha descripción se realiza detallando la relación entre los artefactos generados en cada etapa del desarrollo de un producto de software.

Se presenta un modelo integral que muestra los artefactos y las relaciones necesarias entre estos, siguiendo por el detalle de la relación entre cada uno de ellos: LEL, UC, Incidencia y Código fuente (Ver Fig. 2).

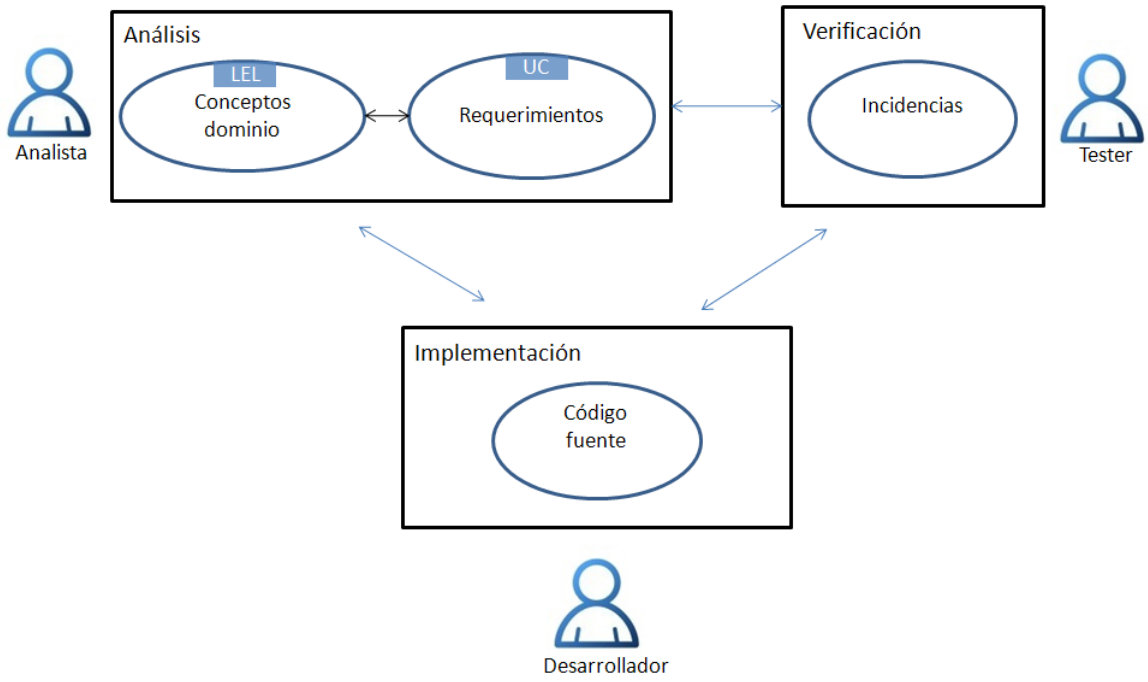


Figura 2. Modelo conceptual de traza

El modelo planteado abarca:

- La creación y administración del diccionario LEL (artefacto de análisis) y de Casos de Uso (artefacto de análisis) a cargo del rol analista dentro del producto.
- La implementación de una estrategia de derivación para transformar automáticamente LEL a UCs.
- La creación y administración de artefactos de verificación (incidencias), facilitada por la herramienta de tracking, a cargo de quien tenga el rol tester del producto.
- La creación de artefactos de implementación (código fuente) es responsabilidad del rol desarrollador dentro del producto.
- La relación bidireccional entre la etapa de análisis y verificación que ocurre cuando se reporta una incidencia sobre un requerimiento determinado y a su vez cuando se indica en

un requerimiento particular que el mismo presenta una incidencia previamente reportada en la misma herramienta.

- La relación bidireccional entre la etapa de implementación con las etapas de requerimientos y verificación queda a cargo del desarrollador y de la funcionalidad que pueda llegar a ofrecer la herramienta de tracking seleccionada.

3.1 Esquema de integración

En la presente tesis se plantea un esquema de integración y trazabilidad con dos partes fundamentales:

(i) Interrelacionar las herramientas de uso cotidiano, utilizadas en cada etapa de cualquier proyecto de software, a modo de hacer un seguimiento de los requerimientos desde su nacimiento hasta su mantenimiento en el producto de software.

(ii) La integración de los artefactos mencionados anteriormente: la captura de requerimientos utilizando LEL y Casos de uso, el código fuente y las incidencias halladas durante la verificación de implementación de los requerimientos de análisis en una misma herramienta: la herramienta de tracking.

La integración de los artefactos mencionados en la parte (ii) se logra con las interrelaciones entre las herramientas mencionadas en (i) alcanzando de esta forma, la trazabilidad buscada en el presente trabajo de tesis (Ver Fig. 3).

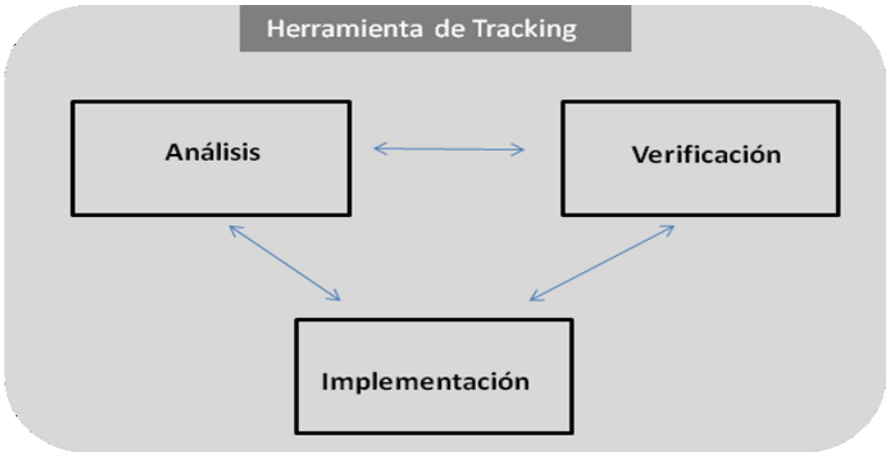


Figura 3. Trazabilidad que espera lograrse en el trabajo de tesis

A continuación se detalla cada etapa perteneciente al modelo planteado en la figura y las relaciones entre ellas.

La etapa de análisis comprende los conceptos del dominio capturados mediante LEL y los requerimientos definidos a través de Casos de Uso. Además, esta etapa incluye la integración automática entre ambos artefactos brindada por el enriquecimiento de la estrategia de derivación propuesta por [Antonelli 2012] para transformar los conceptos de dominio (LEL) en requerimientos

(Casos de Uso), lo que permite conocer también, qué símbolo le da origen a cada Caso de Uso derivado. (Ver Fig. 4)

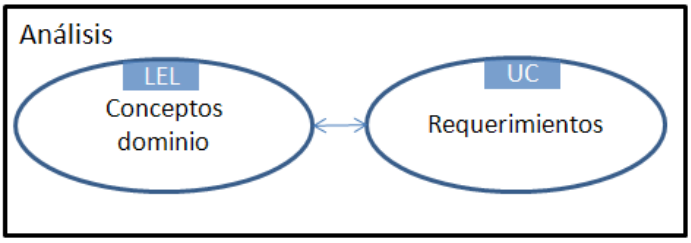


Figura 4. Integración de artefactos de análisis: LEL-UCs

La etapa de implementación queda a cargo de los desarrolladores. Sin embargo, en este modelo se plantea la utilización de un servidor de versionado donde pueda alojarse el código fuente. (Ver Fig. 5)

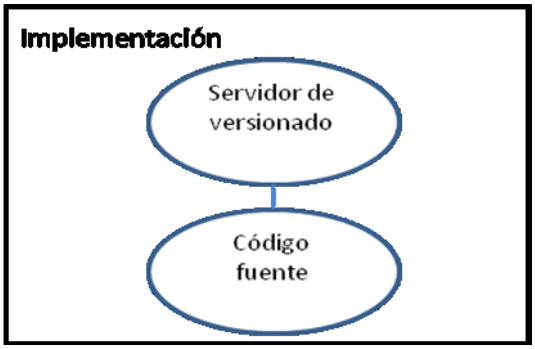


Figura 5. Utilización de servidor de versionado en modelo de trazabilidad

La etapa de verificación comprende a las incidencias que puedan surgir durante las pruebas de la implementación de los requerimientos. Las mismas se registran en la herramienta de tracking para su correspondiente seguimiento. (Ver Fig. 6)

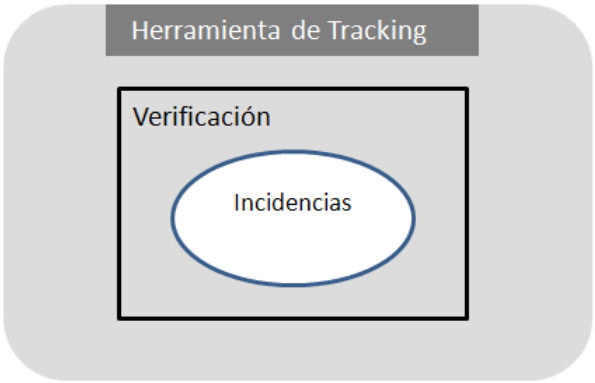


Figura 6. Incidencias en herramienta de tracking

3.2 Análisis: Relación LEL - UC

Antonelli en [Antonelli 2012] propone una estrategia para obtener un conjunto preliminar de requerimientos funcionales, especificados a través de UCs, a partir del LEL. Estos requerimientos podrán ser enriquecidos y validados en fases posteriores así como también podrán incorporarse nuevos requerimientos que no formen parte de la estrategia de derivación, es decir, que no tienen origen a partir del diccionario LEL.

La esencia de la relación se basa en adquirir conceptos del dominio mediante LEL y lograr los Casos de Uso que deriven del LEL.

Los Casos de uso representan interacciones dentro de la aplicación, mientas que los símbolos de tipo verbo representan acciones dentro del alcance de la misma. Esto muestra que cada símbolo de tipo verbo se puede derivar en un Caso de Uso.

La identificación del UC se estable mediante el nombre del verbo. El objetivo en cada símbolo de de tipo verbo se representa con la noción, la cual se utilizará para describir el objetivo en el contexto del Caso de Uso. El comportamiento (Impactos) en un símbolo de tipo verbo, describe las acciones necesarias para lograr el objetivo, de esta forma se puede derivar este comportamiento para definir el escenario principal del UC.

Es necesario definir un rol que lleve a cabo las acciones del UC. Los símbolos de tipo sujeto están relacionados naturalmente con los verbos, dado que los impactos en el sujeto incluyen las acciones que los verbos realizan. De aquí puede observarse que los actores primarios son los sujetos que realizan acciones indicadas por los verbos.

Los símbolos de tipo estado son los candidatos para ser las pre y postcondiciones del UC. Para ello, es necesario identificar los estados del LEL que están relacionados (mencionan en sus impactos) con el verbo que representa al UC y así derivarlos a las pre y postcondiciones del Caso de Uso. Puede ocurrir que LEL no tenga estados relacionados con cada verbo, en estas situaciones las pre y postcondiciones quedaran sin definir en el UC.

En la figura que se muestra a continuación (Ver Fig. 7) puede observarse, a modo de ejemplo, cómo queda generado un Caso de Uso a partir de un símbolo del diccionario LEL.

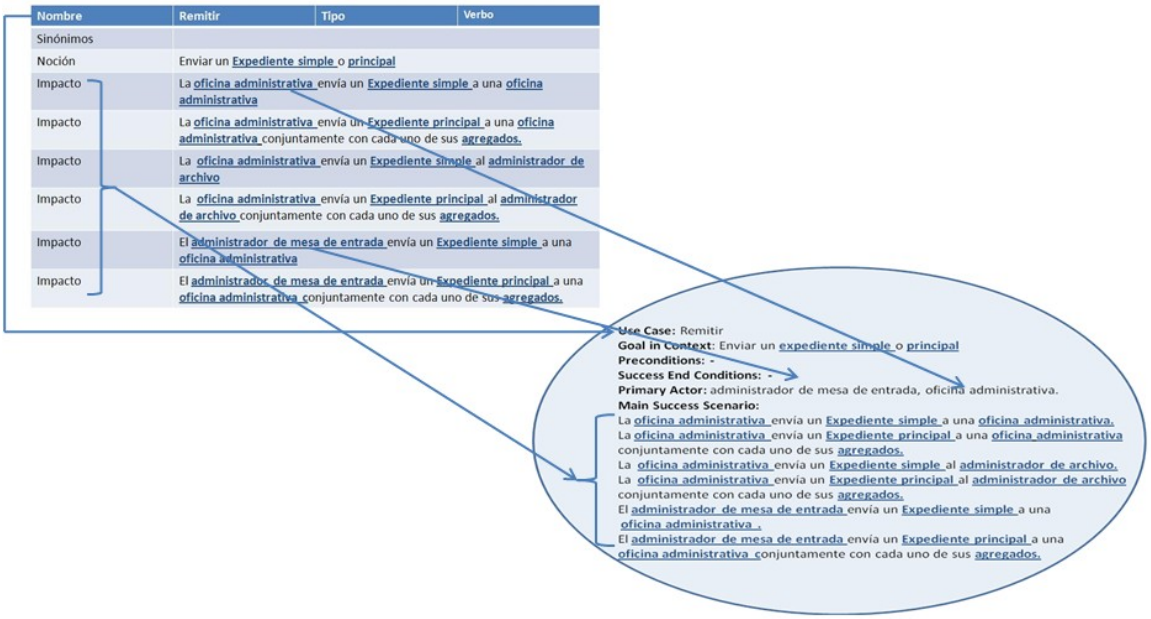


Figura 7. Derivación LEL - Casos de Usos [Antonelli 2012].

El proceso de derivación está constituido por los siguientes pasos:

- Cada símbolo de tipo **Sujeto** pasará a ser un Actor.
    - El nombre del símbolo representará el nombre del Actor.
    - La noción del símbolo pasará a ser la descripción del Actor.
- Se identifican todos símbolos de tipo verbo que haya en el diccionario LEL.
- Cada símbolo de tipo **Verbo** pasará a ser un Caso de Uso.
    - El nombre del símbolo representará el nombre del Caso de uso.
    - La noción del símbolo pasará a ser el objetivo del Caso de Uso.
    - Los impactos de cada símbolo de tipo verbo pasarán a formar parte del escenario principal del Caso de Uso.
    - Los símbolos de tipo sujeto que se encuentran en los impactos de cada símbolo de tipo verbo pasarán a ser los actores principales del Caso de Uso.
    - Los símbolos de tipo estado, que entre sus impactos mencionen a un símbolo de tipo verbo son candidatos a ser pre o postcondiciones. Si se menciona a otro estado, estamos en presencia de una postcondición, caso contrario, una precondición.

    En la Figura 8 puede observarse el proceso completo antes detallado.

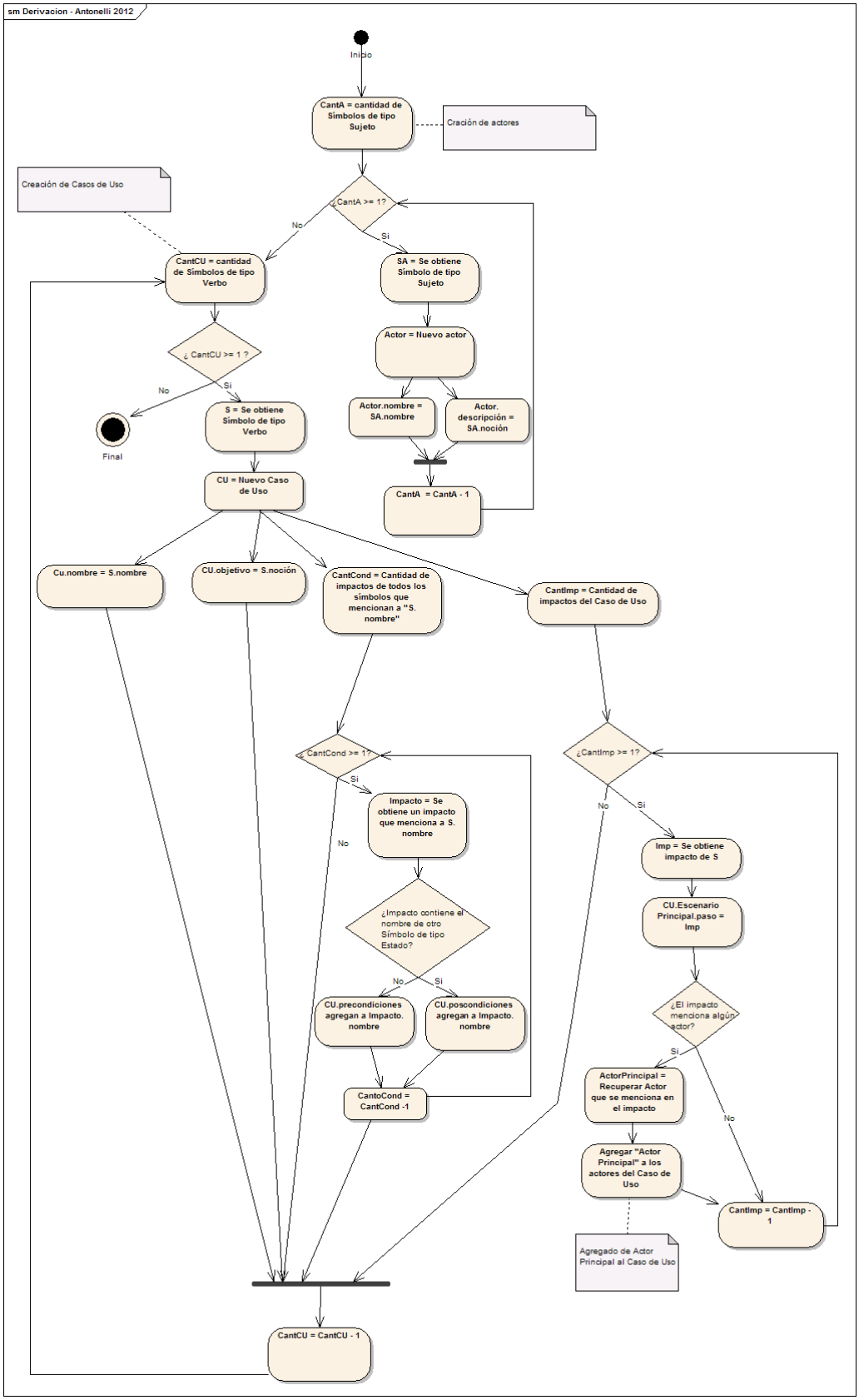


Figura 8. Proceso de derivación de LEL a UC definido por [Antonelli 2012].

Para contribuir a la comprensión del marco teórico de esta investigación, se creó un ejemplo práctico utilizando C&L, una herramienta online que permite registrar diccionarios LEL. Luego, se derivó el diccionario obtenido utilizando la técnica de derivación LEL – UC propuesta por [Antonelli 2012]

El ejemplo corresponde a un Sistema de **Seguimiento de Expedientes** para la resolución de trámites administrativos de la salud pública basados en la DECRETO-LEY 7.647/70.

Los expedientes son caratulados en la Mesa de entradas por su administrador para luego ser remitidos a las distintas oficinas administrativas donde los reciben, los controlan, y eventualmente los agregan o desglosan para nuevamente ser remitidos y continuar su camino administrativo.

Una vez que el expediente termina su correspondiente recorrido es enviado al archivo donde el administrador de archivo se encarga de culminar el trámite archivándolo.

Durante cualquier etapa del proceso, un interesado puede consultar el estado del trámite.

El sistema es de una complejidad razonable y presenta una nutrida diversidad de situaciones, razón por la que resultó adecuado para aplicar la metodología.

Los símbolos encontrados son los siguientes: Expediente, Expediente Normal, Expediente Principal, Expediente Agregado, Expediente archivado, Agregar, Desglosar, Recibir, Remitir, Caratular, Archivar, Consultar, Administrador de Mesa de Entradas, Interesado, Oficina Administrativa y Administrador de Archivo. (Ver Tablas 3 a 18).

Tabla 3. Símbolo Expediente

Nombre	Expediente	Tipo	Objeto
Sinónimos	Trámite, Carátula, Trámite Administrativo, Expediente simple		
Noción	Información organizada en fojas que requiere la aprobación de más de una **oficina administrativa**		
Impacto	El **administrador de mesa de entrada** **carátula** un **Expediente**		
Impacto	El **administrador de mesa de entrada** **remite** un **Expediente**		
Impacto	La **oficina administrativa** **recibe** un **Expediente**		
Impacto	La **oficina administrativa** **remite** un **Expediente**		
Impacto	La **oficina administrativa** **agrega** un **expediente simple** a otro **expediente simple** o a un **expediente principal** . El primero pasa a estar **agregado** y el segundo pasa a estar **principal** si aún no lo estaba.		
Impacto	La **oficina administrativa** **desglosa** un **expediente agregado** a un **expediente principal** volviendo a ser **expediente simple** y el **expediente principal** pasa a **simple** si ya no tiene más **agregados**		
Impacto	El **interesado** **consulta** en qué **oficina administrativa** se encuentra un **Expediente**		
Impacto	El **administrador de archivo** **recibe** un **Expediente**		
Impacto	El **administrador de archivo** **archiva** un **Expediente**		

Tabla 4. Símbolo Expediente Agregado

Nombre	Expediente Agregado	Tipo	Estado
Sinónimos			
Noción	**Expediente** que se encuentra contenido dentro de un **expediente principal**.		
Impacto	El **interesado** **consulta** en qué **oficina administrativa** se encuentra un **expediente agregado**		
Impacto	El **interesado** **consulta** cuál es el **expediente principal** de un **expediente agregado**		

Tabla 5. Símbolo Expediente Principal

Nombre	Expediente Principal	Tipo	Estado
Sinónimos			
Noción	**Expediente** que contiene **expedientes agregados**.		
Impacto	El **interesado** **consulta** en qué **oficina administrativa** se encuentra un **expediente principal**		
Impacto	El **interesado** **consulta** los **expedientes agregados** a un **expediente principal**		
Impacto	La **oficina administrativa** **desglosa** un **expediente agregado** al **expediente principal** volviendo a ser **expediente simple** y el **expediente principal** pasa a **simple** si ya no tiene más **agregados**		
Impacto	La **oficina administrativa** **remite** al **expediente principal** conjuntamente con cada **expediente agregado** que contenga.		
Impacto	El **administrador de mesa de entrada** **remite** al **expediente principal** conjuntamente con cada **expediente agregado** que contenga.		
Impacto	La **oficina administrativa** **recibe** al **expediente principal** conjuntamente con cada **expediente agregado** que contenga		
Impacto	El **administrador de archivo** **recibe** al **expediente principal** conjuntamente con cada **expediente agregado** que contenga		
Impacto	El **administrador de archivo** **archiva** al **expediente principal** conjuntamente con cada **expediente agregado** que contenga		

Tabla 6. Símbolo Expediente Normal

Nombre	Expediente Normal	Tipo	Estado
Sinónimos			
Noción	**Expediente** que se encuentra iniciado y habilitado para realizarle cualquier tipo de operación mientras no sea **archivado**		
Impacto	El **interesado** consulta en qué **oficina administrativa** se encuentra un **expediente normal**		

Tabla 7. Símbolo Expediente Archivado

Nombre	Expediente Archivado	Tipo	Estado
Sinónimos			
Noción	**Expediente** que se encuentra finalizado e inhabilitado para realizarle cualquier tipo de operación.		
Impacto	El **interesado** **consulta** en qué **oficina administrativa** se encuentra un **expediente archivado**		

Tabla 8. Símbolo Agregar

Nombre	Agregar	Tipo	Verbo
Sinónimos			
Noción	Incluir un **Expediente simple** dentro de otro **Expediente simple** o **principal**.		
Impacto	La **oficina administrativa** debe tener el **Expediente** en su oficina para realizar esta operación		
Impacto	La **oficina administrativa** incluye un **Expediente simple** dentro de otro **Expediente simple** pasando a estar el primero **agregado** y el segundo **principal**		
Impacto	La **oficina administrativa** incluye un **Expediente simple** dentro de un **expediente principal** pasando a estar el primero **agregado**		

Tabla 9. Símbolo Desglosar

Nombre	Desglosar	Tipo	Verbo
Sinónimos			
Noción	Separar un **Expediente agregado** de su **Expediente principal**		
Impacto	La **oficina administrativa** debe tener el **Expediente** en su oficina para realizar esta operación		
Impacto	La **oficina administrativa** separa un **Expediente agregado** de su **Expediente principal** pasando a estar el primero **simple** y el segundo **simple** si ya no tiene más **agregados**		

Tabla 10. Símbolo Caratular

Nombre	Caratular	Tipo	Verbo
Sinónimos			
Noción	Dar inicio a un nuevo **Expediente Simple**		
Impacto	El **administrador de mesa de entrada** busca si el número de **expediente** a asignar pertenece al intervalo de números ya asignados		

Tabla 11. Símbolo Archivar

Nombre	Archivar	Tipo	Verbo
Sinónimos			
Noción	Dar fin a un **Expediente simple** o **principal**		
Impacto	El **administrador de archivo** debe tener el **Expediente** en su oficina para realizar esta operación		
Impacto	El **administrador de archivo archiva** un **Expediente simple**		
Impacto	El **administrador de archivo archiva** un **expediente principal** conjuntamente con cada uno de sus **agregados**		

Tabla 12. Símbolo Administrador de mesa de entrada

Nombre	Administrador de mesa de entrada	Tipo	Sujeto
Sinónimos			
Noción	Persona que trabaja en la mesa de entradas del ministerio de salud		
Impacto	El **administrador de mesa de entrada caratula** un **expediente**		
Impacto	El **administrador de mesa de entrada remite** un **expediente**		

Tabla 13. Símbolo Interesado

Nombre	Interesado	Tipo	Sujeto
Sinónimos			
Noción	Empleado de una **oficina administrativa** del ministerio de salud		
Impacto	El **interesado** **consulta** por un número de **Expediente**		

Tabla 14. Símbolo Administrador de archivo

Nombre	Administrador de archivo	Tipo	Sujeto
Sinónimos			
Noción	Persona que trabaja en la oficina de archivo del ministerio de salud		
Impacto	El **administrador de archivo** **archiva** un **Expediente**		
Impacto	El **administrador de archivo** **recibe** un **Expediente**		

Tabla 15. Símbolo Oficina Administrativa

Nombre	Oficina administrativa	Tipo	Sujeto
Sinónimos			
Noción	Organismo encargado de verificar la información contenida dentro de un **Expediente**		
Impacto	La **oficina administrativa** **recibe** un **Expediente**		
Impacto	La **oficina administrativa** **remite** un **Expediente**		
Impacto	La **oficina administrativa** **agrega** un **Expediente** a otro **Expediente**		
Impacto	La **oficina administrativa** **desglosa** un **expedientes agregado** de un **expediente principal**		

Tabla 16. Símbolo Recibir

Nombre	Recibir	Tipo	Verbo
Sinónimos	Acuse de Recibo		
Noción	Obtener un **Expediente simple** o **principal** que ha sido enviado		
Impacto	La **oficina administrativa** se notifica de que tiene un nuevo **Expediente**		
Impacto	La **oficina administrativa** marca como recibido a un **Expediente Simple**		
Impacto	La **oficina administrativa** marca como recibido a un **expediente** **principal** conjuntamente con cada uno de sus **agregados**		
Impacto	El **administrador de archivo** marca como recibido a un **Expediente Simple**		
Impacto	El **administrador de archivo** marca como recibido a un **expediente** **principal** conjuntamente con cada uno de sus **agregados**		

Tabla 17. Símbolo Consultar

Nombre	Consultar	Tipo	Verbo
Sinónimos			
Noción	Obtener seguimiento de un **Expediente** particular		
Impacto	El **interesado** ingresa un número de **expediente** y obtiene un listado con las oficinas y fechas por las que fue transitando el **Expediente simple**		
Impacto	El **interesado** ingresa un número de **expediente** y obtiene un listado con las oficinas, **expedientes agregados** y fechas por las que fue transitando el **expediente principal**		
Impacto	El **interesado** ingresa un número de **expediente** y obtiene un listado con las oficinas, el **expediente principal** y fechas por las que fue transitando el **expediente agregado**.		

Tabla 18. Símbolo Remitir

Nombre	Remitir	Tipo	Verbo
Sinónimos			
Noción	Enviar un **Expediente simple** o **principal**		
Impacto	La **oficina administrativa** envía un **Expediente simple** a una **oficina administrativa**		
Impacto	La **oficina administrativa** envía un **expediente principal** a una **oficina administrativa** conjuntamente con cada uno de sus **agregados**		
Impacto	La **oficina administrativa** envía un **Expediente simple** al **administrador de archivo**		
Impacto	La **oficina administrativa** envía un **expediente principal** al **administrador de archivo** conjuntamente con cada uno de sus **agregados**		
Impacto	El **administrador de mesa de entrada** envía un **Expediente simple** a una **oficina administrativa**		
Impacto	El **administrador de mesa de entrada** envía un **expediente principal** a una **oficina administrativa** conjuntamente con cada uno de sus **agregados**		

Se identifican 7 Casos de Uso, uno por cada verbo

1. Remitir
2. Recibir
3. Archivar
4. Caratular
5. Desglosar
6. Agregar
7. Consultar

Existen 4 roles bajo los cuales opera el Usuario

1. Oficina administrativa
2. Administrador de Archivo
3. Interesado
4. Administrador de Mesa de Entrada

Estados de un expediente

1. Expediente Normal
2. Expediente Archivado
3. Expediente Principal
4. Expediente Agregado

Casos de Uso derivados a partir del LEL:

Tabla 19. Caso de Uso derivado del símbolo Archivar

Caso de Uso: Archivar **Objetivo**: Dar fin a un Expediente simple o principal **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Administrador de archivo **Escenario Principal** El administrador de archivo debe tener el Expediente en su oficina para realizar esta operación. El administrador de archivo archiva un Expediente simple. El administrador de archivo archiva un expediente principal conjuntamente con cada uno de sus agregados.

Tabla 20. Caso de Uso derivado del símbolo Remitir

Caso de Uso: Remitir **Objetivo**: Enviar un Expediente simple o principal **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Oficina administrativa, Administrador de Mesa de Entrada **Escenario Principal** La oficina administrativa envía un Expediente simple a una oficina administrativa. La oficina administrativa envía un expediente principal a una oficina administrativa conjuntamente con cada uno de sus agregados. La oficina administrativa envía un Expediente simple al administrador de archivo. La oficina administrativa envía un expediente principal al administrador de archivo conjuntamente con cada uno de sus agregados. El administrador de mesa de entrada envía un Expediente simple a una oficina administrativa. El administrador de mesa de entrada envía un expediente principal a una oficina administrativa conjuntamente con cada uno de sus agregados

Tabla 21. Caso de Uso derivado del símbolo Recibir

Caso de Uso: Recibir **Objetivo**: Obtener un Expediente simple o principal que ha sido enviado **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Oficina administrativa, Administrador de archivo **Escenario Principal:** La oficina administrativa se notifica que tiene un nuevo Expediente. La oficina administrativa marca como recibido a un Expediente Simple La oficina administrativa marca como recibido a un expediente principal conjuntamente con cada uno de sus agregados. El administrador de archivo marca como recibido a un Expediente Simple. El administrador de archivo marca como recibido a un expediente principal conjuntamente con cada uno de sus agregados

Tabla 22. Caso de Uso derivado del símbolo Agregar

Caso de Uso: Agregar **Objetivo**: Incluir un Expediente simple dentro de otro Expediente simple o principal. **Precondiciones:** -- **Postcondiciones:** -- **Actores Principales:** Oficina administrativa **Escenario Principal:** La oficina administrativa debe tener el expediente en su oficina para realizar esta operación. La oficina administrativa incluye un Expediente simple dentro de otro Expediente simple pasando a estar el primero agregado y el segundo principal. La oficina administrativa incluye un Expediente simple dentro de un expediente principal pasando a estar el primero agregado.

Tabla 23. Caso de Uso derivado del símbolo Consultar

Caso de Uso: Consultar **Objetivo**: Obtener seguimiento de un Expediente particular **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Interesado **Escenario Principal:** El interesado ingresa un número de expediente y obtiene un listado con las oficinas y fechas por las que fue transitando el Expediente simple. El interesado ingresa un número de expediente y obtiene un listado con las oficinas, expedientes agregados y fechas por las que fue transitando el expediente principal. El interesado ingresa un número de expediente y obtiene un listado con las oficinas, el expediente principal y fechas por las que fue transitando el expediente agregado

Tabla 24. Caso de Uso derivado del símbolo Caratular

Caso de Uso: Caratular **Objetivo**: Dar inicio a un nuevo expediente simple **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Administrador de Mesa de Entrada **Escenario Principal** El administrador de Mesa de Entrada busca si el número de expediente a asignar pertenece al intervalo de números ya asignados

Tabla 25. Caso de Uso derivado del símbolo Desglosar

Caso de Uso: Desglosar **Objetivo**: Separar un Expediente agregado de su Expediente principal **Precondiciones**: -- **Postcondiciones**: -- **Actores Principales**: Oficina administrativa **Escenario Principal** La oficina administrativa debe tener el expediente en su oficina para realizar esta operación. La oficina administrativa separa un Expediente agregado de su Expediente principal pasando a estar el primero simple y el segundo simple si ya no tiene más agregados

3.3 Implementación: Relación UC - Código

La relación que se establece entre UCs y código quedará a cargo del desarrollador, quien mediante una plataforma de desarrollo implementará cada uno de los Casos de Uso definidos en la etapa de análisis.

La herramienta de tracking seleccionada ofrece un módulo de conexión con Subversion para el control de versionado. Esta funcionalidad permite registrar los comentarios realizados en la operación de commit durante el desarrollo, y que mencionan a una incidencia en particular. Del mismo modo, se realizó una extensión para almacenar los comentarios de los commits que mencionan UCs.

Al momento de guardar en el servidor de versionado el código que implementa un Caso de Uso, el desarrollador deberá mencionar su identificador para obtener así la relación.

3.4 Verificación: Relación Incidencia – UC – Código

Las incidencias que surjan durante la etapa de verificación (testing) deben ser reportadas en una herramienta de tracking y queda a cargo del tester realizar esta tarea.

Al momento de guardar en el servidor de versionado el código que resuelve una incidencia, el desarrollador deberá indicar el identificador de la incidencia para obtener la traza entre la incidencia y el código fuente.

La posibilidad de conocer las incidencias que se detectaron sobre un requerimiento en particular, así como también qué artefactos de desarrollo se involucraron para resolver incidencias reportadas, dependen de la funcionalidad que pueda brindar la herramienta de tracking utilizada para el seguimiento de cada proyecto. En el presente trabajo de tesis se realizó la extensión de la herramienta de tracking seleccionada con la posibilidad de mantener la traza entre Incidencia, Caso de Uso y Código.

3.5 Conclusión

Para cada una de las etapas del ciclo de vida de un proyecto se utilizan herramientas que colaboran en la construcción de cada artefacto de software.

Lograr la interconexión entre las herramientas utilizadas en las distintas etapas del proyecto permite alcanzar la integración buscada entre:

- los requerimientos (artefactos de la etapa de análisis),
- el código fuente (artefacto de la etapa de implementación)
- las incidencias (artefacto de la etapa de verificación).

Para contribuir a la comprensión del marco teórico de esta tesis, se presenta un ejemplo concreto del diccionario LEL utilizando C&L, una herramienta online para tal fin. Luego, se derivó dicho diccionario a Casos de Uso utilizando la técnica de derivación propuesta por [Antonelli 2012].

La utilización de la estrategia de derivación entre artefactos de la etapa de análisis permite transformar conceptos del dominio (LEL) en requerimientos (UCs). Esta relación unidireccional puede convertirse en una traza bidireccional si el Caso de Uso conociera al símbolo que lo generó. Más adelante en este trabajo, se describe cómo se llevó a cabo la extensión en la herramienta de tracking seleccionada para lograr la traza bidireccional entre los artefactos de análisis.

Relacionar los artefactos de desarrollo con la funcionalidad (Caso de Uso) o Incidencia registrada en la herramienta de tracking es tarea exclusiva del desarrollador. Sin embargo, en el presente trabajo se investigó cómo se realiza esta tarea y la posibilidad de mantener la traza bidireccional, a objetivos de lograr un modelo de traza que integre todas las etapas de un proyecto de desarrollo de software.

A continuación se presenta el modelo integral de la traza que muestra los artefactos y las relaciones necesarias entre los mismos. (Ver Fig. 9)

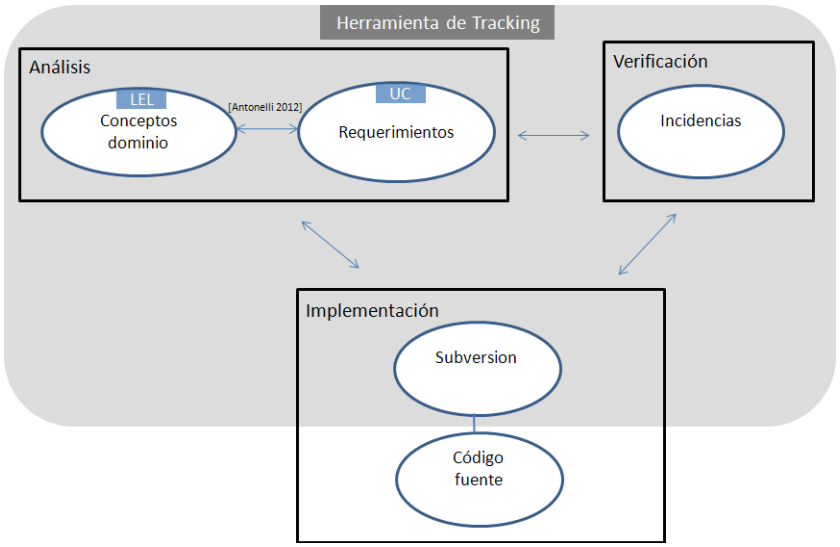


Figura 9. Modelo conceptual de traza

4 Herramienta

Este capítulo presenta los motivos por los cuales se seleccionó la herramienta de tracking MantisBT, continuando con la descripción de la configuración que debe llevarse a cabo para crear una extensión en dicha herramienta, lo cual fue utilizado para crear el plugin Honey.

Se muestra, en detalle, el modelo de trazabilidad bidireccional que espera lograrse con el presente trabajo, indicando las funcionalidades incorporadas en Honey para lograr dicho modelo.

Finalizando, se describe paso a paso, cómo configurar el módulo de MantisBT que permite la integración con Subversion, cómo instalar el plugin Honey en la herramienta de tracking y cómo debe utilizarse esta extensión a través del manual de usuario.

4.1 Descripción general

La herramienta de tracking seleccionada para lograr el modelo de trazabilidad entre las distintas áreas de un proyecto de desarrollo de software fue MantisBT, cuya descripción se detalló en sección 1.2.3.1. Los motivos por los cuales se eligió dicha herramienta son los siguientes:

- Está desarrollada en PHP, pudiéndose utilizar en cualquier plataforma que lo corra.
- Es Open Source, permitiendo la facilidad de extensión.
- Se destaca por su facilidad y flexibilidad de instalar y configurar.
- Se pueden registrar diferentes tipos de incidencias (Bugs, Task, etc).
- Posee integración con SVN mediante un módulo específico para tal fin, quedando registrado en la herramienta el historial de cambios en las incidencias con cada operación de commit.

4.2 Proceso para extender MantisBT

Sin necesidad de modificar archivos que forman el núcleo, MantisBT proporciona un sistema de eventos que permiten crear plugins en la herramienta.

Los plugins se definen como subclases de la clase “MantisPlugin” definida en core/classes/MantisPlugin.php.

Existe una serie de métodos ya definidos en la clase MantisPlugin, los cuales deben ser utilizados para las opciones de configuración, la declaración de nuevos comportamientos o la modificación del esquema de la base de datos.

Los plugins se deben ubicar dentro del directorio /plugins.

Todo plugin debe estar contenido en un solo directorio cuyo nombre coincide con el “nombre base” del plugin, así como contener por lo menos un archivo PHP, también con el “nombre base”,

ej:

Example/
Example.php

Este archivo (Example.php) debe contener una clase concreta que derive de MantisPlugin.class; esta clase se debe llamar <nombre base>Plugin, ej.: ExamplePlugin.

Como MantisBT declara como abstracto el método para registrar un plugin (register()), el plugin a desarrollarse debe implementarlo. Este método es utilizado sólo para este propósito (establecer las propiedades del plugin, incluyendo el nombre, descripción, versión, etc.) y no debe utilizarse para otras tareas.

Una vez definida la clase del plugin e implementado el método register(), asignando al menos las propiedades de nombre y versión, el plugin puede considerarse completo. De esta forma, puede ser cargado e instalado a través del administrador de plugins de MantisBT.

Por otro lado, MantisBT proporciona procesos y una jerarquía determinada para agregar páginas y archivos a los plugins.

Consideramos páginas a aquellos archivos PHP que son ejecutados dentro del núcleo de MantisBT, mientras que los archivos serán los que son pasados al navegador del cliente exactamente como aparecen en el sistema de archivos.

Toda nueva página debe ser ubicada en el directorio /pages dentro del plugin; debe ser nombrada utilizando únicamente letras y números y debe tener la extensión “.php”.

Para generar un link, dentro del plugin, a la nueva página se debe utilizar la función plugin_page().

Para incorporar archivos que no sean PHP, como por ejemplo hojas de estilo, se procede de igual manera que con las páginas.

Los archivos deben ser colocados en el directorio /files y sólo deben contener un único punto (“.”) en su nombre.

El link a los archivos se genera a través de la función plugin_page().

Una estructura del sistema de archivos posible se vería de la siguiente manera:

```
Example/
        Example.php
        pages/
                page.php
        files/
                file.css
```

El sistema de plugins brinda una metodología para el manejo de eventos que no requiere llamar directamente a las funciones de la API. Los eventos toman forma de métodos de clase dentro del plugin.

Para declarar un nuevo evento o una serie de eventos que el plugin debe ejecutar hay que sobrescribir el método event() del plugin en cuestión y retornar una arreglo (diccionario) con los nombres de los eventos como clave y el tipo del evento como valor.

Esto permite que cuando se cargue el plugin, se agreguen estos nuevos eventos a la lista de eventos de MantisBT.

Así como para el manejo de los eventos, MantisBT proporciona una metodología simple para las opciones configuración: se declaran y utilizan de la misma manera que los eventos.

El valor de la opción de configuración de un plugin se obtiene mediante el uso de la función plugin_config_get(), y se puede asignar un valor utilizando plugin_config_set().

Para poder manipular la base de datos, MantisBT cuenta con la función schema() escrita con adodb donde se declara la estructura de las tablas.

El esquema de base de datos puede actualizarse al instalar el plugin o a través del administrador de plugins.
Por razones de performance, una actualización del esquema de la base sólo se puede chequear a través del administrador de plugins.

Un esquema de base de datos se define mediante una lista de los cambios de esquema, que se aplican secuencialmente, como añadir tablas, añadir o modificar columnas, etc.
La historia del esquema debe ser lineal, y las entradas anteriores del esquema no se deben cambiar para que el proceso de actualización funcione correctamente.

4.3 Modelo de traza detallado

Interrelacionando herramientas seleccionadas de uso cotidiano para el desarrollo de un producto de software, que intervienen en las distintas etapas de dicho producto, se logra hacer un seguimiento de los requerimientos desde su concepción hasta que son implementados en la etapa de desarrollo y luego verificados en la etapa de prueba.

La elección de las herramientas y artefactos que éstas producen permitió lograr la trazabilidad bidireccional, objetivo del presente trabajo de tesis.

A continuación se presenta el modelo completo de trazabilidad logrado con las funcionalidades incorporadas a MantisBT:

- Creación y gestión de artefactos de análisis (LEL y UCs).
- Relación entre artefactos de análisis. A partir de un concepto del dominio (símbolo LEL) se puede crear un UC quedando registrada la relación entre el UC y el símbolo que lo creó y viceversa.

- Relación entre artefactos de análisis y artefactos de pruebas. En MantisBT se registran las incidencias que se vayan encontrando en la etapa de verificación del producto, donde se prueba la funcionalidad implementada. En el presente trabajo se realizó una extensión con la que se pueden crear Casos de Uso manualmente o a través de una estrategia de derivación. Una vez que el rol tester registra una incidencia, la misma puede ser asociada a un Caso de Uso por medio de una nota. De esta manera, el rol de tester puede indicar qué funcionalidad, o mejor dicho, qué Caso de Uso es el que está fallando y se logra la relación UC-INCIDENCIA que permite conocer todas las incidencias generadas para un Caso de Uso particular.

- Relación entre artefactos de pruebas y artefactos de desarrollo. MantisBT posee un módulo de integración con Subversion que registra notas en las incidencias con el comentario realizado en la operación de commit que las mencionan. Si bien esta funcionalidad es útil, no permite hacer un seguimiento de manera tal que pueda conocerse qué artefactos de desarrollo se involucraron en la resolución de cierta incidencia en el producto. Este seguimiento representa la trazabilidad buscada, y para lograrla se incorporó, en la extensión realizada en el presente trabajo, la posibilidad de que se registre en MantisBT la ruta de los archivos de desarrollo (artefactos de desarrollo) que referencian a una incidencia en la herramienta. Esta referencia queda impactada en la herramienta cuando se hace una operación de commit de uno o más artefactos de desarrollo cuyo comentario menciona el identificador de una incidencia en MantisBT.

- Relación entre artefactos de análisis y artefactos de desarrollo. Con objetivo de trazar los artefactos de desarrollo con los de análisis, se utilizó la extensión desarrollada en el punto anterior para asociar los Casos de Uso con los artefactos de desarrollo que lo implementan. Para establecer la relación es necesario que el comentario del commit mencione al Caso de uso; como resultado quedarán registradas las URLs de los archivos commiteados como nota del UC mencionado.

- La implementación de los artefactos de desarrollo (código fuente) queda a cargo de los desarrolladores.

La siguiente figura representa el modelo de trazabilidad logrado (Ver Fig. 10)

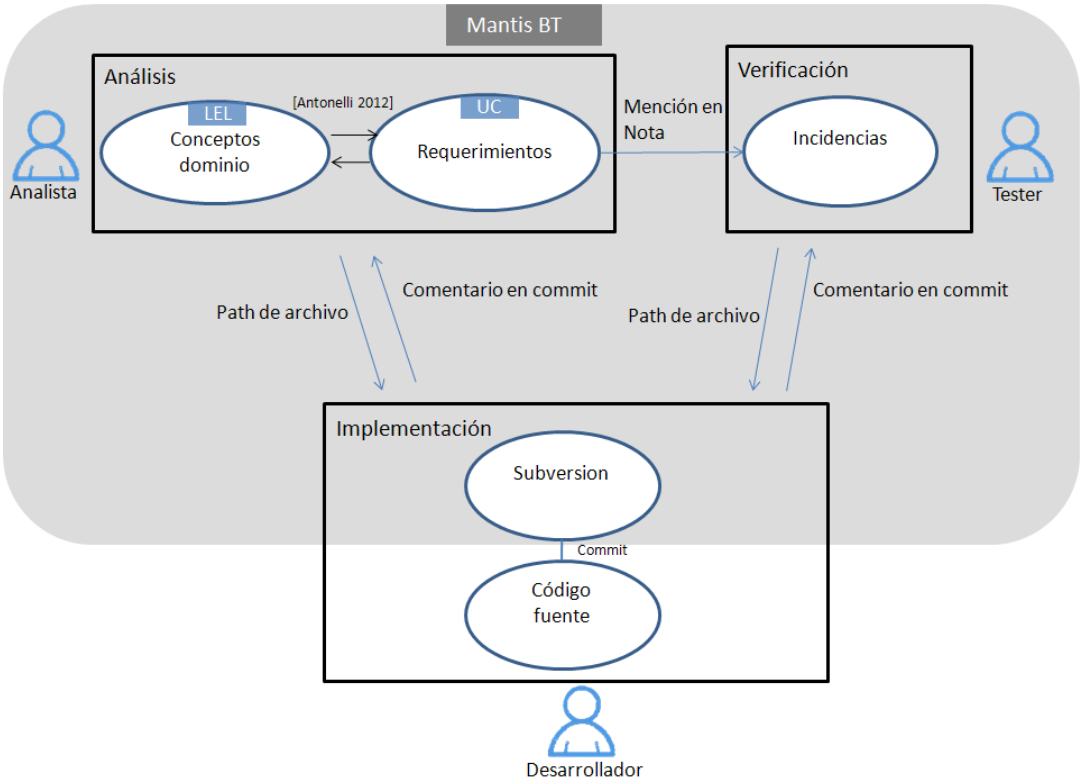


Figura 10. Modelo de traza detallado

4.3.1 Análisis: Relación LEL - UC

Basándonos en la estrategia de derivación propuesta por [Antonelli 2012], la trazabilidad entre los artefactos de análisis se realizó de la siguiente forma:

Se incorporó una nueva funcionalidad en MantisBT, a través de un módulo llamado Honey, que permite ingresar y administrar los distintos símbolos capturados durante el análisis de requerimientos mediante LEL.

Luego, se extendió e implementó el proceso de derivación de LEL a UCs anteriormente descripto, brindando la posibilidad de incorporar a la herramienta de tracking, de manera automática, todos los UCs y actores que puedan ser derivados del LEL, quedando registrado en cada entidad creada (UC y actor) el símbolo que le dio origen. Se puede observar (Ver Fig.11) que con la derivación automática se obtiene la relación bidireccional (trazabilidad) entre las entidades de análisis LEL-UC.

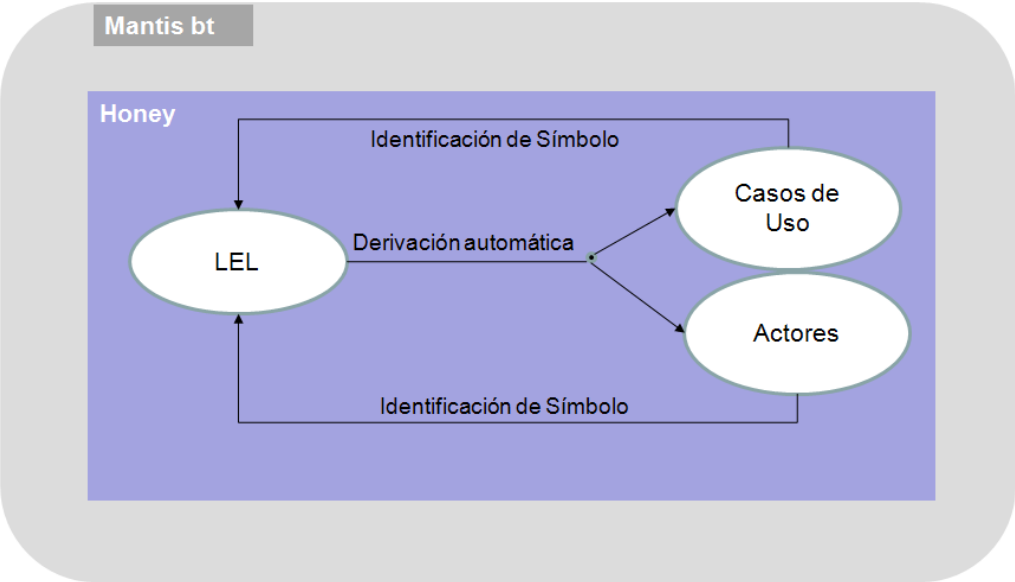


Figura 11. Módulo HONEY

Durante el análisis de la estrategia de derivación utilizada en el modelo de traza, se detectó que este proceso de transformación a pre y postcondiciones de Casos de Uso debía ser enriquecido. El mismo establece que si dentro de los impactos de un símbolo de tipo estado se menciona a un símbolo de tipo verbo y a otro de tipo estado, entonces estamos frente a una postcondición de dicho verbo. Caso contrario es una precondición. Sin embargo se observa en el siguiente ejemplo, un impacto de un símbolo de tipo estado que contiene un verbo y otro estado pero no es una postcondición. (Ver Tabla 26).

Tabla 26. Contraejemplo de postcondición

Nombre	Expediente Agregado	Tipo	Estado
Sinónimos			
Noción	**Expediente** que se encuentra contenido dentro de un **expediente principal**.		
Impacto	El **interesado** **consulta** cuál es el **expediente principal** de un **expediente agregado**		

Notar que el símbolo es de tipo estado, su impacto menciona al verbo **consultar** y al estado **expediente principal** y sin embargo, en el proceso de derivación no sería correcto que el Caso de Uso Consultar tenga como postcondicion “Expediente Agregado,” Por contraejemplos como este, fue necesario evaluar las distintas alternativas para generar Casos de Uso con información consistente con el dominio. La solución lograda consiste en que sea el usuario quien descarte o clasifique, para cada Caso de Uso, aquellos impactos que corresponden a precondición o postcondicion,

Honey provee un mecanismo visual sencillo para llevar a cabo este proceso. (Ver Fig. 12).

Desglosar	Not Condition	Precondition	Postcondition
La oficina administrativa desglosa un expediente agregado al expediente principal volviendo a ser expediente simple y el expediente principal pasa a simple si ya no tiene más expedientes agregados	○	○	◉

Remitir	Not Condition	Precondition	Postcondition
La oficina administrativa remite al expediente principal conjuntamente con cada expediente agregado que contenga.	◉	○	○
El administrador de mesa de entrada remite al expediente principal conjuntamente con cada expediente agregado que contenga.	◉	○	○

Recibir	Not Condition	Precondition	Postcondition
La oficina administrativa recibe al expediente principal conjuntamente con cada expediente agregado que contenga	◉	○	○
El administrador de archivo recibe al expediente principal conjuntamente con cada expediente agregado que contenga	◉	○	○

Consultar	Not Condition	Precondition	Postcondition
El interesado consulta en qué oficina administrativa se encuentra un expediente archivado	◉	○	○
El interesado consulta en qué oficina administrativa se encuentra un expediente normal	◉	○	○
El interesado consulta cuál es el expediente principal de un expediente agregado	◉	○	○
El interesado consulta en qué oficina administrativa se encuentra un expediente agregado	◉	○	○
El interesado consulta en qué oficina administrativa se encuentra un expediente principal	◉	○	○
El interesado consulta los expedientes agregados a un expediente principal	◉	○	○

Save

Figura 12. Definición de pre y post condiciones en Honey

Por otro lado, como los datos proporcionados por el LEL no son suficientes para alcanzar un UC completo, Honey permite enriquecer los UCs derivados con reglas del negocio, interfaces, escenarios alternativos y relaciones con otros UCs.

Además, se incorpora la funcionalidad para crear Casos de Uso sin derivar y administrarlos (agregando notas, modificando datos, etc.), brindando también la posibilidad de relacionarlos con aquellos que fueron derivados (relaciones extend e include). Para lograr dicha funcionalidad se agregó al plugin Honey la administración de reglas y actores (Ver Fig. 13).

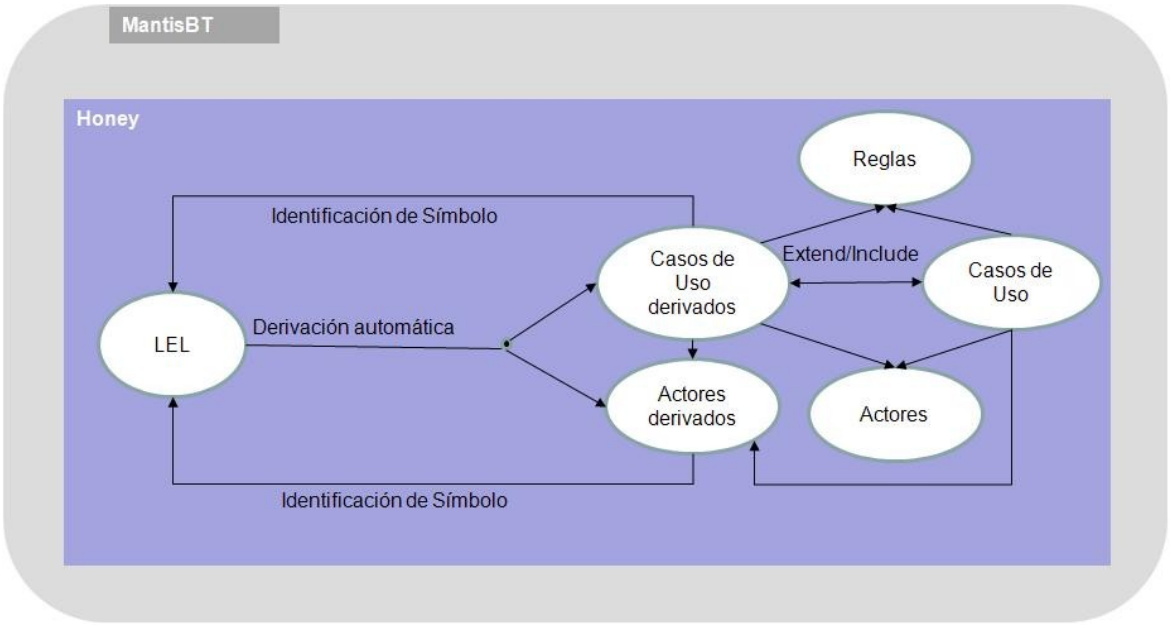


Figura 13. Administración de Casos de Uso, reglas y actores no derivados

Una vez lograda la derivación de LEL a UCs, puede ocurrir que surjan cambios que impacten directamente en el LEL. Esta situación implicaría modificar el diccionario previamente creado y una nueva derivación. Fue necesario tomar una postura ante este caso, la cual consiste en mostrarle al usuario los conflictos y que sea él, con su conocimiento del dominio, quien se encargue de solucionarlos antes de volver a intentar derivar.

Los conflictos pueden darse cuando un UC o actor creado manualmente se relaciona con elementos derivados; ya sea actores o que extiende o incluye a UCs derivados, dado que si se realiza una nueva derivación dichas relaciones pueden quedar obsoletas o con información inconsistente.

Ejemplo: Caso de Uso creado manualmente relacionado con Actor derivado

Sea el Caso de uso "Reservar Expediente" creado manualmente en Honey, cuyos actores principales son "Oficina Administrativa" y "Director Administrativo" originados con la derivación de los símbolos de tipo sujeto "Oficina Administrativa" y "Director Administrativo" respectivamente (Ver Fig. 14)

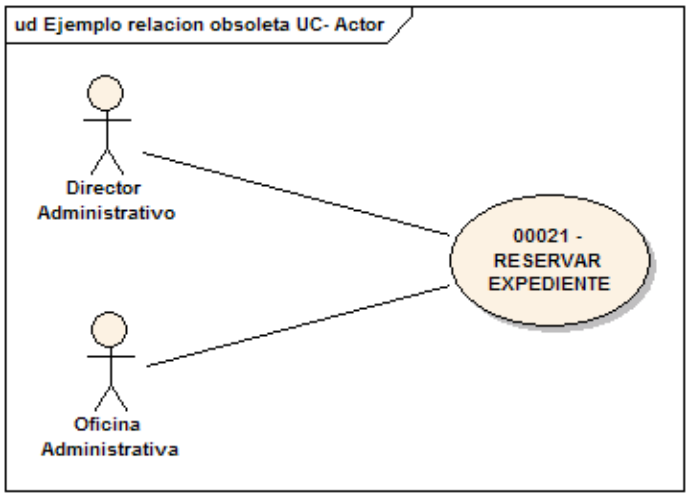


Figura 14. Relación UC manual y Actores derivados

Una norma impone que la Oficina Administrativa no interviene más en el circuito de expedientes, debiéndose eliminar el término o símbolo del LEL.

El analista funcional procede a borrar el símbolo "Oficina Administrativa" de LEL y realiza una nueva derivación. La información quedaría de la siguiente forma (Ver Fig. 15):

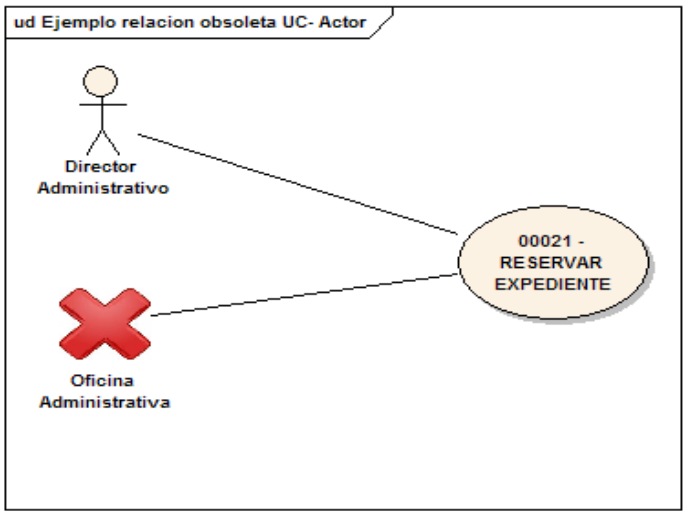


Figura 15. Borrado de Actor "Oficina Administrativa" y derivación en LEL

Puede observarse que el actor "Oficina Administrativa" fue eliminado, quedando el Caso de Uso "Reservar Expediente" relacionado a un actor que ya no existe en el dominio.

Parte de la labor en el presente trabajo de tesis fue evaluar las alternativas que eviten dejar información del dominio inconsistente como se presenta en este ejemplo.

La solución lograda plantea que al intentar eliminar un símbolo de tipo Sujeto o Verbo, el sistema advierte al usuario la existencia de Casos de Uso o Actores derivados de dichos símbolos. Las alternativas que tiene el usuario frente a esta situación son:

- Eliminar el símbolo dejando al actor o Caso de Uso como no derivados.
- Eliminar el Actor o Caso de Uso y luego, el símbolo que creó dicha entidad.

Por otro lado, Honey posee la funcionalidad de almacenar todos los datos generados en cada derivación (UCs, actores y relaciones entre éstos). Esta característica podría permitir, por ejemplo, ver el historial de derivaciones o restaurar a una derivación previa (posible trabajo fututo).

4.3.2 Implementación: Relación UC - Código

La relación que se establece entre el código fuente de un producto y los Casos de Uso que especifican la funcionalidad de dicho producto debe realizarse explícitamente por el desarrollador. Sin embargo, en el presente trabajo de tesis se analizaron herramientas e incorporaron extensiones que facilitan la traza bidireccional entre la especificación de los requerimientos (UCs) y el código fuente.

Parte de la extensión desarrollada con el plugin Honey ayuda al desarrollador en su tarea de establecer las relaciones entre los artefactos de las áreas de Análisis e Implementación. (Ver Fig. 16)

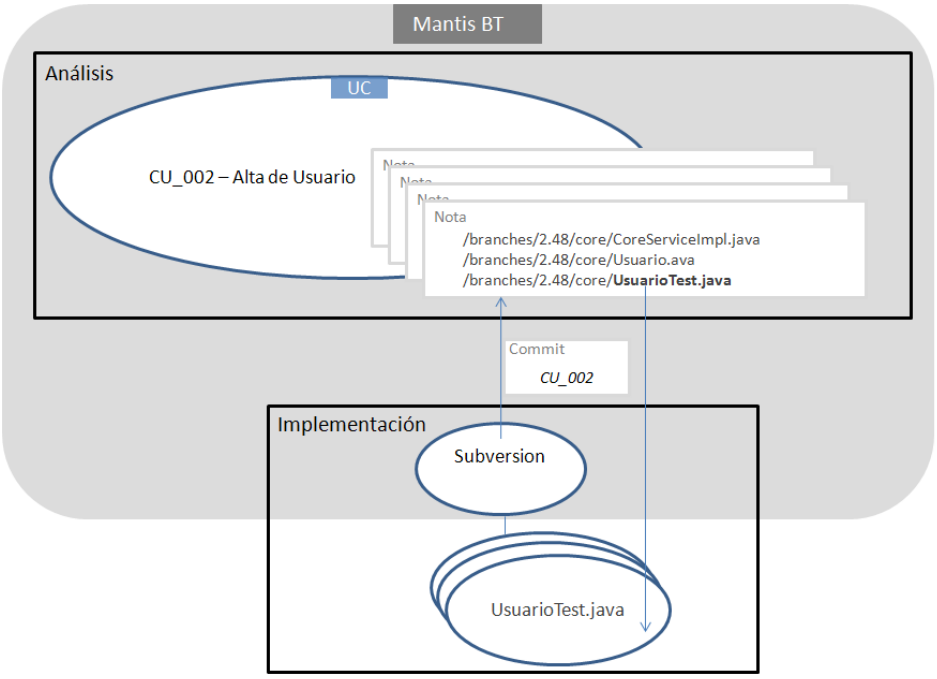


Figura 16. Trazabilidad entre artefactos de análisis e implementación

Al momento de seleccionar una herramienta de tracking a extender, uno de los puntos requeridos dentro de la elección era que la herramienta tenga la posibilidad de conectarse a repositorios de Subversion, de esta manera se conocería la ubicación de los artefactos de desarrollo utilizados durante la implementación del producto. MantisBT cuenta con esta integración funcionando de la siguiente forma:

- El usuario selecciona un artefacto o conjunto de artefactos en su plataforma de desarrollo.
- Procede a guardar los cambios realizados generando una nueva versión del/los artefacto/s mediante la operación de commit.
- La operación de commit solicita el ingreso de un comentario.
- Si dentro del comentario, el usuario utiliza el identificador de una incidencia registrada en MantisBT, la integración de la herramienta con el repositorio de Subversion detecta la identificación, registrando en la herramienta de tracking una nota para dicha incidencia con el comentario que ingresó el desarrollador en la operación de commit. En la siguiente figura se observa este comportamiento. (Ver Fig. 17)

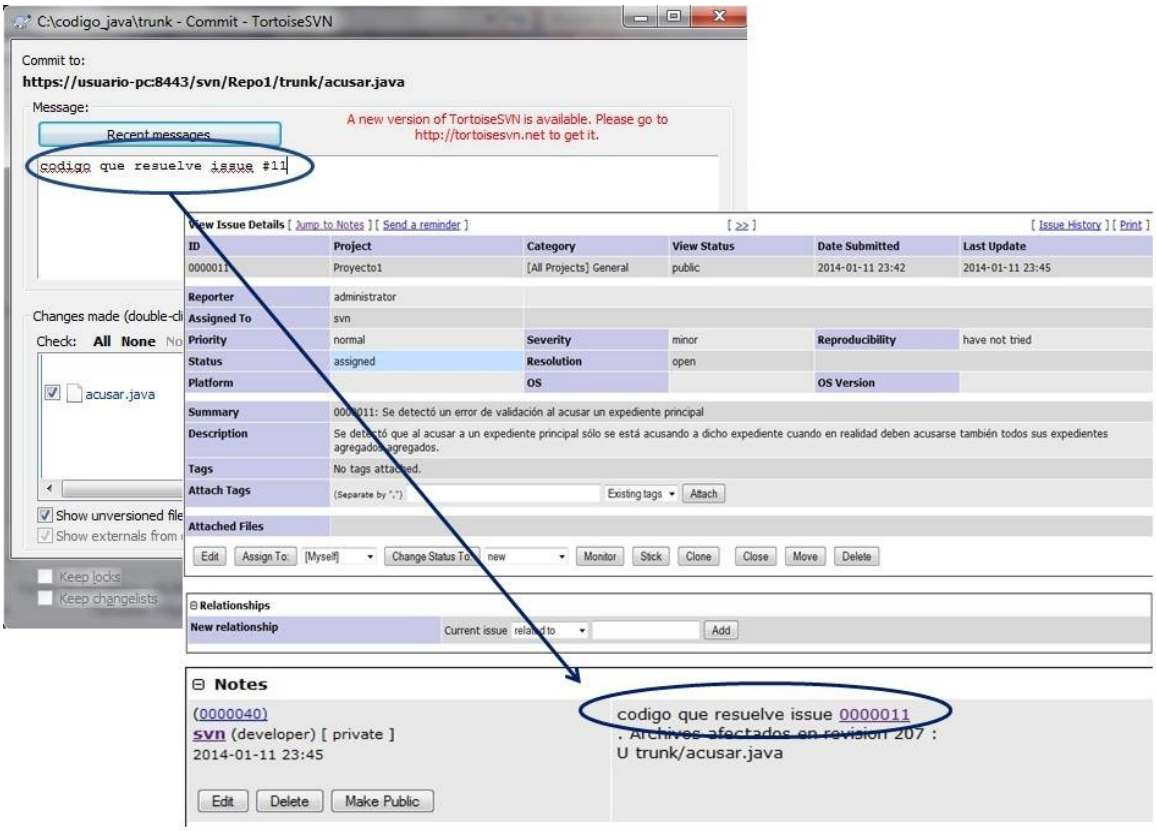


Figura 17. Comentario en operación de commit referenciando a una Incidencia

Este proceso de integración permite registrar el paso a paso de lo que ocurre con incidencias reportadas en la herramienta, sin embargo no brinda la posibilidad de relacionar estos artefactos con los elementos de desarrollo, es decir:

- ¿Qué artefactos de desarrollo se modificaron o agregaron para desarrollar un nuevo UC?
- Si se desea modificar la funcionalidad de un UC ya desarrollado ¿puedo saber qué artefactos de desarrollo involucran ese UC?

Las respuestas a estas preguntas se obtienen manteniendo la trazabilidad entre artefactos de desarrollo y análisis. Para ello se incorporaron en el plugin Honey las siguientes funcionalidades, logrando trazar las etapas de Análisis e Implementación:

- **Posibilidad de identificar Casos de Uso en el commit de un artefacto de desarrollo**: Consiste incorporar en el comentario del commit la identificación de uno o más Casos de Uso registrados en MantisBT (creados manualmente o bien derivados del LEL, utilizando el plugin Honey). (Ver Fig. 18)

Figura 18. Comentario en operación de commit referenciando a un Caso de Uso

- **Registro de Path de archivos en las notas de los artefactos registrados en MantisBT**: Honey guarda todos los paths de archivos commiteados que referencian a incidencias o Casos de Uso creados en la herramienta de tracking. Las rutas de los archivos relacionados con una incidencia o UC también se mantiene visible al usuario de la herramienta. (Ver Fig. 19)

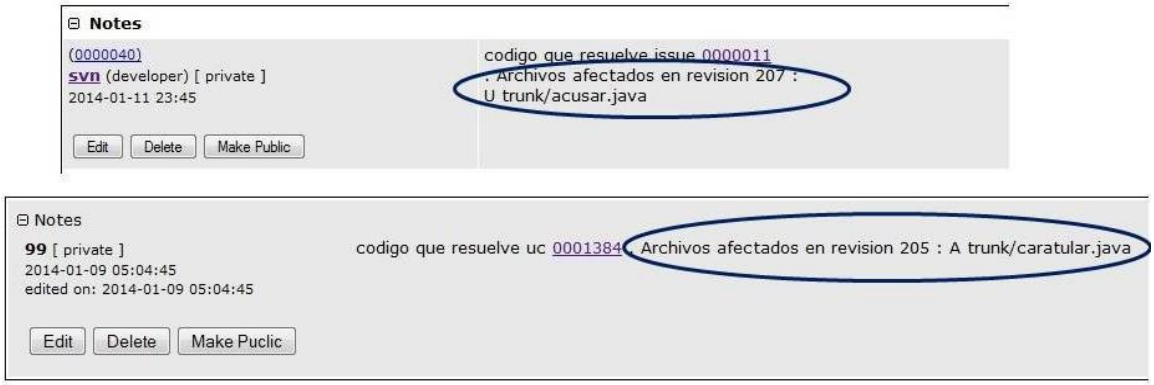


Figura 19. Path de archivos en las notas de incidencias y Casos de Uso

4.3.3 Verificación: Relación Incidencia – UC – Código

Como se indicó anteriormente, MantisBT ofrece parte de la funcionalidad buscada en el presente trabajo, ya que posee un módulo de integración con el repositorio de versionado que permite establecer la relación entre el código fuente y las incidencias reportadas en la herramienta. (Ver Fig. 20)

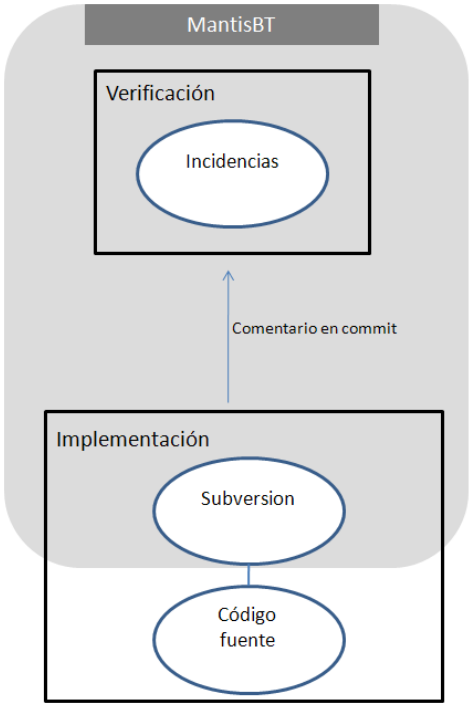


Figura 20. Integración entre artefactos desarrollo y verificación

MantisBT brinda la posibilidad de mencionar incidencias en las notas de otra incidencia, generando una relación directa entre estas. Esta característica facilitó el trabajo de realizar la extensión necesaria en Honey, de manera tal que, dentro de las notas de un Caso de Uso creado en MantisBT se pueda hacer referencia a una o más incidencias, generándose la relación INCIDENCIA – UC (Ver Fig. 21).

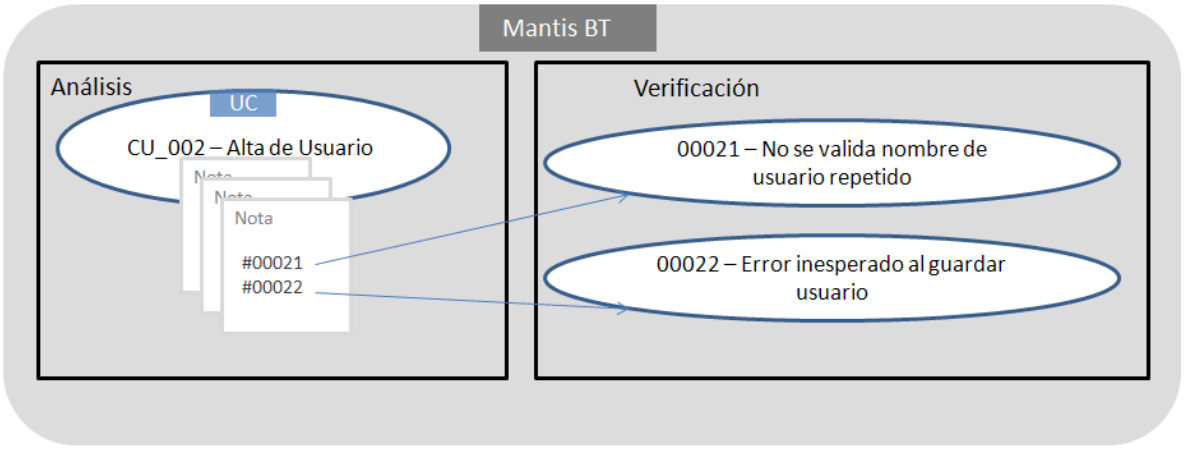


Figura 21. Relación Incidencia – UC

A continuación se presenta la pantalla que evidencia dicha funcionalidad:

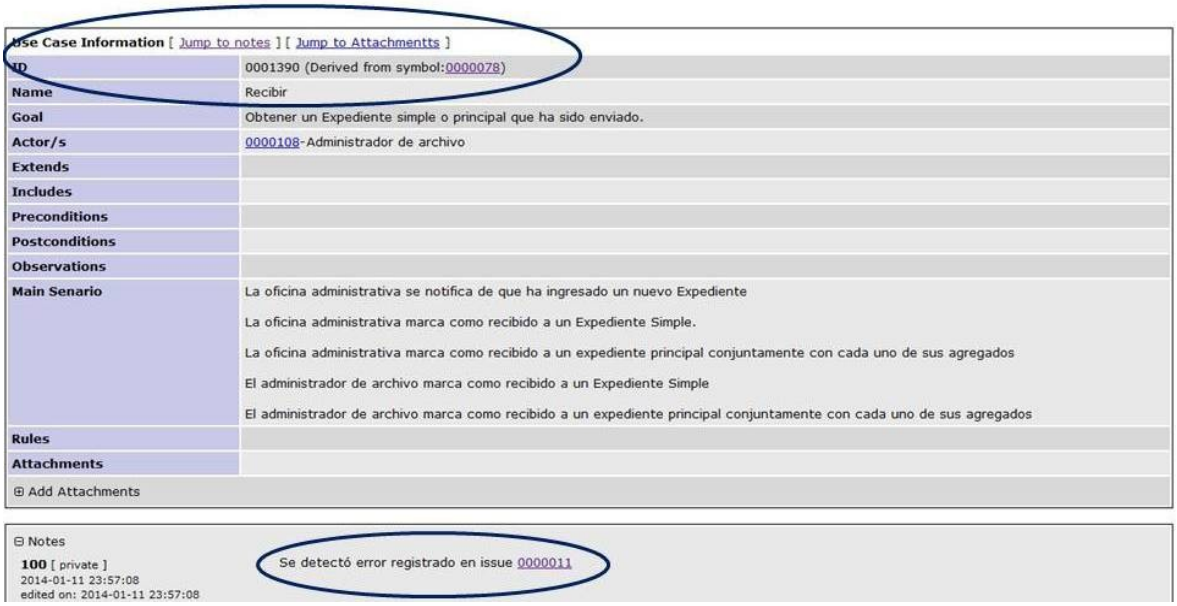


Figura 22. Relación Incidencia – UC en Honey

MantisBT en su integración con Subversion permite, como se explicó anteriormente, que los comentarios realizados en las operaciones de commit que mencionan una incidencia en MantisBT, queden registrados como nota de dicha incidencia. Sin embargo, esto no permite conocer qué artefactos de desarrollo fueron necesarios para resolver determinada incidencia o BUG.

Para poder relacionar los artefactos de desarrollo con las incidencias registradas en MantisBT, se desarrolló en Honey la funcionalidad que permite que los paths de archivos, involucrados en la operación de commit, queden registrados en la herramienta de tracking. Así logra establecerse la traza entre el código fuente del producto y las incidencias reportadas en la herramienta (Ver Fig. 23)

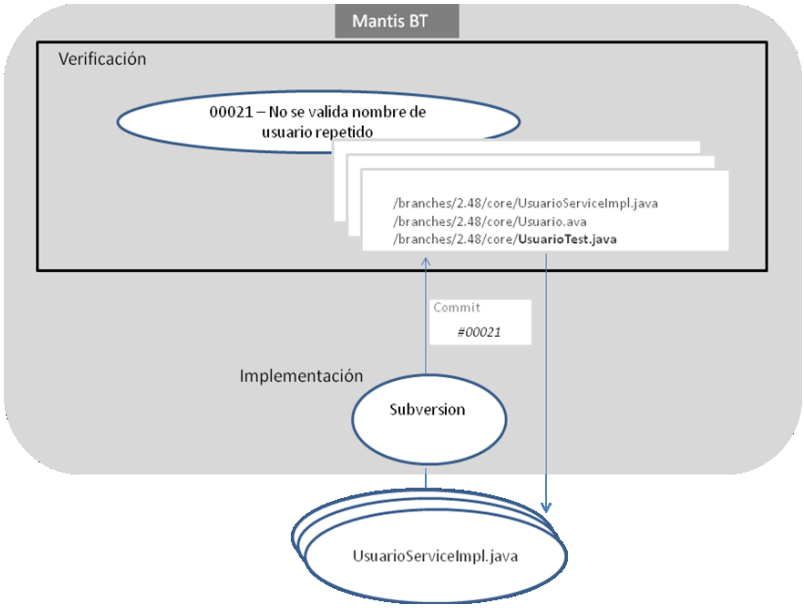


Figura 23. Relación Incidencia – Código

Queda como trabajo futuro la implementación de una funcionalidad que permita referenciar un Caso de Uso dentro de una nota de una incidencia reportada en la herramienta de tracking, Esto implicaría la alteración del código fuente de MantisBT.

4.4 Instalación e integración

Los primeros pasos para un preciso funcionamiento de las herramientas, es su correcta instalación e integración. El Anexo A brinda toda la información necesaria para lograrlo.

Paso 1. Configurar MantisBT para integrar con Subversion.

La integración entre la herramienta de tracking MantisBT y el servidor de versionado Subversion VisualSVN permite la traza entre el código fuente de la aplicación en cuestión y las incidencias de la herramienta de tracking; de manera tal que al realizarse un commit con un mensaje del tipo "arreglada incidencia #1" se marque automáticamente como resuelta dicha incidencia en MantisBT.

Para alcanzar dicho propósito es necesario instalar, en MantisBT, los plugins que realizan la integración. Para ello basta con colocarlos en la carpeta plugins de MantisBT e instalarlos desde la opción de menú: Manage → Manage Plugins.

Los plugins necesarios son:

- MantisBT Core 1.2.15
- MantisBT Formatting 1.0a
- Meta Programing 0.1
- Source Control Integration 0.16.4
- Subversion / WebSVN Integration 0.16
- Subversion Integration 0.16

La figura 24 muestra la pantalla resultante luego de la instalación de los plugins.

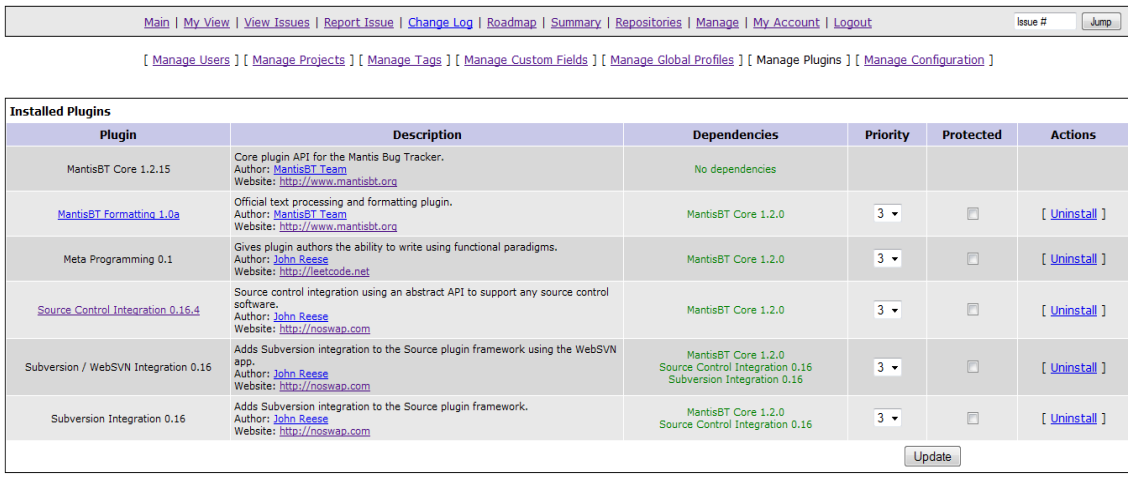


Figura 24. Plugins instalados en MantisBT

El siguiente paso consta de la creación de un usuario en MantisBT con el nombre “svn”, con al menos nivel de desarrollador. Será la cuenta que se conecte con Subversion.

Luego de la instalación del servidor de versionado se habrá creado la carpeta “Repositories” (en la ubicación indicada durante la instalación), la cual contiene una carpeta, con el nombre del repositorio, y dentro de la misma, se encuentra la carpeta hooks con un archivo llamado post-commit que luego utilizaremos.

El resto de la integración necesaria para alcanzar la traza se logra instalando el plugin Honey y configurando el archivo post-commit antes mencionado.

Paso 2. Instalar plugin Honey en MantisBT

La instalación del plugin Honey requiere la realización de los siguientes 3 pasos:

- Colocar la carpeta Honey dentro de la carpeta plugins de MantisBT (Ver Fig. 25).

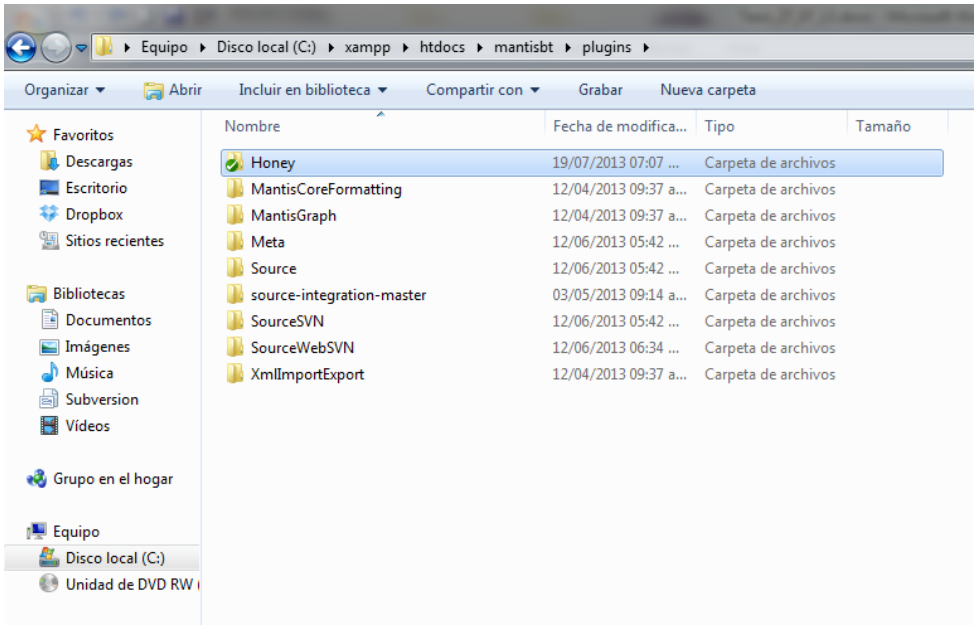
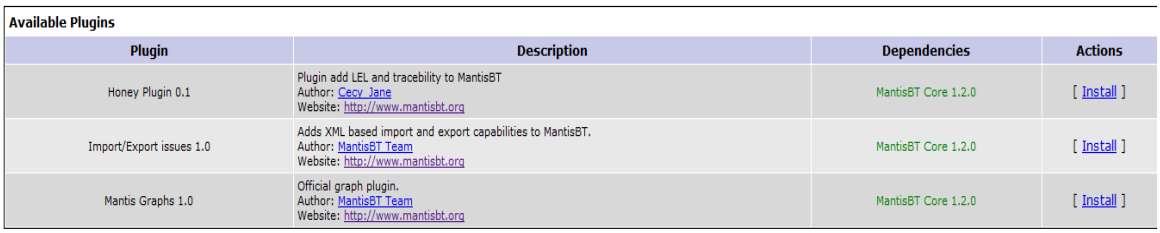


Figura 25. Ruta de carpetas donde debe ir el nuevo plugin

- Instalar el plugin Honey desde el administrador de plugins de la herramienta : Manage → [Manage Plugins]. (Ver Fig. 26).



Available Plugins

Plugin	Description	Dependencies	Actions
Honey Plugin 0.1	Plugin add LEL and tracebility to MantisBT Author: Cecy Jane Website: http://www.mantisbt.org	MantisBT Core 1.2.0	[Install]
Import/Export issues 1.0	Adds XML based import and export capabilities to MantisBT. Author: MantisBT Team Website: http://www.mantisbt.org	MantisBT Core 1.2.0	[Install]
Mantis Graphs 1.0	Official graph plugin. Author: MantisBT Team Website: http://www.mantisbt.org	MantisBT Core 1.2.0	[Install]

Figura 26. Administrador de MantisBT donde se visualiza nuevo plugin

- Visualizar los nuevos ítems de menú: LEL y Use Cases. (Ver Fig. 27).

Figura 27. Visualización de plugin instalado y su correspondiente menú en MantisBT

Paso 3. Configurar Subversion

Para completar la integración, se debe modificar (para Linux) o reemplazar (para Windows) el archivo post-commit del SVN.

Modificaciones en Subversion para Linux

Es necesario modificar el archivo hooks/post-commit.tmpl de manera tal que cada vez que se produzca un cambio, se avise a MantisBT de ello. El archivo deberá quedar como indica la siguiente figura:

```
#!/bin/sh

REPOS="$1"
REV="$2"

auth=$(svnlook author -r $REV $REPOS)
dt=$(svnlook date -r $REV $REPOS)
changed=$(svnlook changed -r $REV $REPOS)
log=$(svnlook log -r $REV $REPOS)
n=$'\n'
textLog=$'<b>Log del cambio:</b>'

textChanged=$'<b>Ficheros afectados:</b>'/usr/bin/php -q
C:/xampp/htdocs/mantis/scripts/checkin.php <<< "Cambio [<b>${REV}</b>] por
<b>$auth</b>, $dt$n$n$textLog$n$log$n$n$textChanged$n$changed"
```

Figura 28. Código del archivo post-commit del servidor SVN para Linux

Importante: En la línea final se hace la llamada al script que recogerá los valores del commit de Subversion. (Se debe cambiar la ruta /home/www-data/mantisbt/plugins/Honey/pages/checkin.php por la ruta de instalación correspondiente a la PC del usuario).

Por último se debe dar permisos al fichero (hay que asegurarse que el usuario con que se ejecuta Subversion sea propietario también):

chmod 744 /home/svn/hooks/post-commit
chown www-data:www-data /home/svn/hooks/post-commit.

Modificaciones en Subversion para Windows

Para Windows se debe crear el archivo post-commit con extensión ".bat".

Para un mejor control ante eventuales errores, crear los archivos:

C:\Temp\log.txt

C:\Temp\author.txt

C:\Temp\output.txt

El código del archivo post-commit.bat deberá ser el que indica la siguiente figura:

```
@ECHO on

SET REPOS=%1
SET REV=%2
SET PHP="C:\xampp\php\php.exe"
SET CHECKIN="C:\xampp\htdocs\mantisbt\plugins\Honey\pages\checkin.php"
SET SVNLOOK="C:\Program Files (x86)\VisualSVN Server\bin\svnlook.exe"
SET LOGFILE=C:\Temp\log.txt
SET AUTHORFILE=C:\Temp\author.txt
SET OUTPUTFILE=C:\Temp\output.txt

%SVNLOOK% log -r  %REV% %REPOS%  > %LOGFILE%
%SVNLOOK% author -r %REV% %REPOS% > %AUTHORFILE%

ECHO . Archivos afectados en revision %REV% : >> %LOGFILE%

%SVNLOOK% changed -r  %REV% %REPOS% >> %LOGFILE%

TYPE %LOGFILE% > %OUTPUTFILE%

%PHP% %CHECKIN% < %OUTPUTFILE% >> %LOGFILE%

ECHO "Ejecutado post-commit de revisión %REV%" >> %LOGFILE%

exit

@ECHO off
```

Figura 29. Código del archivo post-commit del servidor SVN para Windows

Importante: Se deben cambiar las rutas de los archivos utilizados a las rutas de las instalaciones correspondientes a la PC del usuario.

4.5 Manual de usuario de Honey

Operaciones del sistema

A. Crear un símbolo del LEL para un proyecto.

B. Buscar los símbolos del LEL para un proyecto (datos mínimos).

C. Modificar los datos de un símbolo del LEL para un proyecto.

D. Eliminar un símbolo del LEL de un proyecto.

E. Ver los datos completos de un símbolo del LEL para un proyecto.

F. Derivar LEL a Casos de Uso dentro de un proyecto.

G. Crear un Caso de Uso para un proyecto.

H. Buscar los Casos de Uso para un proyecto (datos mínimos).

I. Modificar los datos de un Caso de Uso para un proyecto.

J. Eliminar un Caso de Uso de un proyecto.

K. Ver los datos completos de un Caso de Uso para un proyecto.

L. Crear un actor para un proyecto.

M. Buscar actores de un proyecto.

N. Modificar un actor de un proyecto.

O. Eliminar un actor de un proyecto.

P. Crear una regla para un proyecto.

Q. Buscar reglas de un proyecto.

R. Modificar una regla de un proyecto.

S. Eliminar una regla de un proyecto.

Detalle de las operaciones del sistema

A. Crear un símbolo para un proyecto.

Esta operación permite crear un nuevo símbolo dentro de un proyecto. Puede accederse a través de la opción de menú:

- “LEL→New Symbol”.
- Presiona “Save” al terminar de completar los datos.

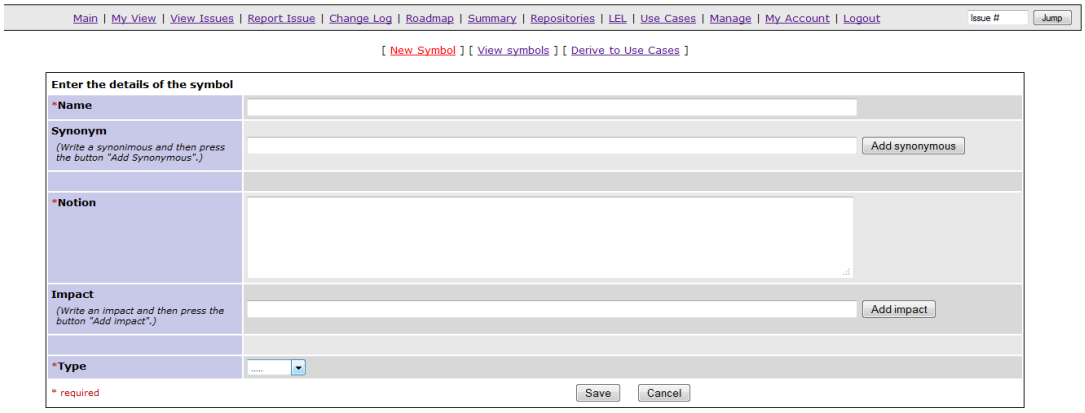


Figura 30. Crear un símbolo en Honey

B. Buscar los símbolos del LEL para un proyecto (datos mínimos).

Esta operación permite ver el listado completo de símbolo del LEL y filtrar dicho listado por ID o nombre de Símbolo. Además permite el orden de los datos por las columnas ID, Nombre o Tipo. Puede accederse a través de la opción de menú:

“LEL” ó “LEL→View Symbols”.

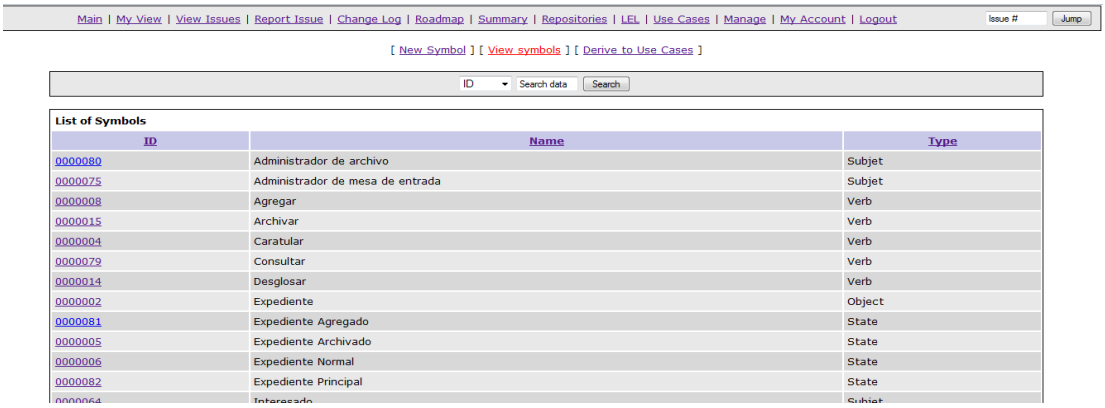


Figura 31. Listado de símbolos de un diccionario LEL en Honey

C. Modificar los datos de un símbolo del LEL para un proyecto.

Esta operación permite buscar y modificar los datos de un símbolo dentro de un proyecto. Puede accederse a través de la opción de menú:

- “LEL” ó “LEL→View Symbols”.
- Seleccionar el símbolo en cuestión.
- Presionar “Edit”, realizar los cambio y “Save”.

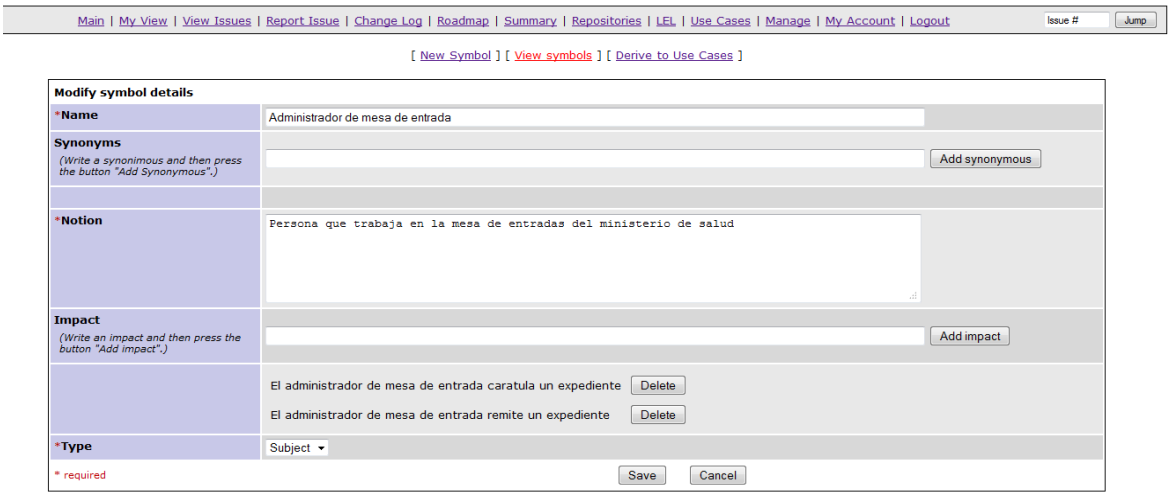


Figura 32. Modificar datos de un símbolo en Honey

D. Eliminar un símbolo del LEL de un proyecto.

Esta operación permite eliminar un símbolo del LEL de un proyecto, previa búsqueda del mismo. Puede accederse a través de la opción de menú:

- "LEL" ó "LEL→View Symbols".
- Seleccionar el símbolo en cuestión de la lista
- Presionar "Delete".

Figura 33. Eliminar símbolo del LEL de un proyecto

E. Ver los datos completos de un símbolo del LEL de un proyecto.

Esta operación permite buscar y ver los datos de un símbolo específico dentro de un proyecto. Puede accederse a través de la opción de menú:

- ""LEL" ó "LEL→View Symbols".
- Seleccionar un símbolo de la lista, el sistema automáticamente mostrará el detalle (Ver Fig. 33).

F. Derivar LEL a Casos de Uso dentro de un proyecto.

Esta operación permite derivar los símbolos del LEL a Casos de Uso dentro de un proyecto. Puede accederse a través de la opción de menú:

- "LEL→Derive to Use Cases"
- El sistema preguntará si desea dejar obsoleta la versión anterior del LEL y en caso de aceptar se crearán los Casos de Uso y actores correspondientes a partir del LEL.

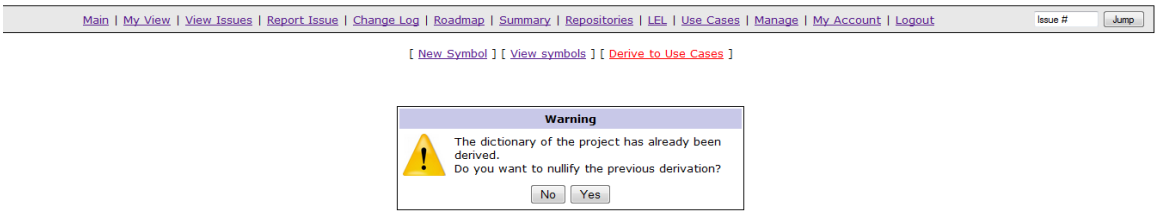


Figura 34. Derivar diccionario LEL a Casos de Uso en Honey

G. Crear un Caso de Uso para un proyecto.

Esta operación permite crear un nuevo Caso de Uso dentro de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→New Use Case".
- Presionar "Save" al terminar de completar los datos.

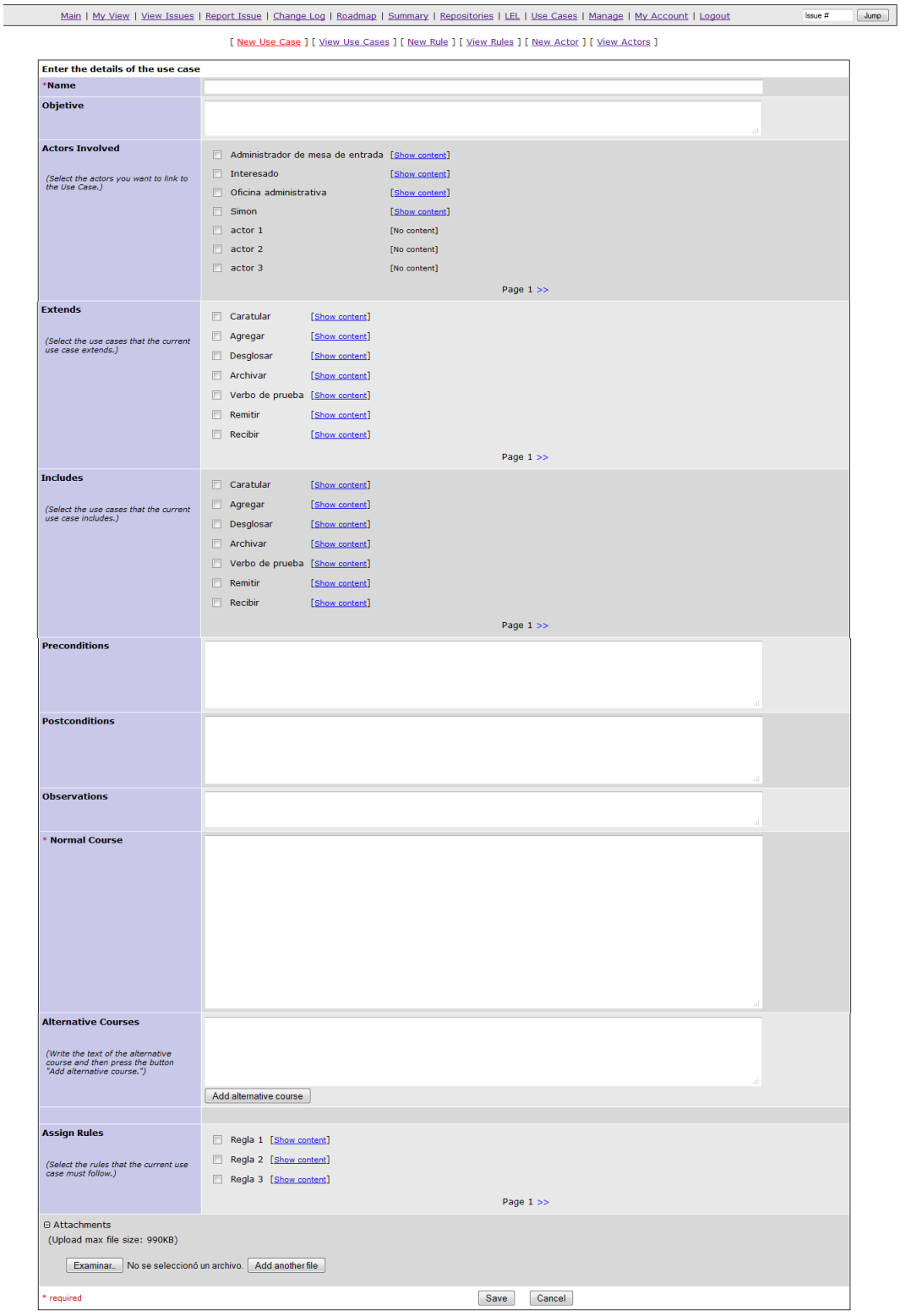


Figura 35. Crear un caso e uso en Honey

H. Buscar los Casos de Uso para un proyecto (datos mínimos).

Esta operación permite ver el listado completo de símbolo de Casos de Uso y filtrar dicho listado por ID o nombre de Caso de Uso. Además permite el orden de los datos por las columnas ID o Nombre. Puede accederse a través de la opción de menú:

"Use Cases" ó "Use Cases→View Use Cases".

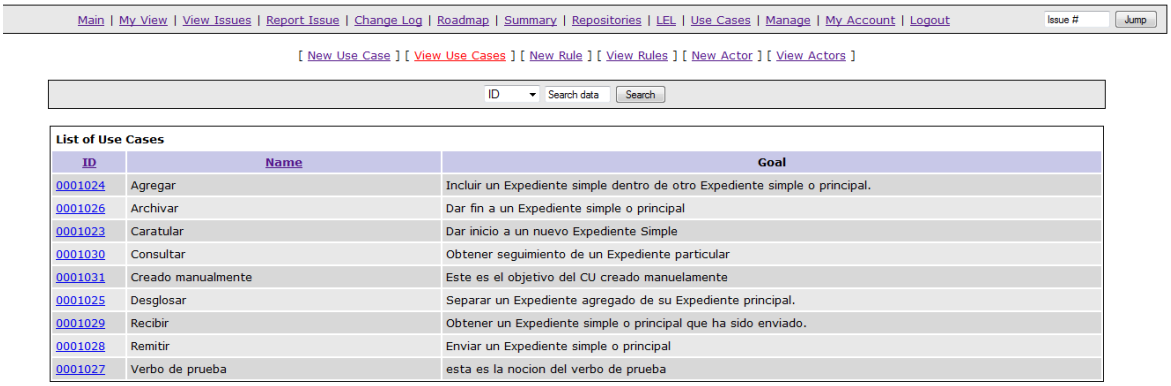


Figura 36. Listado de Casos de Uso en Honey

I. Modificar los datos de un Caso de Uso para un proyecto.

Esta operación permite buscar y modificar los datos de un Caso de Uso dentro de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases" ó "Use Cases→View Use Cases".
- Seleccionar el Caso de Uso en cuestión de la lista.
- Presionar "Edit", realizar los cambio y "Save".

Figura 37. Modificar datos de un Caso de Uso en Honey

J. Eliminar un Caso de Uso de un proyecto.

Esta operación permite eliminar un Caso de Uso de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases" ó "Use Cases→View Use Cases".
- Seleccionar el Caso de Uso en cuestión de la lista.
- Presionar "Delete".

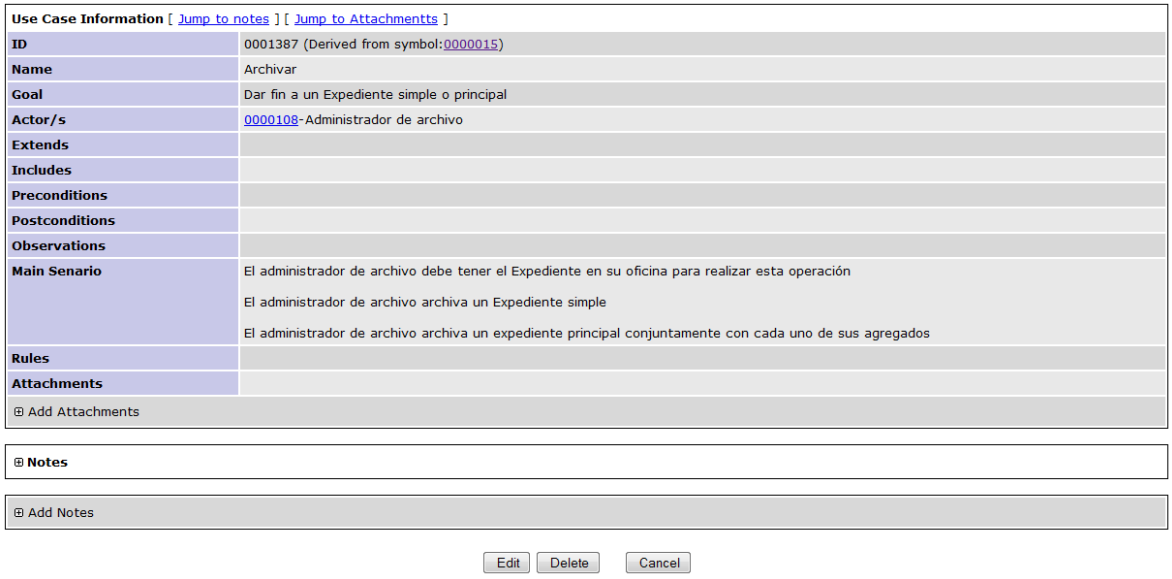


Figura 38. Eliminar símbolo del LEL de un proyecto

K. Ver los datos completos de un Caso de Uso para un proyecto.

Esta operación permite buscar y ver los datos de un Caso de Uso específico dentro de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases" ó "Use Cases→View Use Cases".
- Seleccionar un Caso de Uso de la lista, el sistema automáticamente mostrará el detalle (Ver Fig. 38).

L. Crear un actor para un proyecto.

Esta operación permite crear un nuevo actor dentro de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→New Actor".

- Presionar “Save” al terminar de completar los datos.

Figura 39. Crear un actor de Casos de Uso en Honey

M. Buscar un actor de un proyecto.

Esta operación permite ver el listado completo de actores y filtrar dicho listado por ID o nombre de actor. Además permite el orden de los datos por las columnas ID o Nombre. Puede accederse a través de la opción de menú:

- “Use Cases→View Actors”.

ID	Name	Description
0000088	actor 1	
0000089	actor 2	
0000090	actor 3	
0000091	actor 4	
0000092	actor 5	
0000107	Administrador de archivo	Persona que trabaja en la oficina de archivo del ministerio de salud
0000082	Administrador de mesa de entrada	Persona que trabaja en la mesa de entradas del ministerio de salud
0000097	dsfdf	
0000083	Interesado	Empleado de una oficina administrativa del ministerio de salud
0000084	Oficina administrativa	la ofic
0000085	Simon	ewrwer

Figura 40. Búsqueda de actores en un proyecto Honey

N. Modificar un actor de un proyecto.

Esta operación permite modificar los datos de un actor de un proyecto. Puede accederse a través de la opción de menú:

- “Use Cases→View Actors”.
- Seleccionar de la lista al actor en cuestión.
- Presionar “Edit”, realizar los cambio y “Save”.

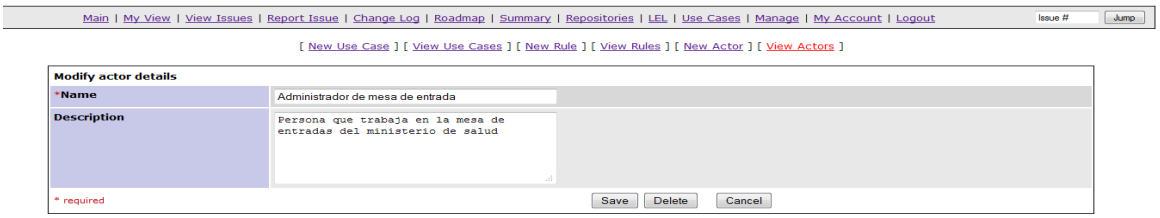


Figura 41. Modificar un actor en Honey

O. Eliminar un actor de un proyecto.

Esta operación permite eliminar un actor de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→View Actors".
- Seleccionar el actor en cuestión.
- Presionar "Delete".

Figura 42. Eliminar un actor en Honey

P. Crear una nueva regla para un proyecto.

Esta operación permite crear una nueva regla dentro de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→New Rule".
- Presionar "Save" al terminar de completar los datos.

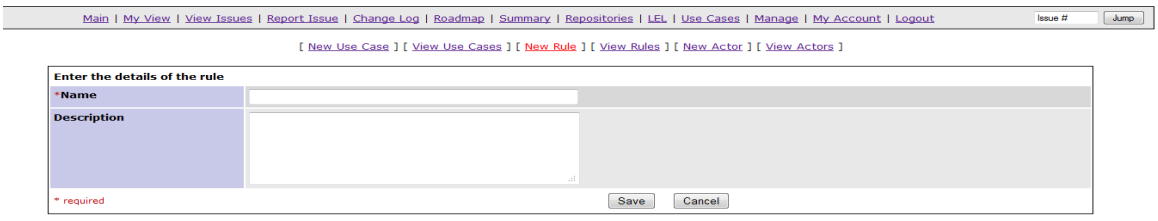


Figura 43. Crear una nueva regla para Casos de Uso en Honey

Q. Buscar una regla de un proyecto.

Esta operación permite ver el listado completo de reglas y filtrar dicho listado por ID o nombre de regla. Además permite el orden de los datos por las columnas ID o Nombre. Puede accederse a través de la opción de menú:

- "Use Cases→View Rules".

Figura 44. Búsqueda de reglas en un proyecto Honey

R. Modificar una regla de un proyecto.

Esta operación permite modificar los datos de una regla de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→View Rules".
- Seleccionar de la lista a la regla en cuestión.
- Presionar "Edit", realizar los cambio y "Save".

Figura 45. Modificar una regla en Honey

S. Eliminar una regla de un proyecto.

Esta operación permite eliminar una regla de un proyecto. Puede accederse a través de la opción de menú:

- "Use Cases→View Rules".
- Seleccionar la regla en cuestión.
- Presionar "Delete".

Figura 46. Eliminar una regla en Honey

4.6 Conclusión

La contribución de este trabajo consiste en la definición de un modelo conceptual de trazabilidad siguiendo por su implementación en una herramienta.

El modelo planteado consiste en:

- Creación y administración del diccionario LEL y Casos de uso dentro de MantisBT.
- Enriquecimiento, implementación y posterior incorporación a MantisBT de la estrategia de derivación propuesta por [Antonelli 2012] para transformar automáticamente LEL en UCs.
- Creación y administración de incidencias, facilitadas por la misma herramienta de tracking.
- Creación de código fuente (que es responsabilidad del rol desarrollador dentro del producto).
- Relación bidireccional entre la etapa de análisis y verificación.
- Relación bidireccional entre la etapa de implementación con las etapas de requerimientos y verificación.

Se presentó un modelo de trazabilidad que comprende aspectos de análisis, implementación y verificación. Parte del mismo está automatizado en MantisBT.

Por otro lado, fue necesaria la incorporación del plugin Honey para completar el modelo de traza. Dicho plugin incorpora (Ver Fig. 47):

- Registro y gestión de conceptos del dominio a través del diccionario LEL.
- Creación automática de Casos de Uso y actores a partir del diccionario LEL, basándose en la estrategia planteada por Antonelli [Antonelli 2012], manteniendo la relación entre Casos de Uso y Símbolo de LEL.
- Creación y administración de Casos de Uso, actores y reglas en la herramienta de tracking.

- Paths de archivos (artefactos de desarrollo) asociados a Incidencias y a Casos de Uso registrados en MantisBT.

- Mención de incidencias dentro de las Notas de los Caso de Uso, quedando registrada la relación entre Incidencias y Casos de Uso.

- Almacenamiento de la información involucrada en cada derivación de LEL a Casos de Uso.

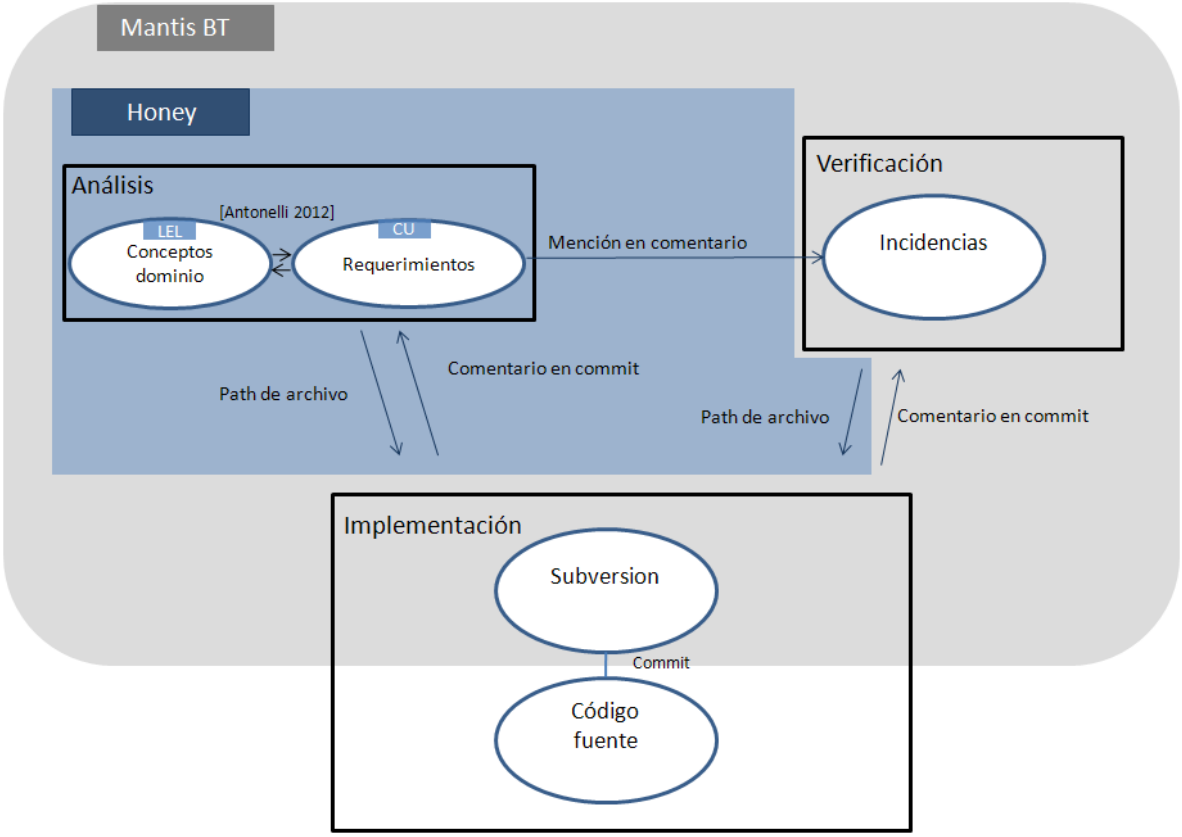


Figura 47. Aporte de Honey al modelo de trazabilidad deseado

La trazabilidad es una actividad importante en el desarrollo de software. Permite analizar el impacto de cambios de requisitos originados por el cliente como así también realizar el mantenimiento. También constituye un elemento importante en el proceso de mejora continua de la organización puesto que es la que permite integrar de forma armoniosa a todos los artefactos que forman parte de cada etapa del proyecto. No obstante, mantener la trazabilidad genera un costo adicional al proceso, pues se carece de herramientas eficaces que automaticen las actividades de registro y seguimiento de las trazas.

Para un software de trazabilidad, la dificultad radica en que no existe un patrón de empaquetamiento e intercambio de datos entre las herramientas existentes. Por este motivo, la información queda aislada, dificultando una visión global, necesaria para la toma de decisiones.

El presente trabajo de tesis dio origen a la creación del plugin Honey como software de trazabilidad, logrando integrar las etapas de análisis, implementación y verificación de un proyecto de software, beneficiando a los integrantes del mismo y colaborando con:

- Las tareas de los coordinadores de proyecto en la obtención de métricas que permiten hacer un seguimiento del producto y controlar los riesgos.

- El esfuerzo de los analistas y testers para controlar la calidad del producto que llega al usuario final.

- El trabajo cotidiano de los desarrolladores, brindando conocimiento de la funcionalidad afectada por los artefactos de desarrollo, evitando posibles errores.

- Las necesidades empresariales para obtener información en tiempo real con el fin de fidelizar a los clientes.

- El desarrollo tecnológico en plataformas informáticas y la obtención de información en la medida de sus movimientos.

5 Usabilidad y accesibilidad del plugin Honey

La presente sección pretende dar cierre a la creación del plugin Honey con las verificaciones de usabilidad y accesibilidad necesarias que garanticen el fácil acceso y buen uso de dicho módulo de MantisBT.

En primera instancia se define el concepto de usabilidad, su importancia y cuáles son las ventajas de que una herramienta de software cumpla parámetros específicos de usabilidad previamente definidos. Se realiza una evaluación de usabilidad, dentro de las evaluaciones de usabilidad existentes (Test de usuarios, Evaluación heurística, Evaluación automática), para lograr que Honey cumpla con los criterios de usabilidad preestablecido.

En segunda instancia se define el concepto de accesibilidad (que es una subdisciplina de la usabilidad), su importancia y las pautas necesarias para que cualquier producto sea accesible por personas discapacitadas. Más adelante se procede a aplicar éstas pautas en MantisBT y Honey.

5.1 Usabilidad

La usabilidad es un concepto aplicable a cualquier elemento en el cual se va a producir una interacción entre un humano y un dispositivo, no se limita a sistemas informáticos exclusivamente.

En el caso de los sistemas informáticos, la usabilidad va a abarcar desde el proceso de instalación de la aplicación hasta el punto en el que el sistema sea utilizado por el usuario, incluyendo también el proceso de mantenimiento.

La usabilidad, en lo que nos compete, puede considerarse como una característica que mide qué tan intuitiva y fácil de usar es un software para el usuario común.

Los 3 grandes aspectos relacionados a la usabilidad son: eficacia, eficiencia y satisfacción.

Usabilidad según la norma ISO 9241-11: “El grado en el cual un producto puede ser usado por unos usuarios específicos para alcanzar ciertas metas especificadas con efectividad, eficiencia y satisfacción en un contexto de uso especificado.”

Usabilidad conforme a la norma ISO/IEC 9126-1 11: “Capacidad de un producto de software de ser entendido, usado y atractivo para el usuario, cuando es usado bajo unas condiciones específicas.”

5.1.1 Atributos de la usabilidad

La usabilidad se asocia con cinco atributos definidos [Nielsen 1993], traducción adaptada de [Baeza 2002]:

Facilidad de aprendizaje (learnability): tiene que ser fácil aprender cómo funciona el sistema, tal que el usuario puede empezar a trabajar con ello lo más rápido posible.

Eficiencia de uso (efficiency): una vez que el usuario ha aprendido a utilizar el sistema, un nivel alto de productividad es posible para completar determinadas tareas.

Facilidad para recordar (memorability): cuando un usuario ha utilizado un sistema tiempo atrás, y tiene la necesidad de reutilizarlo de nuevo la curva de aprendizaje debe de ser significativamente menor que el caso del usuario que nunca haya utilizado dicho sistema.

Pocos errores (low error rate): este atributo se refiere a aquellos errores que comete el usuario al utilizar el sistema. Una aplicación ideal evitaría que el usuario cometiera errores y funcionaría de manera óptima a cualquier petición por parte del usuario. Es vital que una vez que se produzca un error, el sistema se lo haga saber rápida y claramente al usuario, le advierta sobre la severidad del mismo y le provea de algún mecanismo para recuperarse de ese error.

Satisfacción (satisfaction): este atributo se refiere a la impresión subjetiva del usuario respecto al sistema.

[Cato 2001] sugiere además los siguientes atributos:

Control: los usuarios deben de sentir que tienen el control por sobre la aplicación, y no al revés.

Habilidades: los usuarios deben de sentir que el sistema apoya y complementa sus habilidades y experiencia – el sistema tiene respeto por el usuario.

Privacidad: el sistema ayuda a los usuarios a proteger su información o la de sus clientes.

Es muy importante señalar que los atributos antes mencionados necesitan tener una ponderación acorde a la actividad que se quiera realizar con un sistema. Algunos sistemas darán una mayor importancia a ciertos atributos por sobre algunos otros. Todo dependerá de:

Los usuarios: ¿quién está utilizando el producto? Por ejemplo, ¿son usuarios novatos o con experiencia?

Los objetivos: ¿qué es lo que los usuarios intentan hacer con el producto? Por ejemplo ¿El producto ofrece lo que los usuarios quieren hacer?

El contexto de uso o las circunstancias en las cuales se usará la aplicación: ¿dónde y cómo utilizará el usuario al producto?

Existen otros factores que influyen en forma importante en la Usabilidad de un sitio web, tales como: la velocidad de descarga para mostrarse en pantalla, el manejo equilibrado de las imágenes, el uso de animaciones con criterio de escasez, el minimalismo, la jerarquización de la información, la minimización de los clics necesarios para lograr las acciones, la minimización del uso de las barras de desplazamiento (scroll), la utilización de buscadores dentro del sitio que permitan acceder más rápido a lo que el usuario está buscando, la posibilidad de ver el sitio en distintos idiomas, entre otros.

5.1.2 Importancia de la usabilidad

- Un sitio usable se puede aprender mejor.

- Su aprendizaje perdura más en la memoria.
- Reduce los errores cometidos por los usuarios.
- Aumenta la satisfacción de los usuarios.
- Mejora la experiencia global con el sitio.
- Se puede tener mayor ventaja competitiva.
- Se consigue una reducción de costos.
- Se reduce el número de abandonos del sitio.
- Las tareas se realizan más rápido y reduce pérdidas de tiempo.
- Para desarrolladores y fabricantes: cada dólar invertido en usabilidad, devuelve de 10 a 100 dólares.

5.1.3 Evaluación de la usabilidad

La evaluación de la usabilidad abarca una serie de metodologías y técnicas que ayudan a medir la forma en que los usuarios son capaces de utilizar un sitio web, al mismo tiempo que determinan la manera en que lo hacen. El llevarla a cabo derivará en la creación de mejores productos, por lo que conseguirá que los usuarios realicen sus actividades más fácilmente. De hecho, sin evaluación será imposible saber si un producto cumple las expectativas de sus creadores, o si se adapta a su contexto social, físico y organizativo.

La evaluación se presenta como la etapa más importante en el proceso de Diseño Centrado en el Usuario.

El principal objetivo de la evaluación de la usabilidad consiste en establecer si un sistema cumple las necesidades y expectativas del usuario y si, por lo tanto, éste se encuentra satisfecho.

La ventaja más destacada que aporta la evaluación es la de la localización y definición previa de los problemas, lo que implicará una mejora notable en la calidad del producto visible tanto a corto como a largo plazo.

5.1.3.4 Tipos de evaluación

Test de usuarios: este método de evaluación de la usabilidad, complementario a la evaluación heurística, se basa en la observación y análisis de cómo un grupo de usuarios reales utilizan el sitio web. Se irán anotando todos los problemas de uso reales con los que se encuentren para poder solucionarlos posteriormente.

El experimento estará controlado por el evaluador en un laboratorio de usabilidad dotado de ordenadores con conexión a la red. Consistirá en la observación y registro del comportamiento de los usuarios en tareas previamente acordadas. Por ello es recomendable algún método de grabación de los pasos dados en la realización de las tareas, previa autorización de los usuarios, así como un cuestionario en papel para otro tipo de apreciaciones.

Entre las variables a medir se encuentran el tiempo de realización de la tarea, el tiempo utilizado en recordar la estructura de la página, la satisfacción con la página web o el número de errores.

Los participantes en la evaluación, al menos cinco, no deben contar con más información de la necesaria de forma que actúen con libertad en su interacción con la web. Trabajarán por separado pero es recomendable que durante la prueba consulten todas sus dudas para que el evaluador tenga más datos sobre la usabilidad.

Una vez terminadas las tareas, los usuarios se sentarán con los evaluadores para intercambiar opiniones y poner en común sus dificultades y los puntos que destacan, tanto negativos como positivos.

Evaluación heurística o por criterios: se encuadra dentro de los métodos de inspección de la usabilidad de un sitio web. Los lleva a cabo el experto en usabilidad, y se basan en el recorrido y análisis del sitio. A partir de aquí localizará errores y problemas de diseño. El resultado será un informe que dé cuenta de las mejoras necesarias para que la web sea totalmente usable.

Debido a que está comprobado de diversas personas, aún en base a los mismos principios heurísticos, encontrarán diversos problemas, es recomendable usar un grupo de evaluadores. Dichos evaluadores trabajarán de forma individual para, al final, comunicar sus percepciones mediante informes por escrito en el que enumerarán los problemas y los explicarán de acuerdo a los principios de usabilidad.

Ya que lo que se pretende es evitar repetir los errores en el rediseño del sitio, es conveniente jerarquizar la gravedad de estos errores para ir actuando según lo apremiante que sea. En este sentido habrá que tener en cuenta tres factores:

La frecuencia con que ocurre el problema.

El impacto que tiene el problema en los usuarios.

La persistencia del problema en todo el sitio web.

La evaluación heurística puede aplicarse en cualquier fase del ciclo del desarrollo del proyecto. Es muy adecuada al principio, donde aún no hay demasiadas cosas que se puedan testear con usuarios.

Evaluación automática: este último sistema de evaluación consiste en el uso de software que detecta problemas elementales, como tamaños absolutos de fuentes y de tablas, formato de los textos, tamaño de las páginas, tiempos de descarga o enlaces rotos.

Sin embargo, aunque su ventaja es la rapidez y son considerados un buen punto de partida que ahorran trabajo posterior, este tipo de software no es capaz de detectar cuestiones globales de usabilidad (realmente las más importantes). Éstas sólo pueden ser detectadas por los evaluadores.

Por tanto observamos que el software es útil para todos aquellos aspectos cuantificables, que se pueden considerar de una forma objetiva. Sin embargo, aspectos relacionados con la comodidad, la estética, la legibilidad, la capacidad de actuar fácilmente con el sistema, etc, todos ellos más subjetivos y amplios (como decimos, las verdaderas cuestiones de usabilidad), son difícilmente medibles si no es por medio de personas.

5.1.4 Evaluación de la usabilidad en Honey

- Test de usuarios:

  Desarrollamos Honey respetando el estilo y la estructura básica de MantisBT y a su vez incorporamos características que mejoran la usabilidad del sitio:

  - el uso de animaciones con criterio de escasez.
  - el minimalismo.
  - la jerarquización de la información.
  - la minimización de los clics necesarios para lograr las acciones.
  - la minimización del uso de las barras de desplazamiento (scroll).
  - la utilización de buscadores dentro del sitio que permitan acceder más rápido a lo que el usuario está buscando.
  - Ampliación de la búsqueda por distintos criterios en todos los listados.
  - La opción de ordenar por la columna que el usuario elija en todos los listados.
  - Operaciones principales fácilmente identificables.
  - La opción de cambiar de idiomas (español, inglés).

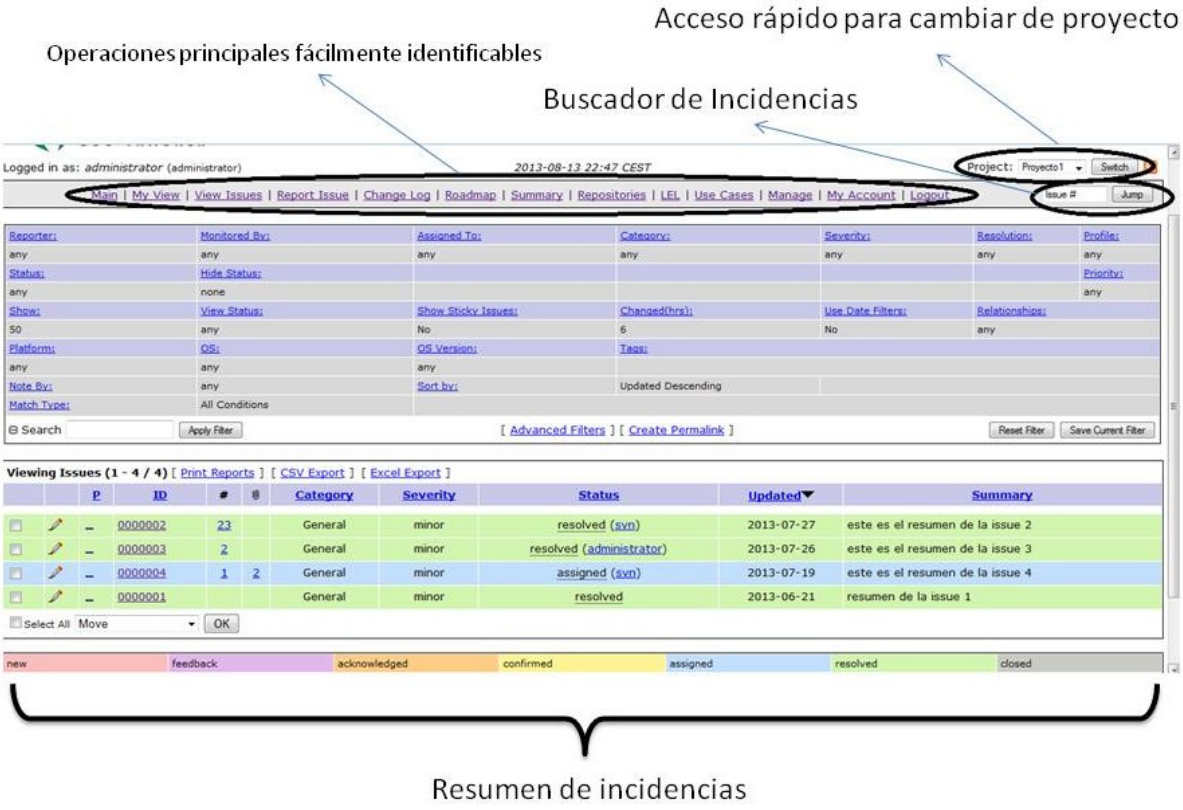


Figura 48. Características de usabilidad que cumple MantisBT

Además, incorporamos un rápido acceso a la información, de manera que si el usuario, por ejemplo, se encuentra generando un Caso de Uso y no recuerda con exactitud las reglas definidas, o las características de los actores, o la definición de un Caso de Uso del cual extiende o incluye, pueda ver dicha información sin tener que perder el hilo de la operación que estaba realizando.

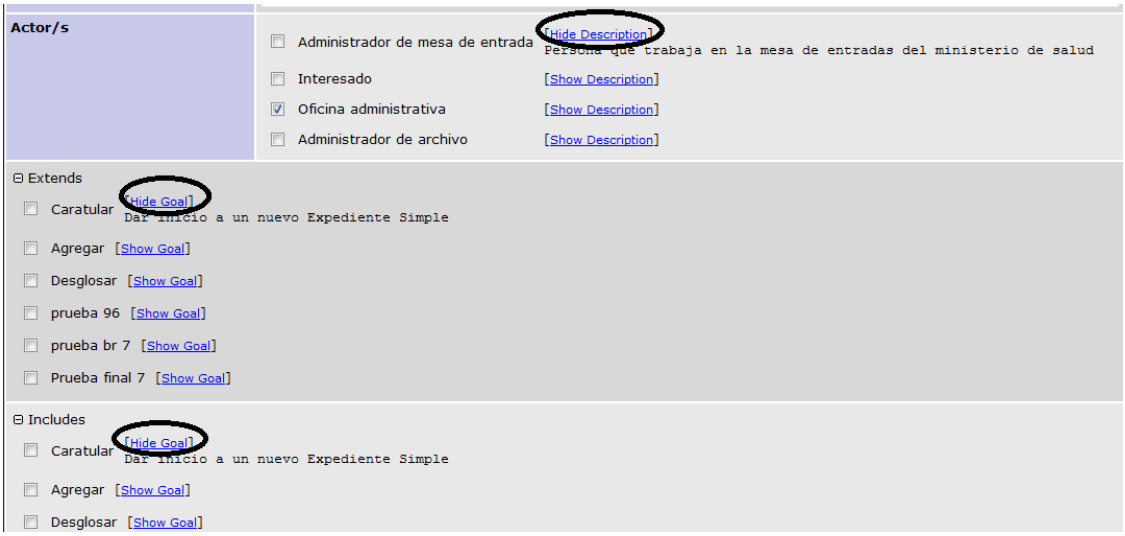


Figura 49. Accesos rápidos en Honey

- **Evaluación automática**

  Se hizo uso de distintas herramientas (ScreenFly, Screen Size Tester, Test Size) que permiten ver un sitio web con las distintas resoluciones de pantalla más utilizadas por los usuarios, por ejemplo: monitores, tablets, teléfonos celulares, televisores, etc., llegando a resultados exitosos.

  Además confirmamos la validez del estilo utilizado en el sitio, requerido por W3G (css-validator).

  Corroboramos que no se detectaron links rotos dentro del sitio (Xenu Link).

- **Evaluación heurística**

  Un experto de usabilidad realizó una evaluación del sistema contra ciertos parámetros o guías generales (heurística) sobre el plugin Honey creado en la herramienta MantisBT.

  A continuación se presenta la evaluación realizada sobre Honey indicando (mediante checkbox seleccionado) los parámetros que cumple la herramienta según el experto y aquellos que debían ser mejorados (mediante checkbox no seleccionado).

Tabla 27. Evaluación de usabilidad heurística sobre plugin Honey – Navegación

USABILIDAD DE INTERFAZ – Plugin Honey

Inspección – Evaluación Heurística con Expertos

Principios heurísticos

1. NAVEGACION

☐ **1.1 ¿Es fácil reconocer en todo momento el lugar en donde estoy parado dentro de la aplicación?**

Comentario del experto: El usuario siempre debe saber exactamente:
• Donde está.
• Donde ha estado.
• Donde puede ir.
Se debe aportar siempre información sobre la posición del usuario en la estructura.
Luego de la evaluación en Honey, el experto indicó que para poder lograr este parámetro heurístico debe incorporarse en el plugin la posibilidad de que los ítems de menú de la herramienta se identifiquen de algún color característico para informar al usuario el lugar donde se encuentra dentro del a aplicación.

☑ **1.2 ¿El usuario tiene el control?**

Comentario del experto: Este punto refiere a si el usuario se debe quedar esperando a que el plugin realice cierto procesamiento o bien que el usuario tenga la sensación de que la herramienta realizo alguna operación sin su consentimiento.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☑ **1.3. ¿El usuario tiene libertad de movimientos? (Tener en cuenta que no debe haber limitaciones incoherentes en la navegación)**

Comentario del experto: Este parámetro heurístico refiere a que el usuario pueda ir desde un punto del plugin a cualquier otro que no esté relacionado con la posición actual dentro de la herramienta.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☐ **1.4 ¿Es flexible y eficiente la navegación? (¿Son cómodos y rápidos los comandos?)**

Comentario del experto: este punto refiere especialmente a si los botones para aplicar los comandos que ofrece la herramienta se encuentran en todos los formularios o pantallas respetando el mismo orden y un tamaño considerable para que el usuario pueda operar cómodamente.
Luego de la evaluación en Honey el experto de usabilidad indico que si bien el plugin debe usar la estética de botones heredada por MantisBT, deben acomodarse los formularios para que todos los botones se encuentren en la misma posición y distancia.

☐ **1.5 ¿Están las salidas de emergencia claramente marcadas?**

Comentario del experto: Este punto refiere básicamente a que todas las operaciones que se realizan en MantisBT tengan la posibilidad de ser canceladas.
Luego de la evaluación en Honey el experto de usabilidad indicó que faltaba la incorporación del botón cancelar en algunos formularios tales como la confirmación al borrar un Símbolo.

☑ **1.6 ¿Es deseable que tenga una estructura de navegación en niveles de detalle progresivo?**

Comentario del experto: Refiere a la necesidad de que la herramienta mantenga barra navegación dado que el usuario puede meterse en ramas muy profundas de funcionalidad.
Luego de la evaluación en Honey el experto de usabilidad observo que no hay operaciones con gran nivel de profundidad que requieran la incorporación de una barra de navegación.

Tabla 28. Evaluación de usabilidad heurística sobre plugin Honey – Herramienta

2. HERRAMIENTA ORIENTADA AL USUARIO Y NO A LA TECNOLOGIA

☑ **2.1 ¿El lenguaje está orientado al usuario?**
Comentario del experto: Este punto se refiere a si la redacción y términos utilizados por la herramienta son los suficientemente claros y amigables al usuario.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☑ **2.2 Los conceptos están orientados al usuario?**
Comentario del experto: Refiere a que los términos utilizados en la aplicación sean compresibles por el usuario.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☑ **2.3 ¿El lenguaje es simple y natural? (Tiene que haber sólo información relevante por el principio de "Less is more")**
Comentario del experto: Toda información de lectura para el usuario debe ser lo más concisa posible.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☐ **2.4 ¿La información está ordenada lógicamente?**
Comentario del experto: El contenido de la herramienta debe estar agrupado de acuerdo a conceptos específicos y principales de la herramienta.
Luego de la evaluación en Honey, el experto de usabilidad observo que el dominio se agrupaba en los conceptos de LEL y Casos de Uso, en los formularios los campos se encuentran agrupados bajo criterios específicos y las tablas con información se mantienen ordenadas por el concepto más comúnmente utilizado. El experto sugirió que en los formularios de búsqueda exista un filtro por cada una de las columnas de la grilla resultado y además se permita ordenar por cualquiera de las columnas en la grilla resultado.

Tabla 29. Evaluación de usabilidad heurística sobre plugin Honey – Consistencia

3. CONSISTENCIA

☑ **3.1 ¿Hay consistencia en el lenguaje? (A igual palabra → igual significado)**
Comentario del experto: un concepto dentro de la herramienta debe mantener con el mismo nombre, no es correcto mencionar al mismo concepto con distintos nombres dentro de la herramienta.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☐ **3.2 ¿Hay consistencia en el comportamiento? (A igual componente → igual comportamiento)**
Comentario del experto: Refiere a que la herramienta tiene que ser lo más predecible posible, por ejemplo, cada vez que se desea realizar la eliminación de una entidad, ¿hay confirmación? En todos los lugares donde se edita una entidad ¿se visualiza el formulario con los datos de ésta?
Luego de la evaluación en Honey, el experto de usabilidad indicó que faltaban formulario donde luego de realizar una operación se solicite confirmación por parte del usuario, por ejemplo: ¿Esta seguro que desea borrar?

☐ **3.3 ¿Hay consistencia en la apariencia? (Ver colores, iconos, fuentes, formatos de ventana, etc.)**
Comentario del experto: En la herramienta todos los carteles de confirmación, carteles de advertencia, links y botones deben conservar la misma estética.
Luego de la evaluación en Honey, el experto indicó que seria conveniente incorporar en los carteles de confirmación iconos que representen la acción del cartel: Error, Advertencia, Confirmación.

☐ **3.4 ¿Hay consistencia en el manejo de errores? (A igual error → igual forma de reportarlo)**
Comentario del experto: El mensaje que se le brinda al usuario ante la presencia de algún error debe ser en todos los casos de error estéticamente igual.
Luego de la evaluación en Honey, el experto señaló que debe modificarse este punto, que se haría en conjunto con la mejora del punto 3.3

☐ **3.5 ¿Hay consistencia en la ayuda? (ver punto 6)**
Comentario del experto: En todos los formularios de la herramienta la ayuda al usuario sobre el uso de ésta debe manejarse de la misma manera.
Luego de la evaluación en Honey, el experto sugirió la incorporación de tooltips en los campos de los formularios que indique como completar éstos y además poner texto más chico de ayuda al lado de campos complejos tales como el agregado de Símbolos o Impactos.

Tabla 30. Evaluación de usabilidad heurística sobre plugin Honey – Reconocer

4. RECONOCER ES MEJOR QUE RECORDAR

☐ **4.1 ¿Se cumple a nivel de los datos? (En el uso de combos, etc.)**

Comentario del experto: Este punto refiere a tener en cuenta el criterio de los datos que van en cada campo y acomodar el mismo según los datos, por ejemplo: si ya se conoce que los datos son un número fijo es conveniente la utilización de combos.
Luego de la evaluación en Honey, el experto observo que en los formularios donde se utilizan reglas de dominio, actores y casos de uso que extienden y usan otro se debe incorporar algún mecanismo de paginado debido a la gran cantidad de información que se maneja en estos campos.

☑ **4.2 ¿Se cumple a nivel de apariencia? (En el uso de iconos, colores, etc. ¿Son fáciles de reconocer?)**

Comentario del experto: Cada icono o color distintivo que se encuentra en la herramienta identifica algo claramente reconocible por el usuario.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

Tabla 31. Evaluación de usabilidad heurística sobre plugin Honey – Errores y Feedback

5. ERRORES Y FEEDBACK

☑ **5.1 ¿Se previenen lo errores?**

Comentario del experto: Este punto refiere a evitar que se muestren al usuario errores del servidor.
Luego de la evaluación en Honey, el experto de usabilidad indico que el plugin cumple con este punto.

☐ **5.2 Los mensajes de error y feedback son:**

5.2.1¿Claros?
5.2.2¿Cortos?
5.2.3¿Auto explicativos?
5.2.4¿Dan la solución al problema?

Tabla 32. Evaluación de usabilidad heurística sobre plugin Honey – Ayuda y Documentación

6. AYUDA Y DOCUMENTACION

☑ **6.1 ¿El acceso a la ayuda es fácil de encontrar?**

Comentario del experto: Refiere a la incorporación de hints y tooltip fáciles de visualizar por el usuario.
Luego de la evaluación en Honey, el experto de usabilidad indico que este punto será cumplimentado con la resolución del punto 3.5.

☐ **6.2 ¿El acceso a la ayuda es rápido?**

Luego de la evaluación en Honey, el experto de usabilidad indico que este punto será cumplimentado con la resolución del punto 3.5.

☐ **6.3 ¿Hay consistencia en la ubicación de los accesos a la ayuda contextual?**

Luego de la evaluación en Honey, el experto de usabilidad indico que este punto será cumplimentado con la resolución del punto 3.5

☐ **6.4 El icono es fácil de reconocer? (Principio "reconocer es mejor que recordar")**

Luego de la evaluación en Honey, el experto de usabilidad indico que este punto no aplica sobre el plugin Honey dado que no hay icono especifico de ayuda y no es necesario.

A continuación se detallan los puntos de heurística de usabilidad que fueron mejorados en el plugin Honey luego de la evaluación anteriormente descripta.

Punto 1.1. ¿Es fácil reconocer en todo momento el lugar en donde estoy parado dentro de la aplicación?

Se indicó en color resaltado el menú o submenús del plugin Honey donde se encuentra el usuario

Figura 50. Honey ajustado a principio de heurística de usabilidad 1.1

Punto 1.4. ¿Es flexible y eficiente la navegación? (¿Son cómodos y rápidos los comandos?)

En los formularios se acomodaron los botones de manera que permanezcan centrados, y en los casos donde exista un botón "Cancelar" el experto de usabilidad sugirió que se mantenga mayor distancia respecto al resto de los botones.

Figura 51. Honey ajustado a principio de heurística de usabilidad 1.4

Punto 1.5. ¿Están las salidas de emergencia claramente marcadas?

Se incorporo el botón "Cancelar" en todos los formularios donde hay accionar del usuario.

Punto 2.4. ¿La información está ordenada lógicamente?

En los formularios de búsqueda se incorporo:

1. Un filtro por cada uno de los campos principales de la entidad buscada. En la figura se observa un combo con los filtros por ID o Nombre de Símbolo.

2. Posibilidad de ordenar la grilla por cualquiera de sus columnas. En la figura la grilla se encuentra ordenada por tipo de símbolo.

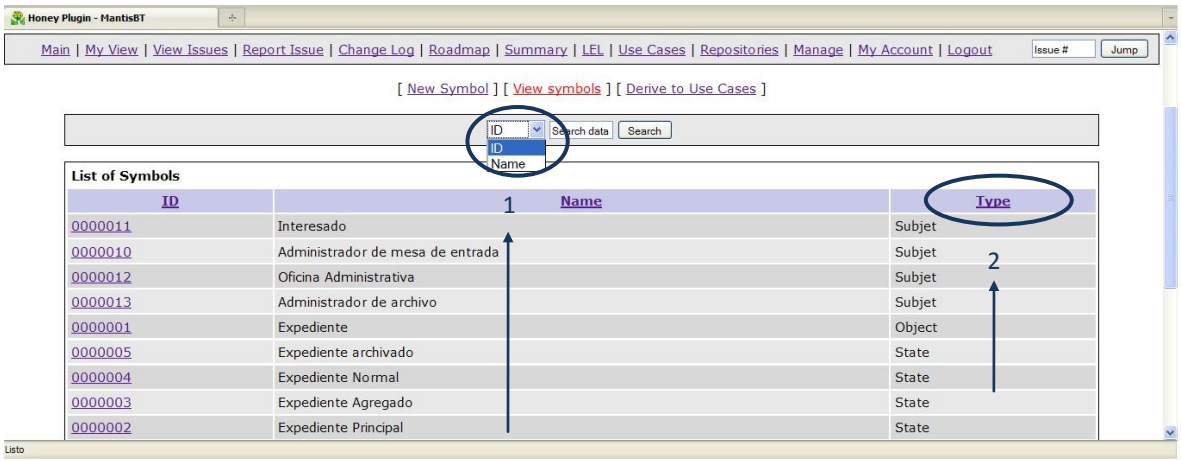


Figura 52. Honey ajustado a principio de heurística de usabilidad 2.4

Punto 3.2. ¿Hay consistencia en el comportamiento? (A igual componente → igual comportamiento)

Se incorporo el siguiente mensaje de confirmación dentro de todas las acciones que lo requerían dentro del plugin Honey

Figura 53. Honey ajustado a principio de heurística de usabilidad 3.2

Punto 3.3. ¿Hay consistencia en la apariencia? (Ver colores, iconos, fuentes, formatos de ventana, etc.)

Se incorporaron los íconos:

1. Advertencia
2. Error
3. Confirmación

Figura 54. Honey ajustado a principio de heurística de usabilidad 3.3

Punto 3.4. ¿Hay consistencia en el manejo de errores? (A igual error → igual forma de reportarlo)

Este punto fue mejorado en conjunto con el punto 3.3.

Punto 3.5. ¿Hay consistencia en la ayuda? (ver punto 6)

Se incorporaron textos de ayuda (tooltips) a los campos de los formularios.

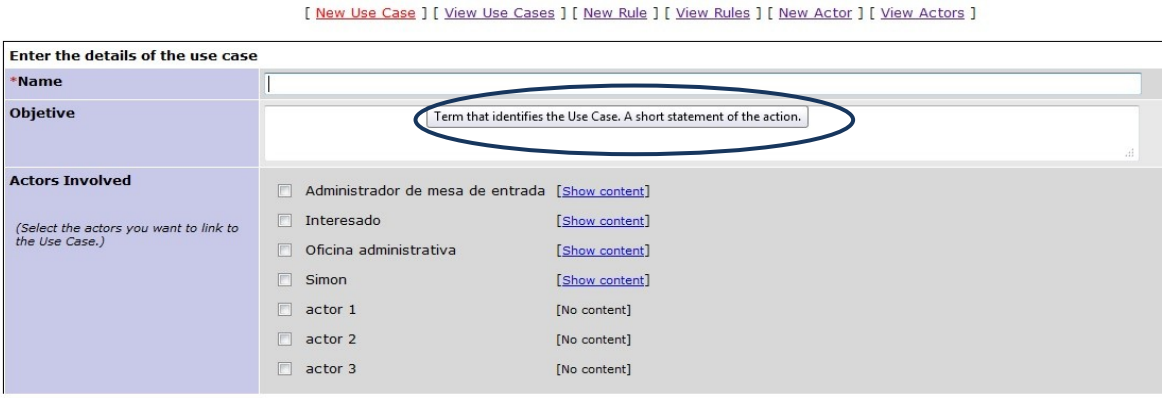


Figura 55. Honey ajustado a principio de heurística de usabilidad 3.5

Punto 4.1. ¿Se cumple a nivel de los datos? (En el uso de combos, etc.)

1. En los campos complejos se incorporo descripción breve de su funcionamiento.
2. Se agrego paginado en campos que manejan gran cantidad de información.

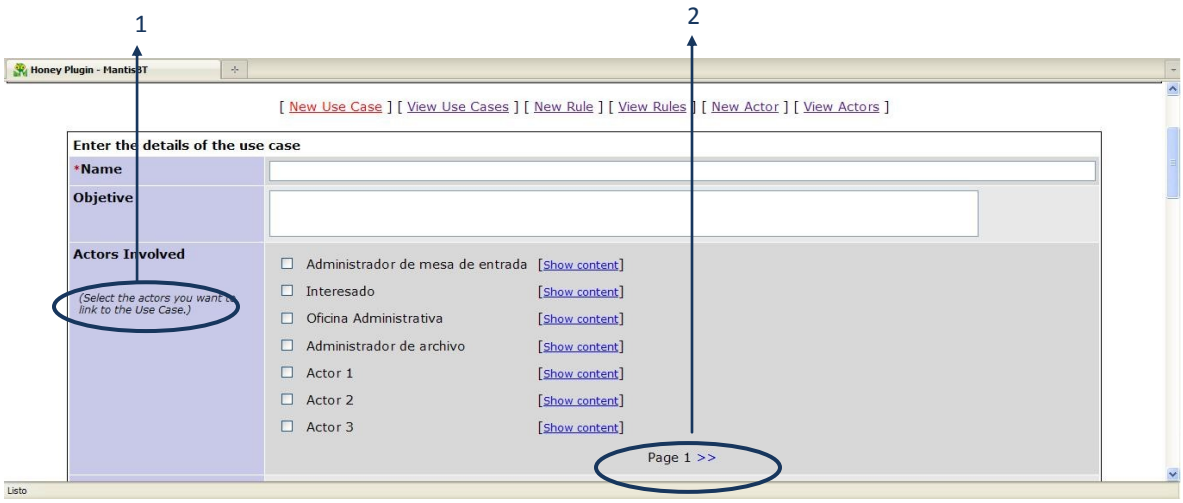


Figura 56. Honey ajustado a principio de heurística de usabilidad 4.1

Punto 5.2. Los mensajes de error y feedback son

5.2.1 ¿Claros?

5.2.2 ¿Cortos?

5.2.3 ¿Auto explicativos?

5.2.4 ¿Dan la solución al problema?

Este punto fue mejorado en conjunto con los puntos 2.3 y 3.4

Punto 6.1. ¿El acceso a la ayuda es fácil de encontrar?

Este punto fue mejorado en conjunto con el punto 3.5

Punto 6.2. ¿El acceso a la ayuda es rápido?

Este punto fue mejorado en conjunto con el punto 3.5

Punto 6.3. ¿Hay consistencia en la ubicación de los accesos a la ayuda contextual?

Este punto fue mejorado en conjunto con el punto 3.5

5.2 Accesibilidad

La accesibilidad es una sub-disciplina de la usabilidad

La accesibilidad, o la usabilidad universal, está obteniendo cada vez mayor importancia en la "ERA" (Europear Research Area) de las tecnologías de la sociedad de la información, cuyo objetivo es que la sociedad sea para todos, a cualquier hora, en cualquier sitio. La accesibilidad se centra en que las aplicaciones software y Web sean usables para todos, incluso para las personas con discapacidades.

La accesibilidad Web significa que personas con algún tipo de discapacidad van a poder hacer uso de la Web. En concreto, al hablar de accesibilidad Web se está haciendo referencia a un diseño Web que va a permitir que estas personas puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos. La accesibilidad Web también beneficia a otras personas, incluyendo personas de edad avanzada que han visto mermadas sus habilidad a consecuencia de la edad.

La accesibilidad Web engloba muchos tipos de discapacidades, incluyendo problemas visuales, auditivos, físicos, cognitivos, neurológicos y del habla.

5.2.1 La importancia de la accesibilidad

La Web es un recurso muy importante para diferentes aspectos de la vida: educación, empleo, gobierno, comercio, sanidad, entretenimiento y muchos otros. Es muy importante que la Web sea accesible para así proporcionar un acceso equitativo e igualdad de oportunidades a las personas con discapacidad. Una página Web accesible puede ayudar a personas con discapacidad a que participen más activamente en la sociedad.

La Web ofrece a aquellas personas con discapacidad una oportunidad de acceder a la información y de interactuar.

Otra consideración importante para las empresas es que la accesibilidad Web es un requisito establecido en algunos casos por leyes y políticas.

5.2.2 Pautas de accesibilidad

- **Proporcione alternativas equivalentes para el contenido visual y auditivo.**

  Los textos alternativos al contenido visual o auditivo benefician a personas ciegas y/o sordas y a aquellos usuarios que deciden anular la descarga de imágenes y/o sonidos (velocidad de acceso a Internet limitada).

  Los equivalentes no textuales, como pueden ser dibujos o vídeos, benefician a personas analfabetas o con dificultades en la lectura.

- **No se base sólo en el color.**

  Los textos y gráficos deben comprenderse sin necesidad de ver los colores. El cumplimiento de esta pauta beneficia a personas con dificultades para ver los colores y a usuarios que utilizan pantallas monocromáticas.

- **Utilice marcadores y hojas de estilo y hágalo apropiadamente.**

  El control de la presentación de los contenidos se debe realizar con hojas de estilo en vez de con elementos y atributos de presentación. Con el uso de marcadores de presentación los usuarios que utilizan software especializado tendrán dificultades para entender la estructura de la página.

- **Identifique el idioma usado.**

  Esta pauta implica usar marcadores que faciliten la pronunciación o interpretación de texto abreviado o extranjero. Se debe indicar el idioma predominante en cada página y marcar aquellas expresiones que se encuentren en otra lengua. De esta forma, los sintetizadores de voz son capaces de cambiar su pronunciación en función del idioma siempre y cuando se usen los marcadores apropiados.

- **Cree tablas que se transformen correctamente.**

  Las tablas sólo se utilizan para marcar información tabular (tablas de datos). El uso de tablas con otros fines crea dificultades para los usuarios que usan lectores de pantalla. De igual forma, las tablas mal estructuradas (por ejemplo, sin encabezados <th>) dificultan la lectura a usuarios que no pueden visualizar la información de forma global: ciegos con lectores de pantalla y/o dispositivos braille, deficientes visuales que utilizan magnificadores de pantalla o usuarios con dispositivos de pantalla pequeña.

- **Asegúrese de que las páginas que incorporen nuevas tecnologías se transformen correctamente.**

  Una página basada en tecnologías modernas tiene que ser accesible al desconectarla o al visualizarla con navegadores antiguos. El usuario puede desconectar las tecnologías más modernas para ganar en rapidez de descarga. Sin embargo, los contenidos deben permanecer accesibles.

- **Asegure al usuario el control sobre los cambios de los contenidos tempo-dependientes.**

  El movimiento de los objetos o páginas, su parpadeo o actualización automática deben ser controlados por el usuario. Las personas con discapacidades cognitivas o visuales no pueden leer textos en movimiento. De forma similar, algunos discapacitados físicos no pueden interactuar con objetos móviles (limitaciones motrices).

- **Asegure la accesibilidad directa de las interfaces incrustadas.**

Cuando un objeto incrustado (flash, applet) tiene su "propia interfaz", ésta (al igual que la interfaz de su navegador) debe ser accesible. Si la interfaz del objeto incrustado no puede hacerse accesible, debe proporcionarse una solución alternativa accesible.

- **Diseñe para la independencia del dispositivo.**

  Esta pauta significa que el usuario puede interactuar con la aplicación de usuario o el documento con un dispositivo de entrada (o salida) preferido - ratón, teclado, voz, puntero de cabeza (licornio) u otro. Si, por ejemplo, un control de formulario sólo puede ser activado con un ratón u otro dispositivo de apuntamiento, alguien que use la página sin verla, con entrada de voz, con teclado o quien utilice otro dispositivo de entrada que no sea de apuntamiento, no será capaz de utilizar el formulario.

- **Utilice soluciones provisionales.**

  Las alternativas accesibles sólo son imprescindibles hasta que los antiguos navegadores y las ayudas técnicas operen correctamente.

- **Utilice las tecnologías y pautas W3C.**

  Cuando no se pueda usar una tecnología W3C o al usarla se obtengan materiales que no se transformen correctamente, se debe proporcionar una versión alternativa. Se recomiendan las tecnologías W3C por incluir características accesibles incorporadas, estar desarrolladas en un proceso abierto consensuado y porque se utilizan como base para crear contenidos accesibles.

- **Proporcione información de contexto y orientación.**

  Esta información ayuda al usuario a comprender páginas o elementos complejos. Se deben agrupar los elementos y ofrecer información contextual sobre la relación entre elementos. Esta acción es fundamental para discapacitados cognitivos y visuales.

- **Proporcione mecanismos claros de navegación.**

  Estos mecanismos facilitan a todos los usuarios la búsqueda de aquella información que necesitan (fundamental para discapacitados cognitivos y visuales). Ejemplos: mapa web, ayuda, barras de navegación, etc.

- **Asegúrese de que los documentos sean claros y simples.**

  La utilización de lenguaje claro y simple facilita la comunicación de información. El acceso a la información escrita puede ser difícil para discapacitados cognitivos o con dificultad de aprendizaje y para personas sordas o que hablan en una lengua extranjera. La comprensión de un documento también depende de la maquetación de la página y de los gráficos (que deben llevar un texto alternativo).

5.2.3 Accesibilidad aplicada a Honey

Teniendo en cuenta las pautas antes mencionadas, enumeraremos a continuación los controles de Honey que ayudaron a mejorar la accesibilidad del sitio:

- que haya un título de la página;
- Proveer texto alternativo para cada elemento visual (atributos alt, longdesc);
- permitir la adaptación del sitio si se utiliza el zoom del navegador;
- controlar que las tareas no se realicen a través de un único periférico;
- controlar cómo funciona la página web cuando se "linealiza" en una columna y se cambia la presentación utilizada por personas con baja visión y/o ciegos
- si se utilizan mapas de imágenes, proveer texto explicativo para cada región.
- Verificar que los elementos multimedia tengas audio explicativo.
- Proveer links o botones auto-descriptivos.
- Utilizar hojas de estilo para controlar la disposición y la presentación
- Utilizar posiciones relativas y no absolutas.
- Especificar el nombre completo de una abreviatura o acrónimo la primera vez que aparece en el documento.
- Utilizar tablas completas (theader, tbody) y sólo para datos, no para organizar el texto (Si se utiliza una tabla para maquetar, no utilizar marcadores estructurales para el efecto visual de formato).
- Utilizar atributo Summary para las tablas.
- Controlar que los elementos frame o iframe tengan los sufijos de extensión, ejemplo: .html, php, asp, etc.
- Evitar la utilización de texto en movimiento.

Por otra parte pudimos comprobar el correcto funcionamiento de nuestro sistema a través de la utilización de herramientas que proporcionan las características de los navegadores más utilizados (IETester, Cross browser testing).

5.3 Conclusión

Para lograr un alto grado de uso (usabilidad) en Honey se expuso la herramienta a una evaluación heurística que se llevó a cabo por un experto de usabilidad, quien recorrió el plugin Honey detectando los puntos débiles que debían ser corregidos en el módulo. Luego se realizaron las correcciones sugeridas por el experto de usabilidad; las cuales son expuestas en el presente trabajo de tesis.

Teniendo en cuenta las pautas de accesibilidad definidas en el presente documento y probando el correcto funcionamiento del sistema a través de la utilización de herramientas especificas, se realizaron las reformas necesarias en Honey; las cuales ayudaron a mejorar la accesibilidad del sitio.

De esta forma, se logró una herramienta accesible, intuitiva y fácil de usar lo cual es de gran importancia para que la aplicación del modelo propuesto sea efectiva.

El principal objetivo de la evaluación de la usabilidad consiste en establecer si un sistema cumple las necesidades y expectativas del usuario y si, por lo tanto, éste se encuentra satisfecho.

La ventaja más destacada que aporta la evaluación es la de la localización y definición previa de los problemas, lo que implicará una mejora notable en la calidad del producto visible tanto a corto como a largo plazo.

6 Conclusiones

En esta sección se sintetizan los aportes de esta tesis, como así también se hace un balance de los aspectos positivos y negativos, para finalmente indicar líneas de futuros trabajos.

6.1 Resumen

La trazabilidad en los Proyectos de Desarrollo de Sistemas de Software es de gran importancia por tratarse de un hilo conductor que cruza a través de las numerosas etapas del proceso de desarrollo, asegurando un vínculo indispensable para el éxito del proyecto y brindando la necesaria garantía de coherencia, completitud y corrección en el producto de software, así como posibilitar su posterior mantenimiento correctivo y preventivo.

La trazabilidad constituye un gran apoyo, que debe formar parte en todas las etapas del desarrollo de un sistema, desde la captura de requerimientos hasta la implementación. Esto contribuirá al seguimiento de un requerimiento hacia adelante o hacia atrás, reflejando los efectos que puede tener, la inclusión o exclusión de un nuevo requisito.

Alcanzar la traza, entre artefactos que surgen de cada etapa del proceso de desarrollo de un proyecto y las métricas que se pueden obtener a partir de la información recolectada por dicha traza, son los motivos que llevaron a desarrollar el presente trabajo.

Independiente del modelo de ciclo de vida seleccionado para un proyecto o producto de software, todos tienen en común las etapas de Análisis, Desarrollo y Pruebas.

En el presente trabajo de tesis se brindaron los elementos necesarios para lograr un modelo de trazabilidad que captura progresivamente el avance entre distintos artefactos que surgen durante cada etapa. Esto implicó la investigación de:

- Léxico Extendido del Lenguaje utilizado para capturar el lenguaje de la aplicación con el que se escribirán los requerimientos. [Antonelli 2012]
- Derivación del diccionario LEL a Casos de uso. [Antonelli 2012]
- La posibilidad de relacionar el código ejecutable, mediante la utilización del repositorio donde éste se encuentra, al Caso de Uso o Incidencia al cual alude.

El modelo propuesto comprende la interconexión entre las herramientas utilizadas en las distintas etapas del proyecto, lo cual permitió alcanzar la integración entre:

- los requerimientos (artefactos de la etapa de análisis),
- el código fuente (artefacto de la etapa de implementación)
- las incidencias (artefacto de la etapa de verificación).

La elección de la estrategia [Antonelli 2012] para la derivación entre artefactos de la etapa de análisis permitió conocer qué símbolo del diccionario LEL crea a un UC o actor particular. Esta

relación unidireccional logró convertirse en una traza bidireccional, dado que, a través de este trabajo de tesis, el Caso de Uso o actor conoce el símbolo que lo generó.

Relacionar los artefactos de desarrollo con la funcionalidad (Caso de Uso) o Incidencia registrados en la herramienta de tracking es tarea exclusiva del desarrollador. Sin embargo, en el presente trabajo se investigó cómo se realiza esta tarea y se implementó la traza bidireccional, a objetivos de lograr un modelo de traza que integre todas las etapas de un proyecto de desarrollo de software.

MantisBT ofrece parte de la funcionalidad para conquistar un fragmento de la traza esperada, tal como Integración con Subversion y facilidad de ser extendida. Utilizando estas dos características se creó plugin Honey para completar el modelo de traza. Este software de trazabilidad incorporó:

- Registro y gestión de conceptos del dominio de un producto particular a través del diccionario LEL.
- Creación automática de Casos de Uso y actores a partir del diccionario LEL basándose en la estrategia planteada por Antonelli [Antonelli 2012], manteniendo la relación entre Casos de Uso y Símbolo de LEL.
- Creación y administración de Casos de Uso, actores y reglas en la herramienta de tracking.
- Almacenamiento de los paths de archivos (artefactos de desarrollo) asociados a Incidencias y Casos de Uso registrados en Mantis.
- Mención de incidencias dentro de las Notas de los Caso de Uso, quedando registrada la relación entre Incidencias y Casos de Uso.

De esta manera, Honey logró integrar las etapas de análisis, implementación y verificación de un proyecto de software, beneficiando a los integrantes del mismo y colaborando con:

- Las tareas de los coordinadores de proyecto en la obtención de métricas que permiten hacer un seguimiento del producto y controlar los riesgos.
- El esfuerzo de los analistas y testers para controlar la calidad del producto que llega al usuario final para.
- El trabajo cotidiano de los desarrolladores, brindando conocimiento de la funcionalidad afectada por los artefactos de desarrollo, evitando posibles errores.
- Las necesidades empresariales para obtener información en tiempo real con el fin de fidelizar a los clientes.
- El desarrollo tecnológico en plataformas informáticas y la obtención de información en la medida de sus movimientos.

Para lograr un alto grado de uso (usabilidad) en Honey se expuso la herramienta a una evaluación heurística que se llevó a cabo por un experto de usabilidad, quien recorrió el plugin Honey detectando los puntos débiles que debían ser corregidos en el módulo. Luego se realizaron las correcciones sugeridas por el experto, la cuales se expusieron en el presente trabajo de tesis.

Teniendo en cuenta pautas de accesibilidad definidas y probando el correcto funcionamiento del sistema a través de la utilización de herramientas especificas, se realizaron las reformas necesarias en Honey que ayudaron a mejorar la accesibilidad del sitio.

6.2 Aspectos positivos y negativos

- Entre las herramientas de tracking investigadas para poder llevar a cabo el presente trabajo, se seleccionó MantisBT por tener parte de la trazabilidad deseada en su módulo de integración con Subversion y por proveer facilidad de extensión; lo que permitió crear el plugin Honey. Sin embargo su estética se destaca como **aspecto negativo** de la herramienta, dado que contiene una interfaz muy elemental basaba en tablas de datos y enlaces.

- Un **aspecto positivo** a destacar es que la herramienta de tracking utilizada a objetivos de lograr el modelo de trazabilidad, es de fácil instalación, al igual que sus extensiones entre las que se encuentra el plugin Honey.

- El modelo de trazabilidad logrado en el presente trabajo no sólo permite hacer el seguimiento de los requerimientos desde su nacimiento hasta su mantenimiento, sino que además este seguimiento se realiza de manera automatizada e integrada en un único software, MantisBT con el plugin Honey. De esta manera se evita el tiempo insumido entre coordinadores, analistas, desarrolladores y testers para lograr la interconexión entre las herramientas de cada área y así obtener la información de trazabilidad. El hecho de que toda la información se maneje en una misma herramienta se denota como **aspecto positivo** del modelo creado en el presente trabajo.

- Honey registra en su base de datos todas las relaciones que se van generando, a lo largo del ciclo de vida del proyecto, entre los artefactos que se producen en cada etapa. Además garantiza la consistencia de estas relaciones. Estos dos **aspectos positivos** de la herramienta son de gran ayuda al momento de obtener métricas, lograr una visión objetiva sobre el estado del proyecto y llegar a conclusiones de calidad. Cuanta más información se almacene respecto al ciclo de vida del proyecto, mayores y mejores métricas obtendremos.

- Cuando se realiza una derivación entre artefactos de análisis (LEL – UC) se generan referencias (relaciones bidireccionales entre Símbolos LEL, Casos de Uso y Actores) que son enriquecidas a medidas que trascurre el proyecto de software, tal como creación manual y relación con nuevos Casos de Uso, Actores, Reglas de dominio, etc. Volver a realizar una generación de Casos de Uso a partir del diccionario LEL (derivación), en una etapa avanzada del proyecto, implica quitar funcionalidad que fue creada por fuera de la derivación. Honey detecta esta situación y evita generar la derivación si es que existen relaciones entre entidades derivadas y entidades creadas manualmente. Esto se detecta como **aspecto negativo** de la herramienta dado que, aunque garantiza la integridad de los

datos, se esperaría que se comporte de manera automática registrando el histórico de todas las relaciones.

6.3 Futuros trabajos

- Identificar métricas comúnmente usadas e incorporarlas a Honey para obtenerlas de manera automática Por ejemplo: En las tareas de MantisBT se registra el tiempo estimado y tiempo insumido. El tiempo estimado en la tarea es calculado en base a la complejidad del Caso de Uso (ejemplo en UCP[3]). La idea es que a medida que se va utilizando la herramienta se vayan registrando los cálculos y luego en cada nueva iteración al cargar un nuevo UC o funcionalidad la herramienta automáticamente estime a juicio experto.

- Proveer diagramas de alto nivel en Honey, como por ejemplo, la generación automática de diagramas de Casos de Uso a partir de los UC, actores y relaciones entre estos.

- Proveer soporte para metodologías ágiles a través de la incorporación de User Stories como unidad de análisis a Honey, para ofrecer trazabilidad automatizada y bidireccional en proyectos de metodologías ágiles.

- Proveer versionado de las derivaciones de LEL a UCs, el cual consistiría en permitirle al usuario descartar la derivación actual y restaurar una anterior, basándose en el historial de derivaciones que provee Honey para tal fin.

- Enriquecer la estrategia de derivación de LEL a UCs, permitiéndole al usuario decidir qué relaciones conservar o descartar frente a una nueva derivación del diccionario.

- Mejorar la relación de la implementación con la verificación y el análisis para facilitar la obtención de métricas, persistiendo, de manera independiente, cada path de archivo involucrado en un commit al repositorio de versionado.

- Crear la relación automática entre la verificación y el análisis mediante la identificación de los requerimientos de Honey en las notas de las incidencias de MantisBT.

[3] Puntos de Casos de Uso (en inglés Use Case, UCP): método de estimación de esfuerzo para proyectos de software, a partir de sus casos de uso. El método utiliza los actores y casos de uso relevados para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado.

7 Referencias Bibliográficas

[ADOdb 2012] ADOdb Data Dictionary Library for PHP, accedido en septiembre, 2012.

[Almazán 2005]: Almazán, Felipe: Las claves de la usabilidad, los gurúes Nielsen y Krug. Biblioteca del Congreso Nacional de Chile. Noviembre (2005).

[Antonelli 1999]: Antonelli, L., Oliveros, A., Rossi, G.: Baseline Mentor, An application that derives CRC Cards from Lexicon and scenario. XXVIII JAIO. WER'99. Argentina. (1999).

[Antonelli 2003]: Antonelli, L., Rossi G., Oliveros, A.: Traceability en la elicitación y especificación de requerimientos, Tesis presentada a la Facultad de Informática de la Universidad Nacional de La Plata como parte de los requisitos para la obtención del título de Magister en Ingeniería de Software, Bueno Aires, Argentina, Febrero (2003).

[Antonelli 2012]: Antonelli, L., Rossi, G., Leite, J.C.S.P., Oliveros, O.: Deriving requirements specifications from the application domain language captured by Language Extended Lexicon. In proceedings of the Workshop in Requirements Engineering (WER), Buenos Aires, Argentina, Abril (2012).

[Baeza 2002]: Baeza, R. y Rivera C.. Ubicuidad y Usabilidad en la Web. Universidad de Chile Blanco, Encalada 2120, Santiago, Chile, Diciembre (2002).

[Boehm 1997]: Boehm, B.W.: Software Engineering, Computer society Press, IEEE (1997).

[Cato 2001]: John Cato: User-centered deb design. Harllow, England. Addison-Wesley, Marzo (2001).

[Chiarle 2007]: Chiarle, L., Martínez, A., Rossi, G., Antonelli, L.: Identificación Temprana de Aspectos de Programación. Noviembre (2007).

[CMMI 2010]: CMMI para Desarrollo, Versión 1.3, Guía para la integración de procesos y la mejora de productos. Tercera edición, Junio (2010).

[Cockburn 2000]: Cockburn, A.: Writing Effective Use Cases. (2000).

[CSS Validator 2013] CSS Validator, http://jigsaw.w3.org/css-validator/validator, accedido en septiembre de 2013.

[Fenton 1991]: Fenton, E. Norman. Software Metrics A rigorous approach .Chapman & Hall, Primera Edición (1991).

[Ferré Grau 2010]: Ferré Grau, X.: Principios Básicos de Usabilidad para Ingenieros Software. Facultad de Informática. Universidad Politécnica de Madrid. Campus de Montegancedo. (2010).

[Fontela 2010]: Fontela, C.: Usabilidad. Facultad de Ingeniería. Universidad de Buenos Aires. (2010).

[Giandini 2008]: Giandini, R., Pons, C.: Una teoría dinámica orientada a objetos como fundamento formal para el proceso de desarrollo de software basado en modelos. Tesis doctoral presentada a la Facultad de Informática de la Universidad Nacional de La Plata Buenos Aires, Argentina, Febrero (2008).

[GS1 Argentina 2003]: Implementación de Trazabilidad EAN.UCC. Tesina de Licenciatura presentada a la Facultad de Informática de la Universidad Nacional de La Plata Buenos Aires, Argentina, Febrero (2003).

[Huang 2012]: Huang, J.: Software and Systems Traceability (2012).

[ITI 2010]Usabilidad en aplicaciones informáticas. Publicación de la Revista del Instituto Tecnológico de Informática. Enero (2010)

[Küng 2011]: Stefan Küng, Lübbe Onken, Simon Large: TortoiseSVN. Un cliente de Subversion para Windows Julio (2011).

[Leite 1989]: Leite, J.C.S.P., Application Languages: A Product of Requirements Analysis, Departamento de Informática, PUC-/RJ, Enero (1989).

[Leite1990]: Leite, J., Franco, A.. "O Uso de Hipertexto na Elicitaçao de Linguagens da Aplicaçao" Anais de IV Simpósio Brasilero de Engenharia de Software, SBC, Mayo (1990).

[Leonardi 1997]: Leonardi, C., Maiorana, V., Balaguer, F.: Una estrategia de Análisis Orientada a Objetos basada en escenarios. Actas de 11 Jornadas de Ingeniería del software. 1IS97. Dpto. de Informática. Universidad del País Vasco. San Sebastián. pp. 87-100. (1997).

[Leonardi 2001] Leonardi, C., Leite J.C., Rossi, G., "Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural", Tesis de maestría, http://wwwdi.inf.puc-rio.br/~julio/teses.htm, Facultad de informática, Universidad Nacional de La Plata,Argentina, Noviembre (2001).

[MantisBT 2012] Dynamic Plugin Requirements, http://www.mantisbt.org/wiki/doku.php/mantisbt:dynamic_plugin_requirements, accedido en octubre de 2012.

[MantisBT 2012] Plugin System Overview, http://www.mantisbt.org/wiki/doku.php/mantisbt:plugins_overview, accedido en diciembre de 2012.

[Nava 2003]: Víctor Manuel Nava Carbellido, Ana Rosa Jiménez Valadez: ISO 9000: 2000: Estrategias Para Implementar La Norma De Calidad Para La Mejora Continua, Limusa Noriega Editores, México, Abril (2003).

[Nielsen 1993]: Nielsen, J.: Usability Engineering, Febrero (1993).

[Nielsen 1994]: Nielsen, J.: Heuristic evaluation. Usability Inspection Methods (1994).

[Nielsen 2000]: Nielsen, J.: Usabilidad. Diseño de páginas Web. Universidad Autónoma de Ciudad Juarez. Programa de Diseño Grafico. Academia de Tecnología. (2000).

[Pressman 1998]: Pressman S. Roger. Ingeniería de Software Un enfoque Práctico. México: Mc Graw Hill, Cuarta Edición (1998).

[Sandoval 2008]: Sandoval, M., Vargas, M.: La trazabilidad en el proceso de requerimientos de software. Presentación en Universidad Nacional, Escuela de Informática, Heredia, Costa Rica, Enero (2008).

[Sandoval 2008] Master Maria Marta Sandoval Carvajal, PMP. Master Maria Adilia García Varga Universidad Nacional, Escuela de Informática. Heredia Costa Rica. La trazabilidad en el proceso de requerimientos de software. (Junio 2008).

[Screenfly 2013] Screenfly, https://quirktools.com/screenfly/, accedido en agosto de 2013.

[Screen Size Tester 2013] Screen Size Tester, http://www.anybrowser.com/ScreenSizeTest.html, accedido en septiembre de 2013.

[Sirius 2013] Sirius: Nuevo sistema para la evaluación de la usabilidad web, http://olgacarreras.blogspot.com.es/2011/07/sirius-nueva-sistema-para-la-evaluacion.html, accedido en octubre de 2013.

[SprinterWeb 2013] Evaluación de usabilidad, http://evaluausabilidad.sprinterweb.net/, accedido en octubre de 2013.

[SVN-MantisBT 2012] Subversion (SVN) Repository Integration with Mantis Bug Tracker, http://www.unitz.com/u-notez/2009/10/subversion-svn-integration-mantisbt, accedido en diciembre de 2012.

[Tabares 2007]: Tabares, M., Barrera, A., Arroyave, J., Pineda, J.: Un método para la trazabilidad de requisistos en el proceso unificado de desarrollo. Publicación en revista EIA, ISSN 1794-1237 Número 8, p. 69-82, Escuela de Ingeniería de Antioquia, Medellín, Colombia, Diciembre (2007).

[TesterSize 2013] Tester Size, http://testsize.com/, accedido en octubre de 2013.

[Tortoise 2013] Tortoise SVN, http://tortoisesvn.net/docs/release/TortoiseSVN_en/index.html, accedido en agosto de 2013.

[Vargas 2009]: Vargas, C., Biagioli, G.: Sistema para auditar el cumplimiento de CMMI-SW nivel 2. Tesina de Licenciatura presentada a la Facultad de Informática de la Universidad Nacional de La Plata Buenos Aires, Argentina, Junio (2009).

[Vergara 2008] Generación automática de métricas en proyector de software a partir de la especificación de requisitos. Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas. Dpto. de Ciencias de la Computación. Tesis para optar el grado de Megister en Ciencias Mención Computación. Santiago de Chile. Abril (2008).

[WCAG 2013] Accesibilidad Web, http://accesibilidadweb.dlsi.ua.es/?menu=pautas-1.0, accedido en marzo de 2013.

[W3C 2013] Introducción a la Accesibilidad Web, http://www.w3c.es/Traducciones/es/WAI/intro/accessibility, accedido en septiembre de 2013.

[W3C 2013] Accessibility Evaluation Resources, http://www.w3.org/WAI/eval/Overview.html, accedido en junio de 2013.

Anexo A (Herramientas necesarias)

El Anexo A presenta herramientas de uso cotidiano que abarcan distintas etapas de un proyecto de software.

En particular, se nombran las herramientas utilizadas para el presente trabajo de tesis y los enlaces de descarga de las mismas y la forma de instalación.

La integración de dichas herramientas, junto con los plugins, permite relacionar información de cada etapa del proyecto, alcanzando la trazabilidad buscada.

Herramientas necesarias:

- Herramienta de tracking.
- Servidor/Cliente de versionado.
- Lenguaje de programación (Server-side).
- Base de datos.
- Servidor HTTP.

Herramientas utilizadas en el presente trabajo de tesis:

- Herramienta de tracking: MantisBT v1.2.15
- Cliente de versionado: TortoiseSVN v1.7
- Servidor de versionado: VisualSVN v2.5.6
- Lenguaje de programación (Server-side): PHP v5.4.25
- Base de datos: MySQL v5.5
- Servidor HTTP: Apache 2.2

URLs para descargar las herramientas:

- MantisBT: http://www.mantisbt.org/download.php
- VisualSVN: http://www.visualsvn.com/server/download/ (Requiere la última versión de TortoiseSVN).
- TortoiseSVN: http://tortoisesvn.tigris.org/

  (Si se está instalando TortoiseSVN 1.8 en Windows XP SP3, se debe tener la última versión de windows installer).
- Xampp v1.8.2 (PHP+Apache+MySQL): http://www.apachefriends.org/en/xampp-windows.html

Pasos para la Instalación de MantisBT:

1. Descomprimimos el paquete Xampp.
2. Colocamos la carpeta mantisbt en la carpeta htdocs del servidor.
3. Abrimos un navegador y colocamos la siguiente url: http://localhost/mantisbt

Veremos la siguiente pantalla:

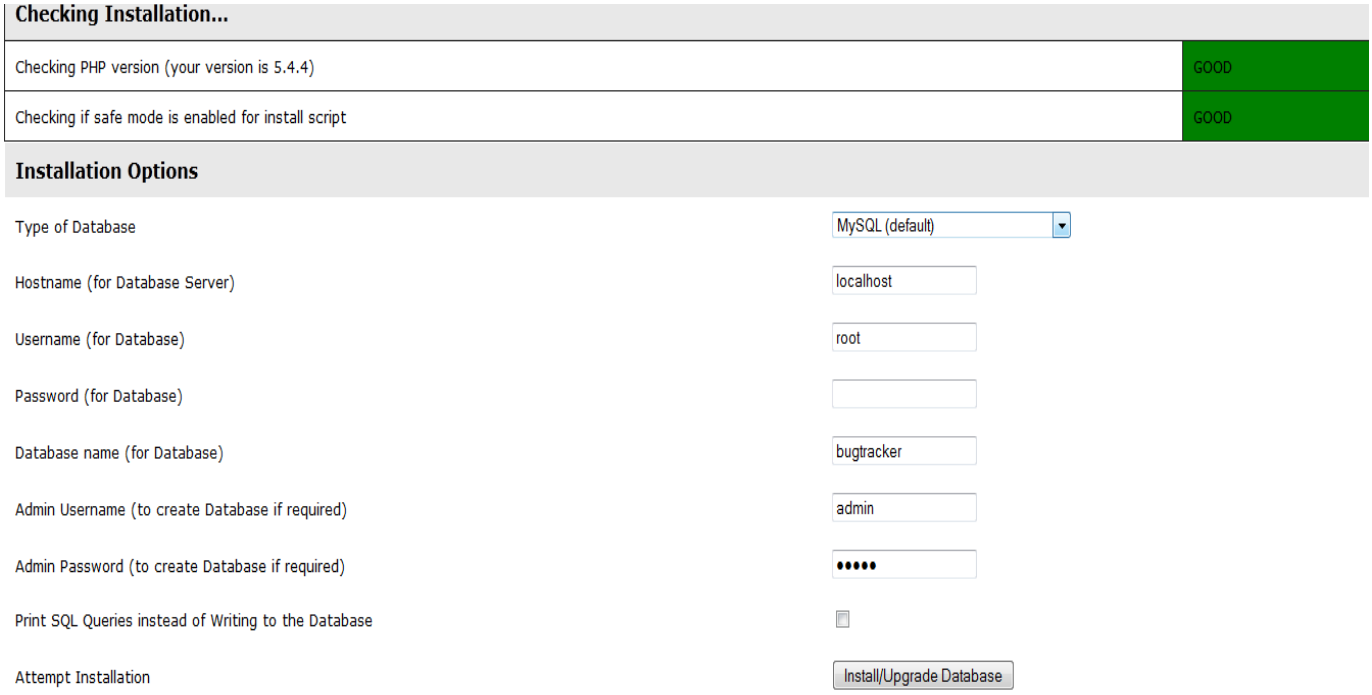


Figura 57. Pantalla de instalación de MantisBT

4. Completamos con los datos requeridos de la base de datos (usuario, contraseña).
5. Ya con MantisBT instalada, podemos acceder a la herramienta utilizando los siguientes datos:

   - usuario: administrator
   - contraseña: root

Plugins de MantisBT necesarios para la integración con Subversion:

- MantisBT Core 1.2.15
- MantisBT Formatting 1.0a
- Meta Programing 0.1
- Source Control Integration 0.16.4

- Subversion / WebSVN Integration 0.16
- Subversion Integration 0.16

Links de descarga de los plugins de MantisBT:

- https://github.com/mantisbt-plugins/source-integration/

- http://git.mantisforge.org/w/meta.git