UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Una herramienta para identificar crosscutting concerns a través de LEL
Autores: Aramayo Adriana Lorena, Rossi Joaquín Ignacio
Director: Rossi Gustavo
Codirector: Antonelli Leandro
Asesor profesional:
Carrera: Licenciatura en Informática

Resumen

Las aplicaciones de software son sistemas muy complejos que poseen un gran número de características (concerns) muy diferentes. En general estas características se organizan de manera separada en distintos módulos, pero la mayoría de las veces estas características se superponen y son transversales entre sí. La última situación es problemática porque las características se encuentran enmarañadas (tangled) y esparcidas (scattered) en diferentes módulos, lo que hace que el diseño y el código sean difíciles de implementar y de mantener. Los enfoques que identifican características transversales (croscutting concerns) en general son muy complejos y se enfrentan con mucha información, por lo tanto es importante contar con herramientas que ayuden a la implementación de los mismos. Esta tesis presenta la herramienta TICC que identifica las características transversales a través del enfoque del Léxico extendido del Lenguaje, consume un LEL previamente generado por otra herramienta, lo analiza e identifica las características transversales.

Palabras Claves

Análisis de Dominio. Característica Transversal. Código enmarañado. Código disperso. Léxico Extendido del Lenguaje. Aspecto. Usabilidad. Accesibilidad. Aplicación Web.

Conclusiones

Se presentó una herramienta que permite la identificación de características transversales a través del uso del Lenguaje LEL. Se profundizó en la automatización y en el enriquecimiento de la estrategia que identifica las características transversales.
Mediante la implementación de la herramienta, se brindo al usuario una mayor interacción con los símbolos y la posibilidad de agruparlos en diferentes estados para efectuar cálculos en condiciones distintas.

Trabajos Realizados

Para el desarrollo de la herramienta TICC primero se analizo el LEL generado en formato XML por la herramienta C&L y la estrategia que identifica las características transversales. Se implemento el parseo de los símbolos del XML y los cálculos de la estrategia en lenguaje Lua. Se integro Lua con la plataforma JAVA. Se generaron vistas para las distintas instancias de la estrategia. Se realizaron casos de estudios con la herramienta y finalmente un análisis de usabilidad y accesibilidad provocando modificaciones en TICC.

Trabajos Futuros

Dado que la herramienta TICC es una aplicación Web, la misma presenta varios puntos en los cuales se puede enriquecer. Como por ejemplo una segunda evaluación de usabilidad para refinar detalles de diseño y aumentar el nivel de accesibilidad. Incorporar la registración y autenticación de distintos usuarios. Brindar una forma colaborativa de análisis de los cálculos por distintos usuarios, a través de vistas y/o archivos de detalle.

Fecha de la presentación: Noviembre del 2013

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción

El desarrollo de sistemas de software es una tarea muy compleja por distintos motivos, entre ellos vale la pena destacar la naturaleza propia del software y las tareas de gestión relacionadas con su construcción.

El objetivo de las tareas de gestión consiste en planificar y controlar todas las actividades que realizan una o más personas para alcanzar un objetivo o tarea. Debido a esto, es que las tareas de gestión son complejas, ya que se trata con personas que tienen distintas capacidades y backgrounds. Esta diversidad hace difícil que se pueda organizar la construcción del software en un cronograma preciso que sea definido al inicio del proyecto. En general, al principio se definen los grandes objetivos, a los cuales se les asigna un tiempo de finalización estimado, pero es muy difícil cumplirlos.

Además, por la naturaleza propia del software es que su construcción es una tarea compleja. Brooks [Brooks 1995] define 4 características que hacen que sea un producto muy particular y así también su construcción:

- **Invisibilidad**: el software no es un producto visible como lo puede ser un edificio, un auto o un avión. Esta característica impone al ser humano una barrera de percepción de tamaño, forma y estructura.

- **Modificabilidad**: el software es construido a través de descripciones, las cuales son altamente maleables. Sin embargo, el hecho de que se puedan reescribir, no significa que se pueda modificar. Entendiendo por modificar a introducir cambios en una forma coherente y consistente. Esta capacidad de modificabilidad crea la ilusión de que es fácilmente cambiable.

- **Conformidad**: el software tiene que adecuarse a su infraestructura, el cual puede ser a su vez otro software, el cual incluso puede no existir aún. Por lo cual, cambios que pudieran ocurrir en su infraestructura originan cambios en el sistema.

- **Complejidad**: el software es complejo por naturaleza ya que debe abstraer la complejidad que existe en su contexto para proveer un soporte o automatizarlo. Si el software se simplificara no cumpliría con su función.

En la construcción de software se debe lidiar con las **características** (*concerns*) del problema a resolver. Con el fin de hacer frente a esta complejidad, la ingeniería de software tradicional propone la estrategia de descomposición, denominada divide y vencerás (divide and conquer). La misma consiste en dividir el problema en subproblemas independientes de forma que en conjunto resuelvan el problema original. Esta forma de resolución es utilizada en el desarrollo de software, en uno de los paradigmas más difundidos: el paradigma procedural. Esta técnica se basa en encapsular cada característica (*concern*) en un bloque de software (módulo). Sin embargo, esta estrategia falla cuando es necesario trabajar con piezas que están dispersas en otras piezas. Esto generalmente ocurre cuando se inyectan atributos de calidad, ya que la

implementación de estos atributos es común a las diferentes piezas. Dos de los ejemplos más comunes son:

I. Auditoría o logging: es cuando se debe realizar un registro de cada una de las actividades del sistema. Por lo que diferentes módulos deben incluir el mecanismo necesario para registrar dicha actividad.

II. Autorización: diferentes piezas de software pueden requerir operaciones que garanticen que la autorización adecuada está presente en la ejecución de cada pieza.

Como tales, auditoría y autorización son un ejemplo de característica transversal, esto es, una característica que atraviesa diferentes piezas que conforman al todo.

1.1 Aspectos

Con conceptos tales como procedimientos, funciones, módulos, bloques estructurados, tipos de datos abstractos, generalidad y herencia, la Ingeniería de Software logro evolucionar desde sus comienzos. Estos conceptos le proveyeron formas de abstracción para realizar una programación de alto nivel. Sin embargo, los progresos más importantes fueron obtenidos al aplicarlos junto con tres principios estrechamente relacionados entre sí: abstracción, encapsulamiento y modularidad.

Las técnicas de implementación en general tienden a implementar los requerimientos usando metodologías de una sola dimensión, forzando a que todos los requerimientos sean expresados en esa única dimensión. Esta dimensión resulta adecuada para la funcionalidad básica, pero no para requerimientos más complejos, los cuales quedan diseminados a lo largo de la dimensión dominante. Es decir, que mientras el espacio de requerimientos es de n-dimensiones, el espacio de la implementación es de una sola dimensión. Dicha diferencia produce un mapeo deficiente de los requerimientos a sus respectivas implementaciones.

Los requerimientos complejos a los que se refiere en el párrafo anterior son llamadas **características transversales** (*crosscutting concerns*); un ejemplo de ellos es el de la auditoría de la actividad de una aplicación. Este tipo de requerimiento plantea dos tipos que problemas: el código enmarañado (*tangling*) y el código disperso (*scattering*). La comunidad de Desarrollo de Software Orientado a Aspectos (*Aspected Oriented Software Develpment, AOSD*) se ocupa de las características transversales con el fin de atacar los problemas de enmarañamiento y dispersión.

El **código disperso** se encuentra en los casos en los que un requerimiento se encuentra en más de un módulo y su implementación también es replicada en más de un modulo. En cambio el **código enmarañado** ocurre cuando en un módulo se intenta abarcar más de un requerimiento. La combinación de estos síntomas afecta tanto al diseño como a la implementación de software [Dijkstra 1976].

La implementación de código enmarañado trae aparejados dos efectos negativos. El primero es una baja correspondencia entre un concepto y su implementación. El segundo es la distracción en los desarrolladores de la implementación del concepto principal. Estos efectos terminan provocando que la implementación del concepto contenga código poco reusable, de baja calidad y propenso a errores debido a que algún concepto puede ser subestimado. Además provoca que el costo de realizar alguna modificación al código tenga un costo mayor que si los conceptos estuvieran correctamente modularizados.

Como respuesta al problema del código enmarañado y código disperso fue que se introdujo el concepto de aspecto.

Una de las primeras definiciones que aparecieron del concepto de aspecto fue publicada en 1995, y se describía de la siguiente manera: "*un aspecto es una unidad que se define en términos de información parcial de otras unidades*" [Demeter 1995]. Actualmente la definición de aspecto más utilizada es la siguiente: "*Un aspecto es una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa*" [Kiczales 2001].

Separando las características que posee un dominio, obtenemos una disminución en la complejidad al momento de tratarlos y se puede cumplir con requerimientos relacionados con la calidad como adaptabilidad, mantenibilidad, extensibilidad y reusabilidad.

Este principio puede aplicarse de distintas formas. Por ejemplo, separar las fases del proceso de desarrollo puede verse como una separación de actividades en el tiempo y por su objetivo. Definir subsistemas, objetos y componentes son formas de poner en práctica el principio de separación de características.

La programación orientada a aspectos (*Aspect Oriented Programming, AOP*) es una forma de modularizar las características transversales, al igual que la programación orientada a objetos es una forma de modularizar las características tradicionales. AOP provee una forma de modularizar los aspectos en módulos, y luego, a través de un mecanismo conocido como entretejido de aspectos (*weaving aspects*), el código aspectual es inyectado en todos los módulos en donde el aspecto es necesario. AspectJ es una extensión al lenguaje de programación JAVA que agrega capacidades orientadas a aspectos al mismo.

Para entender el concepto de aspecto se considera una aplicación bancaria que posee una clase BankAccount con las siguientes operaciones:

- Extraer (Withdraw)
- Depositar (Deposit)
- Consultar el saldo (getBalance)

```
public class BankAccount {
     private int balance;
     public void deposit(int anAmount) {
          balance = this.getBalance() + anAmount;
     }
 public void withdraw(int anAmount) {
  if (this.getBalance() >= anAmount) {
   balance = this.getBalance() - anAmount;
  }
 }
 public int getBalance() {
  return balance;
 }
}
```

Figura 1.1 Clase BankAccount con su funcionalidad base

En la figura 1.1, se muestra la definición de clase Java [Java] con las tres operaciones. El ejemplo muestra como cada método muestra sola característica encapsulada.

Considere que el banco del ejemplo previo necesita cierto nivel de seguridad, por lo que cada operación debe incluir el comportamiento para proveer autorización. Además, el banco necesita dejar registro de cada operación (log). En la figura 1.2 se incorpora a la funcionalidad básica, el comportamiento necesario para proveer autorización y dejar registro de las operaciones.

La figura 1.2 además, muestra que las acciones para implementar autorizaciones y loggin se repiten en los métodos depositar, extraer y consultar balance. Sin embargo, utilizando los conceptos de la orientación a aspectos, es posible modularizar las características (concerns) de la autorización y loggin de forma que el código se simplifique y haciendo más fáciles los cambios, ya que solo debería modificar en un solo lugar.

```
public class BankAccount {
        private int balance;
        private ArrayList <Subject> owners;
        private ActivityHistory history;
        public void deposit(int anAmount) {
                this.logMessage("deposit requested");
                 if (this.isSubjectAuthorizedToOperate ()) {
                        balance = this.getBalance() + anAmount;
                }
        }
        public void withdraw(int anAmount) {
                this.logMessage("withdraw requested");
                if (this.isSubjectAuthorizedToOperate ()) {
                        if (this.getBalance() >= anAmount) {
                                balance = this.getBalance() - anAmount;
                        }
                }
        }
        public int getBalance() {
                this.logMessage("getBalance requested");
                if (this.isSubjectAuthorizedToOperate ()) {
                        return balance;
                }
         }
        private boolean isSubjectAuthorizedToOperate() {
                return owners.contains (history.activeSession.subject());
        }
        private void logMessage(String message) {
                history.log.add (message);
        }
}
```

Figura 1.2 Clase BankAccount con funcionalidad para autorizar y dejar registro de las operaciones

Para declarar un aspecto en AspectJ es necesario definir dos características: pointcut advice. Un pointcut declara un conjunto de puntos en la ejecución de un programa donde el código de los aspectos es inyectado. Estos puntos no pueden ser cualquier punto en el programa, por el contrario, hay un conjunto de puntos bien definidos que son conocidos como join points. Luego, el código inyectado es denominado advice. La figura 1.3 muestra dos aspectos: autorización y logging. Cada operación de la clase BankAccount(depositar, extraer obtenerBalance) necesita ser autorizado y loggueada, por lo cual, los pointcuts definidos en cada aspecto se refieren a cada operación de la clase BankAccount. Esto se realiza de la siguiente manera: (* BankAccount.* (...)). Esto significa que cualquiera sea la firma (signature) del método (return_type class_name.name_of_the_method (parameters)), si el método es de la clase BankAccount, el mismo debe ser considerado. Además, el pointcut de este ejemplo además hace una declaración necesaria para trabajar dentro del advice con el objeto receptor del mensaje. Esta declaración es: (BankAccount bankAccount) y && target (bankAccount).

Finalmente, es importante mencionar que el aspecto autorización previene la ejecución de la funcionalidad base (*core*) si el sujeto no es autorizado, mientras que el aspecto de logging solo agrega comportamiento pero no altera la funcionalidad base. Por lo cual, las keywords call y execution son agregadas en la declaración del pointcut. Los advices requieren menos explicación. El advice de logging es ejecuta antes de la invocación del método y consiste en registrar (log) el mensaje invocado. El advice de autorización es más complejo ya que verifica si el usuario está habilitado o no para realizar la operación, y previene la ejecución de la operación si el usuario no lo está. Mientras que si lo está el método se ejecuta normalmente. Finalmente, cada aspecto agrega una definición de variable y un método. El aspecto de autorización necesita la variable owners y el método isSubjectAuthorizedToOperate, mientras que el aspecto de logging necesita la variable history y el método logMessage.

```
public aspect Authorization {
 private ArrayList <Subject> BankAccount.owners;
 private boolean BankAccount.isSubjectAuthorizedToOperate() {
  return owners.contains (history.activeSession.subject());
 }
 pointcut accountOperationExecution (BankAccount bankAccount) :
execution (* BankAccount.* (..)) && target (bankAccount);
around (BankAccount bankAccount): accountOperationExecution
(bankAccount) {
  if (bankAccount.isSubjectAuthorizedToOperate ()) {
proceed(bankAccount);
}
}
}
public aspect Logging {
private ActivityHistory BankAccount.history;
 private void BankAccount.logMessage(String message) {
  history.log.add (message);
 }
pointcut accountOperationCall (BankAccount bankAccount) : call
(* BankAccount.* (..)) && target (bankAccount);
before (BankAccount bankAccount):accountOperationCall
(bankAccount) {
Signature sig = thisJoinPointStaticPart.getSignature();
bankAccount.logMessage(sig.getName() + " requested" );
}
}
```

Figura 1.3 Declaración de los aspectos de autorización y logging

La modularización provista por los aspectos provee a las cuentas bancarias con la incumbencia de autorización y logging. Sin embargo, la funcionalidad base (ver Figura 1.1) y los aspectos (ver Figura 1.3) se definen en forma separada. Luego utilizando los mecanismos de entretejido (weaving) de la AOP se genera el programa con los aspectos representado en la Figura 1.2.

1.2 Modelo de Dominio

El análisis del dominio introducido a principios de los 80 es una actividad que incluye el proceso de identificación, captura y organización de la información con el objetivo de generar sistemas reutilizables y adaptables. El análisis del dominio se basa en la reutilización sistemática, es decir, capturando la variabilidad de la información con el objetivo de mejorar la eficacia del desarrollo y su posterior mantenimiento [Arango 1989].

Durante el análisis del dominio se debe “limitar cuidadosamente el dominio que es considerado, las concordancias y las diferencias de los sistemas en el dominio, organizar una comprensión de las relaciones entre los varios elementos en el dominio, y representar esta comprensión de una manera útil” [Nilson 1994]. Las técnicas de análisis en el dominio se centran en aumentar la comprensión del dominio capturando la información en modelos formales [Nilson 1994]. El objetivo fundamental de esta etapa es dar un marco para que permita obtener requerimientos claros, de alta calidad y que satisfagan de la mejor manera posible las necesidades del cliente.

Al momento de realizar un análisis de dominio, nos encontramos con que existen numerosas técnicas o herramientas. Estas técnicas producen grandes y confusos procesos de documentación que llevan a que los analistas generen información poco confiable. Debemos tener en cuenta que algunas técnicas consideradas “livianas” al no tener formalismos pierden el foco del problema.

Pueden identificarse cuatro etapas en el proceso de análisis de un dominio:

1. **Definir el alcance**: el objetivo de esta etapa es definir el alcance del dominio. Definir por otro lado las relaciones entre el dominio y los objetos externos a él, como por ejemplo diversos ambientes de funcionamiento y orígenes de datos. El resultado de esta etapa deberá ser documentado en un modelo de contexto. El modelo de contexto muestra con que otros sistemas interactúa uno en particular.

2. **Análisis del dominio**: el concepto de dominio es utilizado para describir un grupo o conjunto de sistemas o áreas funcionales que muestran características similares. El análisis del dominio pretende detectar esas características de la manera más formal y ágil posible. Durante esta etapa es necesario entender cuáles son los elementos que posee el dominio y la relación entre ellos.

3. **Especificar la estructura**: en esta etapa se pretende definir qué estructura presenta el dominio, es decir, detectar y presentar las áreas definidas en el alcance, es decir, cada uno de los subdominios por los que está formado un dominio de mayor jerarquía. Por ejemplo en una empresa de medicina prepaga pueden detectarse varios subdominios, entre ellos: cobranzas, beneficiarios, facturación, etc., los cuales representan sectores o gerencias muy importantes dentro de la organización.

4. **Construir los componentes**: durante la etapa final se pretende construir los documentos que se hayan planteado como producto final de esta etapa. Durante esta etapa final se deberá entregar la documentación que respalde el trabajo realizado en las etapas anteriores, como por ejemplo diagramas de estado, diagramas de clases, etc.

La investigación en el desarrollo de software centra sus esfuerzos en obtener nuevas técnicas y metodologías para realizar la solución, pero se le da poca importancia al problema en cuestión que debe ser inferido de la solución misma. Esta aproximación orientada a la solución puede servir en un área donde los problemas son bien conocidos. Sin embargo, no es el caso del desarrollo de software en donde el análisis del problema es imprescindible ya que la computación avanza a un ritmo muy acelerado y ataca dominios muy variados [Jackson 1999] [Arango 1989].

Es aquí en donde juega un rol fundamental el Léxico Extendido del lenguaje (*LEL*). El LEL es un modelo contextual que permite capturar el lenguaje de un dominio. A través de la identificación y definición de los símbolos propios de un contexto se logra un mejor entendimiento de éste. La idea bajo el LEL es muy simple, hay que preocuparse por entender el lenguaje del problema, sin preocuparse por entender el problema [Leite 1997].

Generalmente se dice que los requerimientos fueron mal definidos o no se llegó a un entendimiento entre el analista de sistemas y el usuario. La manera de minimizar los errores durante la definición de los requerimientos es tener (por parte del analista) un amplio conocimiento del dominio en el que va a trabajar; de esta forma se minimizarán los errores durante el análisis de requerimientos debido a que tanto el analista de sistemas como los usuarios del dominio hablarán el mismo "lenguaje" y podrán entenderse con mayor facilidad.

El modelo del dominio da el marco necesario para obtener requerimientos de calidad y minimizar la probabilidad de errores durante la siguiente etapa, tal es así que consideramos de vital importancia realizar un exhaustivo análisis del dominio para luego mejorar la calidad de los requerimientos de los usuarios.

1.3 Importancia de Diseñar con Características Transversales

En los lenguajes de programación surgidos en los 90's, se encontraban códigos en los que no había separación de conceptos, datos y funcionalidad. A esta etapa se la conoce como la del código espagueti, ya que se tenía una maraña entre datos y funcionalidad. En la siguiente etapa se pasó a aplicar la llamada descomposición funcional que como se nombró en el capítulo anterior pone en práctica el principio de "divide y vencerás" identificando funciones que se definen en el dominio del problema. La principal ventaja que proporciona esta descomposición es la facilidad de integración de nuevas funciones, aunque también tiene grandes inconvenientes, como son el hecho de que las funciones quedan algunas veces poco claras debido a la utilización de datos compartidos, y el que los datos quedan esparcidos por todo el código, lo cual, normalmente, el integrar un nuevo tipo de datos implica que se tengan que modificar varias funciones.

Intentando solventar estas desventajas con respecto a los datos, se dio otro paso en el desarrollo de los sistemas de software: La programación Orientada a Objetos (*POO*). Esta supuso un gran avance en la ingeniería del software para construir sistemas complejos utilizando el principio de descomposición, ya que en el modelo de objetos subyacente se ajusta mejor a los problemas del dominio real que la descomposición funcional. Pero al igual que la descomposición funcional, las funciones siguen quedando dispersadas por todo el código y tiene el inconveniente de que generalmente para agregar nuevo funcionamiento se deben modificar muchos objetos lo que genera enmarañamiento.

Los problemas anteriormente descriptos se solucionaron utilizando AOP (*Programación Orientada a Aspectos*). AOP propone un desarrollo que se complementa con el paradigma Orientado a Objetos, por lo cual soporta una descomposición orientada a objetos, además de procedural y funcional.

La separación de requerimientos (*concerns*) en todos sus niveles (análisis, diseño, codificación y ejecución) es un requerimiento importante si se pretende el desarrollo de un software que sea reutilizable y de alta calidad.

Las construcciones provistas por los lenguajes de programación, que fueron creados para implementar los modelos generados por las técnicas de diseño existentes, reproducen las jerarquías y; por lo tanto, comparten el defecto de la dispersión de código. En el paradigma de programación imperativa, la descomposición consiste en identificar procedimientos que resuelvan parte del problema, y la jerarquía se da en el árbol de ejecución, según el cual los procedimientos se invocan unos a otros. En el caso de la programación orientada a objetos, la jerarquía generada en la etapa de diseño suele plasmarse en las relaciones de herencia o de composición entre objetos. Por ejemplo, algunos patrones de diseño de uso habitual como el observador (*Observer*), visitante (*Visitor*) y mediador (*Mediator*) exhiben estos problemas, ya

que para aplicarlos es necesario adaptar a ellos más de una clase. El problema aparece cuando una característica afecta a distintas partes del sistema que no aparecen relacionadas en la jerarquía. En ese caso, la única solución suele ser escribir código repetido que resuelva esa incumbencia para cada subsistema.

1.4 Importancia de Identificar Características Transversales en el Dominio

La programación orientada a Aspectos (*AOP*) se basa fundamentalmente en las características transversales de un sistema. Estas características pueden separarse en tipos de acuerdo a los módulos o componentes que atraviesan un sistema de software.

La motivación fundamental para trabajar con características transversales es que nos permite disminuir el tiempo, incrementar la calidad y disminuir el costo de desarrollo de una aplicación.

Para que estos resultados sean obtenidos las características transversales deben ser identificadas antes de ser implementadas, sino las características transversales generarán grandes cambios que a su vez impactarán directamente en el aumento del tiempo y costo del desarrollo.

Por lo anteriormente mencionado, una consideración lógica implica encontrar un proceso o herramienta que nos ayude en la detección de características transversales de forma temprana, lo cual provocaría una disminución en los costos de las etapas posteriores. Es de esperarse que con el correr del tiempo se vaya incrementando la cantidad de aplicaciones que utilicen este paradigma de programación. En este contexto, es imprescindible encontrar una metodología que nos permita identificar, modelar y reutilizar características transversales para ser utilizados en futuras aplicaciones.

La reutilización de los aspectos (características transversales a nivel del código) requiere primeramente la identificación y explotación a través de la especificación de las características transversales. Uno de los procesos comúnmente adoptados para poder llevar a cabo la reutilización es el análisis del dominio. El análisis del dominio puede ser definido como el proceso de identificación, captura y organización del conocimiento del problema a resolver, con el objetivo de que pueda reutilizase en la creación de nuevos sistemas. El resultado del análisis del dominio es un modelo de dominio. Numerosos métodos utilizados en el análisis del dominio se enfocan en el incremento de conocimiento sobre el dominio de manera de poder generar un modelo reutilizable.

1.5 Técnicas de Identificación de características transversales

Existen distintos trabajos que se encargan de identificar características transversales, Aquí se nombraran algunas estrategias que permiten la identificación de características (*concerns*) en etapas tempranas del desarrollo.

1.5.1 Descubriendo aspectos tempranos

Banniassad et al [Banniassad 2006] proponen una forma de organizar los requerimientos descriptos con texto coloquial, como así también presenta una serie de actividades para organizar los requerimientos descriptos tradicionalmente a la forma que ellos proponen.

Banniassad et al hacen una analogía entre los documentos de requerimientos y el código fuente, sugiriendo que los requerimientos se organicen de la siguiente forma. Establecen que por un lado se especifiquen los requerimientos base (*core*), separados de los requerimientos

transversales. Esto es análogo al código fuente donde el código base se ubica separado de los aspectos. Luego, tanto en el modelo de organización de requerimiento como en los aspectos, es necesario vincular de alguna forma ambas descripciones: base y transversal. Es por esto que Baniassad et at. determinan que es necesario especificar requerimientos que impactan (*impact requirements*), los cuales constituyen un vinculo entre los requerimientos transversales y los requerimientos base.

Además de la forma de especificar los requerimientos, Banniassad et al organizan la construcción mediante 4 actividades:

1. Identificación
2. Captura
3. Composición
4. Análisis

La identificación se refiere a analizar el conjunto de requerimientos para separar los requerimientos base de los transversales. Esta identificación puede realizarse utilizando tres técnicas:

1. Identificación de términos transversales (*aspects terms*)
2. Analizando términos que impactan (*impact requirements*)
3. Analizando características dispersas (*scattered concerns*)

Términos transversales (*Aspects Terms*) son términos o expresiones que se refieren a atributos de calidad (requerimientos no funcionales) bien conocidos y cualquier palabra o expresión que haga referencia a alguno de ellos puede ser reconocida automáticamente por estrategias de minería de datos. (*Data Mining*) que se describe más adelante.

Otra forma de separar los requerimientos base de los transversales consiste en analizar a aquellos que requerimientos que impactan (*impact requirements*). Es decir, analizar la relación entre los distintos requerimientos. Esto puede ocurrir porque los requerimientos de alguna forma pueden mencionar otro requerimiento. Este vínculo es la pista de que existe comportamiento aspectual que puede ser detectado por técnicas tales como Theme/Doc que se describe más adelante.

El último método propuesto para identificar los requerimientos es características dispersas (*scattered concerns*), el cual consiste en buscar conceptos o requerimientos que se repitan en el código fuente para identificar código aspectual. En los requerimientos también puede ocurrir ya que generalmente se obtiene y especifica la información en forma desordenada y redundante.

El segundo paso consiste en capturar los requerimientos que fueron identificados mediante la utilización de una o más técnicas descriptas anteriormente. La captura consiste en reorganizar los requerimientos, de forma tal de aislar los requerimientos transversales de los requerimientos base.

Una vez realizada la captura, se cuentan con dos grupos de requerimientos, sin embargo, los requerimientos transversales se caracterizan por influir en más de un requerimiento base, es por esto que es necesario realizar una composición. Esta composición consiste en indicar explícitamente que requerimientos transversales afectan a que requerimientos base.

Una vez realizada esa composición, se produce un modelo de requerimientos que describe los mismos requerimientos que se modelo inicialmente, solo que el mismo se encuentra

reorganizado con los 3 grupos que plantean los autores: requerimientos base, requerimientos transversales y requerimientos que impactan.

Una vez obtenido el modelo es necesario realizar una actividad de análisis sobre el mismo. Con esta reorganización es mucho más sencillo entender los requerimientos y poder realizar un análisis sobre los mismos y a la vez detectar conflictos e inconsistencias.

Esta estrategia tiene como beneficio que los autores proponen una serie de actividades para factorizar los requerimientos y proveen una categorización o guía sobre como identificar la información repetida, sin embargo, no proveen ningún método específico para llevarlo a cabo.

1.5.2 Theme/Doc

La estrategia Theme/Doc [Baniassad 2004] provee vistas de las especificaciones textuales, exponiendo las relaciones entre las distintas piezas del sistema con comportamiento. Estas vistas asisten al desarrollador a determinar cuáles elementos de funcionalidad son transversales y cuales son base.

Esta estrategia está basada en la noción de theme, el cual representa un rasgo (feature) del sistema. Existen dos tipos de themes: **themes bases**, los cuales pueden compartir algunas estructuras y comportamiento con otros themes base y **themes transversales**, los cuales poseen comportamiento que se superpone con funcionalidad de los themes base.

Theme/Doc opera con la premisa básica de que si dos comportamientos se describen en el mismo requerimiento, los mismos están relacionados. La estrategia consta de 4 etapas:

1. Identificar acciones y entidades
2. Categorizar las acciones en themes
3. Identificar themes transversales
4. Analizar themes de forma individual

Identificar acciones y entidades consiste en procesar las expresiones que describen a los requerimientos del sistema. A partir de ciertas claves provistas por el desarrollador, la herramienta identifica acciones claves (*key-actions*) como así también entidades claves (*key-entities*).

El siguiente paso consiste en analizar e identificar las features que tienen la relevancia necesaria para ser modeladas como themes. Con este análisis se construye la vista de acciones (*action-view*). La misma es un diagrama compuesto por dos elementos: acciones (que se modelan con rombos) y requerimientos (que se modelan con rectángulos de puntas redondeadas). Si una acción es mencionada en un requerimiento, se debe trazar una línea entre la acción (rombo) y los requerimientos (rectángulo). Con esta vista, se debe analizar y determinar cuáles acciones deben ser themes o simplemente comportamiento dentro de los themes. El definir qué acciones son lo suficientemente importantes para ser consideradas themes es un proceso altamente intuitivo.

La identificación de themes transversales se realiza a partir de la vista de acciones principal (*mayor action view*), el cual es un diagrama similar a la vista de acciones, con la diferencia que este ilustra a los requerimientos más relevantes, es decir, aquellos que son compartidos por más de un theme. Si un requerimiento es compartido por más de un theme, es una pauta de que pudimos haber identificado una característica transversal en nuestro sistema. Eso se debe a que dos themes no pueden operar sin que cada uno de ellos se apoye en el comportamiento del otro. Sin embargo, antes de determinar que se encontró una característica transversal hay que asegurarse de que no encontramos un requerimiento vagamente descripto que esta

enmascarando varios requerimientos. Por lo tanto, es necesario verificar si el requerimiento no puede ser reescrito en varios requerimientos distintos los cuales referencian solamente a un theme cada uno. Si no es posible una reescritura de requerimientos que provea una relación 1 a 1 entre requerimiento y themes, definitivamente encontramos una característica transversal.

La última etapa consiste en analizar los themes en forma individual. En este punto los themes son acciones relevantes las cuales se deben analizar en relación a los requerimientos asociados con las acciones principales, como así también con las acciones menores. Las entidades clave también se muestran como cuadrados. Esta vista se utiliza para verificar que las relaciones que permitieron la identificación se realizaron correctamente.

A diferencia de la estrategia descripta anteriormente Baniassad describe un método concreto y especifico para identificar las características transversales, sin embargo, depende de la subjetividad del profesional que analiza las acciones para determinar que acciones merecen ser consideradas como Themes.

1.5.3 EA-Miner

Sampaio et al. [Sampaio 2005] [Sampaio 2005b] proponen una herramienta que provee soporte automatizado para identificar y separar características transversales de los no transversales, como así también provee soporte para la identificación de las relaciones que los requerimientos proveen de ellos. Específicamente, los autores hablan de identificar características base, aspectos tempranos (*early aspects*) y relaciones transversales entre las características base y los aspectos temprano.

La herramienta es configurable, por lo cual, las características base pueden tomar distintas formas en función de la técnica que se utilice. Por ejemplo, si el análisis de descomposición es a través de un modelo orientado a puntos de vista (*viewpoints*), una característica base es un viewpoint. Mientras que si la técnica usada está basada en UML, una característica base podría ser un caso de uso (*use case*). Por su parte, los aspectos tempranos son las entidades que semánticamente atraviesan a los concerns base.

EA-Miner utiliza técnicas de procesamiento del lenguaje natural como por ejemplo: parte del discurso (*part-of-speech, POS*) y marcación semántica (*semantic tagging*), concordancia y frecuencia de palabras (*Word frequencies and concordances*), etc. Todo esto lo hace a través de la herramienta WMatrix.

Los autores sostienen que en la etapa de requerimientos existen grandes cantidades de información, por lo que la mejor forma de analizarla es a través de herramientas que identifiquen características transversales. Sin embargo, los autores aclaran que las características identificadas deben ser validadas por algún experto con criterio. Por otro lado, los autores también resaltan el hecho de que esta técnica no utiliza requerimientos sin ningún tipo de estructura.

La estrategia de detección consta de las siguientes etapas.

1. Identificación, EA-Miner parsea archivos de diferentes formatos y estructuras
2. WMatrix realiza el análisis de parte del discurso y marcación semántica a partir de los datos generados en el punto 1. WMatrix retorna el archivo procesado en el cual incluye anotaciones a partir de los análisis realizados.
3. EA-Miner a partir de la información incluida por WMatrix produce un modelo basado en puntos de vista (*viewpoints*), casos de uso, etc.
4. Se debe refinar el modelo obtenido en el punto 3 mediante la actividad de descarte (*screening out*) y generación de especificación, en donde el modelo producido es refinado por parte del usuario a través de la eliminación de entidades irrelevantes

El punto más sensible de esta estrategia es el pre procesamiento que realiza WMatrix el cual consiste en realizar el análisis parte del discurso en donde se identifican los sustantivos y verbos del texto. Luego, WMatrix también realiza un análisis semántico el cual consiste en relacionar palabras y expresiones de varias palabras con conceptos.

El análisis que EA-Miner realiza esta guiado por el pre procesamiento que realiza WMatrix. Para el caso del modelado en puntos de vista (*viewpoints*) por ejemplo, EA-Miner busca sustantivos y reúne a todos los requerimientos en los que está presente el sustantivo para determinar que es una característica base. La herramienta adolece del inconveniente de no detectar que la misma palabra puede tener pequeñas variaciones (ejemplo de singular a plural). Luego, la herramienta se ayuda de un catálogo de requerimientos no funcionales para detectar que palabras de los documentos son aspectos tempranos. Luego, las características transversales las determina, al identificar las relaciones entre las características base y los aspectos tempranos. Esto lo determina por simple intersección entre ambas entidades.

1.5.4 Análisis semántico para la identificación de características transversales

Rago et al. [Rago 2009] indican que la mayoría de los análisis de identificación de características transversales se basan en las búsquedas de verbos para identificar comportamiento transversal. Sin embargo, ellos consideran que sería más adecuado y preciso identificar sobre que objetos actúan estas acciones o verbos, para aumentar la efectividad del enfoque. Esto se debe a que una misma acción aplicada a distintos objetos, puede llegar a considerarse como diferente comportamientos.

De todas formas, es necesario considerar los verbos solos (sin el objeto directo) debido a que en el texto natural el verbo no siempre tiene el sujeto al que afecta. De no tener en cuenta los verbos solo se perdería información. Por otro lado, la estrategia propuesta por los autores considera el uso de UML como base de la aplicación para identificar características transversales.

La base de la estrategia consiste en los siguientes tres pasos:

1. Realizar un análisis léxico, sintáctico y semántico del texto, así como también un estudio estadístico y de ubicación de los textos. Estos análisis consisten básicamente en desambiguar la semántica de las palabras. Estos deben realizarse primero sobre los textos de casos de uso básicos y alternativos, luego sobre los requerimientos no funcionales del sistema. Y finalmente sobre los requerimientos adicionales sobre cada una de las especificaciones.

2. Con la información previa se construye un grafo que muestre la utilización de los nodos, los cuales pueden ser verbos u objetos directos. Esta utilización no se realiza directamente de nodos a nodos, sino que se agrega un nodo intermedio que agrupa pares de verbos semánticamente relacionados y de sustantivos semánticamente relacionados, y la relación se realiza a través de ellos.

3. El último paso consiste en recorrer el grafo construido anteriormente ordenando los nodos que son característica transversal según su relevancia. La identificación de las características candidatas se realiza recorriendo el grafo y contando aquellos grupos de verbos que cortan transversalmente una cantidad de casos de uso que superan un umbral predeterminado. Luego, para cada grupo de verbos considerado transversal se realiza la búsqueda del término (verbo) más representativo del grupo para proponerlo como nombre candidato. Finalmente se debe se deben ordenar las características transversales teniendo en cuenta varios parámetros: el número de casos de uso cortados transversalmente, el numero de cortes transversales totales del grupo de

verbos, la relevancia de ocurrencia del grupo de verbos, la relevancia de ocurrencia de los grupos de objetos directos relacionados con el grupo de verbos en cuestión y por ultimo si el grupo de verbos es considerado funcional o no funcional.

1.5.5 Modularización y composición de características transversales

Rashid et al [Rashid 2003] proponen un enfoque para modularizar funcionalidad base, las características transversales y la forma en la que estos se relacionan. Para realizar esta modularización se debe comenzar identificando y especificando las características (*concerns*) y los requerimientos de los interesados (*stakeholders*). Es muy útil el relacionar las características (*concerns*) a los requerimientos a través de una matriz. Luego, observando la matriz es posible identificar rápidamente que característica atraviesa transversalmente a los requerimientos de los interesados (*stakeholders*).

Una vez que la relación de grano grueso entre las características (*concerns*) y los requerimientos de los interesados (*stakeholders*) se estableció, y de que las características transversales candidatas se identificaron, el siguiente paso es definir las reglas de composición entre las características y los requerimientos para definir entre ellos en un grano más fino. Esto se debe a que las reglas operan con una granularidad de requerimientos individuales y no solo a nivel de los módulos que las agrupa, las reglas de composición implican no solamente el vincular las características transversales y los requerimientos, sino que requieren de acciones y operadores, los cuales se utilizan para describir restricciones (*constraints*) y logros (*outcome*). Tanto las acciones como los operadores son informales, sin embargo, deben poseer un significado y una semántica bien clara. Algunos ejemplos de acciones relacionadas con las restricciones son: imponer (*enforce*), asegurar (*ensure*), proveer (*provide*), aplicar (*apply*), etc. Luego, algunos ejemplos de operadores de restricción son: durante (*during*), entre (*between*), para (for*),* etc. Y finalmente, acciones de logro (*outcome action*) son: satisfecho (*satisfied*) y realizado (*fulfilled*).

Puede suceder que ocurran conflictos entre diferentes característica transversales porque restringen al mismo requerimiento. Es por ello que en este punto es necesario realizar un balance entre las características transversales. Se debe realizar una identificación y resolución de conflictos. Cada característica transversal puede contribuir negativa o positivamente a los otros. Incluso, puede suceder que la contribución sea del tipo “no interesa”, esto es, cuando no se especifica. Para las características transversales que contribuyen negativamente es necesario especificar un valor real entre 0 y 1 y representa la prioridad de una característica transversal en relación al conjunto de requerimientos del interesado (*stakeholder*).

1.5.6 Clasificación comparativa de estrategias de minería de aspectos

Bounour et al. [Bounour 2006] presentan una clasificación de estrategia de minería de aspectos. Algunas de estas técnicas se basan en el análisis estático del código fuente, mientras otras se basan en el flujo de ejecución. Una de las categorías identificadas por Bounour se llama detección de clones (*clone detection*). En esta categoría se incluyen estrategias que realizan un análisis léxico de la estructura de la información. Dentro de este grupo existe una estrategia que consiste en construir un grafo de dependencias del programa (*program dependence graph, PSG*) que contiene información de naturaleza semántica como el control y el flujo de datos de un programa Los links entre los nodos en el PDG representan el flujo de ejecución de las acciones.

El autor también menciona una técnica que combina un árbol de sintaxis abstracta (*abstract syntax tree, AST*) con representación del token del código fuente. Esta técnica usa parsers para obtener una representación sintáctica del código fuente. Luego, el algoritmo de detección de

clones busca sub arboles similares en el árbol. Esta técnica presenta el inconveniente de que solo logra identificar características homogéneas, dado que la estrategia se basa en detectar patrones similares en el árbol.

Bounour et al. También menciona una estrategia llamada análisis de conceptos formales (formal concept análisis, FCA). La cual se basa en encontrar agrupamientos significativos de los elementos que poseen propiedades comunes. Para este fin es necesario ejecutar varios casos de usos y dos restricciones deben ser analizadas para identificar a los conceptos candidatos. La primera restricción establece que los atributos de los conceptos deben pertenecer a más de una clase. Mientras que la segunda establece que los diferentes métodos de una misma clase deben estar contenidos en más de un caso de uso.

La última estrategia descrita por Bounour et al. se llama análisis de fundido (fade in análisis) y está basada en dispersión (*scattering*). Esta estrategia consiste en hacer minería del código fuente para identificar síntomas de dispersión de código. Esta técnica define como característica (*concern*) a un método que tiene múltiples llamados distribuidos.

1.6 Conclusión

En la sección anterior se describieron las técnicas principales para identificar características transversales en etapas tempranas del desarrollo de software. Muchas de estas estrategias se basan en requerimientos o casos de uso aunque existen otras que se ocupan de analizar productos previos a ellos. Cada una de ellas presenta distintos puntos fuertes y débiles:

- Respecto del trabajo de Banniasad [Baniassad 2006] los autores proponen una serie de actividades para factorizar los requerimientos y proveen una guía o categorización sobre como identificar esta información repetida, sin embargo, no provee ningún método concreto y especifico.

- Respecto del trabajo de Baniassad [Baniassad 2004] provee un método concreto y especifico para la identificación de características transversales, sin embargo, la estrategia depende de la subjetividad del profesional que analiza las acciones para determinar que acciones merecen ser consideradas como themes.

- Respecto del trabajo de Sampaio [Sampaio 2005] la estrategia requiere de mucha intervención humana, puesto que si bien propone una herramienta que automatiza la detección, hay mucha intervención humana para analizar los resultados parciales y que la herramienta continúe el proceso.

- Respecto del trabajo de Rago [Rago 2009] se basa en casos de uso ya escritos por lo cual si bien es un momento temprano en el desarrollo, al llegar a los casos de uso, ya se dedico bastante esfuerzo para construirlos.

- Rashid [Rashid 2003] esta técnica agrupa los requerimientos en puntos de vista, sin embargo la técnica es subjetiva dependiendo en gran medida del profesional que la aplica.

Todas estas estrategias tienen en común que dependen del momento en que son aplicadas y de la subjetividad del profesional que la aplica. Estas desventajas se ven atenuadas en la estrategia de detección temprana de características transversales utilizando la estrategia de LEL [Antonelli 2012]. Esta estrategia tiene una forma concreta y especifica de llevarse a cabo, pudiendo ser ejecutada en etapas anteriores a la etapa de requerimientos y volverse a ejecutar en cualquier etapa posterior.

La estrategia propone una serie de cálculos que de realizarse en forma manual resultan engorrosos y propensos a equivocación. A diferencia de los algoritmos tradicionales identifica características transversales basándose en estados y puede llevarse a cabo en una etapa temprana del desarrollo de un software. El lenguaje que utiliza para representar el conocimiento de la aplicación (LEL) posee buena expresividad, pero por sobre todo, utiliza el lenguaje convencional sin utilizar ningún tipo de formalismo, lo cual redunda en beneficios para ser utilizado por todas las personas que participan del desarrollo de software.

1.7 Objetivo

Se propone desarrollar una herramienta que implementa la estrategia de detección temprana de características transversales utilizando la estrategia de LEL [Antonelli 2011]. La herramienta, denominada TICC, implementa los cálculos necesarios para identificar las características transversales mientras que se apoya en una herramienta llamada C&L. C&L permite la creación y edición del LEL. Es decir, TICC recibe el LEL construido a través de C&L [Felicissimo 2004] [Almentero 2009] [C&L 2009] y una vez importado permite definir los estados, realizar los cálculos establecidos por la estrategia de identificación de características transversales de LEL. Finalmente permite visualizar los resultados para eventualmente hacer cambios en las definiciones de estados y volver a realizar los cálculos.

Debido a que el objetivo de la herramienta es que pueda ser utilizado por personas con distintos perfiles, se busco que cumpla con características de usabilidad y accesibilidad nivel I definidos por la W3C [W3C 1996].

1.8 Aportes

La realización de esta tesis proveerá una herramienta para la detección de características transversales implementando la estrategia de detección temprana usando LEL [Antonelli 2011]. Esta herramienta permitirá reducir las posibilidades de cometer errores al realizar los cálculos de forma manual. Esta automatización también permitirá la posibilidad de mejorar la estrategia de derivación de los datos resultantes, como así también la posibilidad de realizar un análisis más completo de los resultados.

Capítulo 2

2. Conocimientos Básicos

En este capítulo se presentan los conceptos básicos para poder explicar la herramienta de detección temprana de características transversales desarrollada en esta tesis.

El capítulo se encuentra divido en tres secciones. La primera presenta la definición de Léxico Extendido por el Lenguaje, el cual sirve como entrada de la estrategia para la identificación de características transversales implementado por nuestra herramienta. La segunda sección describe el algoritmo de detección de características transversales. Finalmente, la última sección describe la herramienta C&L la cual se debe utilizar para generar el Léxicos Extendidos por el Lenguaje.

2.1 Léxico Extendido por el Lenguaje

El léxico extendido por el lenguaje (LEL) es un modelo contextual que permite capturar el lenguaje de un dominio. A través de la identificación y definición de los símbolos propios de un contexto se logra un mejor entendimiento de este. La idea utilizada por LEL es muy simple, hay que preocuparse por entender el lenguaje del problema, sin preocuparse por entender el problema [Leite 1997].

2.1.1 Generalidades

El LEL es similar a un glosario, debido a que el objetivo principal de ambos consiste en registrar signos (palabras o frases) que son particulares a un dominio. A diferencia del diccionario tradicional en donde cada término tiene un solo tipo de definición, en el LEL cada signo es descripto de dos formas: a través de la noción (*notions*) y de los impactos (*behavioral responses*).

La noción es la descripción usual que podemos encontrar en un diccionario, es decir, encontramos el significado del símbolo buscado en lenguaje natural. Es la descripción del signo por medio de sus propiedades intrínsecas. Mientras que los impactos definen la forma en que este se relaciona con los demás. Los impactos exponen la forma en que el signo descripto se relaciona con los demás, o los demás con el signo.

La construcción de un LEL se encuentra regida por dos principios. El principio de circularidad y el principio de vocabulario mínimo. El principio de circularidad establece que en la descripción de la noción e impactos se debe maximizar el uso de signos definidos en el LEL. El principio de vocabulario mínimo, complementa el principio de circularidad, y establece que cuando se utilizan signos externos al LEL deben tener una representación matemática clara (por ejemplo: pertenencia, intersección, etc.).

Ambos principios determinan que el conjunto de signos sea auto contenido. Es decir que un signo este expresado en términos de otros. De esta forma los signos están interrelacionados. Si cada signo se ve como un nodo de información, junto con la relación entre ellos se puede

percibir un grafo. Ahora como cada nodo está conformado por texto, el grafo no es otra cosa que un hipertexto [Leite 1997].

Además de la descripción general que se realiza de los símbolos a través de sus propiedades noción y comportamiento, los símbolos deben categorizarse con el fin de especializar la descripción de los atributos. [Breitman 2003] Breitman define que con tres ontologías básicas se puede describir al mundo. Sus ontologías eran entidades, actividades y afirmaciones. Y con estos tres elementos se puede describir al mundo ya que permiten modelar las cosas, las actividades con las que interactúan las cosas y finalmente las afirmaciones o verdades que las cosas o actividades poseen o persiguen.

En el LEL se definen cuatro categorías. Estas categorías son extensiones de las definidas por Breitman. Ellas son: estado, sujeto, objeto y verbo.

Comparando las categorías del LEL con las definidas por Breitman, se establecen relaciones ya que sujeto y objeto son especializaciones de entidades. Las actividades son equivalentes a los verbos y finalmente las afirmaciones se corresponden con los estados. Los sujetos se corresponden con elementos activos dentro del contexto de la aplicación, mientras que los objetos representan los elementos pasivos. Por otro lado, los verbos son las acciones que realizan los sujetos utilizando los objetos. Finalmente, los estados representan las situaciones en las que se pueden encontrar los sujetos o los objetos previo y luego de realizar las acciones.

En la tabla 2.1, se presentan las cuatro categorías de símbolos y como debe realizar la descripción de los mismos según sus atributos.

Tabla 2.1 Categorías de símbolos y su descripción

Categoría	Características	Noción	Comportamiento
Sujeto	Elementos activos que realizan acciones	Características o condiciones que los sujetos satisfacen	Acciones que el sujeto realiza
Objeto	Elemento pasivo con los cuales se realizan acciones que puede pasar por distintos estados	Características o atributos que los objetos poseen	Acciones que son realizadas con los objetos
Verbo	Acción que realiza un sujeto, servicio que brinda un objeto desencadenante para pasar de un estado a otro	Objetivo que el verbo persigue.	Pasos necesarios para completar la acción
Estado	Situación en la que se encuentra un sujeto o estado	Situación que representa el estado	Acciones necesarias para pasar a un estado previo o subsecuente.

El LEL es una herramienta adecuada y efectiva para modelizar el lenguaje de una aplicación. La utilidad del LEL ha sido observada en los trabajos de Breitman [Breitman 2003] y Gruber [Gruber 1993]. Breitman describe un proceso para estudiar el lenguaje del contexto de la aplicación y a partir del mismo construye una ontología. Gruber por su parte define ontología como “una especificación explícita y formal de una conceptualización compartida”. Si esta conceptualización compartida mencionada por Gruber es una conceptualización compartida del mundo real, lo será sobre la aplicación con la cual se está trabajando y por lo tanto puede ser considerada como modelo de la misma. Por lo tanto, con el LEL se captura el lenguaje del

contexto de la aplicación y a partir de él se construye una ontología la cual puede ser considerada como modelo de la aplicación. Y esto se indica explícitamente en [Breitman 2003] ya que se menciona que "la filosofía subyacente al LEL cae en la categoría de contextualización, de acuerdo a la cual, las particularidades de un contexto de uso de aplicación deben ser comprendidas en detalle antes de que se puedan producir los requerimientos".

El LEL puede ser usado tanto por personas que no tienen las habilidades para definir un dominio como por los expertos. Lo que hace al LEL conveniente para casi todo tipo de persona proviene de tres características significativas: es fácil de aprender, es fácil de usar y posee buena expresividad. Hay experiencias en dominios complejos que validan la conveniencia del LEL. Gil et al [Gil 2000] indican que: "la experiencia de construir un LEL de una aplicación completamente desconocida para los ingenieros de requerimientos y con un lenguaje altamente complejo, puede ser considerada exitosa, desde el momento en que los usuarios fueron los que notaron que los ingenieros de requerimientos habían desarrollado un gran conocimiento sobre la aplicación".

2.1.2 Proceso de construcción de un LEL

La construcción de un 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 en requerimientos. Si algún nuevo símbolo debe ser definido, se repite el proceso [Leonardi 2001][Antonelli 1999].

A continuación, se describen los pasos del proceso:

- **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, auto contenido 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 mucha información. Una característica propia de la expresión oral de los seres humanos es que en forma inconsciente utilizan palabras propias del dominio, es decir aplican el principio de circularidad. En cambio, en la descripción escrita para una narrativa más entretenida se introduce sinónimos, lo que puede dificultar el proceso de identificación de símbolos. Es aconsejable utilizar ambas fuentes de información.

Las personas que son aconsejables entrevistar son:

- Las mas mencionadas dentro del entorno de sistema.
- Las que toman 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 la 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, debido a que las entrevistas libres conducen a un gran volumen de información lo que se hace muy difícil de manejar.

En esta lista inicial se deben tomar aquellos símbolos o frases que se utilizan con mucha frecuencia, aquellas a las que se les hace énfasis (aparecen en negrita, cursiva o subrayado) o aquellas que parecen estar fuera de contexto. El motivo de esta elección de símbolos o frases radica en el objetivo del LEL, el cual consiste en capturar el lenguaje de un dominio. Los términos más utilizados son significativos del dominio, por lo que deben estar definidos. En cambio los términos que parezcan estar fuera de contexto, es debido 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 puede tener cierto significado, pero si se las combina en una frase puede comenzar en esta etapa.

- **Clasificación de la lista de símbolos**

Antes de realizar una descripción de los símbolos, se debe llevar a cabo una clasificación. Esta clasificación consiste simplemente en determinar para cada símbolo si es sujeto, objeto, verbo o estado. La misma tiene como fin el de poder realizar una administración del conjunto de símbolos. Además, dependiendo la categoría a la que pertenezca cada símbolo es la forma en la que debe ser definido. Vale aclarar que esta categorización no es definitiva, sino que es una categorización preliminar y luego en la descripción puede surgir que la categoría determinada no era la correcta

- **Descripción de los símbolos**

Los símbolos se describen a partir del conocimiento obtenido de la lectura de la documentación y de las entrevistas. Para cada símbolo se deben escribir la noción y los impactos teniendo en cuenta la clasificación establecida en la etapa anterior. Por otra parte, suele ocurrir que la descripción de los símbolos no surja del conocimiento que el analista recopilo, sino que necesita recurrir a las fuentes para completar la descripción. Esto podría llevar a que aparezcan nuevos símbolos que necesiten ser definidos.

Algunas pautas generales para la descripción de los símbolos, independientemente de la clasificación inicial que luego será validada y controlada:

- 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 impactos de un símbolo pueden representar diferentes puntos de vista o pueden ser complementos.

- **Identificación de sinónimos a partir de las descripciones**

Esta etapa de creación de un LEL también podría ser descripta como un control sobre el LEL. Esto se debe a que la etapa consiste en realizar una revisión del LEL con el objetivo verificar la existencia de sinónimos. Sin embargo, este no es el único objetivo de esta tarea ya que también se pueden encontrar símbolos con un concepto muy amplio que deba refactorizarse en un concepto genérico con varias especializaciones. También podría suceder que se utilicen términos para describir otros, y estos términos que se utilizan merezcan ser definidos. Además, podría surgir la necesidad de especializar la categorización básica de símbolos.

- **Validación con los interesados (*stakeholders*)**

La última etapa en la construcción de un LEL es la más costosa y sensible, y en cierta forma es utópica, dada que la necesidad de validarlo con las mismas fuentes que permitieron su construcción y que no siempre están disponibles. Sin embargo, siempre es posible solicitar a algún interesado (*stakeholder*) que realice alguna revisión exploratoria con el fin de validar el producto obtenido.

A continuación, se ejemplificaran los pasos descriptos en la construcción de un LEL utilizando y extendiendo el dominio de un banco con el objetivo de esclarecer los pasos definidos anteriormente.

Este dominio consiste en que un cliente tiene una cuenta en un banco, en la cual podía realizar las tres operaciones básicas que son depositar, extraer y consultar el saldo. A su vez, el banco antes de realizar cada una de estas operaciones verifica la identidad del cliente y deja un registro para futuras auditorias.

Finalmente, el banco al momento de abrir una cuenta exige una serie de pasos que deben realizarse para que la cuenta sea operativa. Estos son:

1. El cliente debe realizar una solicitud de apertura de una cuenta. En esta instancia la cuenta no está creada, pero si en un estado de **solicitada**
2. El banco debe procesar la solicitud y efectivamente crear la cuenta, dejándola en un estado **bloqueada**. Esto es la cuenta esta creada pero no activa.
3. El banco verifica que el cliente no tenga deudas y sea solvente.
4. El banco activa la cuenta.

En la tabla 2.2 se muestra el resultado de las dos primeras etapas descriptas en la creación de un LEL que son Identificación de símbolos y clasificación de símbolos.

La tercera etapa en la creación de un LEL es la descripción de los símbolos, como se indico anteriormente en esta descripción se debe completar para cada símbolo cual es su noción y cuáles son sus impactos de acuerdo a la clasificación que se le haya dado al símbolo.

Las descripciones de los símbolos poseen palabras subrayadas, estas palabras son expresiones que también están definidas en el LEL. Simbolizan una especie de link para mostrar que puede ser explorada la definición de otras palabras. Esta definición cíclica esta alentada por el principio de circularidad. A continuación se exponen cuatro ejemplos, los cuales pertenecen a cada una de las clasificaciones de símbolos.

Tabla 2.2 Símbolos del dominio bancario agrupados por categoría

Categoría	Símbolos
Sujetos	Banco Cliente
Objetos	Cuenta Log
Verbo	Abrir una cuenta Activar una cuenta Registrar Corroborar Operar Depositar Extraer Consultar saldo
Estados	Solicitada Bloqueada Activada

El primer símbolo es **cliente** el cual clasificamos como sujeto, por lo que su noción describe quién es el cliente, mientras que los impactos describen las acciones que el cliente puede realizar con su cuenta.

Sujeto: cliente

Noción: Persona que abre una cuenta en el banco.

Impactos:
El cliente puede depositar dinero en su cuenta.
El cliente puede extraer dinero de su cuenta.
El cliente puede consultar el saldo de su cuenta.

Figura 2.1 Descripción de un símbolo de clase Sujeto

El segundo símbolo es **depositar** (ver figura 2.2), que al ser un verbo en su noción describe el objetivo de la acción, mientras que en los impactos describen los pasos necesarios para realizar la acción.

Verbo: depositar

Noción: acción de agregar dinero a una cuenta.

Impactos:
El banco debe corroborar que la persona quien opera es el propietario de la cuenta.
El banco agrega el dinero a la cuenta.
El banco registra la operación en un log.

Figura 2.2 Descripción de un símbolo de clase Verbo

El tercer símbolo es **cuenta** (ver figura 2.3), que fue clasificado como objeto, por lo que su noción describe los atributos que posee la cuenta, y sus impactos describen las operaciones que el cliente puede realizar con su cuenta.

Finalmente se describe el estado **solicitada** (ver Figura 2.4), que por su clasificación como noción describe el concepto del estado, mientras que su impacto describe como el estado se transforma en estado bloqueada.

Objeto: cuenta

Noción: La cuenta posee un saldo y el cliente puede operarla.

Impactos:
El cliente puede depositar dinero en su cuenta.
El cliente puede extraer dinero de su cuenta.
El cliente puede consultar el saldo de su cuenta.

Figura 2.3 Descripción de un símbolo de clase Objeto

Estado: Solicitada

Noción: Situación en la cual el cliente ha decidido abrir una cuenta y el banco está lista para abrirla.

Impactos:
El banco abre la cuenta y el cliente es el propietario de una cuenta bloqueada.

Figura 2.4 Descripción de un símbolo de clase Estado

Por la simplicidad del caso descripto, no existen sinónimos, como tampoco existe la necesidad de realizar una especialización de las categorías (etapas 4 y 5 de la construcción de un LEL). El ejemplo completo se verá en el capítulo 4.

Como se mencionó anteriormente, el trabajo desarrollado en esta tesis propone la identificación de características transversales utilizando LEL, pero no incluye la creación de LELs. Para esto es que se utiliza una herramienta llamada C&L la cual se describe más adelante.

2.2 Identificación de características transversales usando LEL

Considerando al LEL como un modelo adecuado para capturar el conocimiento relacionado con una aplicación, se hace uso del mismo para identificar características transversales en el contexto de la aplicación. La esencia de la estrategia se basa en identificar como características transversales los símbolos que con mayor frecuencia son referenciados desde los impactos de otros símbolos [Antonelli 2010]. La estrategia de identificación de características transversales presenta las siguientes características:

- El análisis solo debe tener en cuenta los impactos del LEL, ya que estos evocan la invocación de procedimientos del código fuente. Los impactos de los verbos en particular se corresponden con un desglose de los pasos o etapas necesarias para lograr el cometido que el verbo determina, lo cual es una analogía directa con la invocación a sub módulos. En el caso de los sujetos y objetos consideramos que los mismos son objetos de un ambiente orientado a objetos, por lo que los impactos al hacer referencia a las acciones que realizan o que les realizan también es comportamiento, es decir, métodos que implementan o implementa otro objeto pero que refiere a acciones relacionadas. Finalmente los estados describen en los impactos las transiciones, que también son las acciones que permiten pasar de un estado a otro.

Por lo cual, para todas las categorías, hay una vinculación con la ejecución de comportamiento en los impactos, mientras que en la noción no sucede.

- El análisis se realiza agrupando símbolos y no a nivel de símbolo particular. En el LEL no hay bloques de construcción como puede ser en los requerimientos, diseño o código. En el LEL la única unidad es el símbolo por lo cual definimos crear grupos de símbolos (*clusters*) a partir de los símbolos estados. Decidimos utilizar los estados para agrupar símbolos, porque los símbolos estados modelan en cierta forma los nodos de una maquina de estado y Mahoney [Mahoney 2005] [Mahoney 2007] clama que el comportamiento core y transversal es a menudo modelado usando escenarios o maquinas de estados. La razón radica en que las aplicaciones son sistemas complejos reactivos con dos niveles: las maquinas de estados describen el comportamiento intra-objetos, mientras que los escenarios describen el comportamiento inter-objetos.

A continuación se describen los símbolos de categoría verbos con su definición completa (noción e impactos). Es importante recordar que la estrategia sólo hace uso de los impactos.

Verbo: Depositar

Noción
Acción de agregar dinero a una cuenta.
Impactos
El banco debe corroborar que la persona quien opera es el propietario de la cuenta.
El banco agrega el dinero a la cuenta.
El banco registra la operación en un log.

Figura 2.5 Descripción del símbolo depositar

Verbo: Extraer

Noción
Acción de tomar dinero desde la cuenta.
Impactos
El banco debe corroborar que la persona quien opera es el propietario de la cuenta.
El banco debe verificar que la cuenta posee suficiente dinero para realizar la extracción.
El banco debe verificar que el propietario de la cuenta no ha realizado más extracciones que la cantidad permitida.
El banco registra la operación en un log.

Figura 2.6 Descripción del símbolo extraer

Verbo: **Registrar**

Noción
Acción realizada por el banco para agregar la operación al log.
Impactos
El banco registra la operación en un log.

Figura 2.7 Descripción del símbolo Registrar

Verbo: Corroborar

Noción
Acción realizada por el banco para asegurar la identidad de la persona que quiere realizar una operación con una cuenta.
Impactos
El banco verifica la identidad de la persona.

Figura 2.8 Descripción del símbolo Corroborrar

Verbo: Consultar el saldo

Noción
Acción de informar el monto de dinero que la cuenta posee.
Impactos
El banco debe corroborar que la persona quien opera es el propietario de la cuenta.
El banco informa el monto.
El banco registra la operación en un log.

Figura 2.9 Descripción del símbolo Consultar Saldo

Observando las definiciones anteriores y teniendo en cuenta solo los impactos descriptos, los verbos **corroborar** y **registrar** son referenciados por los otros tres símbolos, tanto **corroborar** como **registrar** son considerados características transversales.

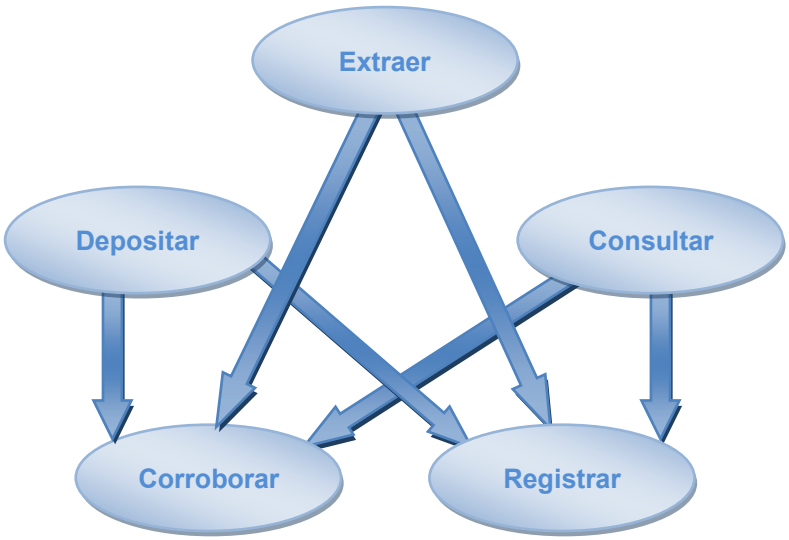

Figura 2.10 Referencias a los símbolos corroborar y registrar

Esta es la base y esencia de la estrategia, sin embargo, esta utiliza símbolos de categorías de estados para agrupar los otros símbolos y el análisis de referencias se realiza a partir de los grupos determinados por los estados.

La forma en la que se encuentran estructurados los símbolos donde en el ejemplo corroborar y registrar son referenciados por depositar, extraer y saldo viene al hecho de que es una forma natural de hacerlo y no requiere un trabajo extra para el encargado de armar el LEL. Si las referencias fueran inversas, los cálculos del algoritmo serian más simples, se vería de la misma forma que los aspectos en el código fuente, pero como se menciono anteriormente traería un costo adicional para el ingeniero encargado del armado del LEL.

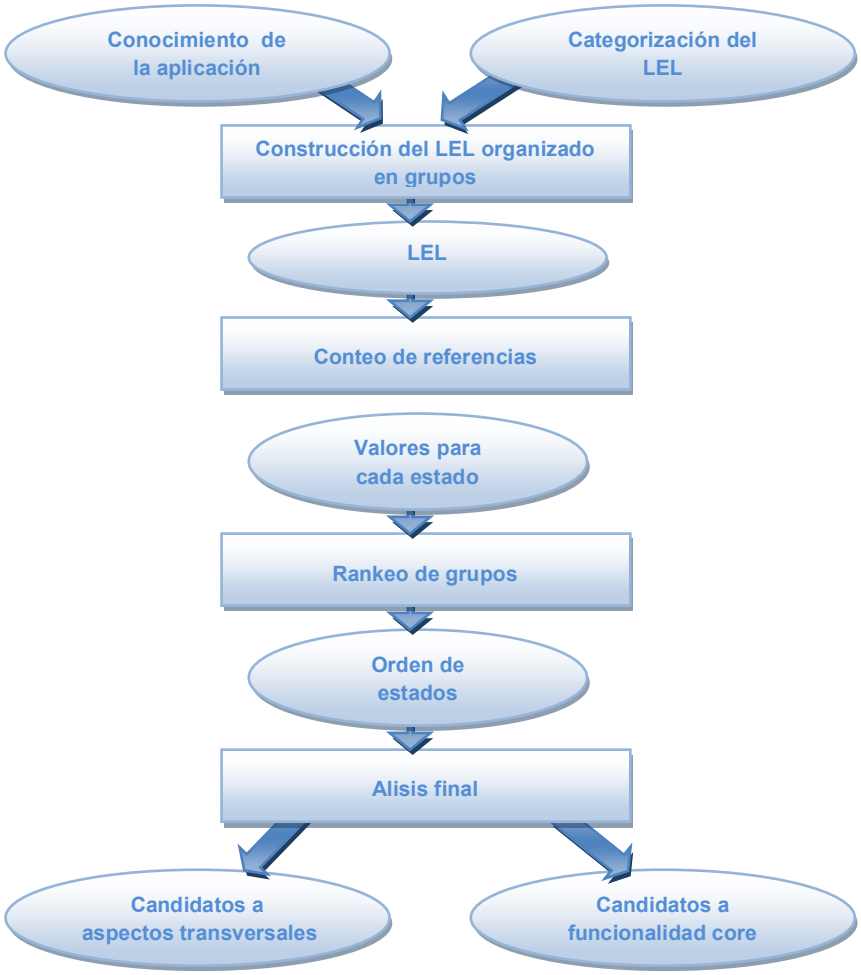

Figura 2.11 Pasos de la estrategia de Identificación de características transversales

El detalle de los pasos desarrollados por la estrategia de detección de características transversales mediante LEL, es el siguiente:

i. **Construcción de LEL organizado en grupos**. Se debe definir la noción y los impactos de cada símbolo y también se debe relacionar cada símbolo con un estado.

ii. **Conteo de referencias**. Se deben contar las referencias de los impactos entre símbolos que pertenezcan a distintos estados.

iii. **Clasificación de grupos**. Se deben ordenar los grupos de símbolos (estados) de mayor a menor de acuerdo a la cantidad de referencias, esto determina la probabilidad de que sean considerados características transversales.

iv. **Análisis Final**. En esta etapa se deben identificar los grupos con mayor probabilidad de que sean consideradas características transversales. Además, se debe realizar un análisis a nivel de símbolo para determinar si hay símbolos que distorsionan el orden de los grupos en su totalidad. Incluso se puede realizar un análisis más fino, para determinar si ciertos símbolos particulares del grupo son los que le dan la condición de característica transversal al estado. En este punto, puede verse que un análisis a dos niveles: a nivel estado y luego a nivel nodo.

En las siguientes secciones se describe en detalle cada uno de los pasos de la estrategia.

2.2.1 Construcción del LEL organizado en grupos

Esta estrategia, requiere que los símbolos del LEL se agrupen en conjuntos determinados por los símbolos estados. Esta organización radica en la necesitad de determinar que bloque de trabajo representa la funcionalidad principal (o core) y que bloque representa funcionalidad aspectual. Como se nombro previamente las aplicaciones de software pueden ser reducidas (descriptas) como maquinas de estados [Mahoney 2005] [Mahoney 2007], y dado que el LEL provee una categoría especifica de estados y estos símbolos son naturalmente identificados y descriptos, la estrategia determina el agrupar los símbolos en estados.

El proceso de identificación y descripción de símbolos en el marco de la estrategia para identificar características transversales es el siguiente:

i. Construcción del LEL.
   Se debe construir el LEL siguiendo los siguientes pasos: (a) identificación de símbolos; (b) categorización de cada símbolo; (c) descripción de los símbolos de acuerdo a la categoría e (d) identificación de sinónimos.

ii. Identificar al símbolo más representativo y significativo de la aplicación.
   Este debe ser un objeto o sujeto el cual es el elemento más relevante en el contexto de la aplicación y que además posee una maquina de estados asociada. Finalmente, una vez que este símbolo es identificado, se debe identificar todos los estados vinculados a este símbolo. Considerando el ejemplo de la aplicación bancaria, los símbolos que se identificaron son: *banco, cliente, cuenta, log, abrir una cuenta, activar una cuenta, registrar, corroborar, operar, extraer, consultar saldo, solicitada, bloqueada* y *activada.* Sin embargo, de todos los símbolos, sólo se debe prestar atención a *banco, cliente, cuenta* y *log,* puesto que los dos primeros son sujetos y los dos últimos son objetos. De estos cuatro, el símbolo principal *cuenta* tiene asociada una máquina de estados. Por último se debe identificar los símbolos de categoría estado que están vinculado con el símbolo principal. Los estados vinculados con *cuenta* para la aplicación bancaria son: *solicitada, bloqueada* y *activada.*

iii. Vincular cada uno de los símbolos del LEL con alguno de los estados determinados en el punto anterior.
   Existen distintas razones para vincular un estado, estas son:
   - El símbolo es instanciado en el estado
   - El acoplamiento entre el estado y el símbolo implica una gran interacción, pesar de que el símbolo no es instanciado en el estado.

   Siguiendo con el ejemplo de la aplicación bancaria, el estado *solicitada* se relaciona con los símbolos *banco*, *cliente*, *cuenta* y *abrir una cuenta*, puesto que todos esos elementos intervienen para solicitar una cuenta. Luego, de solicitar una cuenta se obtiene una cuenta *bloqueada*, en donde no se pueden realizar operaciones más que *activar*, sin embargo, se puede intentar operar con la misma y es por ello que tienen sentido las operaciones *registrar* y *corroborar,* como así también el *log*. Luego, en el estado *activado*, se ubican las 3 operaciones que tienen sentido realizar con la cuenta: *depositar*, *extraer* y *consultar el saldo*. También se ubica el símbolos *operar* para generalizar los otros 3.

2.2.2 Conteo de Referencia

El conteo de referencias consiste en calcular el promedio de referencias que cada grupo de símbolos posee desde los impactos de símbolos que se encuentran relacionados con otro estado.

En otras palabras, se debe obtener el promedio de referencias para cada uno de los grupos de símbolos (determinado por cada estado del símbolo principal y todos los símbolos asociados al estado) contando solo las referencias provenientes de los impactos de símbolos que no pertenecen al grupo que está siendo contabilizado.

Este conteo presenta dos características particulares. Solo se cuentan las referencias entre símbolos que pertenecen a distintos grupos. Esto debido a que se intenta medir el acoplamiento entre distintos grupos de símbolos. Además, se tiene en cuenta el promedio de referencias ya que los grupos pueden tener distinta cantidad de símbolos, lo que puede provocar que un grupo tenga más cantidad de referencias solo por el hecho de tener más símbolos y no por tener un mayor nivel de acoplamiento.

Finalmente, para obtener mayor precisión en el análisis, los promedios de referencias se calculan para todas las categorías, es decir, se deben calcular las referencias producidas sólo por sujetos, sólo por objetos y sólo por verbos.

2.2.3 Ranqueo de Grupos

Van Den Berg et al [Van Den Berg 2005] definen que dos características transversales determinan la presencia de características transversales: dispersión (*scattering*) y enmarañamiento (*tangling*). Van Den Berg dice que la dispersión ocurre cuando "*un elemento origen es relacionado a múltiples elementos destinos*", mientras que el enmarañamiento ocurre cuando "*un elemento destino es relacionado a múltiples elementos destinos*". En esta estrategia tomamos como elemento a cada uno de los grupos, y tomamos las referencias provenientes de los impactos de los símbolos para medir la dispersión y el enmarañamiento.

Se deben considerar como posibles características transversales a aquellos grupos de símbolos que maximicen las siguientes variables:

1. La cantidad de grupos de símbolos distintos que los referencian.
2. Promedio de referencias totales.

Es importante maximizar ambas variables ya que (1) la cantidad de grupos que los referencian indica cuan disperso (*scattered*) está el grupo, mientras que (2) el promedio de referencias indica cuando acoplado (*tangled*, enmarañado) se encuentra el grupo.

Basándonos en las dos variables previamente descriptas, además de grupos con las características buscadas pueden encontrarse grupos de símbolos con las siguientes propiedades:

1. Maximizan la cantidad de grupos de símbolos que lo referencian pero posee un bajo promedio de referencias.

   En este caso puede ocurrir es que un subconjunto de símbolos reciban la mayoría de las referencias, lo que provoca que el promedio de referencias baje a pesar de que se encuentra disperso. En este caso se debería analizar la posibilidad de que el subconjunto que recibe la mayoría de las referencias sea considerado característica transversal en lugar del conjunto completo.

2. Poseen una baja cantidad de grupos de símbolos que los referencian pero un alto promedio de referencias.

   Aquí se presenta un caso donde si bien el grupo de símbolos se encuentra altamente acoplado con una baja cantidad de grupos de símbolos, no está lo suficientemente disperso como para ser considerado característica transversal.

3. Minimizan la cantidad de grupos de símbolos que lo referencian y tienen un bajo promedio de referencias.

   Este conjunto no debe ser considerado candidato a ser característica transversal ya que no se encuentra disperso ni acoplado.

En la figura 2.12 se presenta un diagrama xy con el objetivo de graficar cuando un grupo de símbolos es candidato a ser característica transversal. Este diagrama presenta en su eje x el promedio de referencias del grupo (cuan acoplado esta), mientras que en su eje y se muestra la cantidad de grupos que lo referencian (cuan disperso se encuentra). En el diagrama se encuentra sombreado de acuerdo a las variables antes descriptas, de forma que en su parte más oscura se presentan aquellos grupos que son candidatos a ser característica transversal.

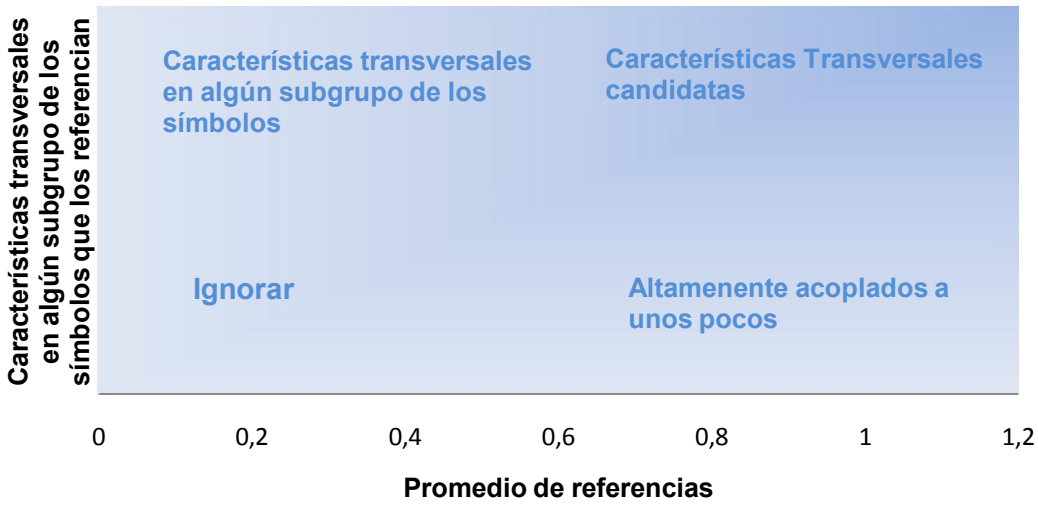

Figura 2.12 Diagrama XY para visualizar cuando un grupo de símbolos es una posible característica transversal

Aplicando el diagrama anteriormente descripto al dominio de la aplicación bancaria, se obtiene la figura 2.13. En este se tiene un punto por grupo de símbolos (3) y todos a la misma altura, esto se debe a que todos los grupos tienen al menos una referencia de los grupos restantes. En cambio, el promedio de referencias (eje y) es el que varia, por lo que el grupo 1 **solicitada**, se encuentra en el extremo derecho, el grupo 2 **bloqueada** se encuentra en el centro de los dos grupos y finalmente el grupo 3 **activada** es el que se encuentra más a la izquierda.

En este caso, el orden de los grupos es claro, ya que la cantidad dispersión (cantidad de grupos distintos desde los que existen referencias a un grupo de símbolos) es la misma para todos los grupos por lo que el orden lo establece el promedio de referencias generales. Por todo esto, el orden para considerarlos características transversales es el siguiente:

1. Solicitada
2. Bloqueada
3. Activada

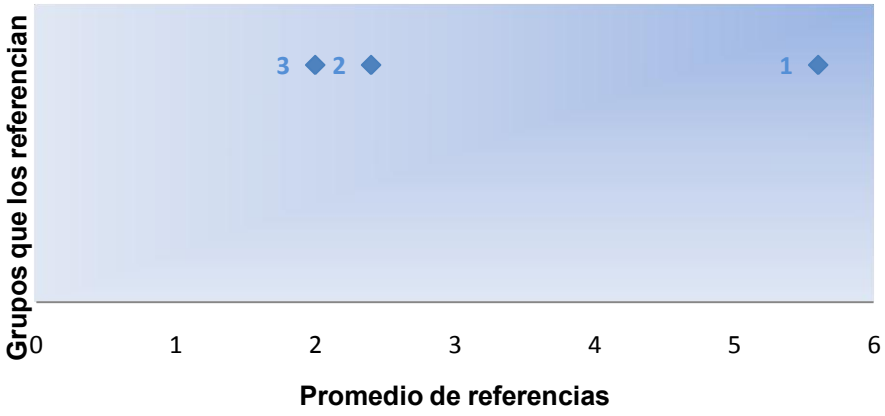

Figura 2.13 Diagrama XY para visualizar en el dominio del banco las posibilidades de los grupos de ser característica transversal

Tabla 2.3 Grupos del dominio bancario ordenados en base a la posibilidad de ser característica transversal

		Grupos que lo referencian	Promedio general de referencias
1	Solicitada	2	5.6
2	Bloqueada	2	2.4
3	Activada	2	2.0

2.2.4 Análisis Final

Finalmente, es necesario llevar a cabo un análisis de los resultados obtenidos en la etapa anterior. El mismo debe consistir en revisar los símbolos y determinar si alguno podría alterar y/o cambiar el ranking de los grupos obtenidos en la etapa anterior. Es importante prestarle atención a aquellos símbolos principales (sujetos, objetos y verbos) los cuales no se modularizarán como características transversales, sino que se hará como funcionalidad core. Estos símbolos podrían tener una gran cantidad de referencias y podrían generar que todo el grupo lo sea. Por lo cual, el símbolo debería ser separado del grupo, no cuentan las referencias que recibe sin embargo si cuentan las referencias que posee.

También es necesario considerar que dentro de un grupo, además de símbolos core existan símbolos aspectuales, los cuales surgen de un análisis más detallado y son más representativos de las características transversales que el mismo grupo.

Si bien los tres grupos poseen igual dispersión (cantidad de grupos que los referencian), solo el grupo bloqueada debe ser considerado como característica transversal. Esto se debe a que tanto el grupo Solicitada (que posee un mayor promedio de referencias) como el grupo Activada poseen elementos core los cuales acaparan la mayor cantidad de referencias en el grupo por lo que deben ser descartados.

Tabla 2.4 Análisis final sobre los grupos del dominio bancario

Id	Estado	Descripción
1	Solicitada	Los símbolos *banco* y *cuenta* son los principales sujetos y objetos de la aplicación y alteran las referencias, es por ello que estos símbolos no deben considerarse. Por lo cual, este grupo no debe ser considerado candidato.
2	Bloqueada	Este grupo posee sólo objetos y verbos, y las referencias están balanceadas entre todos ellos. Por lo cual, este grupo debe ser considerado candidato.
3	Activada	Este grupo posee sólo verbos y las referencias están bastante balanceadas entre todos ellos. Sin embargo, estos verbos son considerados símbolos principales en el lenguaje de la aplicación. Por lo cual, este grupo no debe ser considerado candidato.

2.3 C&L

C&L es una herramienta de software libre que proporciona al usuario un ambiente colaborativo que permite armar y diseñar escenarios y léxicos descriptos en un lenguaje natural semiestructurado.

El objetivo que persigue esta herramienta es el de facilitar la comprensión de una situación específica en una aplicación a través del uso del lenguaje natural, priorizando su comportamiento. Además el de permitir organizar la información a través de una estructura bien definida.

2.3.1 Generalidades

En la creación y edición de proyectos, C&L permite que diferentes usuarios puedan editar un mismo símbolo de forma simultánea. Esto se debe a que esta herramienta presenta dos niveles de acceso:

- Nivel Usuario
- Nivel Administrador

Todos los proyectos tienen un usuario con nivel administrador y n usuarios con nivel usuario. En ambos niveles de accesos se permite ver, agregar, modificar y eliminar símbolos, la diferencia radica en que cuando se modifica un símbolo, el administrador debe aceptar la modificación.

Cada proyecto puede tener uno o más escenarios, el uso de escenarios integrados con el código fuente es una técnica que tiene como objetivo la mejora en la presentación de la documentación, por lo que estructurada y uniformemente facilita su lectura y compresión.

Estos escenarios tienen una estructura que consta de las siguientes propiedades:

- Titulo
- Objetivo
- Contexto
- Recurso

- Autores
- Excepciones o atributos restrictivos

Figura 2.14 Página principal de C&L

Mientras que titulo, objetivo, contexto y excepciones son sentencias declarativas simples, los episodios son sentencias escritas en un lenguaje simple que debe proveer una descripción operacional del comportamiento en la situación planteada.

Un escenario debe satisfacer un objetivo, la forma de lograr este objetivo se describe paso a paso por sus episodios. Los episodios no solo representan el plato principal de las acciones de un escenario, sino que incluyen también las variaciones y otras alternativas posibles. Una excepción puede ocurrir durante aplicación de los episodios, indicando que hay un obstáculo para satisfacer objetivo. El tratamiento de la excepción no necesariamente satisface el objetivo.

Por cada escenario, se agregan símbolos, los cuales son generales para todos los escenarios del proyecto. Las propiedades que tiene cada símbolo son las descriptas por el LEL:

- Nombre
- Noción
- Impacto
- Clasificación

Figura 2.15 Pantalla para agregar un escenario en C&L

Además, de existir se pueden agregar sinónimos del símbolo que se está agregando.

Figura 2.16 Pantalla para agregar un símbolo en C&L

Además de los símbolos, en cada escenario se puede hacer referencia a otros escenarios creados. Para esto debe existir un escenario integrador como muestra la figura 2.17.

Una vez que los símbolos y escenarios se cargaron, C&L hace un refinamiento de los escenarios, resultando en un único documento. El método utilizado para el refinamiento usa la idea de escenarios de integración para proporcionar una visión global del sistema, con el objetivo de facilitar la compresión y gestión de los grupos de escenarios que componen el sistema.

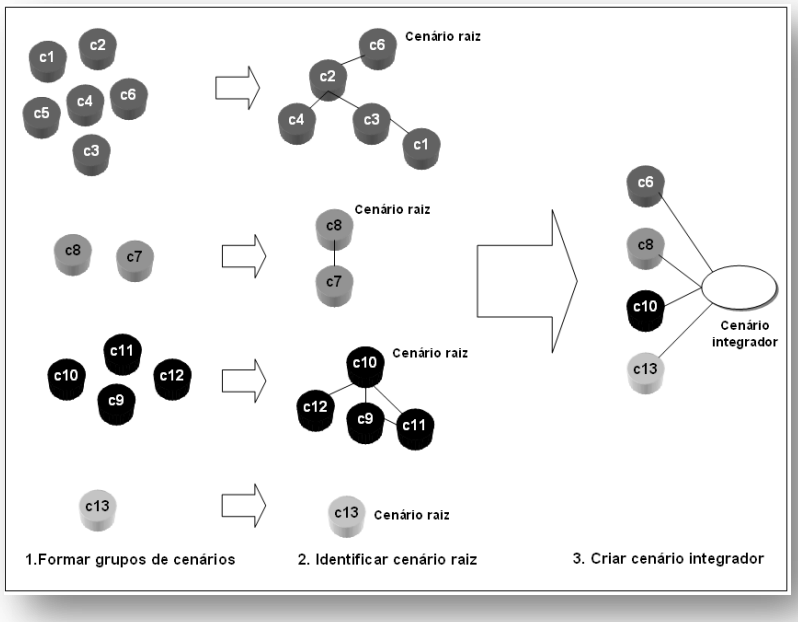

Figura 2.17 Escenario Integrador de C&L

2.3.2 Integración con TICC

Para la implementación de la herramienta desarrollada en esta tesis (Capítulo 3) sólo es de interés el LEL, el cual está naturalmente organizado en un hipertexto. C&L implementa las referencias (*links*) entre los símbolos del LEL y también hacia los escenarios. Estas referencias (*links*) se crean automáticamente cuando el usuario describe un símbolo e ingresa una palabra la cual ya se encuentra definida dentro del LEL.

El prototipo actual de C&L fue desarrollado utilizando el lenguaje Lua. La herramienta C&L opto por utilizar este lenguaje por tres razones:

1. Lua ofrece un excelente soporte para la programación funcional.

2. Lua es un lenguaje que se está utilizando cada vez más y sus principales características (gratis, rápido, extensible, entre otros) son útiles para el desarrollo de un sistema web para C&L.

3. Lua proporciona un excelente apoyo para el uso de expresiones regulares y comparaciones en cadenas, funcionalidad necesaria para la herramienta C&L.

Lua no fue diseñado para el desarrollo de sistemas Web, por lo que la herramienta C&L utilizó la plataforma **Kepler** [Kepler 09]. Esta plataforma proporciona una serie de módulos que facilitan el desarrollo de sistemas Webs basados en código Lua.

C&L presenta una arquitectura modular la cual permite que le sean agregados nuevos plug-ins para proveer de nueva funcionalidad como si fuera funcionalidad nativa del sistema. Por esto y por una futura integración de TICC con C&L es que la funcionalidad base de TICC fue desarrollada en lenguaje LUA.

Capítulo 3

3. Herramienta

El presente capítulo describe la herramienta que brinda un soporte automatizado para la detección de características transversales en etapas tempranas del desarrollo. La herramienta implementa la estrategia de detección temprana de características transversales usando LEL [Antonelli 2011] tomando como datos de entrada los LELs exportados desde la herramienta C&L. La implementación de esta estrategia busca a futuro ser integrada como complemento al software C&L [C&L 2009]. Por esto, la funcionalidad base de la aplicación TICC fue desarrollada utilizando el lenguaje Lua.

El capítulo se encuentra dividido en cuatro secciones. En la primera se describe la arquitectura utilizada así como el porqué de los componentes. En la segunda sección se presenta la implementación de la herramienta. Mientas que en la tercera sección se encuentra la forma de instalar la herramienta. Sobre el final del capítulo, se encuentra la forma de utilizar la herramienta.

3.1 Arquitectura

El desarrollo de la herramienta se baso en una arquitectura de sistemas moderna, de acceso amplio y orientado a servicios utilizando el concepto de Arquitectura Orientada a Servicios de Clientes (*Service Oriented Architecture*, SOA).

SOA es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos de negocios. La utilización de una arquitectura SOA permite que los sistemas sean altamente escalables brindando a su vez una forma bien definida de exposición e invocación de servicios (comúnmente, pero no exclusivamente servicios web).

En particular, para la construcción de TICC, se utilizo una arquitectura de n-capas distribuida, utilizando el lenguaje Java como la tecnología dominante para el soporte de los servicios, siguiendo un patrón de diseño MVC (*Model View Controller*). Además a este modelo se le agrego:

- Independencia o portabilidad de sus componentes de infraestructura de software y hardware.
- Lua como tecnología secundaria.

La separación en capas propuesta permite aislar los distintos aspectos del desarrollo de la herramienta, trabajando de esta manera las capas inferiores proveen servicios a su capa inmediata superior. Esta forma de separación, además de lo descripto y mediante el uso de interfaces permite que las capas puedan reemplazarse teniendo un costo e impacto mínimo respecto de las capas con las que se comunica. Un ejemplo de esto puede ser que la capa de servicios de negocios prepara los datos para entregárselos a la capa de presentación, esta capa se encarga de cómo mostrar los datos que recibe, sin preocuparse de que datos muestra. Las capas en las que se separo la arquitectura de TICC, ordenadas desde la capa superior a la inferior fueron las siguientes:

- Presentación
- Servicios de Negocios
- Modelo de Dominio
- Integración
- Algorítmica

A continuación se detallan cada una de ellas, junto con las tecnologías utilizadas.

Figura 3.1 Modelo de Capas

3.1.1 Capa de Presentación

Esta capa tiene como responsabilidad presentar la funcionalidad ofrecida por el sistema a través de una interfaz gráfica. Este esquema permite desacoplar la lógica de negocios de la presentación permitiendo alterar la interfaz de la aplicación o agregar un medio de presentación alternativo sin necesidad de alterar la lógica de negocios. Esta independencia del modelo de servicios permite acoplar rápidamente nuevas tecnologías de presentación.

La implementación de esta capa se desarrollo usando Spring MVC [Spring 2012], el cual es un framework basado en solicitudes (*request*). Este define interfaces para todas las responsabilidades que deben ser manejadas. Todas las interfaces están altamente relacionadas con la API de Servlet [Java 1997], lo cual asegura que las mismas se mantengan disponibles, mientras que ofrece un framework de nivel abstracto que permita facilitar el uso de la Interfaz de Programación de Aplicaciones (*Application Programming Interface*, API). La facilidad para testear las implementaciones de las interfaces, es una importante ventaja de nivel de abstracción ofrecido por Spring. Esta capa fue enriquecida agregándole controles provistos por JQuery [JQuery 2012], en particular fue muy útil el control que permite arrastrar (*Droppeable control*) objetos/símbolos de la interfaz gráfica ya que permite una interacción simple e intuitiva con el usuario. JQuery es una biblioteca de Javascript [JavaScript 2012] la cual simplifica la forma de interactuar con documentos HTML, manipular arboles DOM, gestionar eventos, desarrollar animaciones e incorporar AJAX [AJAX 2012]. Finalmente, el último componente de esta capa en ser introducido son las bibliotecas gráficas de Google [Google 2012], las cuales son utilizadas para gráficas estadísticas sobre los resultados obtenidos por la herramienta.

3.1.2 Capa de Servicios de Negocios

Esta capa es central en la arquitectura de la aplicación, porque es la capa que lleva a caracterizarse como SOA. Permite abstraer todas las complejidades propias del dominio del sistema y brindársela como funciones de grano grueso para que sean utilizadas por la capa de presentación.

Cada servicio de esta capa se encarga de integrar cada una de las capas inferiores y delegar al modelo de dominio la responsabilidad de la lógica de negocios.

En un sistema JEE como se propone, esta capa estará actuando en el contexto de servidor de aplicación (*application-server*) y con un conjunto de tecnologías, herramientas y frameworks.

Adicionalmente en esta capa se hace un uso extensivo de patrones arquitecturales como Fachada (*Session Facade*) y Servicio de Aplicaciones (*Application Service*), donde Fachada tiene como objetivo el exponer ante los usuarios componentes de la capa de negocio evitando que accedan directamente a los objetos de esta capa. Mientras que el patrón Aplicación de Servicio cumple la finalidad de centralizar la lógica de negocio y apartarla de los objetos de negocio. El uso de estos patrones es para realizar una correcta organización de la lógica y su buena performance de acceso.

La capa de servicios será implementada sobre el framework de aplicaciones conocido como Spring [Spring 2012] que se caracteriza por ser un contenedor liviano sobre el cual los servicios de negocio son implementados como objetos Java simples (*Plain Old Java Object, POJO)*. Con POJO, nos referimos a una instancia de una clase que no extiende ni implementa nada en especial.

3.1.3 Capa de Modelo de Dominio

Esta capa crea y contiene una red de objetos interconectados donde cada objeto encapsula la estructura y el comportamiento de una entidad de negocio.

La lógica de negocios en un sentido amplio es un conjunto de métodos o procedimientos utilizados para administrar una función específica de negocio. El proceso de diseño orientado a objetos (*Object Oriented Design, OOD*), sobre el cual el sistema está basado, permite descomponer esa función de negocios en un conjunto de elementos denominados objeto de negocios (*Business Objets*). La representación de un problema de negocios involucra varios objetos de negocio que interactúan para proveer la funcionalidad de negocios requerida. El conjunto de reglas específicas de negocio ayudan a identificar la estructura y el comportamiento de los objetos de negocio.

Los componentes de esta capa estarán implementados como objetos PersistentObject. Esta clase pose las características básicas de un objeto persistente. Permitiendo el acceso a datos de forma transparente e independiente de la implementación del framework de persistencia de datos (por ejemplo Hibernate).

3.1.4 Capa de Integración

La capa de integración es la encargada de conectar la capa JEE de la herramienta con la capa de Scripting Lua. Para lograrlo fue necesario el uso de dos frameworks, Maven [Maven 2012] y Jnlua [JnLua 2012].

El framework Maven fue utilizado en las capas anteriormente descriptas como forma de organización, enumerando y gestionando las distintas bibliotecas de software libre que fueron necesarias para desarrollar algunas de las características de la herramienta, ejemplo de esto

son las bibliotecas para generar objetos en formato Json a partir de un objeto Java (Jackson JSON Mapper). En esta capa, además de la función que cumplió en las capas previas fue necesario realizar una compilación e instalación del framework JnLua debido a que para que JnLua pueda ejecutarse desde un proyecto administrado por Spring, este debe ser parte de su repositorio local.

El framework JnLua es el encargado de traducir código Java a código Lua y viceversa. Debido a que el lenguaje Lua usa bibliotecas escritas en C, el lenguaje natural para combinarlo es C o .NET y no existe una forma o framework transparente que lo combine con Java. Por esto fue que se analizaron distintas alternativas para esta capa, tales como LuaJ [Luaj 2007], Kahlua2 [Kahlua 2012], LuaJava [LuaJava 2011] y JnLua. La elección por este último se debió a que los nombrados anteriormente presentaban las siguientes características:

- **Kahlua2**: este framework es el que mayor transparencia provee en la transformación de objetos, ya que el paso de un lenguaje a otro es muy simple y soporta toda la funcionalidad que Java provee. Fue descartado debido a que Kahlua2 no tiene soporte del 100% del código Lua. El framework no ejecuta código Lua nativo sino que es una implementación de Lua sobre código Java. La elección de este framework provocaría que no se pudiera dejar una implementación en Lua de la estrategia de identificación de características transversales. Dificultando una posible integración con C&L.

- **LuaJava**: esta fue la primera herramienta que salió para trabajar con código Lua y Java al mismo tiempo, su principal objetivo fue poder enriquecer código Lua agregándole código Java y no a la inversa. Además no provee soporte para la transformación de listas, lo que es una característica necesaria en la implementación de la estrategia de detección temprana de características transversales. Este framework se encuentra obsoleto y trabaja con Lua 5.1.

- **Luaj**: es un framework que estaba basado en LuaJava. Mantiene el soporte y la API de LuaJava y además provee soporte para JSE (Java Standar Edition) y JME (Java Micro Edition). Y a diferencia de Luajava agrega soporte para Lua 5.2. Fue descartado porque si bien provee soporte para datos del tipo lista, las mismas no cuentan con un soporte adecuado para listas de objetos definidas por el usuario.

Finalmente, se termino optando por **JnLua**, que si bien no provee una transparencia como la que si da Kahlua, este permite una interacción completa entre datos nativos pertenecientes al código Lua y el código Java en ambos sentidos. Además provee un manejo transparente de los errores los errores generados en la ejecución de código Lua, los cuales son interceptados por Java.

3.1.5 Capa Algorítmica

Esta capa es la encargada de interpretar el LEL seleccionado y transformarlo en datos que la herramienta reconozca, así como también de implementar la estrategia de detección de características transversales.

La implementación de esta capa se realizo utilizando el lenguaje Lua, así como el framework Lua Rocks [Lua Rocks 2012] del cual se utilizaron bibliotecas de manejo de archivos. La selección del lenguaje Lua no fue arbitrario, debido a que esta selección tiene como propósito a futuro una posible incorporación de estos algoritmos a la herramienta C&L, de forma que además de poder crear y editar los LEL, también pueda detectar características transversales.

Esta incorporación de nueva funcionalidad es posible debido a que C&L fue desarrollado en el lenguaje LUA y permite nueva funcionalidad si esta es desarrollada en forma de plugin.

3.1.6 Tecnologías Adicionales

Además de las tecnologías utilizadas en cada una de las capas anteriormente descriptas, la herramienta también utiliza:

- **Servidor Jetty**
  La arquitectura desarrollada tiene como objetivo ejecutarse en un contexto web para que cualquier persona pueda usarlo y no sea necesaria su instalación. Con este fin se utilizo un servidor Jetty [Jetty 2012] que es un servidor HTTP basado en Java y contenedor de Java Servlets y Java Server Pages (JSP) que fue publicado bajo la licencia de software libre Apache 2.0. El objetivo que persigue Jetty es el de ser un servidor web sencillo, eficiente y fácil de ser incorporado en distintas aplicaciones.

- **XML** (*Extensible Model Languaje*)
  Comprende un conjunto de reglas para codificar documentos de una forma entendible por la computadora. La W3C produjo una especificación XML 1.0 la cual sienta las bases para la construcción de documentos XML. Los mismos son archivos de texto con un fuerte soporte de Unicode para facilitar el uso por los distintos lenguajes. Aunque el diseño de XML se centra en documentos, es ampliamente utilizado para la representación de estructuras de datos, por ejemplo, servicios web.
  La herramienta C&L utiliza la tecnología XML para exportar el LEL y TICC toma el archivo resultado de C&L y en base a su contenido determina que nodos son potenciales características transversales.

3.2 Implementación de TICC

Para explicar de forma clara la implementación de TICC, se va a utilizar la misma subdivisión de capas descripta anteriormente. Está forma de división para explicar la aplicación es conveniente ya que cada capa cumple una función específica y totalmente distinta que el resto de las capa.

3.2.1 Capa de Presentación

La capa de presentación en TICC fue desarrollada usando el framework Spring MVC, el cual es una implementación de Spring del patrón (*pattern*) MVC. Los elementos que conforman esta capa son:

- **Modelo** (*Model*): encapsula los datos de la aplicación y generalmente puede estar constituida por objetos POJO.
- **Controlador** (*Controller*): es el responsable de procesar las solicitudes de los usuarios y de la construcción del modelo apropiado para luego pasarlo a la vista para la representación.
- **Vista** (*View*): es la responsable de procesar los datos del modelo y generar la vista.

A continuación se describirá una implementación de ejemplo tomada de la aplicación TICC.

3.2.1.1 Definiendo el controlador

El Controlador (la C de MVC) es la componente que se encarga, como su nombre indica, de manejar las peticiones (HTTP Request) que recibe la aplicación web por parte del usuario. Decide qué hacer con ella y normalmente delega a los objetos de servicios (capa de servicios) la lógica de negocio. Además determina a dónde tiene que redirigir al usuario o si creará directamente un resultado (un HTTP Response) para él mismo. Es decir encapsula la lógica de navegación.

En este caso presentaremos un controlador responsable de generar la vista para los símbolos que fueron generados a partir del archivo XML.

```
@Controller [1]
@RequestMapping(value = "/results")[2]
@SessionAttributes({ "symbols", "results" })
public class ResultsTableConcernsController {
        @Autowired [3]
        private SymbolService symbolService;

        public SymbolService getSymbolService() {
                return symbolService;
        }

        public void setSymbolService(SymbolService symbolService) {
                this.symbolService = symbolService;
        }

        @RequestMapping(value = "/table"[5], method = RequestMethod.GET [6]) [4]
        public String viewTableResults(Model model, HttpSession session) {
                List<Node> symbols = (List<Node>)
                                                session.getAttribute("symbols");
                List<ResultState> results =
        this.getSymbolService().getCrossCuttingConcernsResult(symbols);

                model.addAttribute("results", results); [7]
                return "results/concerns-table-result";[8]
        }
}
```

Figura 3.2 Ejemplo de controlador de la capa de presentación.

La anotación (*annotation*) **@Autowired** [3] es un recurso de Spring que le pide al contenedor (Java Beans Container) que nos provea de una instancia del bean SymbolService para poder responder a la petición que nos llega desde el navegador web al Controlador. Spring buscará en el contenedor un bean del tipo SymbolService, e incluso si su nombre no es el mismo que el de la variable. Cuando se cree la instancia del Controlador, Spring le "inyectará" la instancia del servicio SymbolService.

La anotación **@Controller** [1] indica a Spring que la clase va a cumplir el papel de un controlador.
La anotación **@RequestMapping** se utiliza para asignar una URL a toda una clase o a algún método particular. El primer uso de la anotación **@RequestMapping** [2] es a nivel de clase e indica que todos los métodos incluidos en el controlador son relativas a la ruta **/results**. Mientras que el segundo uso de la anotación **@RequestMapping** [4] es a nivel de método. El atributo **value** [5] indica la URL a la que se asigna el método y el atributo **method** [6] nos permite especificar el tipo de petición http (GET, POST, DELETE). Por lo tanto el método

viewTableResults() será el encargado de responder a aquellas peticiones por GET que el servidor reciba con la URL /table [5].

Existen otros puntos importantes a destacar sobre el controlador definido anteriormente:

- La lógica de negocio necesaria podrá ser definida en cualquier método del controlador.

- Según la lógica de negocio, se podrá crear un modelo en el método, para poder asignarle diferentes atributos. Dichos atributos serán accedidos por la vista con el fin de presentar el resultado final. En el ejemplo, el método viewTableResults () crea un modelo con los atributos “results” [7] ya que la información asociada a cada uno de ellos es necesaria para presentar la vista.

- Un método del controlador pude devolver un string que contendrá el nombre de la vista que será usada para mostrar el modelo. En el ejemplo, el método viewTableResults() retorna "graph/graph-define-states" [8], el cual es el nombre lógico de la vista.

3.2.1.2 Creando la vista jsp

Spring MVC soporta muchos tipos de vistas para diferentes tecnologías de presentación. Estos incluyen - JSP, HTML, PDF, hojas de cálculo Excel, XML, plantillas Velocity, XSLT, JSON, Atom, RSS feeds, JasperReports etc.

La aplicación TICC usa la tecnología de presentación de plantillas **JSP** (*Java Server Pages*) que son escritas con **JSTL** (*JSP Standard Tag Library*). JSTL no es más que un conjunto de librerías de etiquetas simples y estándares que encapsulan la funcionalidad principal que es usada comúnmente para escribir páginas JSP. Las etiquetas JSTL están organizadas en 4 librerías:

- **core**: Comprende las funciones script básicas como loops, condicionales, y entrada/salida.
- **xml**: Comprende el procesamiento de xml.
- **fmt**: Comprende la internacionalización y formato de valores como de moneda y fechas.
- **sql**: Comprende el acceso a base de datos.

La página asociada al controlador descripto previamente será descripta a través de sus diferentes bloques que la conforman.

3.2.1.3 Librerías de etiquetas

Es importante incluir en la página JSP, las librerías necesarias para la escritura de las mismas. En la página JSP se importa cada librería JSTL que la página necesita. Para ello se agregaron las directivas **taglib** apropiadas al inicio de la página JSP.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> [1]

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %> [2]

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %> [3]

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%> [4]
```

Figura 3.3 Ejemplos de Tags JSP

En la figura anterior se importaron el core de JSTL [1] y la fmt [2]. Y las librerías de etiquetas que Spring MVC proporciona para el manejo de formularios [3] e internacionalización [4].

3.2.1.4 Recursos de Hojas de estilo y Javascript

En el encabezado (*head*) de la página jsp se incluyen las hojas de estilos y los archivos javascript que necesita la página jsp.

```
<head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title><spring:message code="concerns.table.results.title"/></title>

        <link type="text/css" href="<c:url value="/resources/css/custom-blue/jquery-ui-
        1.9.0.custom.css"/>" rel="Stylesheet" /> [5]

        <link type="text/css" href="<c:url value="/resources/css/custom-
        blue/common.concerns.css"/>" rel="Stylesheet" />     [6]

        <script type="text/javascript" src="<c:url value="/resources/scripts/jquery/jquery-
        1.8.2.js"/>"></script> [7]

        <script type="text/javascript" src="<c:url value="/resources/scripts/jquery-ui/jquery-ui-
        1.9.0.js"/>"></script>[8]

        <script type="text/javascript">
                $(function() {
            $( document ).tooltip();[9]
          });
        </script>
</head>
```

Figura 3.4 Ejemplo de importación de recursos

Con la etiqueta link se vinculan las hojas de estilo con la página. Para la página de resultados se vincula tanto la hoja de estilo definida por la librería Jquery [5] como la hoja de estilo de la aplicación TICC [6].

Luego se importan los archivos de la librería Jquery, tanto la que incluye las funciones básicas de Jquery [7] como la que proporciona efectos de interfaz web [8].

Finalmente se incrusta código javascript [9], el cual permite asociar a cada elemento (de la página) que tiene tanto el atributo alt como title, el tooltip para brindar información del mismo.

3.2.1.5 Presentación de los resultados

En la figura 3.5 se muestra el bloque de código de una página jsp que implementa la lógica necesaria para mostrar la información que fue asociada, previamente en el controlador antes descripto.

Spring MVC provee la etiqueta **<spring:message>** [10] (ver Figura 3.5), la cual permite acceder a las palabras que se definieron en el archivo message.properties (archivo de internacionalización). El atributo code indica la clave asociada al valor internacionalizado.

```
<table class="bordered">
  <thead>
    <tr>
      <th>
     [10]<spring:message code="concerns.table.states.name"/>
      </th>
      …
    </tr>
  </thead>
 <tbody>
 <c:forEach items="${results}" var="resultState"> [11]
    <tr>
      <td>${resultState.name}</td> [12]
      <td class="number-cell">
                ${resultState.avgSubjects}
      </td>
          …
  </tr>
 </c:forEach>
 </tbody>
</table>
```

Figura 3.5 Ejemplo table JSP

En el archivo message.properties, Figura 3.6, se tiene las siguientes claves para cada uno de los mensajes internacionalizados:

```
concerns.table.states.name= State
concerns.table.subject.average= Average of references to subjects
concerns.table.objects.average= Average of references to objects
concerns.table.verbs.average= Average of references to verbs
```

Figura 3.6 Configuración de propiedades de Spring

Para poder añadir una sección que muestre la información de cada resultado de un estado, usamos la etiqueta JSTL **<c:forEach/>** [11] (ver Figura 3.5) . Esta etiqueta nos permite iterar sobre la variable results y por cada elemento, resultState, se muestra su información [12] (ver Figura 3.5).

3.2.2 Capa de Servicios de Negocio

El objetivo de esta capa es la de procesar la información de la capa de modelo para ser expuesta a la capa de presentación, de modo que la información expuesta sea la respuesta precisa a una funcionalidad particular de la capa de presentación. Este procesamiento tiene como objetivo que la capa de presentación sea una vista tonta (view dummy).

En los procesamientos de esta capa es común que la capa de presentación se comunique con la capa de servicios mediante objetos DTO que internamente son convertidos a objetos del modelo de dominio en la capa siguiente. En particular en TICC en estas capas no fue necesario el uso de objetos DTO debido a la simplicidad del modelo y que no existían propiedades que no debieran mostrarse en la capa de presentación.

En TICC esta capa debe ser implementada por una clase que extienda la interface SymbolService. Esto tiene como objetivo que si se decidiera cambiar la implementación de la capa de servicios, esta siga proveyendo los mismos servicios que la implementación anterior de forma que la comunicación con la capa de presentación no se vería afectada, la definición de esta interface puede verse en la figura 3.7.

```
public interface SymbolService {
/**
 * Obtiene todos los símbolos existentes.
*/
[1] List<Node> getAllSymbols(String fileName);
/**
 * Obtiene los resultados finales.
*/
 List<ResultState> getCrossCuttingConcernsResult(List<Node> nodes);

/**
 * Busca el nodo cuyo identificador se corresponda con el identificador recibido como
parámetro
*/
Node searchNodeById(List<Node> symbols, Long idNode);
/**
 * Busca aquellos nodos cuya clasificación sea igual a la recibida como   parámetro
*/
List<Node> searchNodesByType(List<Node> symbols, String type);
/**
 * Busca aquellos nodos que pertenecen al estado recibido como parámetro
*/
List<Node> searchMembersForState(List<Node> symbols, Long idNode);

/** Busca aquellos nodos que pertenecen al estado recibido como parámetro y además
están disponibles
 */
List<Node> searchAvailableMembersForState(List<Node> symbols, Long idNode);
}
```

Figura 3.7 Interface SymbolService para implementar la capa de servicio

La clase que implementa la interface se denomino SymbolServiceImpl, esta es instanciada de forma dinámica por Spring al momento de iniciar la aplicación. A modo de ejemplo puede verse la implementación del servicio getAllSymbols en la figura 3.8, el cual tiene como objetivo que a partir del path (ruta) de la ubicación de un LEL, el servicio retorne una lista con los símbolos que el LEL contiene.

```
public List<Node> getAllSymbols(String fileName) {
        try {
        return
        this.getNodeDTOConverter().
                convert(this.getLuaProxy().getNodes(fileName));
        } catch (Exception e) {
                logger.error("Couldn't get symbols from file: " + fileName,
e);
        }
        return null;
}
```

Figura 3.8 Implementación del servicio getAllSymbols

En [2] de la figura 3.8 el servicio invoca al método getNodes de LuaProxy, esta clase pertenece a la capa de integración y retorna un DTO por cada símbolo del LEL, mientras que convert [1] perteneciente a la capa de modelo, transforma los DTOs en nodes.

3.2.3 Capa de Modelo de Dominio

Como su nombre lo indica esta capa contiene las entidades del dominio (ver figura 3.9).

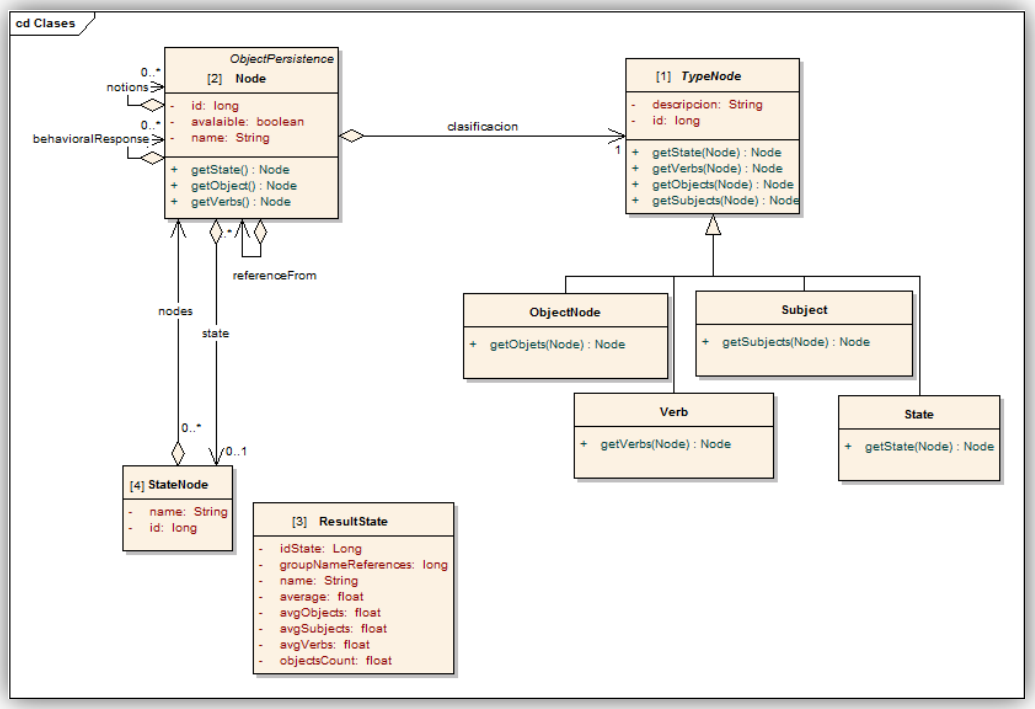

Figura 3.9 Modelo de Dominio

La clase Node [1], representa a cada uno de los símbolos pertenecientes a un LEL:

- Id: identificador del símbolo
- Available: si su valor es verdadero, significa que las referencias que provienen de este símbolo deben tenerse en cuenta, false en caso contrario.
- Behaviorals: lista de nodos de los símbolos a los que se referencia desde la propiedad “behavioral_response” del LEL.
- state: es el estado del símbolo estado al cual se asocio el símbolo.
- Name: nombre del símbolo.
- Notions: lista de nodos de los símbolos referenciados desde la propiedad “notions” del LEL.
- Classification: se utiliza para distinguir el tipo de símbolo. Esta debe ser una especificación de la clase TypeNode [2], la cual puede ser: “verb (verbo)”, “state (estado)”, “objectNode (objecto)” o “subject (sujeto)”.

Mientras que la clase **StateNode**, se utiliza para agrupar los símbolos una vez que la elección de cómo agruparlos fue realizada. Cada instancia toma el nombre e identificador del símbolo de tipo estado que se encuentra en el grupo (solo existe un símbolo estado en cada grupo de símbolos).

La clase **ResultState** [3] representa los datos retornados por la ejecución de la estrategia de identificación de características transversales (estos se calculan en la capa algorítmica). Cada instancia de esta clase, tiene el mismo identificador (id) que el símbolo estado que agrupaba al conjunto de símbolos.

Además de los objetos de dominio, esta capa contiene clases helpers que se usan para transformar datos, estos son:

- **NodeDTOConverter**: se utiliza al momento de pasar los símbolos retornados por la capa algorítmica a la capa de servicios. Este transforma una lista de NodesDTO a Nodes.
- **NodeConverter**: se usa para transformar los nodes usados en la capa de servicio y presentación a NodesDTO que utiliza la capa algorítmica.
- **ReferenceCountingReportConverter**: transforma los ReferenceCountingReportDto usados en la capa algorítmica en ResultState utilizados por la capa de servicios.

3.2.4 Capa de Integración

Esta capa es la encargada de transformar los objetos Java a Lua, invocar y ejecutar las funciones Lua (implementadas en la capa algorítmica) y finalmente transformar los datos Lua retornados a objetos Java. Para esto, JnLua provee una secuencia de seis acciones a realizar:

1. Lo primero que se debe hacer al trabajar con JnLua consiste en crear una instancia de la clase LuaState. Esta clase es el núcleo de JnLua. Y cada instancia de esta clase representa un bloque de ejecución de código Lua, similar al puntero de ejecución que provee el lenguaje C.

2. Una vez creada la instancia se le debe pasar el bloque de código que contiene a la función a ejecutar, esto se logra invocando al método **load**() de la instancia, al que se le pasa un String con el nombre del bloque de código a ejecutar y el bloque de código a ejecutar en forma de String. Este bloque de código, permite la ejecución tanto de código Lua como código Java. Así por ejemplo, el String “function add(a, b) return a + b end“ es un ejemplo de bloque valido. Una vez definida la función, se debe ejecutar la función **call** con los parámetros 0 (cero) y 0 (cero). Esta función, con los parámetros en cero, realiza una verificación de que el código ingresado tiene la estructura válida para ser ejecutado.

3. El tercer paso consiste en preparar la función definida antes para su ejecución, esto significa, que se debe pasar el nombre de la función a ejecutar (esta debe estar dentro del bloque de código incluido en el punto 2) e incluir en la instancia de LuaState los parámetros que necesitemos en la ejecución de la función. Para realizar la primera de estas dos tareas se debe utilizar la funcion **getGlobal**() que recibe como parámetro un String con el nombre de la función a ejecutar. Para incluir los parámetros en la ejecución del código hibrido, se utiliza la función **Push_{tipo_de_dato}**, siendo tipo de dato alguno de los tipos primitivos de Java, o **JavaObject** para representar a cualquier subclase de Object. Así por ejemplo **pushJavaObject**() será la función para incluir objetos o **pushInteger** para pasar tipos primitivos int, un caso particular son los String que si bien no son datos Java primitivos, tiene su propio método que es **pushString**(). Cualquiera de las funciones push, recibe dos parámetros, uno con el parámetro propiamente dicho y otro con la posición que este va a ocupar en la ejecución.

4. El siguiente paso consiste en ejecutar la función, esto se realiza mediante la función **call**(), esta recibe como parámetros la cantidad de parámetros de entrada que debe tener la función a ejecutar definida en 3 y la cantidad de parámetros de salida esperados. Los resultados quedan almacenados en la instancia de **LuaState**.

5. Se deben recuperar los resultados de la instancia de **LuaState**, mediante la función **getResult**.

6. Como último paso y sin importar si existieron problemas, se deben liberar los recursos utilizados por la instancia de **LuaState**, para lo cual simplemente se debe ejecutar el método **close**() de **LuaState**.

En TICC fue necesario desarrollar dos bloques de código para invocar funciones Lua, en ambos casos puede verse la convivencia de código Java y código Lua.

El primer bloque invoca una función que parsea un LEL. Esta función recibe como parámetro un String con el Path del archivo, ejecuta la función Lua que parsea el LEL y retorna una lista de nodos Lua que son transformados a NodesDTO Java. El bloque de código se ve en la figura 3.10.

El segundo bloque, transforma Nodes Java a Nodes Lua, ejecuta la función que determina los grupos que son características transversales y transforma los resultados Lua en

```
function parse(filePath)
  List = java.require('java.util.ArrayList')
  NodeDTO = lava.require('ar.edu.unlp.info.model.dto.NodeDTO')
  lista = List:new()
  nodo = NodeDTO:new() " +
  nodosProcesados = parseFile(filePath)
  for i,node in ipairs(nodosProcesados) do
    nuevoNodo = NodeDTO:new()
    nuevoNodo:setId(node.id)
    nuevoNodo:setName(node.name)
    nuevoNodo:setClassification(node.classification)
    nuevoNodo:setState(node.category[1])
    nuevoNodo:setIdState(node.idCategory)
    nuevoNodo:setAvalaible(node.available)
    for j,nodeIndex in ipairs(node.behavioral) do
      nuevoNodo:addNodeToBehaviorals(nodeIndex)
    end
    for j,nodeIndex in ipairs(node.notions) do
      nuevoNodo:addNodeToNotions(nodeIndex)
    end
    for j,insideNode in ipairs(node.referenceFrom) do
      iNode = NodeDTO:new()
      iNode:setId(insideNode.id)
      iNode:setClassification(insideNode.classification)
      iNode:setState(insideNode.category[1])
      iNode:setIdState(insideNode.idCategory)
      iNode:setName(insideNode.name)
      iNode:setAvalaible(insideNode.available)
      nuevoNodo:addNodeToReferenceFrom(iNode)
  end
  lista:add(nuevoNodo)
end
return lista
end
```

Figura 3.10 Transformación de datos Java en datos Lua

3.2.5 Capa Algorítmica

La funcionalidad que provee esta capa es la de llevar a cabo la estrategia de detección de características transversales. Su implementación fue realizada completamente en lenguaje Lua, por lo que fue necesario transformar parte del modelo de datos existente en Java a lenguaje Lua. A diferencia de Java, Lua es un lenguaje procedural, por lo que el modelo fue implementado usando estructuras al estilo de C. Un ejemplo de esto es la implementación de la clase Node (ver figura 3.12).

La funcionalidad base de TICC fue dividida en dos funciones principales, el parseo del LEL y la estrategia de identificación de las características transversales. Esta separación se debe principalmente a tres factores, una correcta separación modular de las distintas funcionalidades base, la independencia de la aplicación web y a que la ejecución de las funciones en TICC requiere de interacción con el usuario.

```
function parseFile(filePath)
	for i,node in ipairs(xfile) do
		if (node:tag() == "lexicon") then
			local lexicoNode=Node:new()
			lexicoNode.available = true
			local id = node:find("id_lexicon")
			if id then
				lexicoNode.id= id[1] --Set node's id
			end
			local name = node:find("name")
			if name then
				lexicoNode.name = name[1] --Set node's name
			end
			local classification = node:find("classification")
			if classification then
				lexicoNode.classification = classification[1] -- Set nodes'classification
				if (classification[1] == 'State') then
					nameCurrentState = name
					idCurrentState = lexicoNode.id
				end
			end
			if nameCurrentState then
				lexicoNode.category= nameCurrentState
				lexicoNode.idCategory = idCurrentState
			end
			local notions = node:find("notion")
			if notions then
				for n,notion in ipairs(notions) do
					if (type(notion)=="table" and notion:tag() == "lexicon_link")
then
						lexicoNode:addNotion(notion.target)		--Add
lexicon's id. (atributte: target)
					end
				end
			end
			local behaviors = node:find("behavioral_response")
			if behaviors then
				for j,behavior in ipairs(behaviors) do
					if (type(behavior)=="table" and behavior:tag() ==
"lexicon_link") then
						lexicoNode:addBehavioral(behavior.target) --Add
lexicon's id. (atributte: target)
					end
				end
			end
			table.insert(lexicos,lexicoNode) -
		end
	end
	for i,node in ipairs(listal) do
		for j,node2 in ipairs(listaJ) do
			if (node.id ~= node2.id) then
				for k,id_node in ipairs(node2.behavioral) do
					if node.id == id_node then
						node:addReferenceFrom(node2)
				…
		…
	return lexicos
end
```

Figura 3.11 Función de parseo de LEL

```
-- struct of Node
Node = {
	id = nil,
	name = nil,
	category = nil, -- State
	idCategory = nil, -- Id State
	notions = {},
	behavioral = {},
	referenceFrom = {},
	available= true,
	classification = nil -- subject, object, verb, etc..
}
-- create an instance of Node
function Node:new(o)
	o = o or {}
	setmetatable(o,self)
	self.__index = self
	return o
end
-- add behavioral at the last of the array
function Node:addBehavioral(e)
	local index = #self.behavioral
	if index == 0  then -- if the array doesn't exist or is empty
		index = index + 1
		self.behavioral = {}
	end
	table.insert(self.behavioral,e)
end
-- add an edge at the last of the array
function Node:addNotion(e)
	local index = #self.notions
	if(index == 0) then -- if the array doesn't exist or is empty
		index = index + 1
		self.notions = {}
	end
	table.insert(self.notions,e)
end
-- remove from the edge's list the edge in the parameter position
function Node:removeNotionAt(i)
	self.notions[i] = nil
end

-- remove from the edge's list the edge in the parameter position
function Node:removeBehavioralAt(i)
	self.behavioral[i] = nil
end
```

Figura 3.12 Implementación de Node en lenguaje Lua

La función que realiza el parseo se denominó "**parseFile**" (Figura 3.11), esta recibe como entrada un String con la ruta del LEL a parsearse y retorna una lista de Nodes. La implementación de la función puede dividirse en tres partes:

1. Cargar el LEL cuya ruta fue pasada como parámetro

   Se encarga de cargar en memoria el archivo xml. Para esto se utilizo una función de luaXml llamada xml.load (filePath) perteneciente a la herramienta LuaRocks [LuaRocks 2011], al cual se le pasa el path absoluto de donde se encuentra el archivo xml.

2. Crear la estructura de símbolos provenientes del LEL

LuaXml provee una función para leer los archivos XMLs, la cual permite leer las estructuras superiores que se encuentren en el archivo. En este caso, se sabe que un símbolo perteneciente al LEL se lo distingue con la etiqueta **lexicon** y tiene la estructura que se ve en la figura 3.13. Luego, para cada **lexicon**, se crea un **Node** (ver figura 3.12) copiándose las propiedades id_lexicon, name y classification y asignándoselos a las propiedades id, name y classification respectivamente. Luego, para cada etiqueta **lexicon_link** que se encuentre dentro de las etiquetas **behavioral_response** y **notion**, se agregan los target de cada uno a su respectiva lista dentro del node, estos target, tienen correspondencia directa con el identificador del símbolo al que referencian dentro del LEL.

```
<lexicon>
        <id_lexicon>1</id_lexicon>
        <name> Signed </name>
        <notion>
                Situation where the client has decided to open an account and the bank is
                ready to open it
        </notion>
        <behavioral_response>
                The <lexicon_link target="2">bank</lexicon_link>
                <lexicon_link target="5">open the account</lexicon_link>and the
                <lexicon_link target="3">client</lexicon_link>is the owner of a
                <lexicon_link target="6">blocked</lexicon_link>
                <lexicon_link target="4">account</lexicon_link>.
        </behavioral_response>
        <classification> State </classification>
</lexicon>
```

Figura 3.13 Ejemplo de estructura de un símbolo en el LEL

3. Determinación de las referencias de un nodo

La estrategia de identificación de características transversales tiene en cuenta las referencia recibidas por cada símbolo y no la cantidad de grupos que referencia el símbolo (como nos provee el LEL), por esto, para facilitar los cálculos se creó una propiedad especial llamada **referenceFrom** (referencias desde), la cual es una lista de referencias a aquellos símbolos, que en su propiedad **behavioral** contienen el identificador del símbolo original

La segunda función implementada en esta capa es la que identifica las características transversales. Esta función recibe como parámetro de entrada la lista de Nodes generadas por la función **parseFile** y retorna una lista de **ReferenceCountingReport** ordenada de mayor probabilidad de ser característica transversal a menor.

La estructura de ReferenceCoutingReport, tiene las siguientes propiedades que son necesarias para poder comparar e identificar los diferentes grupos, esto son:

- Id : El identificador del grupo
- Name: El nombre del grupo
- GroupReferences: El promedio de referencias que reciben sus nodos

- GroupNameReferences La cantidad de grupos desde los que existen referencias
- AvgObjects: El promedio de referencias hacia los objetos del grupo
- AvgSubjects: El promedio de referencias hacia los sujetos del grupo
- AvgVerbs: El promedio de referencia hacia los verbos del grupo

Esta función tiene tres partes bien definidas:

1. Crear una lista de nodos agrupadas por estado
2. Realizar los cálculos para cada grupo de símbolos.
3. Ordenar los grupos de símbolos, poniendo primero a aquellos que tienen mayor probabilidad de ser característica transversal.

La primer parte de la función consiste en agrupar los Nodes recibidos por el estado que tienen asociado. Si bien el grupo de símbolos toma el nombre del símbolo estado, la estructura de un símbolo estado no es diferente del resto de los símbolos, por esto fue necesario crear una estructura capaz de representar y contener al grupo de símbolos. Con este objetivo fue que se creó la estructura auxiliar **StateReferences**, que tiene como propiedades al identificador del símbolo estado que identifica al grupo, el nombre del símbolo estado y la lista de Nodes que integran el grupo (incluido el símbolo estado). En la función, se recorre el listado de nodos, creando un **StateNode** por cada símbolo Estado que se encuentre en el dominio y agregando los símbolos al grupo que está asociado. El código sobre esta primera parte de la función, se encuentra en la función **createGroups**, la cual puede verse en la siguiente figura.

```
function createGroups(nodeList)
	local result = {}
	for i,node in ipairs(nodeList) do
		local stateNode = getState(node.idCategory,result)
		if(stateNode==nil) then
			stateNode = StateReferences:new()
			stateNode.id = node.idCategory
			stateNode.name = node.category
			table.insert(result,stateNode)
		end
		stateNode:add(node)
	end
	return result
end
function getState(idCategory, grouplist)
	for i,group in ipairs(grouplist) do
		if(group.id == idCategory) then
			return group
		end
	end
	return nil
end
```

Figura 3.14 Función de TICC para agrupar símbolos por estado

La segunda parte del algoritmo retorna una lista de **ReferenceCoutingReport** y es donde se realizan los cálculos de referencias recibidas sobre cada grupo de símbolos, Esta parte del algoritmo consiste en recorrer los **StateReferences** creados en la primera parte, y por cada uno de ellos y para cada **Node** habilitado perteneciente al grupo, recorrer la propiedad **referenceFrom**, contando la cantidad de símbolos pertenecientes a otros grupos que referencian al símbolo, contabilizando que tipo de símbolo se está referenciando y la cantidad total de referencias.

Una vez recorrido todos los **Nodes** pertenecientes a un **StateReference**, se debe crear una variable de tipo **ReferenceCoutingReport**, al cual se le completaran las propiedades con datos pertenecientes al **StateReference** y con los datos obtenidos al recorrer los nodos.

La ultima parte de la función que obtiene las características transversales consiste en ordenar las los **referenceCountingReport** obtenidos antes. Estas variables deben ordenarse de forma que en los primeros lugares queden aquellos grupos que maximizan la dispersión y el enmarañamiento. Con este objetivo, la función divide los grupos de símbolos en dos listas: una lista con aquellos grupos de símbolos que son referenciados por más de un grupo de símbolos y otra lista con los grupos restantes. Luego, realiza un ordenamiento de burbuja para cada una de las listas, dándole un orden menor a aquellos grupos de símbolos que maximicen: cantidad de grupos de referencia multiplicada por cantidad de referencia general. En caso de empate, el orden lo define aquel grupo que sea referenciado por la mayor cantidad de grupos. Una vez hecho esto se une las listas y se retorna el resultado. El código que ordena la lista de **ReferenceCountingReport** (ver figura 3.15. y 3.16).

```
function sortConcerns(concerns)
	local groupsOfReport = doubleBubbleSort(getGroupsOfReport(concerns))
	-- create and sort the final list
	local result = {}
	-- in first place put the groups with more than one group reference
	for i,iconcern in ipairs(groupsOfReport[1]) do
		result[#result+1] = iconcern
	end
	-- in the last place put the groups with one group reference
	for i,iconcern in ipairs(groupsOfReport[2]) do
		result[#result+1] = iconcern
	end
	return result
end
function getGroupsOfReport(originalList)
	-- contains the list of states with one or zero reference
	local oneList = {}
	local moreThanOneList = {}
	if(originalList ~= nil) then
		for i,group in ipairs(originalList) do
				if(group.groupNameReferences > 1) then
					moreThanOneList[#moreThanOneList+1] = group
				else
					oneList[#oneList+1] = group
				end
		end
	end
	return {moreThanOneList, oneList}
end
```

Figura 3.15 Código de ordenación de ReferenceCountingReport (parte 1)

```
function doubleBubbleSort(nodeList)
	local moreThanOne = bubbleSort(nodeList[1])
	local one = bubbleSort(nodeList[2])
	return {moreThanOne,one}
end

function bubbleSort(concerns)
	local count = 0
	if(concerns~=nil)then
		count = #concerns
	end
	repeat
		local swapped = false
		count = count-1
		for i=1,count do
			if(lesserThan(concerns[i],concerns[i+1])) then
				local swap = concerns[i]
				concerns[i] = concerns[i+1]
				concerns[i+1] = swap
				swapped = true
			end
		end
	until (swapped==false)
	return concerns
end
-- return true when report1 is minor than report2, in other case return false
function lesserThan(report1, report2)
	if((
	(report1.groupNameReferences)*(report1.average))<=
	((report2.groupNameReferences)*(report2.average))
		) then
	if(((report1.groupNameReferences)*(report1.average))<
	 ((report2.groupNameReferences)*(report2.average))) then
	-- report1 < report2
	return true
	else
	-- report1 = report2
	return (report1.groupNameReferences < report2.groupNameReferences)
		end
	else
		return false
	end
end
```

Figura 3.16 Código de ordenación de ReferenceCountingReport (parte 2)

Finalmente a modo de poder testear las funciones Lua de forma independiente del resto de la aplicación, se creó una función Lua, la cual trabaja de forma similar a un UnitTest de Java, esta función recibe como parámetro el path local de un LEL e invoca a la función que parsea el LEL, y con los resultados invoca a la función que determina las características transversales. Estos resultados son comparados con los obtenidos a través de los cálculos manuales.

3.3 Instalación

Dada la arquitectura planteada en la herramienta, existen ciertos requerimientos y particularidades que deben cumplirse en la instalación de la herramienta:

1. La herramienta debe correr sobre el sistema Windows 7 o superior.

2. Se debe agregar una biblioteca binaria Lua 5.2 [Lua 2012], la cual debe ser agregada a la carpeta System32 de Windows. En una instalación por defecto de este sistema operativo, la biblioteca debería agregarse en C:\windows\system32.

3. Es necesario que se cree una variable de entorno llamada LUA_PATH. Esta debe apuntar a la carpeta que contiene los archivos LUA, LUAC y DLL de la herramienta.

4. La instalación del framework JnLua está dividida en dos partes, una parte se encuentra dentro de la herramienta, la otra parte se incluye en la carpeta que contiene al sistema operativo. En el caso más común esta biblioteca se debe copiar a la carpeta C:\windows.
5. Finalmente se debe desplegar el war "**ticc.war**" el cual contiene la aplicación. Para esto es necesario un servidor HTTP. El desarrollo y las pruebas de la herramienta se realizaron bajo un servidor Jetty, pero también puede usarse un servidor Tomcat ya que son compatibles.

3.4 Uso de TICC

La herramienta como se nombro anteriormente tiene como fin la identificación de características transversales, sin embargo, no tiene la capacidad de edición del LEL, el cual debe generarse con la herramienta C&L.

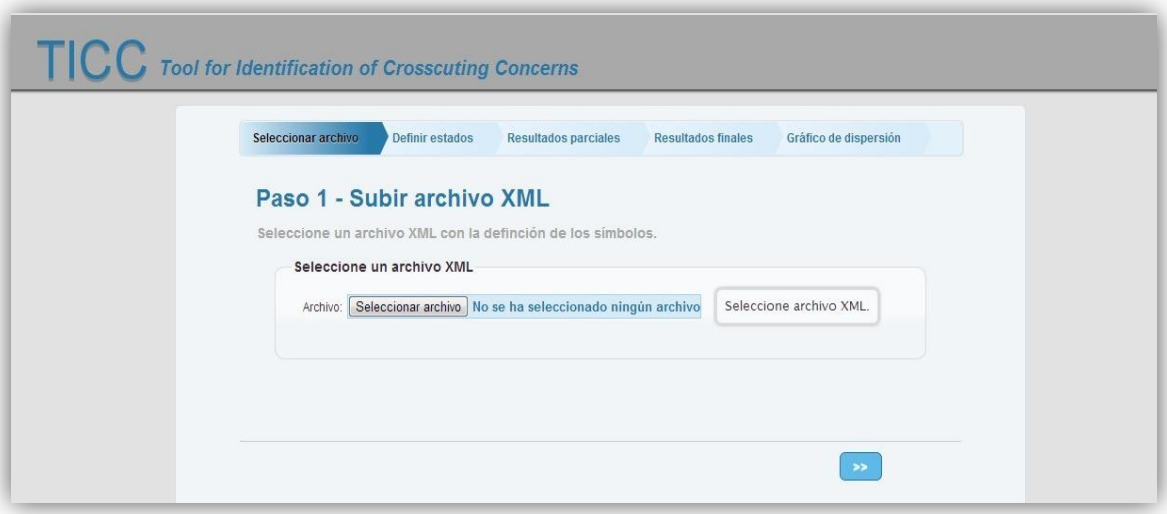

Figura 3.17 Pantalla inicial de TICC

Uno de los objetivos buscados en el desarrollo de la herramienta fue que esta fuera simple e intuitiva, por esto es que el usuario de TICC solo tiene que realizar tres operaciones antes de obtener los resultados, estas operaciones se realizan todas entre la primera y segunda pantalla.

La primera acción que se debe realizar es seleccionar el archivo que contiene el LEL (ver Figura 3.18), este debe haber sido generado por la herramienta C&L o tener el formato que se explico en capítulos anteriores.

Una vez seleccionado el archivo (ver figura 3.19), se continua cliqueando sobre el botón “>>” (siguiente).

```
<project>

<lexicon>
<id_lexicon>1</id_lexicon>
<name>client</name>
<notion>
        Person that opens an <lexicon_link target= 2> account </lexicon_link> with the bank.
</notion>
<behavioral_response>
        The client can deposit money into his <lexicon_link target= 2> account </lexicon_link>.
        The client can withdraw money from his <lexicon_link target= 2> account </lexicon_link>.
        The client can consult balance from his <lexicon_link target= 2> account t</lexicon_link>.
</behavioral_response>
<classification>subject</classification>
</lexicon>

<lexicon>
<id_lexicon>2</id_lexicon>
<name>account</name>
<notion>
        The account has a balance and the <lexicon_link target= 1> client </lexicon_link> can
        operate on it.
</notion>
<behavioral_response>
        The <lexicon_link target= 1> client </lexicon_link> can deposit money into his account.
        The <lexicon_link target= 1> client </lexicon_link> can withdraw money from his account.
        The <lexicon_link target= 1> client </lexicon_link> can consult balance from his account.
</behavioral_response>
<classification>object</classification>
</lexicon>
</project>
```

Figura 3.18 Formato del XML que recibe TICC

A continuación el usuario debe relacionar cada elemento con un estado, es importante aclarar que todos los elementos deben estar relacionados con un estado. Los símbolos pueden pertenecer a distintas categorías:

- Elementos
- Objetos
- Verbos

La pertenencia de los símbolos en cada grupo viene determinada dentro del archivo XML que fue seleccionado en el paso anterior.

Figura 3.19 Pantalla inicial para agrupar símbolos en estados

Para que sea intuitivo, la forma de agregar un elemento a un estado implementado fue que el elemento se arrastre de derecha a izquierda, hacia el estado al que se lo quiere agregar. La figura 3.19 muestra la pantalla de definición de estados cuando recién se accede a ella y la figura 3.20 muestra cuando todos a todos los elementos se les asigno un estado.

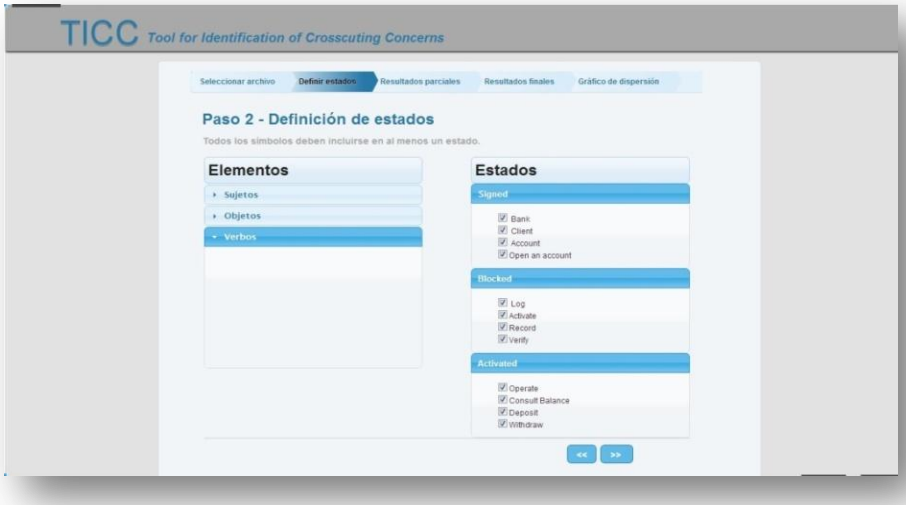

Figura 3.20 Pantalla inicial con todos los nodos agrupados e identificados

El usuario puede identificar los símbolos principales dentro del dominio de la aplicación, los cuales no deben ser considerados para contar las referencias que ellos reciben desde otros símbolos, pero si deben ser considerados para contar las referencias hacia otros símbolos. Esta identificación de los símbolos pertenecientes al nucleó de un dominio, es la segunda acción que debe realizar el usuario, esto lo realiza tildando/des tildado el checkbox que tiene cada elemento a su izquierda, si no tiene el check significa que es un símbolo del nucleó del dominio y la cantidad de símbolos que lo apuntan no se tendrán en cuenta. Vale la pena

aclarar, que aunque un símbolo sea considerado del núcleo del dominio y no tenga el check seleccionado, igual debe ser arrastrado hacia su estado porque podría variar los resultados obtenidos.

Cabe destacar también que las dos acciones que debe realizar el usuario en esta pantalla no necesariamente debe realizarlas en forma secuencial, debida a que también las pueden realizar en forma paralela.

Una vez que los símbolos fueron configurados, es decir, una vez que cada símbolo fue relacionado con un estado y fueron deshabilitados los símbolos necesarios, es posible desencadenar el cálculo de referencias para poder identificar las características transversales ordenadas por posibilidad de ser consideradas como tales, desde la más probable hasta la menos probable. El orden es determinado a través del producto entre el número de grupos que los referencian y los promedios generales.

Además del orden, se pueden comparar los promedios de referencias hacia distintas las clases de símbolos desde cada uno de los estados. Esto se puede ver en la figura 3.21.

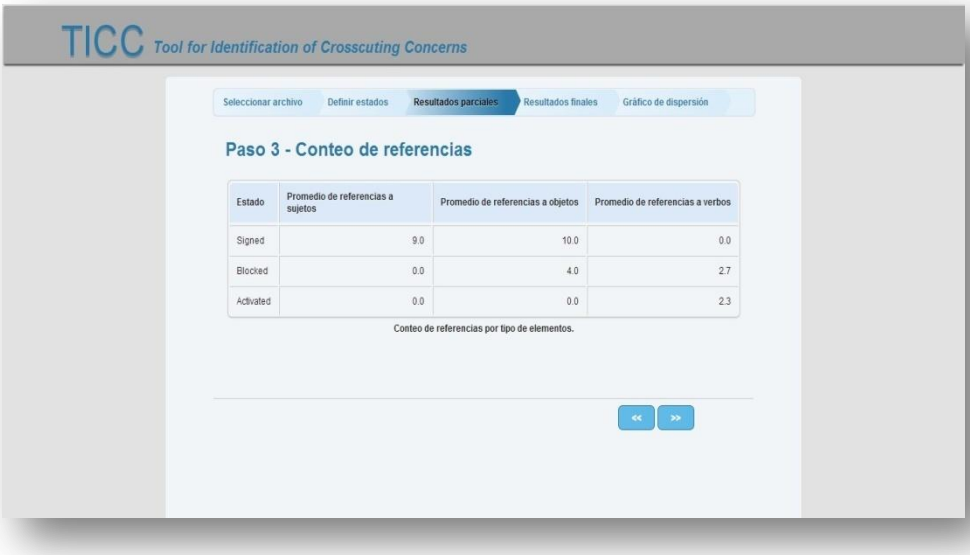

Figura 3.21 Se ordenan los estados, cuyos símbolos concentran la mayor cantidad de referencias

En la siguiente pantalla, se deja en evidencia la cantidad de estados desde los que se referencia a cada estado, dato que es fundamental para determinar la posibilidad que tiene de ser característica transversal, así como también un promedio general de referencias recibidas. Además, se agrego la posibilidad de obtener un detalle de los símbolos que se agruparon en cada uno de los estados junto con la cantidad de referencias obtenidas desde cada uno de los restantes estados. Esto se puede ver en la figura 3.22.

En la figura 3.22 también se puede ver por ejemplo que el símbolo “Bank” es muy referenciado desde el resto de los estados del dominio, esto se debe a que es un símbolo del dominio que no fue considerado como tal al inicio del uso de la herramienta. Esto claramente afecta el resultado obtenido, por lo que es necesario volver a la pantalla donde se agrupan los símbolos y deshabilitarlo, esto puede hacerse mediante los botones de navegación.

Figura 3.22 Detalle de uno de los estados

Finalmente, se agrego un gráfico XY, el cual muestra la dispersión de los grupos en función de dos variables. Esto se puede apreciar en la figura 3.23.

En el capítulo siguiente, con la ayuda de distintos dominios se probara, que los resultados esperados por el algoritmo de detección y los resultados arrojados por la herramienta son consistentes.

Figura 3.23 Gráfico XY de dispersión de estados.

Capítulo 4

4. Validación de la Herramienta

En este capítulo se realiza una comparación entre los resultados que arroja la herramienta TICC, con los obtenidos realizando los cálculos de la estrategia de detección temprana de características transversales usando LEL [Antonelli 2012]. Para realizar esta comparación se van a utilizar 3 dominios: un dominio bancario, el dominio de un sistema anti evasión de impuestos y el dominio de un portal web.

Para comparar los diferentes dominios el capítulo fue dividido en 3 secciones, donde en cada sección muestra un dominio distinto con diferentes características. En cada dominio se realizara una descripción de los símbolos, un ranqueo de grupos y un análisis final de los resultados arrojados por la herramienta TICC.

4.1 Dominio Bancario

El dominio se basa en un banco el cual permite que sus clientes operen con sus cuentas. Sobre estas cuentas se pueden realizar tres operaciones, estas son: extraer dinero, depositar dinero y consultar el saldo. Si bien estas operaciones son sencillas, el banco necesita proveer cierto nivel de seguridad en las mismas, es por ello que ante cada una de estas operaciones, el banco debe verificar la identidad del cliente. Luego de verificada la identidad y de realizada la operación, el banco mantiene un registro de la operación para posterior revisión de auditoría si fuera necesario.

Tabla 4.1 Símbolos correspondientes a la aplicación bancaria

Categoría	Símbolos
Sujetos	Banco
	Cliente
Objetos	Cuenta
	Log
Verbos	Abrir una Cuenta
	Activar una Cuenta
	Registrar
	Corroborar
	Operar
	Depositar
	Extraer
	Consultar Saldo
Estados	Solicitada
	Bloqueada
	Activada

Además de las operaciones para operar con una cuenta existen las operaciones necesarias para abrir una cuenta. La apertura de una cuenta no es inmediata, sino que hay distintos pasos que se deben seguir hasta que la cuenta esta totalmente operativa. En primer lugar el cliente solicita la apertura de la cuenta, por lo cual, el primer estado por el que pasa la cuenta (en realidad no es una cuenta porque en esta instancia no está creada), es el de solicitada (Signed). Luego, en algún momento entre todas las tareas que tiene que realizar el banco procesa la solicitud de creación de cuenta, y efectivamente crea una cuenta, sin embargo, esta cuenta no está disponible para ser operada por el cliente. Esta cuenta se encuentra en estado bloqueada (bloqued / recently created) hasta que ciertas verificaciones de deudas y solvencias sean realizadas. Finalmente luego de realizadas estas verificaciones, la cuenta queda finalmente activada (activated).

La tabla 4.1 muestra los símbolos que deben ser descriptos para capturar la esencia del contexto de la aplicación.

4.1.1 Análisis de Símbolos

El primer paso descripto en la estrategia de identificación de características transversales usando LEL consiste en la creación del LEL, el cual se realiza usando la herramienta C&L. Mientras que el segundo paso consiste en identificar los estados por los cuales los símbolos de la aplicación pueden atravesar. En este caso, el símbolo más relevante del dominio es cuenta. Y los estados asociados son solicitada, bloqueada y activada.

Este paso, es muy importante, ya que las características transversales se calculan a nivel de estados. En la herramienta, este paso se realiza automáticamente ya que los estados se muestran en la parte derecha de la pantalla, mientras que los símbolos cuya categoría es objeto, sujeto o verbo pueden verse en la parte izquierda de la pantalla.

La figura 4.1 muestra de forma gráfica las relaciones que tienen cada uno de los símbolos del domino hacia otros símbolos.

En una primera ejecución de la herramienta sobre este dominio, así como para los cálculos manuales se tendrán en cuenta a todos los símbolos del LEL (en la herramienta todos los símbolos estarán tildados), esto puede verse en la siguiente figura. Los símbolos pueden no tenerse en cuenta, si se les quita la selección junto a cada símbolo.

El tercer paso consiste en vincular cada uno de los símbolos del LEL (de las categorías sujeto, objeto y verbo) con alguno de los estados determinados en el paso anterior. En la tabla 4.2 se encuentran cada uno de los símbolos con su correspondiente estado. El estado solicitada representa los símbolos que intervienen para solicitar una cuenta. Luego de solicitar una cuenta se obtiene una cuenta bloqueada, en donde no se pueden realizar operaciones más que activar, sin embargo se pueden intentar operar con la misma y es por eso que tienen sentido las operaciones registrar y corroborar, como así también el log.

Sobre el estado activado, se ubican las tres operaciones que tienen sentido realizar con la cuenta: depositar, extraer y consultar el saldo. También se ubica el símbolo operar para generalizar los otros tres.

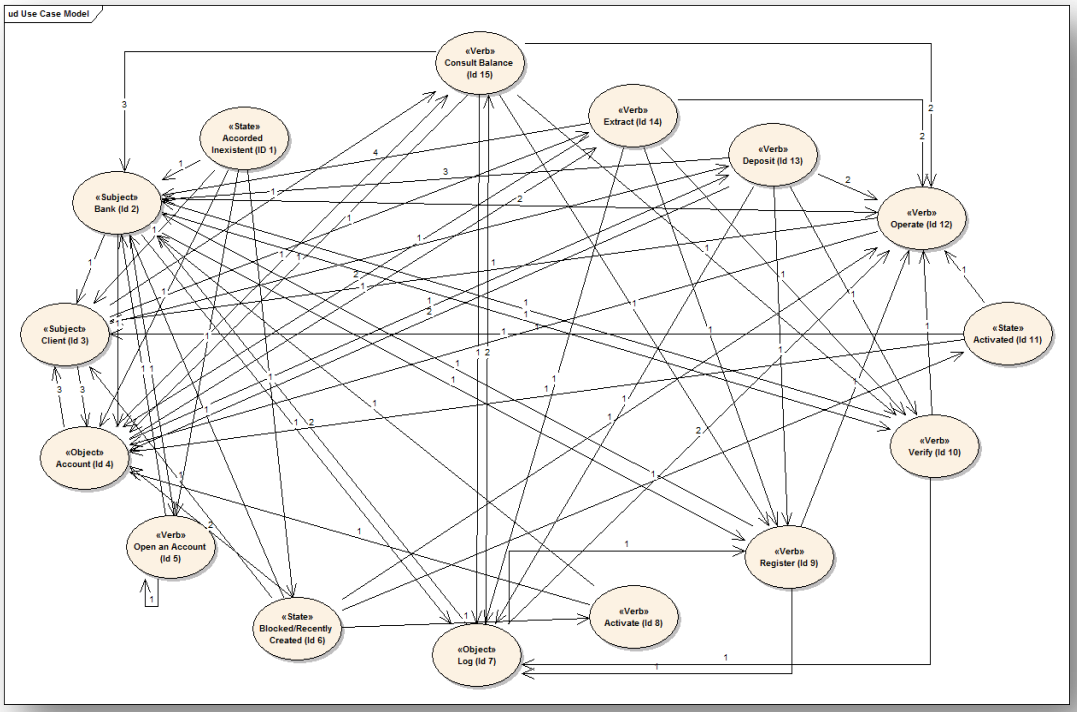

Figura 4.1 Símbolos y Referencias del Dominio del Banco

Figura 4.2 Definición de Estados y pertenencia de los símbolos

Contando las referencias que existen entre símbolos pertenecientes a distintos estados se puede conocer el acoplamiento entre módulos. Sobre la cantidad obtenida no se tiene en cuenta la cantidad absoluta sino el promedio así como desde la cantidad de estados desde los que se le hace referencia. Vale la pena también aclarar que solo se cuentan las referencias desde los impactos de los otros símbolos.

Tabla 4.2 Agrupamiento de símbolos del dominio bancario

Estado	Sujeto	Objeto	Verbo
Solicitada	Banco	Cuenta	abrir una cuenta
	Cliente		
Bloqueada		Log	Activar
			Registrar
			corroborar
Activada			Operar
			depositar
			Extraer
			consultar el saldo

Tabla 4.3 Conteo de referencias entre los distintos símbolos del dominio banco

		Solicitada					Bloqueada					Activada				
		Solicitada	Banco	Cliente	Cuenta	Abrir una cuenta	Bloqueada	Log	Activar	Registrar	Corroborar	Activar	Operar	Depositar	Extraer	Consultar el Saldo
Solicitada	Solicitada						1									
	Banco							1		1	1		1			
	Cliente													1	1	
	Cuenta													1	1	
	Abrir una Cuenta															
Bloqueada	Bloqueada		1	1	2							1	1			
	Log															
	Activar		1		1											
	Registrar		1										1			
	Corroborar		1	1												
Activada	Activar			1	1											
	Operar			1	1											
	Depositar		3		2			1		1	1					
	Extraer		4		2			1		1	1					
	Consultar el Saldo		3		1			1		1	1					

La tabla 4.3 muestra las referencias que cada símbolo recibe agrupándolos por estado. La tabla 4.4 presenta un detalle a nivel de estado de los datos calculados a partir de la tabla anterior.

Tabla 4.4 Conteo de Referencias Detallado

	Grupos que los referencian	Promedio general de referencias	Promedio de Referencias a Sujetos	Promedio de Referencias a Objetos	Promedio de Referencias a Verbos
Solicitada	2	6.0	10	10	0
Bloqueada	2	2.6	0	4	2.67
Activada	2	2.6	0	0	3.0

Para explicar la tabla 4.4 se va a tomar como ejemplo el grupo solicitada (correspondiente a la primera fila), la primera columna grupos que lo referencian, posee el valor 2, esto quiere decir que existen al menos 2 referencias desde símbolos que pertenecen a estados distintos que solicitada (y de estados distintos entre ellos). Luego, la columna Promedio general de referencias da una medida de la cantidad de referencias que el grupo posee dividido la cantidad de símbolos del grupo. Las tres columnas finales también presentan promedios pero acotados a una categoría específica de símbolo: sujeto, objeto y verbo. La última columna promedio de referencias a verbos posee el valor 0 (cero) debido a que el grupo solicitada posee un solo verbo el cual es "abrir una cuenta" y este no es referenciado por ningún otro símbolo, por esto es que tiene el valor 0. La columna promedio general de referencias a objetos posee un valor 10, porque el grupo solicitada tiene un solo objeto "cuenta", el cual posee 3 referencias desde el grupo bloqueada y 7 referencias desde el grupo actividad, esto da un total de 10 referencias, y como se dijo existe un solo objeto en el grupo, por lo que su promedio es 10. Por su parte, la columna promedio de referencias a sujetos posee el valor 10, porque solicitada tiene 2 sujetos (banco y cliente); banco posee 6 referencias desde bloqueada y 10 referencias desde actividad; mientras que cliente posee 2 referencias desde bloqueada y 2 referencias desde activada. El total de referencias para banco es 16 y el total de referencias para cliente es 4, la suma de ambos es 20 por lo que el promedio general para dos símbolos es 10. Finalmente el promedio general para solicitada es de 6, puesto que la cantidad total de referencias del grupo es de 30: 10 referencias a objetos, 20 referencias a sujetos; y el grupo posee 5 símbolos: solicitada, banco, cliente, cuenta y abrir cuenta, por lo cual el cociente 30 sobre 5 da como resultado 6.

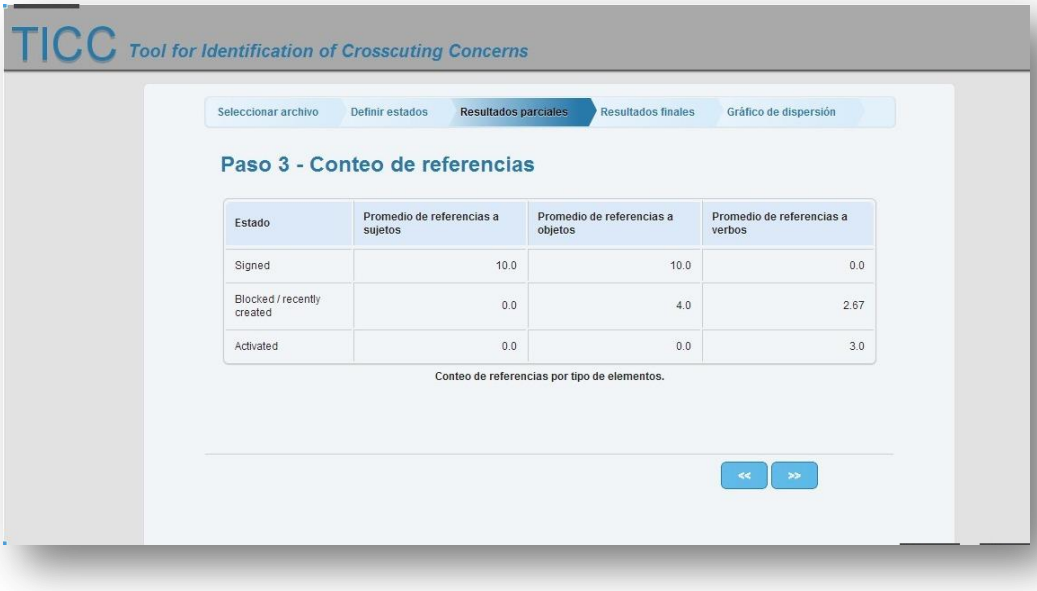

Figura 4.3 Promedio de Referencias a clases de nodos del Banco

Los mismos datos que presenta la tabla 4.4 son presentados en dos pantallas distintas de TICC. La figura 4.3 presenta las tres últimas columnas de la tabla, es decir, muestra los promedios generales de referencias hacia objetos, sujetos y verbos. Mientras que las columnas de la figura 4.4 presenta la cantidad de símbolos de distintos estados desde los que se le hace referencia junto con el promedio general. Vale recordar que en esta pantalla también se puede ver el detalle sobre la cantidad de referencias que recibió cada símbolo de cada uno de los grupos.

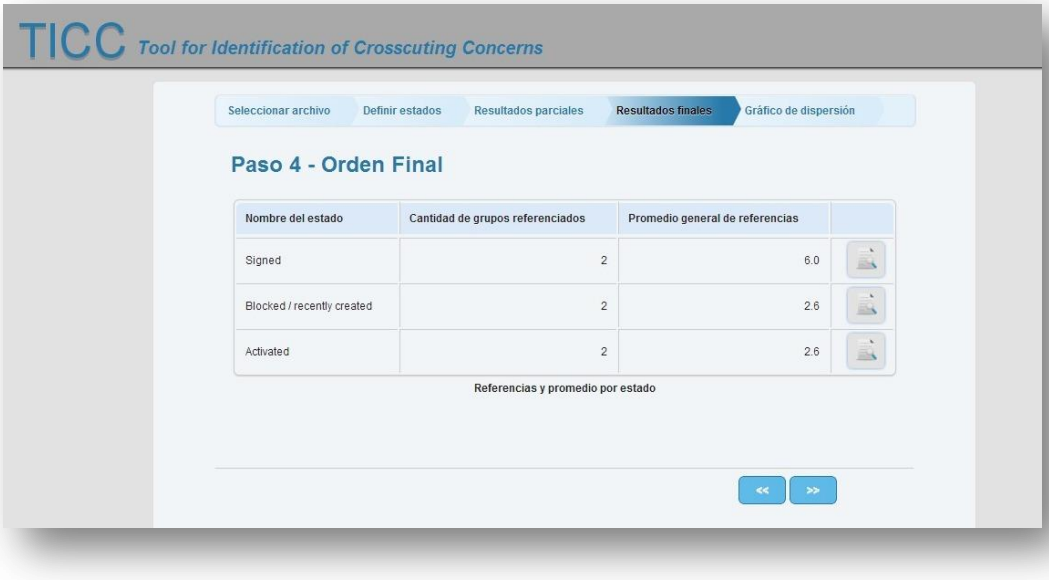

Figura 4.4 Orden final de los estados del banco

4.1.2 Ranqueo de Grupos

Al igual que en la estrategia, el orden presentado por los estados en la herramienta es determinado por las dos características que determinan la presencia de características transversales, que son: dispersión (*scattering*) y enmarañamiento (*tangling*).

Para esto, los estados candidatos:

I. Deben poseer muchos grupos que lo referencien.
II. Deben poseer un alto promedio de referencias.

Mientras que (i) indica cuando disperso esta el grupo, (ii) indica cuan enmarañado esta el grupo. Por esto es que es importante que ambas variables se maximicen.

El dominio analizado posee solo tres estados y todos tienen la misma cantidad de grupos que los referencien (ver primera columna en la figura 4.4 y la tabla 4.4), pero varían en el promedio general de referencias, siendo solicitada el que tiene el mayor promedio. Por esto es que el primer grupo candidato a ser característica transversal es Solicitada, seguida por bloqueada y finalmente Activada.

4.1.3 Análisis Final

La estrategia como último paso propone un análisis de los símbolos pertenecientes a los estados candidatos a ser característica transversal, esto se debe a que en todo dominio existen símbolos que son considerados “core” o principales. Estos símbolos en general tienen muchas referencias y

pueden generar que todo el grupo lo tenga, alterando el resultado obtenido por la estrategia.

Como este análisis varía en función de cada dominio, no se encuentra automatizado por la herramienta La herramienta si permite la posibilidad de hacer o re hacer los cálculos sin considerar a algunos símbolos para que no afecten los resultados. Esto en la herramienta se realiza destilando el check a la izquierda de cada símbolo al momento de agruparlos. Es decir, podrían volverse a realizar los cálculos, sin tener en cuenta a aquellos símbolos principales del dominio.

A modo de finalizar el ejemplo, el grupo bloqueada es el único que debe ser considerado característica transversal. Si bien los tres grupos poseen el mismo número de grupos que los referencian, el grupo solicitada tiene un alto valor en promedio general de referencias, las cuales logra a través de símbolos principales, por lo cual no debe ser considerado como característica transversal. El grupo activada que se encuentra en tercer lugar en función del promedio general de referencias, también posee símbolos principales y no debe ser considerado. Por lo cual, el grupo bloqueada es el único a considerarse característica transversal.

4.2 Sistema Anti Evasión de Impuestos

La descripción del dominio es la siguiente: cuando un ciudadano no cumple con sus deberes impositivos el organismo demandante del gobierno realiza las acciones necesarias para obligar al deudor a cumplir con sus obligaciones. El curso normal de acciones implica:

- Establecimiento de deuda
- Procedimiento a nivel administrativo.
- Procedimiento a nivel judicial.
- Procedimiento a nivel cautelar.
- Cancelación de la deuda.

Los cinco pasos indicados anteriormente son actividades complejas las cuales incluyen las siguientes tareas:

i. Establecimiento de la deuda: situación en la cual el contribuyente ha incurrido en una deuda y el ha rechazado o desaprovechado todas las oportunidades posteriores que se le brindaron para ponerse al día.

ii. Procedimientos a nivel administrativo: en esta etapa, se realiza un análisis costo/beneficio para determinar si el esfuerzo de reclamar la deuda es provechoso o no. Si la relación coste / beneficio resulta positiva, también es necesario determinar si es una deuda critica o no (ya sea por el monto, por las características del deudor, etc.) con el fin de tomar las medidas preventivas.

iii. Procedimientos a nivel judicial: consiste básicamente en cuatro etapas: hacer la presentación en un juzgado, exhortarlo a pagar, tomar las medidas preventivas y finalmente lograr la sentencia. Dado que las medidas preventivas si pudieron haber llevado a cabo el paso anterior, la secuencia de acciones no es única.

iv. Procedimientos a nivel cautelar: básicamente consiste en esperar a que el deudor voluntariamente y espontáneamente se presente a pagar su deuda. Se espera a que el deudor lo realice, puesto que previo a este paso, se tomaron todas las medidas legales para cerrarle las puertas a cualquier acto legal o administrativo, por lo cual, si el mismo se ve en esta necesidad, no le quedara otra opción más que saldar su deuda. Sin embargo, también hay alternativas, por ejemplo, si el deudor poseyera alguna propiedad (o bien mueble o inmueble) el organismo demandante podría hacer uso de los mismos para cubrir el monto adecuado.

v. Cancelación de la deuda: es la etapa final en la cual, la deuda es recuperada y las medidas cautelares son anuladas.

Si bien el flujo normal y esperado se corresponde con los cinco pasos anunciados y descriptos previamente, pueden presentarse situaciones las cuales pueden alterar el flujo normal de eventos. Las situaciones son las siguientes:

vi. Deuda reclamada por error: ocurre cuando se reclama una deuda, pero en realidad es un error porque la misma no existe, es decir, el deudor realmente no fallo en pagar sus obligaciones. Es un simple error administrativo. Sin embargo, aunque la deuda no exista, dado que el deudor tuvo oportunidades previas para aclarar la situación de todas formas deberá abonar los gastos administrativos que se ocasionaron.

vii. Plan de facilidad de pago: es un mecanismo que permite al deudor pagar su deuda en pagos mensuales. Es un compromiso que asume y puede desencadenar el pago completo de la deuda, o también podría suceder que el deudor abandone el plan de facilidades por lo cual, incurriría nuevamente en una deuda.

viii. Subasta: situación en la cual el deudor posee bienes (muebles o inmuebles) por lo cual, se subastan los mismos con el fin de recuperar la deuda.

ix. Medidas preventivas administrativas: situación en la cual no es beneficioso el llevar un acto judicial con el fin de reclamar la deuda y obtener su pago. Por lo cual, solo se realizan medidas preventivas a nivel administrativo con el fin de restringir las acciones del deudor para forzarlo a pagar su deuda.

4.2.1 Análisis de Símbolos

Si bien la descripción provista del contexto de la aplicación es breve, puede observarse que las nueve situaciones planteadas se corresponden con estados en los que pueden encontrarse el reclamo de la deuda. Esto es importante tanto en la construcción del LEL que constituye el primer paso de la estrategia, como al momento de agrupar los símbolos en la herramienta, ya que estos símbolos por ser principales deben deshabilitados para obtener mejores resultados. Por lo descripto el elemento principal es “deuda”, y los estados asociados por los que pasa son los nuevos estados enunciados.

El segundo paso del algoritmo de detección de características transversales es la identificación de los símbolos del dominio. Estos pueden verse en la tabla 4.5 ya agrupados al estado al que se los va a asociar.

Tabla 4.5 Grupos y símbolos del dominio anti evasión de impuestos (parte 1)

Estado		Sujetos	Objetos	Verbos
1	Establecimiento de Deuda	Contribuyente/Ciudadano	Tributo/ Impuesto	Tributar / Pagar / Pagar Tributo / #Cobrar
		Deudor	Deuda	Contraer Deuda / Adquirir Deuda
			Titulo Ejecutivo/Titulo administrativo	Intimación prejudicial
			Comprobante	Crear Titulo Ejecutivo
				Cobrar Deuda / Salvar Deuda
2	Procedimiento a nivel administrativo	Fiscalía / Fisco	Bien	Evaluar Costo - Beneficio
		Registro	Bien Mueble	
		Registro Propiedad	Bien Inmueble	
		Registro Automotor		

Tabla 4.5 Grupos y símbolos del dominio anti evasión de impuestos (parte 2)

	Estado	Sujetos	Objetos	Verbos
3	Procedimiento a nivel Judicial	Juez	Expediente Judicial	Paso Judicial
		Abogado	Carta Documento	Iniciar / Radicar
		Apoderado Fiscal	Juicio	Intimar
		Delegado Fiscal		Obtener Sentencia
4	Procedimiento a Nivel Cautelar			Renovar embargo, reinscribir medida cautelar
				Realizar remate
5	Cancelación de Deuda	Banco		Calcular monto a abonar
				Pagar en efectivo, depositar dinero
				Verificar que se haya efectuado el depósito
				Emitir comprobante de pago de moratoria
6	Deuda Reclamada por Error			Trabar medida cautelar administrativa, inhibición
				Trabar medida cautelar, embargar
				Reinscribir medida cautelar administrativa
				Levantar medida cautelar administrativa
				Levantar medida cautelar
7	Plan de Facilidades de Pago			Entrar en moratoria
				Salir de moratoria
				Pagar cuota, abonar cuota
8	Subasta	Martillero	Precio base	Designar martillero
		Participante de la subasta, postor		Publicar remate
				Rematar
9	Medidas Preventivas Administrativas			Presentar comprobantes de pago

4.2.2 Conteo de Referencias

El siguiente paso del algoritmo de detección de características transversales consiste en contar la cantidad de referencias que recibe cada conjunto de símbolos (agrupados por estados) desde símbolos pertenecientes a otros grupos.

La tabla 4.6 es un resumen a nivel de grupo, donde podemos ver la cantidad de referencias que reciben cada conjunto de símbolos desde el resto de los grupos, estos resultados fueron obtenidos realizando los cálculos manualmente. En esta tabla se ve en la primera columna la cantidad de grupos desde los que existe al menos una referencia al conjunto de símbolos agrupados por un estado. En la segunda columna aparece el promedio general de referencias

del conjunto. En las últimas tres columnas se muestran los promedios de referencias que tienen los distintos tipos de símbolos (Sujetos, Objetos y Verbos).

Los resultados obtenidos por la herramienta coinciden con los que se obtienen de forma manual. Los datos pueden compararse observando las tres últimas columnas de la tabla 4.6 los cuales son los resultados del cálculo manual, contrastando los resultados con los arrojados por la figura 4.5 que pertenecen a TICC.

Tabla 4.6 Conteo de Referencias agrupado por estado para el dominio activación de impuestos

		Número de Grupos que lo referencian	Promedio General de Referencias	Promedio de Referencias a Sujetos	Promedio de Referencias a Objetos	Promedio de Referencias a Verbos
1	Establecimiento de Deuda	7	3.7	10	2.3	1.2
2	Procedimiento a nivel administrativo	7	4.3	6	4.7	0
3	Procedimiento a nivel Judicial	8	2.0	2	3.3	1.5
4	Procedimiento a Nivel Cautelar	1	0.7	0	0	0
5	Cancelación de Deuda	4	2.0	2	0	1.3
6	Deuda Reclamada por Error	1	3.0	0	0	2.0
7	Plan de Facilidades de Pago	4	2.3	0	0	1.7
8	Subasta	2	1.0	0	0	2.0
9	Medidas Preventivas Administrativas	4	2.0	0	0	2.2

Figura 4.5 Resultados arrojados por TICC para el Dominio anti evasión de impuestos

4.2.3 Ranqueo de Grupos

Continuando con los pasos descriptos en el algoritmo de detección de características transversales, el siguiente paso consiste en ordenar los grupos de símbolos poniendo en orden descendente a aquellos que maximizan la dispersión y enmarañamiento de sus símbolos. Es decir, poniendo primero a aquellos grupos de símbolos que tienen mayor probabilidad de ser característica transversal. Para el dominio del sistema anti evasión de impuesto, el orden obtenido manualmente es el que se puede ver en la siguiente tabla.

Tabla 4.7 Orden final del Dominio anti evasión de Impuestos

		Números de grupos que los referencian	Promedio general de referencias
1	Procedimiento a nivel administrativo	7	4.3
2	Establecimiento de la deuda	7	3.7
3	Procedimiento a nivel judicial	8	2.0
4	Plan de facilidades de pago	4	2.3
5	Cancelación de la deuda	4	2.0
6	Medidas preventivas administrativas	4	2.0
7	Subasta	2	1.0
8	Deuda reclamada por error	1	3.0
9	Procedimiento a nivel cautelar	1	0.7

Mientras que los resultados arrojados por TICC son los que se ven en la figura siguiente

Figura 4.6 Orden final del dominio anti evasión de impuestos arrojado por TICC

4.2.4 Análisis Final

El paso final del algoritmo consiste en realizar un análisis de los resultados obtenidos. Esta tarea no puede ser automatizada ya que el análisis es particular a cada caso. Con el objetivo de tratar de facilitar este análisis, es que TICC genera un gráfico de dispersión como el que se ve en la figura 4.7. Esta figura, puede leerse de la siguiente forma: los grupos que se encuentran en la parte superior de la diagonal principal, son los máximos candidatos a ser característica trasversal. En particular para el dominio que se está analizando, en el gráfico se ven dos puntos, los cuales representa a los grupos “Procedimiento a nivel administrativo” y “Establecimiento de Deuda” (de derecha a izquierda). Luego deben ser considerados los grupos “Procedimiento a nivel judicial” y “Deuda reclamada por error”, que si bien salen de la diagonal principal, presentan el mismo promedio de referencias que “encarcelamiento de deuda” y “plan de facilidades de pago”, los primeros tienen una mayor dispersión o cantidad de grupos de los que son referenciados. Finalmente se ubican los grupos “Medida preventiva administrativa”, “subasta” y “Procedimiento a Nivel Cautelar” que no deben ser considerados característica trasversal por su bajo promedio de referencias y cantidad de grupos que los referencian.

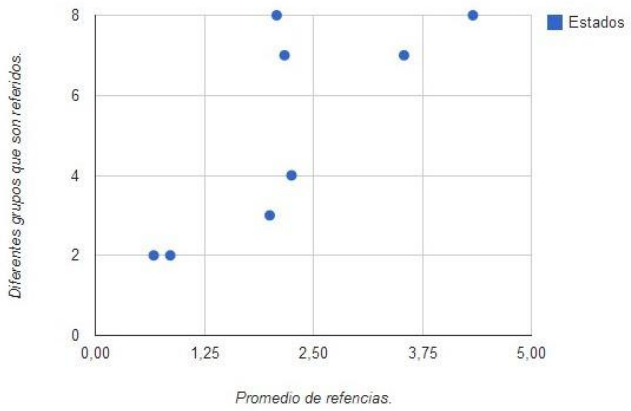

Figura 4.7 Gráfico de Dispersión para el Dominio anti evasión de impuestos

4.3 Portal Web

El último dominio a analizar consiste en un sistema web que se encarga de publicar noticias diariamente. Estas noticias son desarrolladas por redactores quienes luego van a ser supervisados por editores. La publicación de las noticias no es trivial, puesto que se prepara un formato HTML para un navegador web, pero también se preparan otros formatos para teléfonos celulares y dispositivos móviles.

Además de construir los distintos formatos para cada noticia es necesario realizar algún tipo de categorización sobre las mismas. Por lo tanto, la publicación de las notas desde que el redactor las crea, hasta que el usuario final puede leerlas desde la web pasa por distintos estados. Los mismos son: redactada, aceptada, html construido, versiones auxiliares construidas, categorizada, taggeada, incluida en el calendario y publicada.

El primer estado es redactada y representa la situación en donde el redactor preparo su nota y la dejo lista para que el editor haga su revisión. Luego de la revisión, el editor puede descartar la nota, y si eso sucede la nota pasa al estado rechazada, el cual es un estado final, es decir, una vez que la nota fue calificada como rechazada la misma es descartada y nunca se la

considerara para su publicación. Sin embargo, si la noticia pasa la revisión del editor y es considerada para su publicación, pasa al estado aceptada y desde allí es necesario crear las versiones para publicarla. El próximo estado es HTML construido y luego de construir el HTML se construyen las versiones auxiliares. Por lo cual, pasa al estado versiones auxiliares construidas. Los próximos pasos para la publicación de la noticia son opcionales, es por ello que se pueden obviar, realizar alguno de ellos, o realizar todos. Los siguientes estados son: categorizada, taggeada e incluida en el calendario. Luego de todos estos pasos se alcanza el estado publicado.

4.3.1 Análisis de Símbolos

Para el dominio descripto, se construyo el LEL correspondiente, agrupando los símbolos en función de los estados y a partir de ellos se contaron las referencias necesarias para la estrategia propuesta. Esto constituye el primer paso del algoritmo de detección de características transversales. La tabla 4.8, en su columna de mas a la izquierda muestra los estados que se encuentran definidos en el LEL.

El segundo paso del algoritmo de detección de características transversales consiste en la identificación del símbolo principal. En la descripción anterior queda en evidencia que el símbolo que pasa por distintos estados es Contenido, por esto es el símbolo principal del sistema. Mientras que los estados por los que pasa un contenido son redactada, rechazada (es un estado final), aceptada, HTML construido, y versiones auxiliares construidas. Luego están los estados opcionales por los que puede pasar: categorizada, taggeada e incluida en el calendario.

Continuando con el siguiente paso del algoritmo de detección de características transversales, este consiste en agrupar los símbolos a algún estado. En la tabla 4.8 se pueden ver todos los símbolos del dominio, la categoría de cada uno y el estado al que fueron asociado.

Tabla 4.8 Símbolos del dominio del Portal Web (parte 1)

Estado	Símbolo	Categoría
Redactado	Redactor	Sujeto
	Gestor	Sujeto
	Editor	Sujeto
	Contenido	Objeto
	Titulo	Objeto
	Desarrollo	Objeto
	Crear Contenido	Verbo
	Aceptar Contenido	Verbo
	Rechazar Contenido	Verbo
Rechazado	Limpiar Historial	Verbo
Aceptado	Usuario Final	Sujeto
	Publicador	Sujeto
	Publicar Contenido	Verbo
	Construir Versiones Publicables del Contenido	Verbo
HTML Construido	HTML	Objeto
	Planilla	Objeto
	Black Word List	Objeto

Tabla 4.8 Símbolos del dominio Portal Web (parte 2)

Estado	Símbolo	Categoría
Versiones Auxiliares	PDF	Objeto
	Imprimible	Objeto
	Mobile	Objeto
	SMS	Objeto
	Formato Básico	Objeto
	Construir Auxiliar	Verbo
	Construir PDF	Verbo
	Construir Imprimible	Verbo
	Construir Mobile	Verbo
	Construir SMS	Verbo
	Aplicar Formato Básico	Verbo
Indexado	BD Índices	Objeto
	Indexar	Verbo
	Buscar Candidatos	Verbo
	Buscar Candidatos Titulo	Verbo
	Buscar Candidatos Desarrollo	Verbo
	Buscar Texto Enfatizado	Verbo
Calendario	Calendario	Objeto
	Agregar al Calendario	Verbo
Tags	Tags	Objeto
	Taggear	Verbo
Publicado	Hacer visible al usuario	Verbo

4.3.2 Conteo de Referencias

Una vez identificado el símbolo principal y sus respectivos estados, el paso que sigue consiste en contar la cantidad de referencias que recibe cada grupo de símbolos desde los símbolos de los restantes estados.

Realizando de forma manual los cálculos de forma manual, obtenemos los resultados que se ven en la tabla 4.9. Comparando los resultados con los arrojados por TICC (ver figura 4.8), se puede ver que en ambos casos los estados Redactado y Aceptado son los estados más referenciados con 5 y 3 respectivamente, pero el estado Aceptado tiene mayor probabilidad de ser característica transversal ya que presenta un mayor promedio general de referencias. El otro grupo de símbolos a analizar es HTML construido que es referenciado desde 3 grupos de símbolos, pero tiene una baja cantidad de referencias, esto puede por el promedio general de referencias ya que cada símbolo tiene casi una referencia.

Los mismos resultados que se obtuvieron mediante el cálculo manual (ver tabla 4.9) se obtuvieron utilizando la herramienta TICC, estos resultados pueden verse en las figuras 4.8 y 4.9. En la primera figura pueden apreciarse las tres últimas columnas de la tabla 4.9, es decir: el promedio de referencias hacia objetos, sujetos y verbos. Mientras que en la figura 4.9, se ve el promedio de referencias general del grupo y la cantidad de grupos desde los que existen referencias hacia el grupo identificado por la fila.

Tabla 4.9 Conteo de Referencias del Portal Web

		Numero de Grupos que lo referencian	Promedio general de referencias	Promedio de referencias a sujetos	Promedio de referencias a Objetos	Promedio de referencias a Verbos
1	Redactado	5	2,5	2	6,3	0
2	Rechazado	1	0,5	0	0	0
3	Aceptado	3	13,4	32	0	1
4	Html Construido	3	0,9	0	0,3	1,4
5	Versiones auxiliares construidas	1	0,2	0	0	0,2
6	Categorizado	1	0,6	0	0	0,2
7	Incluido en calendario	1	1,3	0	0	1
8	Taggeado	1	1	0	0	0,5
9	Publicado	1	2,5	0	0	1

4.3.1 Ranqueo de Grupos

El paso siguiente del algoritmo, consiste en ordenar los grupos, poniendo en la parte superior a aquellos grupos de símbolos que tienen mayor probabilidad de ser característica transversal. Es decir, se pondrán primeros a aquellos símbolos que maximicen el scattering y el tangling. El orden obtenido por TICC se ve en la figura 4.9, mientras que el orden obtenido realizando los cálculos manuales se puede ver en la tabla 4.10.

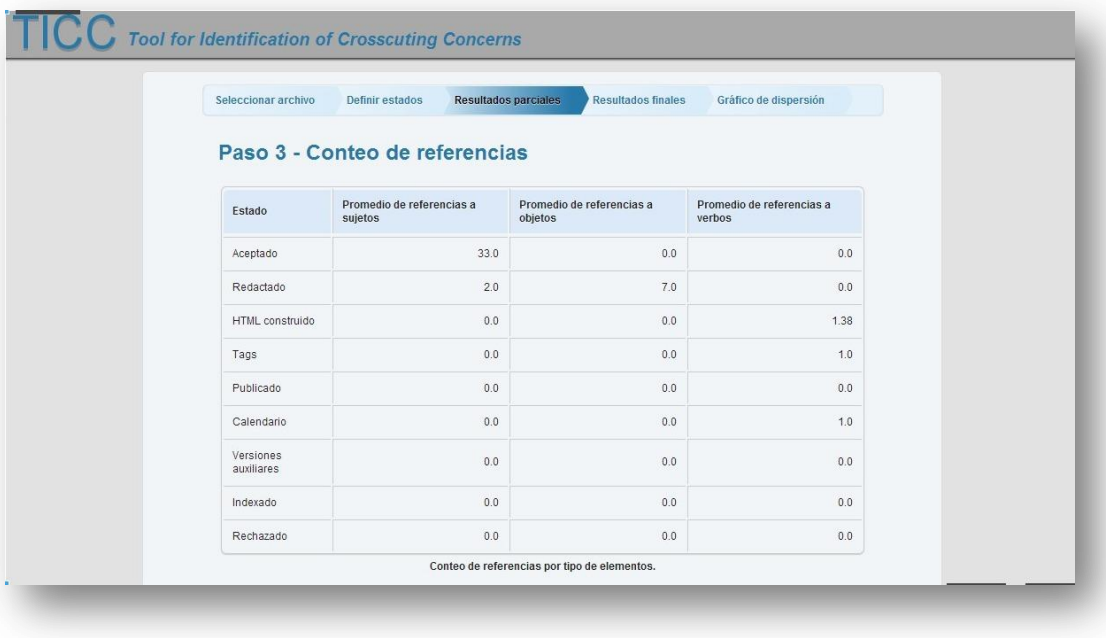

Figura 4.8 Conteo de referencias hacia tipos de dominio del dominio del portal web

Figura 4.9 Orden final propuesto por TICC para el dominio del portal Web

Analizando el orden obtenido en TICC, puede verse que si bien el estado redactado es referenciado desde una mayor cantidad de grupos que aceptado, este tiene menor probabilidad de ser característica transversal debido a que cada símbolo tiene como promedio 13 referencias desde otros grupos, mientras que los símbolos del estado redactado reciben tres referencias desde los otros estados.

Tabla 4.10 Orden final obtenido mediante el cálculo manual del dominio del portal web

		Número de grupos que lo referencian	Promedio general de referencias
3	Aceptado	3	13,4
1	Redactado	5	2,5
4	HTML Construido	3	0,9
9	Publicado	1	2,5
7	Incluido en calendario	1	1,3
8	Taggeado	1	1
6	Categorizado	1	0,6
2	Rechazado	1	0,5
5	Versiones auxiliares construidas	1	0,2

4.3.2 Análisis Final

Con los resultados arrojados por la figura 4.9 y la tabla 4.10, podemos decir que definitivamente el grupo 3 “Aceptado” ocupa el primer lugar de los candidatos a des que posee el valor más alto en promedio de referencias y tiene el segundo valor más alto en grupos que lo referencian con 3. Le siguen el grupo 1 “Redactado” con referencias desde 5 grupos, pero un promedio de referencias por símbolo significativamente menor que el grupo 3, por esto es que queda segundo. El tercer lugar lo ocupa el grupo 4 “HTML construido” el cual tiene referencias desde 3 grupos de símbolos (menos que el que quedo en segundo lugar) y un promedio de referencias por símbolo de .9. Los restantes grupos poseen referencias desde solo un grupo de símbolos y tienen un promedio de referencias bajo, por lo que se descartan como posible característica transversal.

TICC agrega un gráfico de dispersión (ver figura 4.10), donde los grupos de símbolos que son candidatos a característica transversal se agrupan sobre el eje diagonal.

Aunque las figuras 4.9 y 4.10 nos orienten respecto de los posibles grupos a ser característica transversal, es necesario realizar un análisis a nivel de símbolos, este, es el último paso del algoritmo de detección de características transversales. El cual debe ser realizado por el analista que use la herramienta.

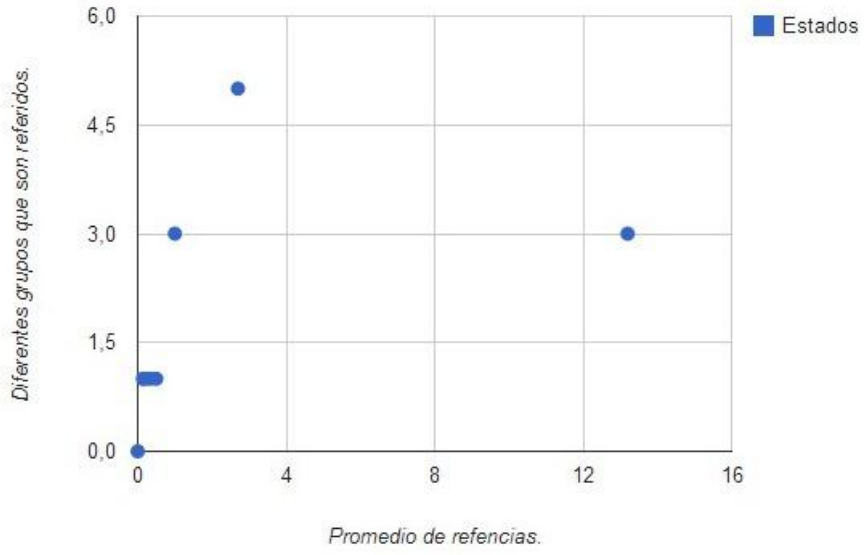

Figura 4.10 Gráfico de Dispersión del Dominio del Portal Web

4.4 Conclusión

En el presente capítulo se ha presentado el uso de la herramienta TICC a través de tres casos de estudio, con el fin de demostrar que la herramienta aplica la estrategia de identificación de características transversales correctamente.

En los tres dominios que se presentaron, se obtuvieron los mismos resultados mediante la herramienta TICC que realizando los cálculos manualmente. El contraste de los dos resultados (manual y automatizado) fue necesario para comprobar la efectividad de la herramienta TICC en la implementación de la estrategia.

Además de optimizar los tiempos de los cálculos, la herramienta TICC proporciona un enfoque iterativo a la estrategia para poder activar y desactivar símbolos y calcular las características transversales con un costo mínimo.

Dado que la herramienta TICC es útil e implementa correctamente la estrategia para identificar características transversales, también requiere que la interacción con la misma sea efectiva. Por ello es que en los próximos capítulos se describen las acciones que se llevaron a cabo para lograr obtener una aplicación web con un correcto diseño que cumple dos aspectos deseables: la usabilidad y la accesibilidad. La **usabilidad** busca mejorar la experiencia del usuario al usar las aplicaciones web, mientras que la **accesibilidad** intenta que distintos usuarios pueda acceder la información.

Capítulo 5

5. Usabilidad

El presente capítulo describe la evaluación de usabilidad que se ha realizado en la aplicación TICC. Esta evaluación fue necesaria para lograr que la aplicación TICC tenga un diseño que sea fácil de aprender, de usar y sea atractivo para los usuarios a la cual está dirigida. Estas cualidades permiten que la aplicación cumpla con la característica de usabilidad.

Este capítulo se encuentra organizado de la siguiente forma. En primer lugar se detalla el concepto de usabilidad (definición, atributos e importancia de la misma). Luego se presenta las técnicas disponibles para la evaluación de usabilidad de una aplicación Web. Finalmente se describe la técnica de evaluación de usabilidad aplicada a TICC y las modificaciones que surgieron a partir de la misma.

5.1 Definición

En principio la usabilidad no es un concepto que se aplique directamente al mundo de las aplicaciones web, tiene más bien que ver con todo aquel objeto útil o artefacto que es utilizado por una persona para un fin dado. Esto significa que la usabilidad es aplicable a cualquier elemento en el cual se va a producir una interacción entre un humano y un dispositivo.

La usabilidad de los productos de software constituye una característica vital que puede llegar a marcar su éxito o su fracaso, independientemente de que el programa tenga una eficiencia interna satisfactoria. Es con este tipo de software con el cual el usuario demanda mayor facilidad de uso.

Para el término **usabilidad** existen diferentes definiciones, las cuales dependen del enfoque que se le quiera dar: facilidad, calidad de uso, funcionalidad y/o eficiencia.

Figura 5.1 Tres grandes aspectos que miden la Usabilidad

La Wikipedia [Wikipedia 2013a] define usabilidad como: “la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto”. Esto es, cuanto más fácil de usar sea un programa o un dispositivo, mayor será su “usabilidad”.

La Organización Internacional para la estandarización (International Organization for Standardization, ISO) propone dos definiciones de usabilidad:

- ISO/IEC 9126 [ISO 1998a]: “La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso”. Esta definición hace énfasis en que la usabilidad no sólo depende del producto sino también del usuario.

- ISO/IEC 9241 [ISO 1998b]: "Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico". Es una definición centrada en el concepto de calidad en el uso, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad.

Jackob Nielsen [Nielsen 2012], definió usabilidad como "El atributo de calidad que mide lo fáciles que son de usar las interfaces Web”.

Otra definición clarificadora es la de Redish [Redish 2000], para quien es preciso diseñar aplicaciones web para que los usuarios sean capaces de “encontrar lo que necesitan, entender lo que encuentran y actuar apropiadamente… dentro del tiempo y esfuerzo que ellos consideran adecuado para esa tarea”.

Usabilidad significa, por lo tanto, centrarse en la audiencia potencial de las páginas, estructurar la aplicación web de acuerdo a sus necesidades y organizar la navegación de manera que le permita encontrar lo que busca. Requiere construir el sistema interactivo que mejor responda a las tareas que se vayan a realizar.

5.2 Atributos de la Usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para poder estudiarla se descompone habitualmente en los siguientes cinco atributos básicos [Nielsen 1995]:

1. **Facilidad de Aprendizaje** (*Learnability*): Tiene que ser fácil de aprender cómo funciona el sistema, tal que el usuario puede empezar a trabajar con el mismo lo más rápido posible. Esto significa que nuevos usuarios deberían aprender fácilmente a usar el sistema.

2. **Eficiencia** (*Efficiency*). Una vez que el usuario ha aprendido a utilizar el sistema, un nivel alto de productividad es posible para completar determinadas tareas. Es decir, el sistema debería ser eficiente para su uso cuando el usuario ha aprendido a usarlo.

3. **Facilidad para recordar** (*Memorability*). Cuando un usuario ha utilizado un sistema tiempo atrás, y tiene la necesidad de volver a usarlo, de nuevo la curva de aprendizaje debe ser significativamente menor que el caso del usuario que nunca haya utilizado el sistema. Esto es, el sistema deberá ser fácil de recordar incluso después de algún periodo sin ser utilizado.

4. **Prevención de error** (*Errors*). 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 funcionaria de manera optima a cualquier petición por parte del usuario. Es vital que una vez que se produzca un error el sistema, se lo haga saber rápidamente y claramente y le provea de algún mecanismo para recuperarse de ese error.

5. **Satisfacción** (*Satisfaction*). Este atributo se refiere a la impresión subjetiva del usuario con respecto al sistema, lo cual significa de que si el sistema es agradable en su uso.

Algunos de estos atributos no contribuyen a la usabilidad del sistema en la misma dirección, pudiendo ocurrir que el aumento de uno de ellos tenga como efecto la disminución de otro. Por ejemplo, esto puede ocurrir con la facilidad de aprendizaje y la eficiencia. Es preciso realizar el diseño del sistema cuidadosamente si se desea tanto una alta facilidad de aprendizaje como una alta eficiencia; siendo el uso de aceleradores (combinaciones de teclas que ejecutan operaciones de uso habitual) la solución más común para conjugar ambos atributos de Usabilidad.

La usabilidad del sistema no es una simple adición del valor de estos atributos, sino que se define para cada sistema como un nivel a alcanzar para algunos de ellos.

5.3 Importancia de la Usabilidad

Cuando se aplica usabilidad a una aplicación Web se facilita el acceso a la información a los usuarios. Por lo tanto el desarrollo de una aplicación Web requiere tener en cuenta a qué publico va dirigido y que se quiere conseguir con el mismo. El principio estratégico que debe guiar todo diseño de una aplicación Web debe tener como referencia las necesidades de los usuarios y las demandas de los posibles clientes.

Los beneficios de la Usabilidad son amplios y tienen impacto tanto desde el punto de vista de la imagen de la aplicación web como desde el punto de vista económico.

Entre los distintos beneficios que reporta una aplicación Web usables, los más destacados son:

- Reducción de los costes de aprendizaje: Si algo es sencillo, rápidamente se aprenderá a usarlo.

- Usuarios más satisfechos: la satisfacción de los usuarios es un resultado directo de las posibilidades que tengan de conseguir sus objetivos con el mínimo esfuerzo posible.

- Usuarios más fieles: la facilidad de uso produce una utilización mayor tanto en frecuencia como en amplitud de funcionalidades utilizadas.

- Disminución de los costes ayuda al usuario: una aplicación más fácil de usar genera menos problemas a los usuarios y por tanto estos consultan menos, reduciendo las necesidades de soporte y ayuda.

- Optimización de los costes de diseño y rediseño de los aplicaciones web: lo usable es lo que tiene éxito, ya que resiste muy bien al paso del tiempo.

- Menor costo de mantenimiento: los problemas de Usabilidad surgen inmediatamente a la luz a través de las llamadas a soporte y quejas de los usuarios, lo que genera un ciclo permanente de modificaciones. Sin dudas es mejor hacer las aplicaciones más usables al momento de construirlas.

- Aumento de la tasa de conversión de visitantes a clientes de la aplicación Web.

- Mejora la imagen y el prestigio de la aplicación Web y en consecuencia de la empresa u organización.

- Mejora la calidad de vida de los usuarios: ya que reduce su estrés, incrementa la satisfacción y la productividad.

Todos estos beneficios implican una reducción y optimización general de los costes de producción, así como un aumento en la productividad. La usabilidad permite mayor rapidez en la realización de tareas y reduce las pérdidas de tiempo.

5.4 Evaluación de la Usabilidad

La principal actividad en el proceso de usabilidad es la evaluación [Ferré 2001].

La evaluación de la usabilidad permite conocer el nivel de usabilidad que alcanza el prototipo actual del sistema, e identificar los fallos de usabilidad existentes.

La evaluación involucra un conjunto de métodos y técnicas que analizan la usabilidad de un sistema en diferentes etapas del ciclo de vida del software. Debe realizarse durante todo el ciclo de vida del desarrollo del sistema de software, incluyendo técnicas que cuenten con la presencia de usuarios representativos finales.

Usualmente, la evaluación se realiza mediante test de usabilidad, complementados con evaluaciones heurísticas. A continuación se describen las técnicas para evaluar un sistema.

5.4.1 Test de Usabilidad

Los test de usabilidad son la práctica de usabilidad más extendida. Consisten en presentar al usuario una serie de tareas a realizar, y pedirle que las realice con el prototipo del sistema. Las acciones y comentarios de usuario se recopilan para un análisis posterior.

Para conseguir unos test de usabilidad con resultados fiables, las condiciones del test y del lugar donde éste se realiza deben ser lo más parecidas posibles al entorno de uso previsto para el sistema.

Es importante realizar este tipo de test de usabilidad ya que no es posible conocer el nivel de usabilidad de un sistema si no se prueba con usuarios reales.

En primer lugar, es preciso decidir con qué distintos grupos de usuarios se va a probar el sistema, y cuántos participantes de cada grupo se van a obtener para realizar los test. A continuación, se deben diseñar las tareas de test cuya realización se va a pedir a los participantes; normalmente se sacan del resultado del análisis de tareas, intentado enmarcarlas en un contexto de uso real. Hay que decidir también otros detalles, como la posibilidad de pedir ayuda al evaluador, qué tipo de información se dará a los participantes acerca del sistema antes de comenzar con el test en sí, o si se dará la posibilidad al participante de acceder libremente al sistema para obtener opiniones acerca de su impresión global.

Otra variante consiste en realizar cada test con dos usuarios para observar los comentarios que intercambian. Es habitual pedir al participante que “piense en voz alta” mientras intenta llevar a cabo las tareas; este procedimiento permite identificar partes del sistema con un nivel pobre de usabilidad.

Una vez que se han realizado todos los test, los datos recogidos son analizados y los resultados se aplican en próximo prototipo del sistema.

5.4.2 Evaluación Heurística

Un experto o grupo de expertos en usabilidad realiza una evaluación heurística de la aplicación Web. El experto realizará una crítica basada en su experiencia de diseño, o en guías de diseño de usabilidad ampliamente aceptadas, como las descritas por Shneiderman [Shneiderman 1998] o Nielsen [Nielsen 1990].

Los expertos proporcionan una información distinta a la obtenida de usuarios finales mediante test de usabilidad. Las sugerencias de modificaciones por parte de un experto suelen tener más valor que las realizadas por usuarios, por ser más viables y más precisas acerca de los problemas subyacentes de usabilidad (falta de consistencia, navegabilidad pobre, etc.).

Es importante realizar una evaluación heurística ya que los expertos identifican ciertos problemas de usabilidad que requerirían un gran número de test de usabilidad para ser correctamente identificados y solucionados.

Se explica al experto el modo de funcionamiento del sistema, y se le entrena en las funciones principales. Se le informa acerca del dominio de aplicación y el experto revisa el sistema según la conformidad del mismo a guías de diseño. Una vez finalizada la revisión el experto elabora un informe con los problemas identificados y sugerencias de diseños alternativos (opcionalmente), el cual es entregado al equipo de desarrollo.

Ambos métodos de evaluación de usabilidad deben hacerse en las primeras fases de desarrollo. Es preferible hacer primero la evaluación heurística, para detectar problemas graves de usabilidad, y tras el primer rediseño hacer el test de usabilidad, para encontrar problemas de usabilidad más difíciles de detectar.

5.5 Evaluación de Usabilidad en TICC

La **Evaluación Heurística** es una técnica de evaluación de usabilidad, efectuado por un experto en usabilidad y se basa en el recorrido y análisis de la aplicación web, identificando errores y problemas de diseño (elementos estructurales, esquema de colores, tipología, componentes interactivos y operativos).

Este método tiene como ventaja la facilidad y rapidez con la que se puede llevar a cabo. Es recomendable aplicar este método en etapas tempranas del desarrollo de una aplicación web o cuando el mismo va a salir a producción. Por todo esto es que en esta primera etapa de validación de usabilidad de TICC, se opto por aplicar la técnica de evaluación heurística.

5.5.1 Heurísticas de Nielsen

Es importante describir cuales son las heurísticas (principios) de usabilidad a través de los cuales se evaluó la usabilidad de TICC. A continuación se detallan las diez heurísticas de Nielsen [Nielsen 1995b]:

i. **Visibilidad del estado del sistema:** El sistema siempre debe mantener a los usuarios informados del estado del sistema, con una retroalimentación apropiada y en un tiempo razonable. Por ejemplo, cuando en una interfaz tipo web mail se adjuntan ficheros a un mensaje, el sistema debe informar del hecho mostrando un mensaje de espera.

ii. **Utilizar el lenguaje del usuario:** El sistema debe hablar el lenguaje de los usuarios, con las palabras, las frases y los conceptos familiares, en lugar de que los términos estén orientados al sistema. Evitar tecnicismos incomprensibles o mensajes crípticos. Además, se deberá seguir las convenciones usadas en el mundo real.

iii. **Control y libertad por parte del usuario**: Los usuarios eligen a veces funciones del sistema por **error** y necesitan a menudo una "salida de emergencia" claramente marcada, esto es, salir del estado indeseado. Por ejemplo la representada por la opción para "saltar" animaciones de introducción (normalmente Flash).

iv. **Consistencia y estándares**: Los usuarios no deben tener que preguntarse si las diversas palabras, situaciones o acciones significan la misma cosa. Un ejemplo de consistencia, es la de no utilizar dos rótulos distintos para referirse a un mismo contenido, o no usar estilos diferentes dentro de la misma aplicación web. Además la aplicación web debe seguir estándares o convenciones de diseño ampliamente aceptados. Cuanto más se parezca un diseño y su funcionamiento al resto de las aplicaciones web, más familiar y fácil de usar resultará para el usuario.

v. **Prevención de errores**: Mejor que un buen mensaje de error es un diseño que prevenga que ocurra el error. Se deben eliminar situaciones propensas a errores o verificarlas y presentarlas al usuario con una confirmación antes de que ejecute la acción.

vi. **Minimizar la carga de memoria del usuario**: El usuario no debería tener que recordar dónde se encontraba cierta información, o cómo se llegaba a determinada página. Se debe minimizar la carga de memoria del usuario haciendo visible las diferentes opciones, acciones, enlaces y objetos.

vii. **Flexibilidad y eficiencia de uso:** Las acciones para el uso del sistema deben ser visibles o fácilmente accesibles siempre que se necesiten. Se debe proporcionar atajos o aceleradores para usuarios expertos.

viii. **Estética y diseño minimalista**: Los diálogos no deben contener información que sea relevante o que rara vez sea de utilidad. Cualquier tipo de información que no sea relevante para el usuario y que sobrecargue la interfaz debe ser eliminada.

ix. **Ayudar al usuario a reconocer, diagnosticar y recuperarse de los errores**: Los mensajes de error deben expresarse en un lenguaje claro, se debe indicar exactamente el problema y deben ser constructivos. Por ejemplo, cuando un usuario introduce una consulta en un buscador y no obtiene ningún resultado, se debe informar al usuario sobre cómo solucionar el problema, por ejemplo con mensajes del tipo "introduzca algún sinónimo" o "quiso Ud. decir...". Además no se debe borrar el contenido de la caja de búsqueda para que el usuario pueda rehacer la consulta.

x. **Ayuda y Documentación:** Aunque siempre es mejor que una aplicación web se pueda utilizar sin necesidad de ayuda o documentación, puede ser necesario disponer de ayuda y documentación. Cualquier información tiene que ser fácil de buscar, centrada en las acciones del usuario y que no sea muy extensa. En cuanto la aplicación web ofrezca algunas características fuera de la norma, sea extenso o cuente con procesos

de interacción complejos (como el rellenado de un formulario) será necesario prestar información de ayuda y dar documentación a los usuarios.

5.5.2 Evaluación Heurística aplicada a TICC

Cabe recordar que, antes de esta instancia de evaluación de usabilidad la versión original de la aplicación se denominaba Tool for Identification of Crosscutting Concerns with LEL (TICCWL).

La evaluación heurística de usabilidad en TICC la llevo a cabo una experta en usabilidad, la cual, en base a su propia experiencia y fundamentándose en las reconocidas heurísticas de usabilidad [Nielsen 1990] y apoyándose en guías de criterios, examino la aplicación web TICC

En una primera instancia la experta navego a través de las distintas páginas que conforman la aplicación TICC. Una vez conocidas todas las páginas de la aplicación, la experta efectuó preguntas referentes al propósito general de la aplicación. Una vez que la experta comprendió el propósito de la aplicación procedió a realizar un segundo recorrido por la misma.

En la segunda instancia de navegación por la aplicación, fue analizando detalladamente cada una de las páginas de la aplicación. Esta vez por cada uno de ellas fue describiendo en forma simplificada cada uno de los errores de usabilidad detectados.

Después de la evaluación y la detección de los errores de usabilidad, la experta realizo un último recorrido por la aplicación TICC, para describir en forma precisa y minuciosa los errores de usabilidad. Cada uno de estos errores fue registrado para poder luego poder llevar a cabo las modificaciones propuestas por la experta. Provocando así que la versión original de la aplicación web cambie a una nueva versión que es usable.

5.5.3 Resultado de la evaluación Heurística aplicada a TICC

Como consecuencia de los errores de usabilidad detectados durante la evaluación heurística realizada por la experta, la misma propuso modificaciones para ser aplicadas a TICC. A continuación se detallaran cada uno de los errores de usabilidad detectados, donde para cada uno de ellos se indicara el número de heurística violado, descripción del error de usabilidad y la modificación realizada [Hommy 1995].

5.5.3.1 Nombre de la Aplicación

Heurística violada:

Heurística VI – *"El nombre del sistema debe ser fácil de identificar, pronunciar y recordar por el usuario."*

Error de usabilidad:

El nombre Tool for Identification of Crosscutting Concerns with LEL y su sigla TICCWL no ayuda a la memoria del usuario.

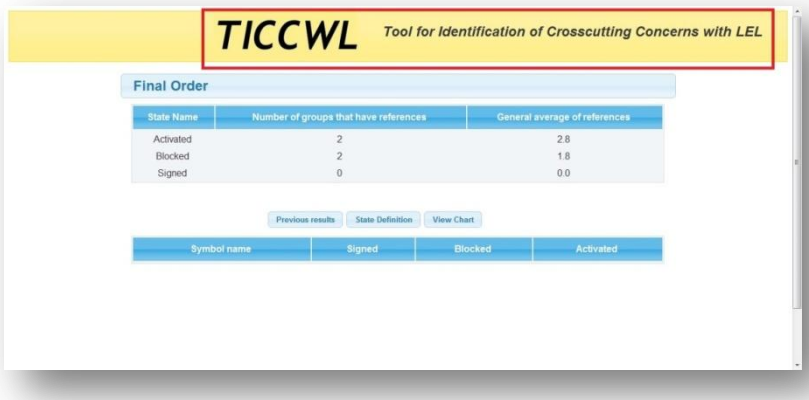

Figura 5.2 Nombre largo de la aplicación

Modificación:

Los usuarios tienen una memoria a corto plazo limitada. Se propuso el nombre "Tool for Identification of Crosscutting Concerns" con su respectiva sigla, TICC.

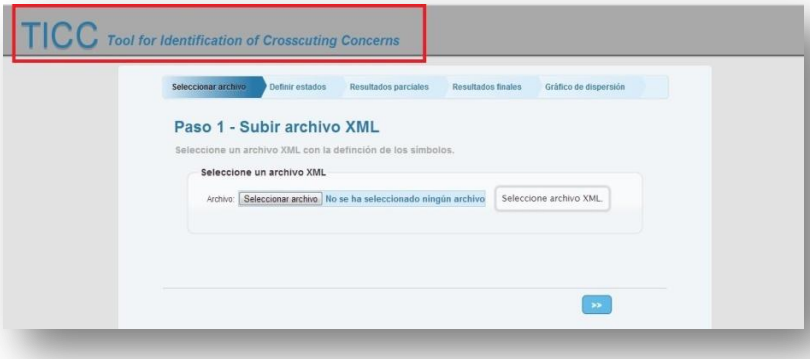

Figura 5.3 Nuevo nombre de la aplicación.

5.5.3.2 Ubicación de la identidad de la aplicación

Heurística violada:

Heurística IV - *"Se debe seguir los convencionalismos de diseño ampliamente aceptados."*

Error de usabilidad:

En una primera versión de la aplicación, se centro la información de identidad de la aplicación. La información de la identidad de la aplicación (logotipo y nombre) debe ser lo suficientemente evidente como para identificarla y si incluye texto, ha de ser fácilmente legible.

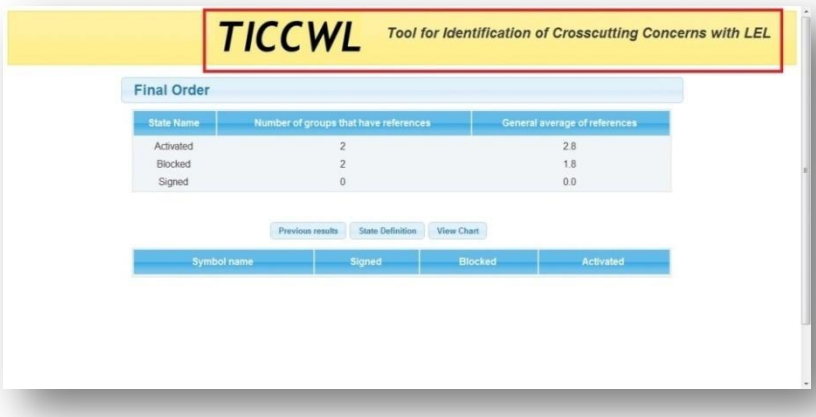

Figura 5.4 Ubicación incorrecta de la información de la aplicación.

Modificación:

La información de identidad de la aplicación siempre ha de estar ubicado en la parte superior de la página y, preferiblemente alineada en su parte izquierda.

Se reubico la información de la aplicación TICC hacia la parte izquierda de la página y se alineo el texto asociado al logotipo de la aplicación.

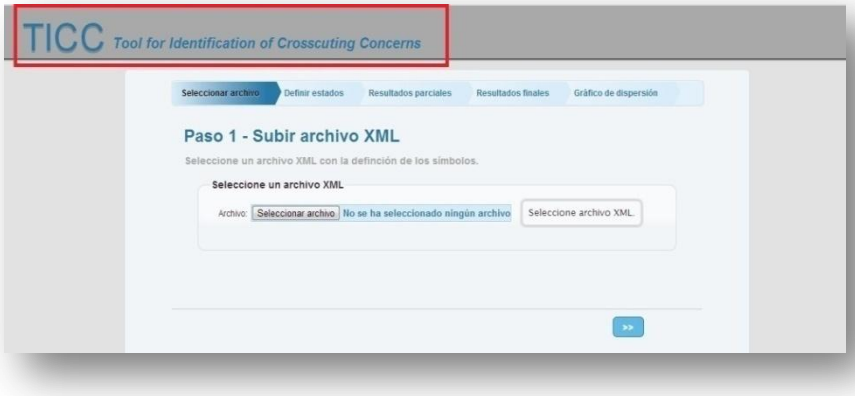

Figura 5.5 Nueva ubicación de la información de la aplicación.

5.5.3.3 Tablas de la aplicación

Heurística violada:

Heurística II - *"La lectura de la información debe ser efectuada en un orden natural."*

Error de usabilidad:

La información debe aparecer en un orden lógico y natural para el usuario. Centrar los textos en las tablas no es una forma correcta de lectura de la información.

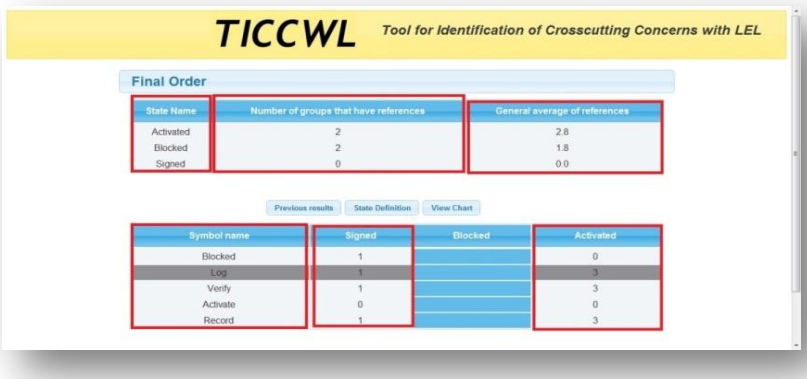

Figura 5.6 Alineación incorrecta de textos.

El usuario está acostumbrado a leer de izquierda a derecha y de arriba abajo.

Modificación:

Las palabras deben estar alineadas a izquierda, mientras que los textos numéricos deben alinearse a derecha. Se modificaron las alineaciones de los textos en todas las tablas del sistema, manteniendo una consistencia para la lectura de la información contenida.

Figura 5.7 Alineación de textos en tablas.

5.5.3.4 Estructura de la Navegación

Heurística violada:

Heurística I - *"Visión de la situación de la aplicación. "*

Error de usabilidad:

La aplicación web debe contar con una estructura de navegación clara para el usuario. La desorientación del usuario durante su navegación por la aplicación es uno de los mayores problemas en las aplicaciones web.

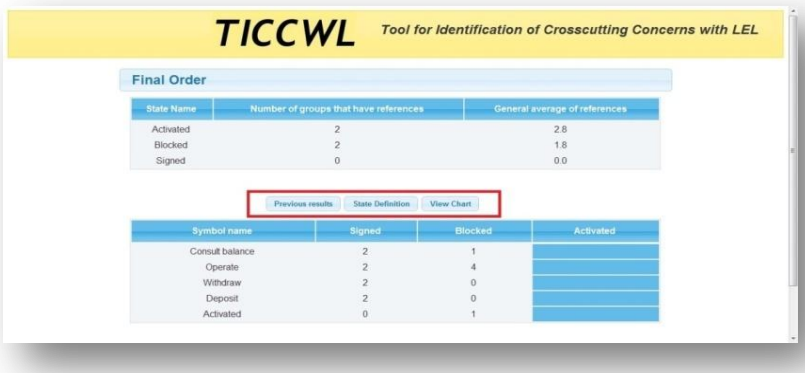

Figura 5.8 Estructura de navegación no definida.

El objetivo que debe perseguir una aplicación web es la de orientar al usuario en su navegación.

Modificación:

Las aplicaciones web están compuestas por diferentes unidades o páginas conectadas. Para navegar debe existir una estructura clara, que no haya que descubrir y que sea consistente en todas las páginas, que sea familiar y facilite su aprendizaje. Como la aplicación TICC consta de una funcionalidad de muchos pasos y su navegación es del tipo lineal, se cambio la estructura de navegación a un wizard.

Figura 5.9 Estructura de navegación- Wizard de la aplicación TICC.

5.5.3.5 Consistencia de la Navegación

Heurística violada:

Heurística IV VI - "*Acciones de navegación ambiguas y complejas.*"

Error de usabilidad:

Las acciones de navegación que se encuentran en la aplicación web TICC, usan terminologías complejas y son distintas a través de las distintas páginas. Esto afecta a la interactividad del usuario con la aplicación web.

Figura 5.10 Acciones de navegación ambiguas.

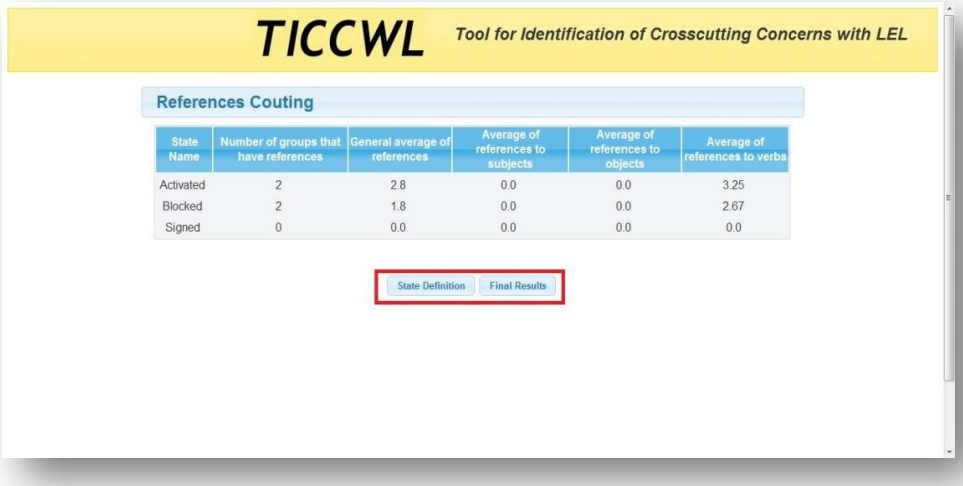

Figura 5.11 Acciones de navegación ambiguas.

Las acciones o enlaces para la navegación a través de la aplicación web deben ser consistentes en todas las páginas que lo conforman.

Modificación aplicada:

Situar en todas las páginas al menos una acción que permita continuar la navegación. Siempre se debe utilizar las mismas referencias iconográficas y/o textuales para representar las mismas acciones a lo largo de la aplicación web.

Se modificaron en todas las páginas de TICC cada uno de las acciones de navegación, por los caracteres << y el >> para indicar página anterior y página siguiente respectivamente.

Figura 5.12 Acciones de navegación unificadas.

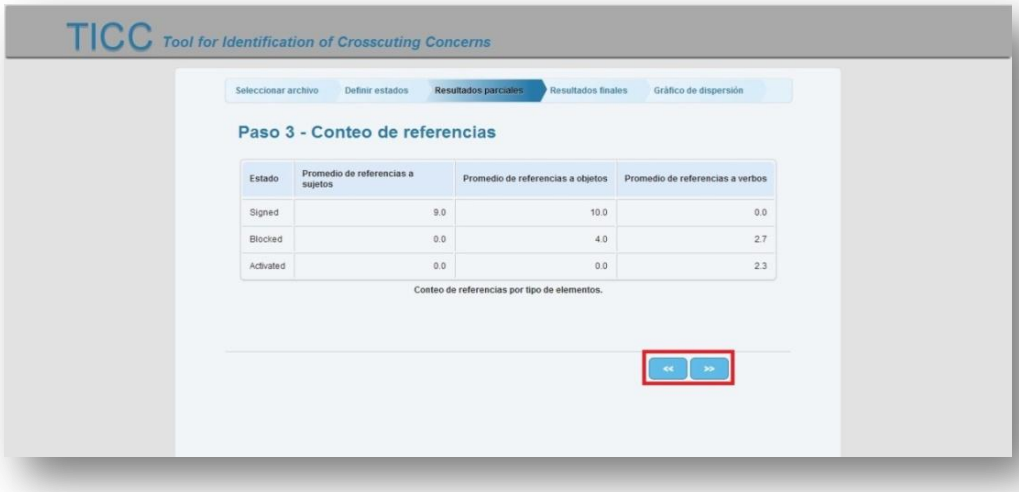

Figura 5.13 Acciones de navegación unificadas.

5.5.3.6 Ubicación de la navegación

Heurística violada:

Heurística IV - *"Respetar los convencionalismos de ubicación de las acciones de navegación."*

Error de usabilidad:

La ubicación de las acciones de navegación centradas en la página no respeta las convenciones de diseño ampliamente aceptadas, para que al usuario le sea más familiar y fácil de usar.

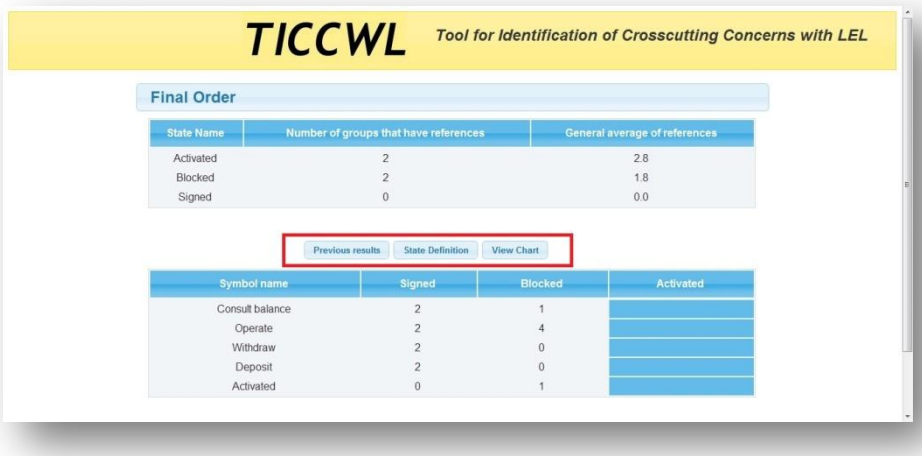

Figura 5.14 Ubicación errónea de las acciones de navegación.

Modificación aplicada:

La navegación se suele basar en colocar los ítems en un raíl derecho, fichas superiores o categorías centrales. Se modifico la ubicación de los acciones de navegación, en todas las páginas de la aplicación TICC, para que los mismo estén alineados a la derecha de cada uno de ellas.

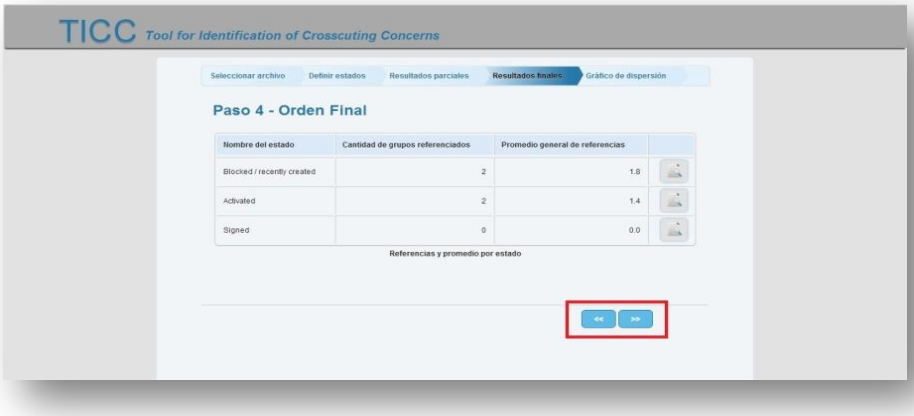

Figura 5.15 Alineación a derecha de las acciones de navegación

5.5.3.7 Estructura visual consistente

Heurística violada:

Heurística IV y VII - *"Mantener una consistencia en la disposición y visibilidad de los elementos."*

Error de usabilidad:

La ubicación de las acciones de navegación entre elementos, en este caso entre que tablas es un error que no permite al usuario mantener una jerarquía visual correcta. Se debe mantener una estructura visual constante, donde los diferentes elementos mostrados en la pantalla no varíen en su ubicación entre páginas y mantengan siempre la misma jerarquía visual.

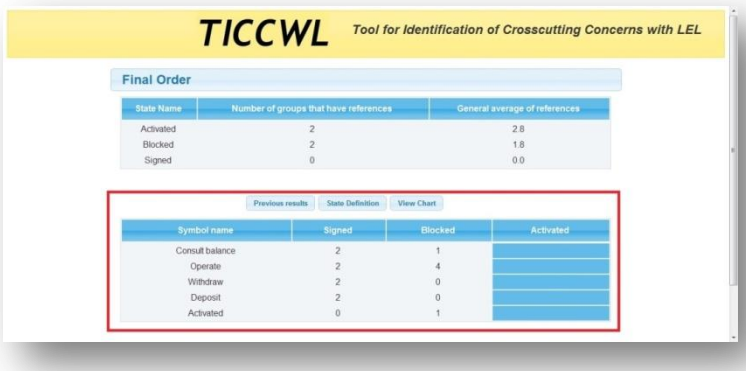

Figura 5.16 Consistencia de la jerarquía visual errónea.

Modificación aplicada:

La aplicación web debe ser consistente en la disposición de los elementos de la página, para ofrecer un entorno homogéneo que ayude a potenciar una comunicación efectiva con el usuario. Se reubico tanto la visualización de la tabla de detalles de un estado como las acciones de navegación, para lograr una disposición de elementos (entre páginas) homogénea.

Figura 5.17 Disposición homogénea de los elementos.

5.5.3.8 Acciones Asociadas a los elementos

Heurística violada:

Heurística I y III - *"Visibilidad de la aplicación y control por parte del usuario."*

Error de usabilidad:

El ubicar la acción de "detalles de un estado" alineada con las acciones de navegación es un error que viola la heurística I. Además, el seleccionar un "estado" de la tabla para luego ver su detalle, es un funcionamiento confuso para el usuario inexperto.

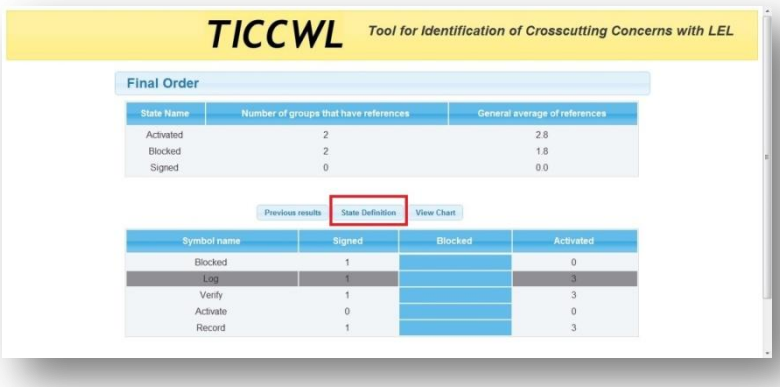

Figura 5.18 Ambigüedad en el impacto de las acciones.

El usuario siempre debe saber que es lo que está haciendo y debe tener en claro en todo momento sobre que componente ejerce una acción.

Modificación aplicada:

En la página de los resultados finales, se reubico la acción de ver detalles de un estado. Se asocio a cada uno de los estados que se visualizan, un icono Ver detalle. Al seleccionar dicha icono/acción sobre un estado dado, se muestra dinámicamente el detalle del mismo.

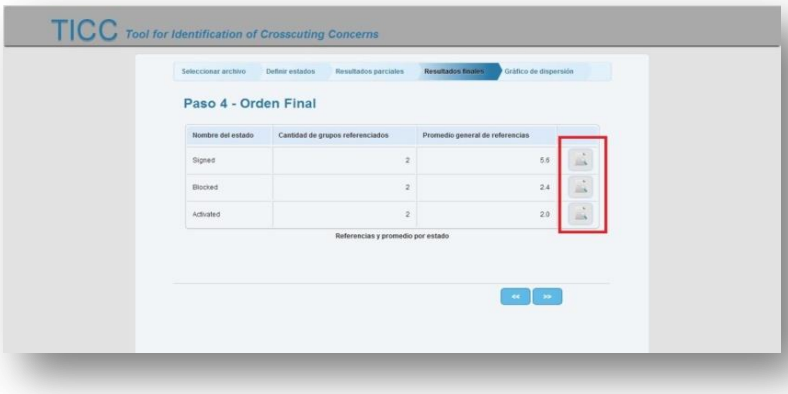

Figura 5.19 Incorporación del icono “ver detalle” en cada uno de los estados.

Con esto se logra crear una jerarquía visual clara que se repite en cada página, dividiendo las diferentes zonas de la misma de forma definida y constante. Se elimina la superposición de elementos del mismo nivel y se deja en claro en todo momento sobre qué componentes podrá el usuario ejercer una acción.

5.5.3.9 Tamaño de las páginas

Heurística violada:

Heurísticas I y III – Visibilidad y control por parte del usuario.

Error de usabilidad:

Los usuarios suelen escoger las opciones que son visibles en la pantalla, por lo tanto el uso del scroll (desplazamiento hacia bajo) en páginas que no lo requieren es un problema que no permite

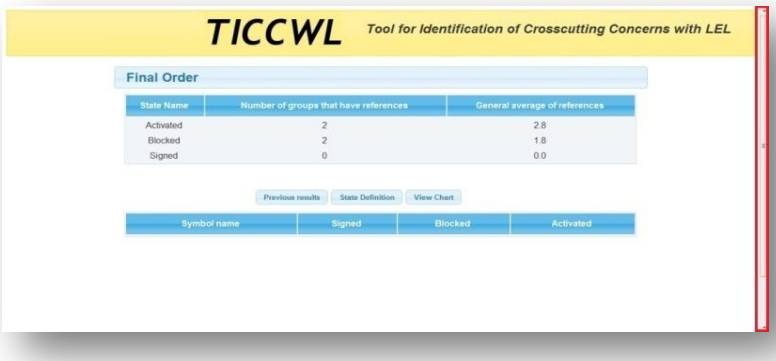

Figura 5.20 Página con scroll innecesario.

Las personas no suelen hacer scroll y además el uso de scroll hace perder el tiempo a los navegantes.

Modificación aplicada:

Se debe minimizar el uso del scroll. Se elimino el scroll en las páginas donde el mismo no era necesario. Para el caso de la página de definición de estado, se lo mantuvo dado que pueden llegar a existir muchos símbolos para ser organizados en estados.

Figura 5.21 Eliminación del scroll en la página de subida de archivo

5.5.3.10 Colores de la aplicación

Heurística violada:

Heurística VIII - "*La estética y el diseño deben ser minimalistas.*"

Error de usabilidad:

Las tonalidades y la combinación de amarillo y celestes no eran recomendables para la aplicación, dado que no eran colores armónicos.

Se debe seleccionar de uno a tres colores armónicos, para que las emociones y sensaciones del usuario sean agradables.

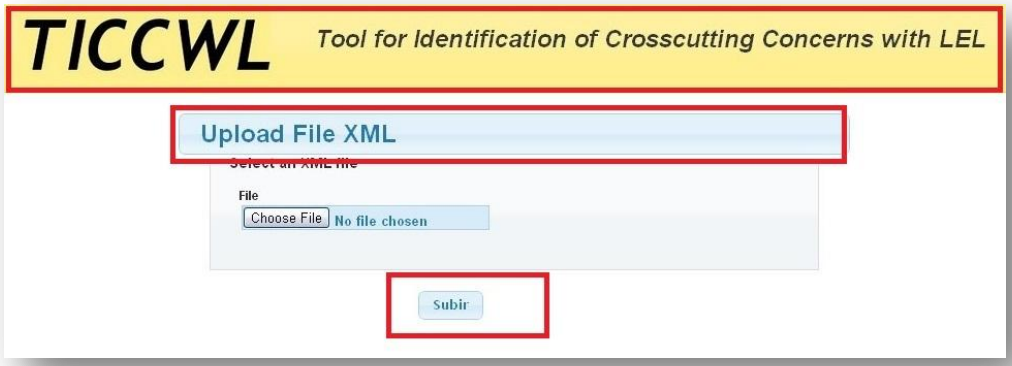

Figura 5.22 Contraste y combinación errónea de colores.

Modificación aplicada:

Se cambio la aplicación a las tonalidades de celestes y grises, para que sea armónica en cuanto a sus colores.

Figura 5.23 Nueva combinación de colores y tonalidades.

5.5.3.11 Diseño consistente de la aplicación

Heurística violada:

Heurística IV - "*Seguir estándares o convenciones de diseño ampliamente aceptados.*"

Error de usabilidad:

El uso del estilo curvo y estilo recto en distintos elemento y en distintas páginas hacen que la aplicación web no cuente con un diseño y estética consistente.

El diseño y la estética debe ser el mismo en toda la aplicación web (entre elementos y páginas) para poder lograr tener una aplicación consistente.

Figura 5.24 Distintos estilos de diseño.

Modificación aplicada:

Se aplico todo el estilo curvo a todos los elementos de todas las páginas para mantener una consistencia de diseño.

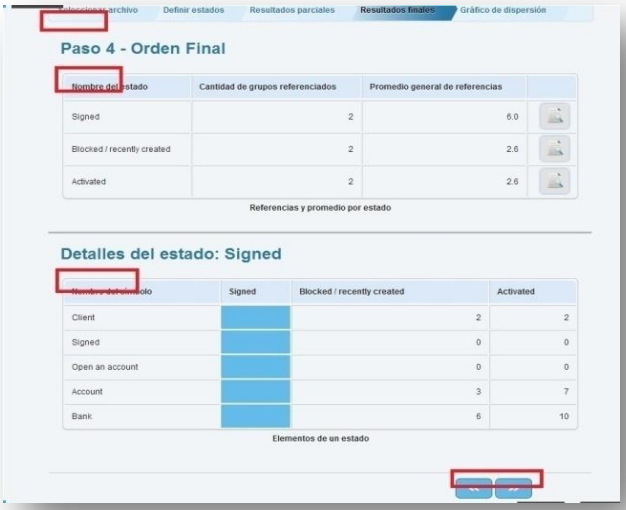

Figura 5.25 Estilo curvo de diseño.

5.5.3.12 Tooltips para las acciones

Heurística violada:

Heurística V - *"Se deben eliminar situaciones propensas a errores."*

Error de usabilidad:

No se tiene información en las distintas acciones o instrucciones que existen en la aplicación. Esto no permite que las acciones estén descriptas de una manera clara para aquellos usuarios inexpertos.

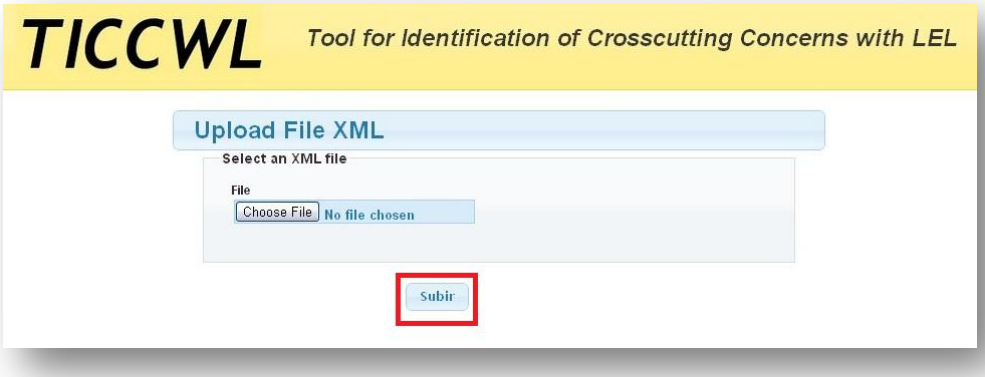

Figura 5.26 Ausencia de información para las acciones.

La aplicación web debe tener un diseño cuidadoso que evite la ocurrencia de errores.

Modificación aplicada:

El tooltips es un mecanismo que permite brindar información al usuario. Se incorporo para todos las acciones del sistema e imágenes los tooltips para proveer mayor información al usuario.

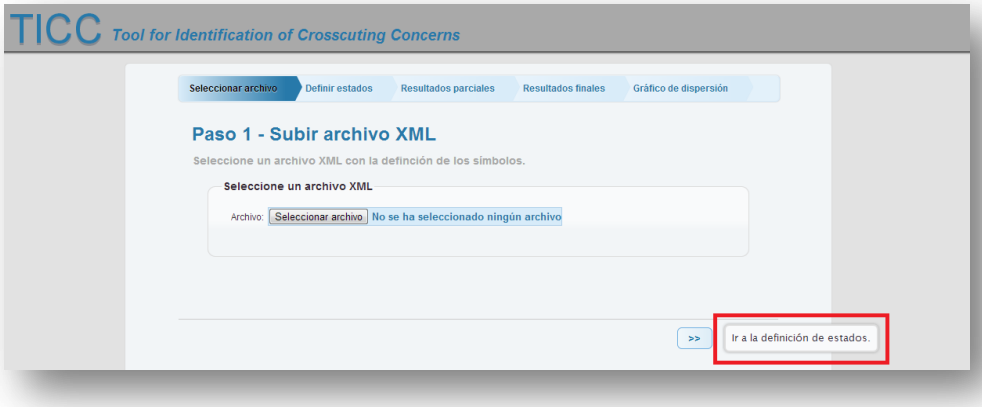

Figura 5.27 Incorporación de tooltips para las acciones.

5.6 Conclusión

En primer lugar se ha intentado dar una justificación de porque es importante y necesario tener en cuenta aspectos de usabilidad cuando se desarrolla una aplicación Web. La usabilidad es una característica de calidad fundamental de cualquier aplicación. Otorgándole un valor añadido que se plasmará tanto en el grado de utilización de la aplicación como en el grado de satisfacción de los usuarios.

Seguidamente se introdujo las técnicas existentes para la evaluación de usabilidad de las aplicaciones web, ya que la evaluación de la usabilidad es un proceso fundamental a tener en cuenta en el diseño de aplicaciones web.

En este capítulo hemos visto la evaluación de la aplicación TICC a través del método de evaluación heurística. Esta técnica es un método de inspección de sencilla aplicación que puede utilizarse en cualquier etapa del desarrollo de la aplicación web. La misma nos permitió

encontrar grandes y graves problemas de usabilidad que tenía el diseño de la aplicación. A partir de las modificaciones efectuadas a la aplicación se tuvo como resultado un mejor diseño orientado a la usabilidad.

Como trabajo futuro, resta realizar una segunda aplicación de la evaluación heurística y finalmente realizar los test de usabilidad para poder detectar los problemas de usabilidad más difíciles de encontrar.

Finalmente queremos enfatizar en que para lograr obtener una aplicación web con un correcto diseño, la misma debe cumplir con dos aspectos deseables: la usabilidad y la accesibilidad. La **accesibilidad** intenta vencer las discapacidades del usuario para acceder la información; mientras que la **usabilidad** busca mejorar la experiencia del usuario al usar las páginas web. Un diseño accesible preparado para personas, discapacitadas o no, debe ser usable. Por otro lado la usabilidad también se ve beneficiada por una buena accesibilidad. Entonces si una aplicación web no es usable, no puede considerarse accesible, y viceversa.

En el siguiente capítulo se presentara el aspecto de accesibilidad y la validación del nivel de accesibilidad para la aplicación TICC.

Capítulo 6

6. Accesibilidad Web

Un aspecto que hace a una aplicación Web fácil de usar es que además sea accesible. Una aplicación web es accesible cuando puede ser usada en forma eficaz por el mayor número posible de personas, independientemente de sus limitaciones personales.

En el presente capítulo se introduce el concepto accesibilidad en las aplicaciones Web y se describen las pautas de accesibilidad de contenidos web elaboradas por el Consorcio de la Web.

Posteriormente se realiza una descripción de cada una de las recomendaciones de la "Guía breve para crear sitios web accesibles" [W3C 2013b] y su aplicabilidad en la aplicación TICC.

Finalmente se afronta la necesidad de realizar evaluaciones automáticas del nivel de accesibilidad de las aplicaciones web, evaluaciones que no pueden automatizarse en su totalidad. Se introduce cada una de las herramientas de validación automática y se muestra la utilización de las mismas en la aplicación TICC.

6.1 Definición

A nivel más genérico el término accesibilidad se define "como el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o física" [Wikipedia 2013b]. Es indispensable e imprescindible, ya que se trata de una condición necesaria para la participación de todas las personas independientemente de las posibles limitaciones funcionales que puedan tener.

El término de accesibilidad web suele vincularse erróneamente únicamente con los requisitos que debe reunir una aplicación Web para que las personas con discapacidad puedan alcanzar la información que contiene. Las personas con discapacidad son más sensibles a la falta de accesibilidad, pero una aplicación web accesible permitirá una mejor interacción con todo el mundo.

La Wikipedia [Wikipedia 2013c] define que la **accesibilidad web** "*indica la capacidad de acceso a la web y a sus contenidos por todas las personas, independientemente de la discapacidad (física, intelectual o técnica) que presenten o de las que se deriven del contexto de uso (tecnológicas o ambientales)*". Esta cualidad está íntimamente relacionada con la usabilidad.

El Consorcio de la World Wide Web *(World Wide Web Consortium, W3C)* [W3C 2013a] define accesibilidad como "*el acceso universal (acceso de todos) a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios*".

Decimos entonces que la **accesibilidad web** tiene como objetivo lograr que las páginas web que conforman a una aplicación web sean utilizables por el máximo número de personas, independientemente de sus conocimientos o capacidades personales e independientemente de las características técnicas del equipo utilizado para acceder a la Web. Logrando así que el contenido de la aplicación Web pueda ser operado y recibido de múltiples modos.

Desde esta propuesta, quien diseña una aplicación web debe pensar en los posibles visitantes de su página y considerar a quienes accederán desde su discapacidad o desde contextos diversos y considerar estas diversas situaciones durante el diseño, de modo que el trabajo que realicen resulte de mayor alcance.

6.2 Relación entre Usabilidad y Accesibilidad Web

Desde el punto de vista conceptual ambos términos buscan el mismo objetivo: que el usuario pueda hacer mejor uso del software, en este caso de las páginas web que conforman a una aplicación web. La accesibilidad y la usabilidad son aspectos deseables (o incluso exigibles legalmente, en el caso de la accesibilidad) para tener una aplicación web con un correcto diseño de web.

La accesibilidad intenta vencer las discapacidades del usuario para acceder a la información; mientras que la usabilidad busca mejorar la experiencia del usuario al usar la aplicación web.

Resulta complicado encontrar una adecuada separación entre ambos conceptos, porque existe una correlación entre ambas. Una aplicación accesible preparada para personas, discapacitadas o no, debe ser usable.

Si existen dificultades en la navegación por una inadecuada usabilidad de la aplicación web, el problema se agravará para personas discapacitadas o que requieren de alguna asistencia para usar la aplicación web. Una aplicación cuyo menú sea poco usable presentará importantes complicaciones para una persona que esté en plenitud de sus facultades y que pueda utilizar un “ratón”, pero su uso le resultará casi imposible a una persona que tiene problemas físicos y debe utilizar un licornio o un sistema de reconocimiento de voz.

La usabilidad también se ve beneficiada por una buena accesibilidad. Una página accesible está concebida principalmente para aquellas personas que tienen dificultades de acceso, lo que obliga en primera instancia a realizar un esfuerzo extra al diseñar la interfaz, la cual además de ser rápida e intuitiva deberá disponer de atajos que faciliten la navegación.

La accesibilidad web sería entonces la capacidad de una aplicación web, para facilitarles a los usuarios (independientemente de sus niveles de discapacidad física o tecnológica) el acceso a la misma y a sus contenidos. Y la usabilidad sería una forma de medir lo fácil, rápido y agradable que resulta utilizar dicha la aplicación web. Ambos conceptos convergen en la búsqueda de la facilidad de acceso y consulta, por parte de los usuarios, a una aplicación web.

6.3 La Importancia de la Accesibilidad

En los últimos años la web ha pasado a ser la mayor fuente de información gratuita disponible para todo el mundo y un medio destacado de participación en la sociedad. Se ha convertido en un recurso fundamental para distintas aéreas, posibilitando un acceso a la información sin precedentes. La aplicación de la accesibilidad, además de permitir y mejorar el acceso de las personas con discapacidad a los contenidos Web, conlleva también otros beneficios:

- **Aumenta el número de potenciales visitantes**: esta es una razón muy importante para una empresa que pretenda captar nuevos clientes. Cuando una aplicación web es accesible no presenta barreras que dificulten su acceso, independientemente de las condiciones del usuario. Una aplicación web que cumple los estándares es más probable que se visualice correctamente en cualquier dispositivo con cualquier navegador.

- **Disminuye los costes de desarrollo y mantenimiento**: aunque inicialmente aprender a hacer una aplicación web accesible supone un coste (igual que supone un coste aprender a utilizar cualquier tecnología nueva), una vez se tienen los conocimientos, el coste de desarrollar y mantener una aplicación web accesible es menor que frente a una no accesible, ya que una aplicación web accesible es una aplicación bien hecha, menos propensa a contener errores y más sencilla de actualizar.

- **Mejoramos la eficiencia y el tiempo de respuesta**. Las páginas están limpias de código inútil o poco eficiente, pesan menos, por lo que el tiempo de carga es mucho menor.

- **Reduce el tiempo de carga de la aplicación web y la carga del servido: A**l separar el contenido de la información sobre la presentación de una página web mediante CSS se logra reducir el tamaño de las páginas web y, por tanto, se reduce el tiempo de carga de las páginas web.
- **Aumenta la usabilidad de la aplicación web**: las aplicaciones web accesibles son en general más “usables” para todo el mundo. Conceptos como la sencillez, facilidad de manejo y navegación, y eficiencia, se manejan en ambas disciplinas.

- **Demuestra responsabilidad social:** Ayuda a mejorar la imagen de nuestra empresa. La eliminación de barreras que dificulten el acceso a nuestra web demuestra una preocupación y atención hacia todos los clientes.

- **Evidencia el cumplimiento de la ley:** En algunos países la accesibilidad es ley. En argentina ya se dispone de una ley de accesibilidad.

La Discapacidad es una parte, una amplia e importante parte, pero parte al fin, de la gran cantidad de beneficiarios de la Accesibilidad Web, ya que esta también incluye el considerar a las personas cuyas infraestructuras de comunicación o capacidades no son avanzadas discapacidad por distancia o falta de recursos así como también a las muy avanzadas discapacidad por exceso de recursos. Por eso la Accesibilidad nos beneficia a todos.

6.4 Pautas de Accesibilidad

Con el fin de que los desarrolladores web sepan cómo diseñar y construir aplicaciones web accesibles, diferentes conjuntos de pautas o guías de accesibilidad han sido desarrollados por diversos organismos. En la actualidad, las pautas más importantes son las pautas de accesibilidad al contenido web desarrolladas por el Consorcio de la World Wide Web *(World Wide Web Consortium, W3C)* a través de la Iniciativa para la Accesibilidad a la Web *(Web Accessibility Initiative, WAI)*.

W3C [W3C 2013a] es una organización internacional en la que las organizaciones miembro, personal a tiempo completo y público en general, trabajan en conjunto para desarrollar estándares Web. A comienzos de 1998, lanzó WAI [WAI 2013] que se enfoca en extender los protocolos y formatos de datos para hacer la Web más Accesible.

La WAI aúna los conocimientos tecnológicos e investigaciones en temas relativos a la accesibilidad; para crear pautas de accesibilidad en la Web y poder garantizar que éstas sean adecuadas para las tecnologías del W3C (tales como HTML, XML, SVG, etc.). WAI ha

desarrollado diversas guías y documentos para impulsar la creación de sitios web accesibles, tanto para los contenidos web, como para las herramientas de autor (herramientas que generan contenido Web accesible), agentes de usuario (hacer accesible los navegadores, reproductores multimedia y otras tecnologías asistidas) y especificaciones técnicas del W3C.

Entre todos ellos, destacan las Pautas para la Accesibilidad al Contenido Web *(Web Content Accessibility Guidelines, WCAG)*, que incluye las características que debe cumplir una web para que sea accesible para todos. En 1991 la WAI publicó la versión 1.0 [WCAG 1.0 2013]. Con el paso del tiempo se han convertido en un referente internacionalmente aceptado. En diciembre del 2008 las WCAG 2.0 [WCAG 2.0 2013] fueron aprobadas como recomendación oficial.

Las Pautas WCAG 1.0, que propone la WAI están organizadas en catorce pautas, que son los principios generales para el diseño accesible. A continuación se enumera cada una de ellas:

1. Proporcione alternativas equivalentes para el contenido visual y auditivo.
2. No se base sólo en el color.
3. Utilice marcadores y hojas de estilo y hágalo apropiadamente.
4. Identifique el idioma y los cambios de idioma.
5. Cree tablas que se transformen correctamente.
6. Asegúrese de que las páginas que incorporen nuevas tecnologías se transformen correctamente.
7. Asegure al usuario el control sobre los cambios de los contenidos tempo-dependientes.
8. Asegure la accesibilidad directa de las interfaces incrustadas.
9. Diseñe para la independencia del dispositivo.
10. Utilice soluciones provisionales para que las ayudas técnicas y los antiguos navegadores operen correctamente.
11. Utilice las tecnologías y pautas W3C (de acuerdo con las especificaciones) y siga las pautas de accesibilidad.
12. Proporcione información de contexto y orientación.
13. Proporcione mecanismos claros y coherentes de navegación.
14. Asegúrese de que los documentos sean claros y simples en cuanto a su comprensión.

Cada pauta a su vez está asociada a uno o más puntos de verificación que describen cómo aplicar esa pauta a las características particulares de las páginas web. En total hay 65 puntos de verificación.

Cada punto de verificación está asignado a uno de los tres niveles de prioridad que indica cómo afecta a la accesibilidad de un sitio Web si dicho punto de verificación no se cumple. Es decir, según cuál sea su impacto en la accesibilidad:

- **Prioridad 1**: son aquellos puntos que un desarrollador web **tiene que cumplir** ya que, de lo contrario, ciertos grupos de usuarios **no podrían acceder** a la información de la aplicación web.

- **Prioridad 2**: son aquellos puntos que un desarrollador web **debería cumplir** ya que, si no fuese así, a ciertos grupos de usuarios les sería **muy difícil** el acceso a la información.

- **Prioridad 3**: son aquellos puntos que un desarrollador web **debería cumplir** ya que, de otra forma, algunos usuarios experimentarían ciertas **dificultades para acceder** a la información.

En función de estos puntos de verificación se establecen los niveles de conformidad:

- **Nivel de Conformidad "A" (A):** todos los puntos de verificación de prioridad 1 se satisfacen.

- **Nivel de Conformidad "Doble A" (AA):** todos los puntos de verificación de prioridad 1 y 2 se satisfacen.

- **Nivel de Conformidad "Triple A" (AAA):** todos los puntos de verificación de prioridad 1, 2 y 3 se satisfacen.

Cada uno de estos grados de cumplimiento lleva asociado un icono de conformidad, definido por el W3C y que cada sitio web puede colocar en cuanto se cumplan los requisitos de accesibilidad pertinentes, con el fin de auto declarar su grado de accesibilidad.

Figura 6.1 Iconos de los grados de cumplimiento de WCAG 1.0

Las pautas describen cómo hacer páginas Web accesibles sin sacrificar el diseño, ofreciendo esa flexibilidad que es necesaria para que la información sea accesible en diferentes situaciones y proporcionando métodos que permiten su transformación en páginas útiles e inteligibles. Una aplicación web es Accesible si su contenido pueda ser operado y recibido de múltiples modos y si cumple con las Pautas propuestas por la WAI.

6.5 Guía breve para crear sitios accesibles

Desarrollar un sitio web accesible puede ser algo sencillo o complejo y depende de muchos factores como por ejemplo, el tipo de contenido, el tamaño y la complejidad del sitio, así como de las herramientas de desarrollo y el entorno.

La W3C provee una "Guía breve para crear sitios web accesibles" [W3C 2013b]. La cual no es una guía completa, simplemente es un extracto de las "Pautas de accesibilidad al contenido en la Web" *(Web Content Accessibility Guidelines)* [WCAG 1.0 2013]. Si al diseñar un sitio Web tenemos en cuenta este sencillo decálogo, tendremos la seguridad de haber satisfecho los requerimientos básicos del diseño accesible de sitios Web.

Esta guía posee tan solo 10 recomendaciones que recogen los fundamentos claves para diseñar de forma accesible un sitio Web. A continuación se enumera cada una de las recomendaciones de la guía y se describe su implementación para poder cumplirla.

6.5.1 Consejo 1: Imágenes y animaciones

"Use texto alternativo, atributo ALT, para describir la función de cada elemento visual."

- SHAPE ="default": define la acción por defecto para toda la imagen.
- SHAPE ="rect": define una región rectangular; el atributo COORDS contiene las coordenadas de la esquina superior izquierda y de la esquina inferior derecha del rectángulo.
- SHAPE ="circle": define una región circular; el atributo COORDS contiene las coordenadas del centro del círculo y del radio.
- SHAPE ="poly": define una región poligonal; el atributo COORDS contiene las coordenadas de cada uno de los puntos que forman el polígono; el último punto tiene que coincidir con el primero para que el polígono se cierre.

Para que el mapa de imagen sea accesible, se tiene que proporcionar un texto alternativo con el atributo alt para cada etiqueta < AREA >. Como cada zona activa realiza la misma función que un enlace, el texto alternativo tiene que ser eficaz, en especial, el texto alternativo tiene que tener sentido cuando se lea fuera de contexto. Desgraciadamente, la etiqueta < AREA > no posee el atributo longdesc para proporcionar una descripción más larga que con el atributo ALT.

El elemento MAP puede ser asociado con otros elementos (IMG, OBJECT, INPUT). Un mapa de imágenes se asocia a un elemento a través del atributo usemap del elemento.

Un lector de pantalla que esté preparado para interpretar los mapas de imagen comunicará al usuario el texto alternativo de cada una de las zonas sensibles definidas en el mapa de imagen. Si el lector de pantalla no interpreta correctamente el atributo usemap asociado a una imagen, la W3C propone el uso de la etiqueta OBJECT en vez de la etiqueta IMG ya que la etiqueta <object> permite incluir contenido alternativo que se muestra en el caso de que el agente de usuario (navegador) no pueda mostrar la etiqueta<object>.

6.5.2 Consejo 2: Mapa de Imágenes

*"Use el elemento **MAP** y texto para las zonas activas."*

Los mapas de imágenes nos permiten especificar regiones en una imagen u objeto y asignar una acción específica a cada región (p.ej. abrir un documento, ejecutar un programa, etc.). Cuando la región es activada por el usuario, se ejecuta la acción.

Un mapa de imágenes se crea asociando un objeto con una especificación de las áreas geométricas sensibles del objeto.

Existen dos tipos de mapas de imágenes:

- **En el lado del cliente**. Cuando un usuario activa una región de un mapa de imágenes en el lado del cliente con un ratón, las coordenadas en píxeles son interpretadas por el agente de usuario. El agente de usuario selecciona el vínculo especificado por la región activada y lo sigue.
- **En el lado del servidor**. Cuando un usuario activa una región de un mapa de imágenes en el lado del servidor, las coordenadas en píxeles son enviadas al agente del lado del servidor especificado por el atributo HREF del elemento A. El agente del servidor interpreta las coordenadas y realiza alguna acción.

Se prefieren los mapas de imágenes en el cliente, dado que son accesibles a las personas que utilizan agentes de usuario no gráficos y permiten saber en todo momento si el apuntador está

sobre una región activa o no. Por lo tanto, se tienen que emplear los mapas de imagen en el lado del cliente para evitar problemas de accesibilidad.

Un mapa de imagen en el cliente se define con la etiqueta <MAP> y cada una de las zonas activas se define con la etiqueta<AREA>. La etiqueta < AREA > posee los atributos SHAPE y COORDS (su valor depende del valor de SHAPE), que se emplean para definir la forma geométrica de la zona activa, y el atributo HREF, que se emplea para indicar la acción asociada:

- SHAPE ="default": define la acción por defecto para toda la imagen.
- SHAPE ="rect": define una región rectangular; el atributo COORDS contiene las coordenadas de la esquina superior izquierda y de la esquina inferior derecha del rectángulo.
- SHAPE ="circle": define una región circular; el atributo COORDS contiene las coordenadas del centro del círculo y del radio.
- SHAPE ="poly": define una región poligonal; el atributo COORDS contiene las coordenadas de cada uno de los puntos que forman el polígono; el último punto tiene que coincidir con el primero para que el polígono se cierre.

Para que el mapa de imagen sea accesible, se tiene que proporcionar un texto alternativo con el atributo alt para cada etiqueta < AREA >. Como cada zona activa realiza la misma función que un enlace, el texto alternativo tiene que ser eficaz, en especial, el texto alternativo tiene que tener sentido cuando se lea fuera de contexto. Desgraciadamente, la etiqueta < AREA > no posee el atributo longdesc para proporcionar una descripción más larga que con el atributo ALT.

El elemento MAP puede ser asociado con otros elementos (IMG, OBJECT, INPUT). Un mapa de imágenes se asocia a un elemento a través del atributo usemap del elemento.

Un lector de pantalla que esté preparado para interpretar los mapas de imagen comunicará al usuario el texto alternativo de cada una de las zonas sensibles definidas en el mapa de imagen. Si el lector de pantalla no interpreta correctamente el atributo usemap asociado a una imagen, la W3C propone el uso de la etiqueta OBJECT en vez de la etiqueta IMG ya que la etiqueta <object> permite incluir contenido alternativo que se muestra en el caso de que el agente de usuario (navegador) no pueda mostrar la etiqueta<object>.

6.5.3 Consejo 3: Multimedia

"Proporcione subtítulos y transcripción de los archivos de sonido, descripción de los videos y versiones accesibles en el caso de usar formatos no accesibles."

Los elementos multimedia pueden ocasionar graves problemas de accesibilidad, ya no sólo a las personas con algún tipo de discapacidad, sino a todo el mundo en general. Esto se debe a que al ser elementos que no son HTML requieren, en la mayoría de los casos, la instalación de un visor específico (plug-in, add-in o extensión) que sea capaz de interpretar el elemento multimedia.

Por tanto, como regla general, no se debe abusar de los elementos multimedia y se debe analizar la inclusión del mismo en el sitio web, si es un elemento esencial que no se puede eliminar o sustituir por otro más accesible.

Si nos centramos en los problemas de accesibilidad que pueden tener las personas con discapacidad, los elementos multimedia más problemáticos son los vídeos, las grabaciones sonoras y las animaciones.

Respecto a los vídeos y las grabaciones sonoras, en ambos casos se tienen que proporcionar una transcripción de los diálogos y una descripción de los sonidos. En el caso de los vídeos también se tiene que proporcionar una descripción del vídeo en sí (de la imagen).

6.5.4 Consejo 4: Enlaces de Hipertexto

"Use texto que tenga sentido cuando se lea fuera de contexto. Por ejemplo, evite usar el "pincha aquí"."

Algunos navegadores y algunos los programas de ayuda que emplean las personas con discapacidad (por ejemplo, los lectores de pantalla) ofrecen al usuario la posibilidad de mostrar, normalmente una ventana aparte, la lista de enlaces que contiene una página web. Esta lista de enlaces normalmente permite activar un enlace y navegar a la página de destino. Si el texto de un enlace no tiene sentido fuera de su contexto, el enlace no tendrá sentido en esta lista de enlaces.

Por otro lado, si los enlaces poseen un estilo especial para resaltarlos, los usuarios suelen fijar su atención en ellos, por lo que es importante que el texto de los enlaces sea lo más claro y significativo. Para lograr escribir enlaces correctos, se tienen los siguientes consejos breves:

- Tiene que ser claro y corto, pero no tan corto que sea casi inapreciable para el usuario cuando lo escuche.

- Tiene que tener sentido fuera de contexto, por ejemplo, cuando se lea él sólo o como parte de una lista de enlaces de una página.
- Tiene que tener sentido en el contexto, por ejemplo, cuando se lea como parte del resto de la página.

En algunos tipos de sitios web, como portales de noticias, de productos o blogs, es muy común mostrar un listado de elementos (noticias, productos o entradas) con un cierto texto breve y con un enlace para obtener el texto completo asociado al elemento (la noticia completa, la descripción completa del producto o la entrada completa). En estos enlaces se suelen emplear textos como "Más información", "Más info", "Más detalle", "Leer más", "Ver más" o incluso simplemente unos puntos suspensivos "...". Estos textos no son descriptivos y no indican el contenido que se va a encontrar, además los mismo pueden aparecer muchas veces en una página. Todo esto plantea un problema de accesibilidad.

Para poder solucionar este problema de accesibilidad es necesario que se use distinto contenido en cada enlace, además se debe especificar el atributo TITLE. Si utilizamos el atributo TITLE en los enlaces y se informa del destino del enlace, los usuarios de navegadores visuales se benefician porque al situar el cursor del ratón sobre un enlace aparece una nota con el valor del atributo TITLE.

6.5.5 Consejo 5: Organización de las Páginas

"Use encabezados (H1,H2, H3,…), listas y estructura consistente. Use hojas de estilos en Cascada (CSS) para la maquetación y estilo, donde sea posible."

Las páginas web tienen que estar correctamente estructuradas. Para ello, se tienen que emplear las etiquetas de HTML que definen la estructura de una página, como son: <TITLE>, <H1>, <H2>, ..., <UL>, <OL>, <P>, <BLOCKQUOTE>.

Título de la página. Toda página web debe tener un titulo y el mismo debe ser definido con la etiqueta < TITLE > y debe resumir el contenido o la función de la página. El título tiene que ser descriptivo pero a la vez breve.

El título de la página se emplea en los marcadores de los navegadores y en los resultados que muestran los buscadores. El título de la página es lo primero que leen los lectores de pantalla y lo primero que aparece en los navegadores no visuales. Por ello, es importante describir correctamente el contenido de la página. Pero como se lee en todas las páginas, debe ser corto y no repetitivo (no debe ser el mismo en todas las páginas). Es conveniente incluir el nombre del sitio web para definir el contexto de la página, pero el nombre debe ser breve y significativo.

Encabezados de la página. El contenido de las páginas se tiene que estructurar con las etiquetas de encabezado <H1>, <H2>,... La mayoría de los lectores de pantalla y algunos navegadores permiten al usuario desplazarse dentro de una página web "saltando" de un encabezado a otro encabezado. Eso permite llegar de una forma más rápida a la información que se busca.

Muchos desarrolladores de páginas web emplean las etiquetas <DIV> y <SPAN> para definir encabezados (títulos) de sección y estructurar el contenido. Sin embargo, estas etiquetas no proporcionan contenido semántico, por lo que se debe evitar su uso. Lo correcto es utilizar las etiquetas de encabezado (<H1> ... <H6>) y definir su presentación visual con CSS.

Existen algunas reglas de aplicación general para la aplicación de los encabezados:

- Se debe utilizar un único encabezado <H1> para el título principal del sitio web, como si fuera el título de un libro.
- Se debe utilizar diferentes encabezados <H2> para cada apartado principal de la página (barra de navegación, contenido principal, pie de página), como si fuera la tabla de contenidos del libro.
- Emplea el resto de encabezados (<H3> ... <H6>) para añadir mayor nivel de detalle en la estructura de la página. No todos los apartados pueden necesitar tanto nivel de detalle. Se debe intenta no crear estructuras con mucha profundidad, normalmente no es necesario llegar a <H5> y <H6>.
- Se de ser consistente en el uso de los encabezados y no se debe dejar huecos: por ejemplo, no se debe pasar de <h2> a <h4> sin utilizar <h3>.
- Si se necesita que los encabezados no se vean en un navegador visual, se debe utilizar CSS para esconderlos. Para ello, no se debe usar el display: none, sino que se debe desplazar el encabezado fuera del área de visualización con posicionamiento absoluto. Por ejemplo, position: absolute; top: -500px;.

Tablas. Para maquetar una página web nunca se debe utilizar las tablas, ya que suponen un grave problema de accesibilidad. Una página web se puede y debe maquetar con CSS y obtener el mismo resultado que se obtendría con tablas.

6.5.6 Consejo 6: Figuras y Diagramas

*"Descríbalos brevemente en la página o use el atributo **LONGDESC**."*

Este consejo está relacionado con el consejo 1 Imágenes y animaciones.

El atributo longdesc complementa al atributo ALT y se emplea para ofrecer una descripción más larga del elemento que la proporcionada por el atributo alt. Las etiquetas de XHTML que admiten este atributo en la versión 1.0 son: IMG, IFRAME y FRAME.

Mientras que el atributo ALT contiene el texto alternativo de la imagen, el atributo LONGDESC contiene una dirección de Internet a otra página web o a la misma página web donde se encuentra la descripción larga de la imagen.

El atributo longdesc es invisible e inaccesible en algunos navegadores, ya que los mismo no proveen un soporte para el mismo. Las personas que no utilizan lectores de pantalla no podrán acceder a dicha descripción. Las únicas personas que pueden acceder al atributo longdesc fácilmente son los que utilizan **lectores de pantalla modernos**. Los lectores de pantalla antiguos no apoyan este atributo. Incluso entre aquellos que utilizan la última versión del lector de pantalla, hay muchos que no están familiarizados con el atributo longdesc (ya que se utiliza con poca frecuencia), y **no saben cómo acceder a la descripción** a pesar de que su lector de pantalla lo soporta.

La forma más sencilla de hacer accesibles las descripciones largas es que sean obvias y al alcance de todos, independientemente si tienen una discapacidad o no. Por ello es recomendable incluir la descripción en la misma página o crear un enlace que permita ofrecer una descripción más larga de las figuras y/o diagramas, además de especificar el valor asociado al atributo LONGDESC.

6.5.7 Consejo 7: Scrips, applets y plug-ins

"Ofrezca contenido alternativo en el caso de que las características activas no sean accesibles o no tengan soporte."

Algunos de los navegadores que emplean las personas discapacitadas no son capaces de interpretar el código de script (JavaScript) o algunos elementos multimedia como applets programados en Java u objetos realizados con Macromedia Flash que requieren de un plug-in. Además, aún en el caso de que pudiesen interpretarlos, sería muy difícil proporcionar una representación alternativa (por ejemplo, una representación textual para una animación de un applet).

Por regla general, el HTML Dinámico (DHTML) no funcionará con un navegador no visual y no será accesible. Cualquier efecto que se base en mostrar u ocultar capas como respuesta a un evento del ratón, como por ejemplo, menús desplegables o información adicional al pasar el ratón por encima de un elemento, no será accesible y es necesario proporcionar una alternativa.

La solución más adecuada es lograr que una página web funcione correctamente sin necesidad de tener que ejecutar el código JavaScript. Esta técnica se conoce como JavaScript no molesto. El JavaScript no molesto es un paradigma floreciente en el uso del lenguaje de programación JavaScript, utilizado en la Web. Sus principios generalmente incluyen:

- La separación de la funcionalidad JavaScript (lógica - "capa del comportamiento") de las capas de estructura/contenido (HTML) y de presentación de un página (CSS).

- Uso de buenas prácticas a fin de evitar los problemas de incompatibilidad de la programación tradicional en JavaScript (tales como inconsistencias entre navegadores y falta de escalabilidad).

El objetivo final es que las páginas web sean totalmente funcionales para aquellos usuarios que no puedan o no quieran hacer uso de JavaScript. Esta técnica se basa en lo que se conoce en inglés como progressive enhancement y graceful degradation, dos estrategias que permiten que un sistema informático (en este caso, una página web) funcione correctamente aun en el caso de que falte algún tipo de componente. Mientras que con progressive enhancement se parte de una versión básica completamente operativa (se parte de una página web compatible con la mayoría de los navegadores y con el menor uso posible de tecnologías complementarias como CSS o JavaScript), con graceful degradation se parte del extremo contrario: se crea una página web para los últimos navegadores, con la posibilidad de que funcione en navegadores antiguos.

6.5.8 Consejo 8: Marcos (Frames)

"Use el elemento NOFRAMES y etiquete con los atributos TITLE o NAME."

Los marcos (*frames*) son un elemento del HTML que siempre han causado problemas. Tanto es así que en XHTML 1.0 Strict y Transitional no se pueden emplear y tenemos que utilizar XHTML 1.0 Frameset si queremos tener marcos

Los principales problemas, en el uso de marcos, es que algunos buscadores no están preparados para buscar en un sitio web desarrollado con marcos, si hay algún error se puede llegar a una situación de múltiples marcos anidados y están desaconsejados por el W3C.

De cara a lograr la máxima accesibilidad posible, es mejor evitar el uso de marcos. Hoy en día, existen diferentes técnicas y la mayoría de las herramientas de autor están preparadas para incluir automáticamente un elemento, como un menú de navegación o un pie de página en un lugar determinado de todas las páginas, con lo que se consigue la misma función que con el uso de marcos pero sin sus inconvenientes.

En el caso, que aún se requiera el uso de los marcos. Se debe proveer para cada uno de ellos un título y una descripción. Para ello, se tienen que emplear los atributos TITLE y longdesc en la etiqueta <FRAME />. El atributo longdesc realiza el mismo papel que con la etiqueta <IMG /> y que se explicó en el consejo 6 Figuras y diagramas.

Además, se tiene que proporcionar una versión alternativa sin marcos para aquellos agentes de usuario (navegadores) que no sean capaces de interpretar los marcos. Para ello se tiene que emplear la etiqueta <NOFRAMES>.

6.5.9 Consejo 9: Tablas

"Realícelas de manera que se puedan leer línea a línea. Incluya un resumen. Evite el uso de tablas para dar formato a las páginas."

Una tabla de datos definida con la etiqueta <TABLE> es muy fácil de entender si se puede ver toda ella en su conjunto, pero es muy difícil de entender si sólo se puede ver un dato aislado cada vez. Este problema lo sufren los usuarios que emplean navegadores no visuales, ya que ellos tienen que recorrer las tablas de manera lineal pero la naturaleza de una tabla es bidimensional, por lo que pierden la visión global de la tabla y pierden las referencias que permiten interpretar el contenido de cada celda de la tabla.

Para evitar este problema, se tienen que etiquetar correctamente las tablas para definir su título, se tiene que incluir un resumen que describa brevemente el contenido de la tabla, se tienen que definir los encabezamientos de las columnas y las filas y se tienen que emplear

etiquetas y atributos especiales para asociar las celdas de encabezamiento y las celdas de datos para los encabezamientos más complejos.

Título. Una tabla debe tener un título que proporcione una descripción breve de la tabla. Para evitar que existan dudas, el título se tiene que definir dentro de la tabla. Para definir correctamente el título de una tabla, se tiene que emplear la etiqueta <CAPTION>.

De acuerdo con la especificación de caption en HTML, la etiqueta <CAPTION> es opcional y tiene que ser el primer elemento que contenga una tabla (justo después de la etiqueta <TABLE>) y sólo puede haber un título en una tabla.

Resumen. El resumen permite definir una descripción larga de la tabla que complemente al título de la tabla. La descripción tiene que incluir una explicación sobre el contenido y sobre la estructura de la tabla (por ejemplo, debe explicar el propósito de la tabla, el número de filas y de columnas que contiene y una descripción de los encabezamientos). Además, también puede explicar la relación que guarda la tabla con el resto de la página.

En HTML, el resumen de una tabla se define con el atributo SUMMARY de la etiqueta <TABLE>. Según la especificación de summary, los navegadores no visuales, como los lectores de pantalla o los basados en el uso de una línea braille, deben transmitir el contenido de este atributo a los usuarios.

Encabezados. Al recorrer una tabla de forma lineal, se pierde la visión global y es muy difícil identificar el significado de un dato. De forma aislada, el contenido de una celda puede no tener sentido si no se sabe en qué fila y en qué columna está situado. Para evitar esta situación, podemos usar los encabezamientos (o encabezados), que permiten asociar un dato con su encabezado.

Un encabezado de una tabla se define con la etiqueta <TH>. Esta etiqueta es similar a la etiqueta <TD> (se puede usar una en el lugar de la otra) y por tanto, ambas definen una celda de una tabla, pero <TH> indica que la celda tiene una función especial y contiene un encabezado. Con la etiqueta < TH> se pueden definir tanto encabezados verticales como horizontales.

Encabezados más complejos. Para tablas con encabezamientos más complejos, donde pueden existir varios niveles de encabezamiento, podemos emplear los atributos **scope** y **headerspara** definir la relación que existe entre las celdas de encabezamiento y las celdas de datos.

En tablas sencillas ambos atributos se pueden emplear de forma equivalente, pero para tablas más complejas se tiene que emplear el atributo headers. En principio, el uso de los atributos **scope** y **headers** no afecta a la presentación visual de la tabla.

El atributo **scope** define el conjunto de celdas para las cuales la celda sobre la que se aplica proporciona información de encabezamiento. Puede tomar cuatro posibles valores:

- row: La celda proporciona información de encabezamiento para el resto de celdas de la fila que la contiene.
- col: La celda proporciona información de encabezamiento para el resto de celdas de la columna que la contiene.
- rowgroup: La celda proporciona información de encabezamiento para el resto del grupo de filas que la contiene.
- colgroup: La celda proporciona información de encabezamiento para el resto del grupo de columnas que la contiene.

El atributo **headers** permite definir una lista de celdas de la tabla que proporcionan información de encabezamiento para la celda actual. El valor de este atributo es una lista separada por espacios en blanco de los identificadores de las celdas de encabezamiento; las celdas de encabezamiento se identifican con el atributo id. Por tanto, el atributo **headers** permite definir encabezamientos más complejos que con el atributo **scope**.

6.5.10 Consejo 10: Revise su trabajo

"Valide el código HTML. Use las herramientas de evaluación, puntos de comprobación y pautas de la W3C."

En primer lugar hay que revisar la sintaxis y semántica de los lenguajes empleados en la creación de las páginas web. Para revisar y validar el lenguaje XHTML se puede emplear la página Markup Validation Service de W3C [MVS 2013].

También se debe revisar y validar el lenguaje CSS. Para revisar y validar el lenguaje CSS se puede emplear la página CSS Validation Service [CSSVS 2013] también de W3C.

Para revisar los puntos de comprobación y las pautas de WCAG 1.0 no existe una herramienta automática que pueda sustituir la revisión de un experto humano. Sin embargo, existen algunas herramientas que pueden ayudar al proceso de revisión, principalmente con aquellos puntos de verificación que sólo supongan la existencia de ciertos atributos o el correcto uso de algunas etiquetas de XHTML.

Esta guía es únicamente una serie de consejos que ayudan cuando se tiene un "primer contacto" con la accesibilidad web. Su cumplimiento no garantiza que se que se alcance algún nivel de adecuación (A, AA, AAA).

6.6 TICC y la aplicación de la guía breve para crear sitios accesibles

Muchas de las características accesibles de una aplicación web se implementan de forma sencilla si se planean desde el principio del desarrollo de la aplicación Web o al comienzo de su rediseño. Es por ello que la aplicación TICC fue modificada siguiendo las recomendaciones de la "Guía breve para crear sitios web accesibles" [W3C 2013b].

A continuación se describe por cada una de los consejos de la "Guía breve para crear sitios web" según el caso los cambios que se efectuaron en la aplicación TICC.

6.6.1 Consejo 1: Imágenes y animaciones

*"Utilice texto alternativo, atributo **ALT**, para describir la función de cada elemento visual."*

En la aplicación TICC solo se hizo uso de una imagen para proveer la funcionalidad de ver los detalles de un estado. En el siguiente código se muestra el cambio realizado sobre la etiqueta IMG:

```
<c:forEach items="${finalResults}" var="finalState">
    <tr>
        <td id="state_${finalState.name}">${finalState.name}</td>
        <td class="number-cell">${finalState.groupNameReferences}</td>
        <td class="number-cell">${finalState.average}</td>
        <td class="number-cell" id="state_${finalState.idState}">
            <a id="image-container">
                <img
                    alt="<spring:message code='concerns.table.actions.view.details.alt'/>" [1]
                    src="<c:url value="/resources/image/view-detail.png"/>"/>
            </a>
        </td>
    </tr>
</c:forEach>
```

Figura 6.2 Definición del atributo alt a las imágenes

A cada uno de los elementos <IMG> se le asocio el atributo ALT con su valor internacionalizado [1]. En el archivo messages.properties (archivo para mensajes de internacionalización), se define la clave asociada al atributo ALT y su valor.

concerns.table.actions.view.details.alt = Ver símbolos del estado.

Figura 6.3 Internacionalización del atributo alt de las imágenes.

6.6.2 Consejo 2: Mapa de Imágenes

*"Utilice el elemento **map** y texto para las zonas activas"*

Este consejo no pudo ser aplicado sobre algún elemento de la aplicación TICC, ya que la misma no cuenta con una imagen que necesite ser descripta por regiones como así tampoco asociarle acciones a cada región.

6.6.3 Consejo 3: Multimedia

"Proporcione subtítulos y transcripción de los archivos de sonido, descripción de los videos y versiones accesibles en el caso de usar formatos no accesibles."

En la aplicación TICC no fue necesario la incorporación de elementos multimedia, es por ello que este consejo no pudo ser aplicado en la misma.

6.6.4 Consejo 4: Enlaces de Hipertexto

"Utilice texto que tenga sentido cuando se lea fuera de contexto."

La aplicación TICC cuenta con unos enlaces que nos permiten navegar a través de la aplicación. A cada uno de ellos se le incorporo el atributo TITLE con información que permita saber la funcionalidad del mismo.

A continuación se presenta, como ejemplo, uno de los cambios realizado en los enlaces de navegación de la página de "Cálculo de referencias".

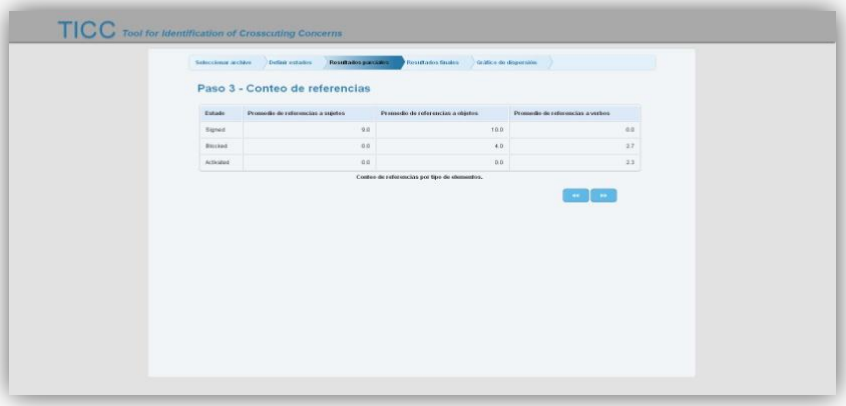

Figura 6.4 Página de cálculo de referencias

En el siguiente código se puede observar el atributo TITLE asociado a cada uno de los enlaces:

```
<a title= '<spring:message code="concerns.table.graph.definition.title"/>' [1]
   href='${context}graph/define-states'>
        <spring:message code="concerns.table.graph.definition"/>
</a>
<a title= '<spring:message code="concerns.table.final.results.title"/>' [2]
   href='${context}results/final'>
        <spring:message code="concerns.table.final.results"/>
</a>
```

Figura 6.5 Definición del atributo title para los enlaces

Los valores asociados al atributo TITLE [1] y [2] se definieron para que los mismos sean internacionalizables. Los valores de cada clave de internacionalización son las siguientes:

```
concerns.table.final.results.title= Ir a los conteos finales.
concerns.table.graph.definition.title= Volver a la definición de estados.
```

Figura 6.6 Internacionalización del atributo title para los enlaces

6.6.5 Consejo 5: Organización de las Páginas

"Utilice encabezados, listas y estructura consistente. Use CSS para la maquetación donde sea posible."

En cuanto al título de la página, la aplicación TICC ya contaba con la definición del mismo en todas las páginas que la conforman.

Como en todos los casos anteriores, el título fue definido para permitir la internacionalización del mismo.

Las páginas de la aplicación TICC fueron estructuradas utilizando las etiquetas de encabezado. Por lo que también siguen los lineamientos indicados para los encabezados de las páginas.

Las páginas de TICC fueron maquetadas a través del uso de las etiquetas DIV y SPAN. Solamente se utilizaron las tablas para mostrar aquella información de tabla de datos.

6.6.6 Consejo 6: Figuras y Diagramas

"Descríbalos brevemente en la página o use el atributo LONGDESC."

En la aplicación TICC, se cuenta con un gráfico estadístico (ver figura 6.7) que no posee una descripción detallada de lo que el gráfico representa. Lo cual no sigue las recomendaciones del consejo anteriormente descripto.

Se tuvo que incorporar en la página que muestra el gráfico estadístico un nuevo enlace que navega a una nueva pestaña para mostrar la descripción detallada del mismo.

Luego se creó una nueva página en donde se incluye la descripción detallada del gráfico de dispersión. La información de esta página como todas las demás de la aplicación TICC, esta internacionalizada. En la figura 6.8, se muestrea la página de descripción del gráfico.

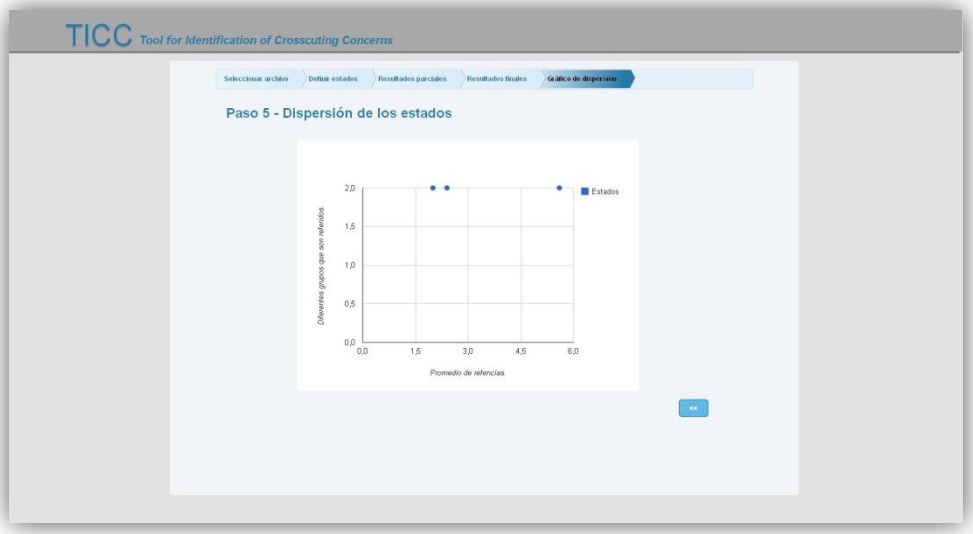

Figura 6.7 Gráfico de dispersión sin descripción asociada

Figura 6.8 Mensaje de texto alternativo para el gráfico de dispersión

6.6.7 Consejo 7: Scripts, applets y plug-ins

"Ofrezca contenido alternativo en el caso de que las características activas no sean accesibles o no tengan soporte."

En la aplicación TICC no se cuenta con la presencia de applets y plug-in, por lo que en este caso no fue necesario proveer contenido alternativo para los mismos. Dado que la aplicación TICC busca satisfacer la prioridad 1, los elementos scripts utilizados solamente fueron revisados y disminuidos.

6.6.8 Consejo 8: Marcos (Frames)

"Utilice el elemento noframes y etiquete con los atributos tittle o name."

Dado que en la aplicación TICC no fue necesario el uso de FRAME y además por conocimiento en el desarrollo de páginas Web, no se introdujo la posibilidad de usar la etiqueta FRAME. Por todo esto es que el consejo ha sido aplicado sin necesidad de cambios en la aplicación.

6.6.9 Consejo 9: Tablas

"Realícelas de manera que se puedan leer línea a línea. Incluya un resumen."

En la aplicación TICC, se modificaron todas las tablas ya que ninguna de ellas contaba con un título y un resumen asociada a la misma. Así por ejemplo en la tabla que muestra los distintos promedios para cada uno de los elementos (ver figura 6.9), podemos notar la ausencia de la etiqueta <CAPTION> y del atributo SUMMARY:

```
<table class="bordered"> --Falta el atributo SUMMARY
        <thead> --Ausencia de la etiqueta CAPTION
          <tr>
                <th>  <spring:message code="concerns.table.states.name"/></th>
                <th>  <spring:message code="concerns.table.subject.average"/></th>
                <th>   <spring:message code="concerns.table.objects.average"/></th>
                <th>   <spring:message code="concerns.table.verbs.average"/></th>
          </tr>
        </thead>
        <tbody>
        <c:forEach items="${results}" var="resultState">
                <tr>
                        <td>${resultState.name}</td>
                        <td class="number-cell">${resultState.avgSubjects}
                        </td><td class="number-cell">${resultState.avgObjects}
                        </td>
                                <td class="number-cell">
                                ${resultState.avgVerbs}
                        </td>
                </tr>
        </c:forEach>
        </tbody>
</table>
```

Figura 6.9 Tabla sin definición del caption y summary

Dado que esta tabla no cumple con el consejo, se la modifico de la forma que se ve en la figura 6.10 y 6.11.

En la figura 6.11, podemos ver que tanto el valor asociado al atributo summary ([1] y [2]) y el valor de la etiqueta caption [3] han sido definidos para poder ser internacionalizados. Sus valores, se definieron en el archivo de mensajes (message.properties).

```
concerns.table.results.summary = Promedio de referencias discriminadas por los
distintos tipos de elementos: Sujeto, Objetos y verbos.
concerns.table.results.caption = Promedio de referencias por tipo de elementos
```

Figura 6.10 Mensajes internacionalizados para Summary

Cabe recordar que el mismo cambio se aplico sobre la tabla que muestra los resultados finales y la tabla que muestra los detalles de los elementos que conforman un estado dado.

```
<c:set var="resultTableSummary" value="<spring:message
code='concerns.table.results.summary'/>"></c:set>    [1]
<table class="bordered" summary="${resultTableSummary}"> [2]
        <caption><spring:message code="concerns.table.results.caption"/>[3]
        </caption>
        <thead>
          <tr>
                <th> <spring:message code="concerns.table.states.name"/></th>
                <th> <spring:message code="concerns.table.subject.average"/></th>
                <th> <spring:message code="concerns.table.objects.average"/>
                <th> <spring:message code="concerns.table.verbs.average"/></th>
          </tr>
        </thead>
        <tbody>
                        <c:forEach items="${results}" var="resultState">
                <tr><td>${resultState.name}</td>
                    <td class="number-cell">${resultState.avgSubjects}</td>
                    <td class="number-cell">${resultState.avgObjects}</td>
                    <td class="number-cell">${resultState.avgVerbs}
                </tr>
        </c:forEach>
        </tbody>
</table>
```

Figura 6.11 Tabla con definición del caption y summary

6.6.10 Consejo 10: Revise su trabajo

"*Valide el código HTML. Use las herramientas de evaluación, puntos de comprobación y pautas de la W3C.*"

En la siguiente sección se describirán las distintas herramientas de evaluación existentes, para finalmente describir la revisión efectuada en la aplicación TICC.

6.7 Herramientas de evaluación de accesibilidad Web

Evaluar la accesibilidad a lo largo del desarrollo de un sitio web permite encontrar problemas de accesibilidad desde el inicio, justo a tiempo para poder solucionarlos. El objetivo de la evaluación de accesibilidad es determinar el grado de cumplimiento de las pautas de Accesibilidad.

El proceso de evaluación de la Accesibilidad de todo contenido Web consta de dos fases:

1. En primer lugar se debe realizar un **análisis automático** que detecte los problemas de accesibilidad. Las herramientas automáticas han de entenderse como una ayuda en el proceso de evaluación y no como un análisis completo ni infalible. No debemos olvidar que éstos y todos los componentes técnicos son de gran ayuda pero es necesaria la intervención de un experto humano para determinar si realmente cumple con las normas establecidas.

2. Como complemento de la evaluación automática ha de realizarse una **evaluación manual** para identificar todos aquellos problemas que no pueden ser comprobados en la primera fase y revisar aquellos dudosos que requieren de pruebas adicionales para su comprobación completa.

En la siguiente sección se describen las herramientas para revisar los sitios web y determinar los temas de accesibilidad que han sido resueltos y los que no. Estas pruebas deberías resaltar la mayoría de los aspectos de accesibilidad y son valiosas para reducir un gran número de barreras de accesibilidad.

6.7.1 Validadores automáticos

Un validador automático es un programa que revisa una página web y nos informa de los resultados de esa revisión. En el caso de un validador automático de accesibilidad, la aplicación revisa diversos puntos conflictivos con la accesibilidad y presenta un informe con los resultados. Estos puntos conflictivos se basan en las pautas de accesibilidad del W3C.

Existen dos tipos de validadores automáticos, dado que existen dos tipos de errores:

- Errores de código (o de HTML o XHTML): nos indica si el sitio web sigue los estándares dictados por el W3C.

- Errores de accesibilidad: son problemas en una página que afectan el acceso de algunas personas, entre otros, de gente con problemas de discapacidad.

Los validadores automáticos más destacados tienen en cuenta ambos tipos de errores, pero esto no siempre es así.

Una pequeña pero importante observación es que no existe un validador automático que pueda revisar todos los posibles problemas de accesibilidad de una página web. Esto se debe a que algunos aspectos de accesibilidad sólo pueden ser revisados por un experto en el tema ya que dependen de cuestiones que un programa no puede detectar y, por lo tanto, un validador puede dar resultados erróneos.

6.7.1.1 Herramientas de validación de gramática

En la evaluación automática, el primer paso consiste en realizar una comprobación de la gramática de las páginas, tanto del código HTML como de las hojas de estilo, para verificar que están bien formadas y son válidas. La validez gramatical es un requisito de accesibilidad. Es recomendable utilizar las **herramientas de validación** de código proporcionadas por el W3C:

Validador (X) HTML (Markup Validation Service) de W3C Este validador es un servicio online gratuito de validación de código que comprueba la conformidad de los documentos (X) HTML respecto a las gramáticas del W3C y otros estándares (X)HMTL.

Allí sólo se debe escribir la dirección de la página a revisar o, un poco más abajo, nos permite subir un archivo que no esté en línea para que también sea revisado.

Validador de CSS (CSS Validation Service) de W3C Es una herramienta gratuita para validar las hojas de estilo CSS solas o presentes en documentos (X)HTML, comprobando de esta manera si cumplen las especificaciones del W3C. Existe una versión online y una versión descargable multiplataforma.

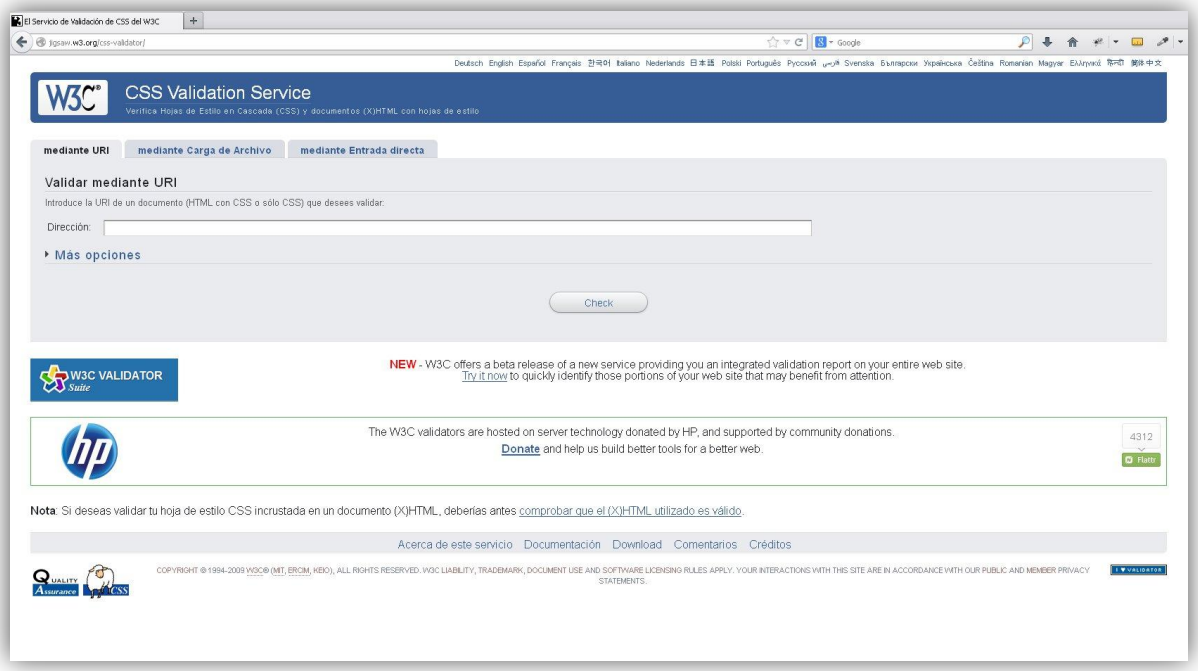

Figura 6.12 Servicio de Validación de CSS [CSSVS 2013]

Allí se debe escribir la dirección de la página donde se encuentra la hoja de estilo y el validador realiza el análisis de forma automática. El código de la hoja de estilo es analizado tanto si ésta se encuentra en un archivo propio (del tipo: estilo.css) como si está dentro de una página Web normal (del tipo: página.html). En este último caso, el validador localiza el trozo de código que corresponde la hoja de estilo (si está correctamente marcado) y lo analiza.

6.7.1.2 Herramientas de evaluación de accesibilidad

Los validadores automáticos de accesibilidad son programas que revisan páginas web en base al cumplimiento de estándares de pautas de accesibilidad. Los mismos realizan la revisión a pedido y entregan informes siguiendo los puntos de Verificación. Existen validadores que trabajan on-line y otros que se pueden descargar a la máquina

Suponen una ayuda en la evaluación de la accesibilidad de los sitios Web, pero hay que tener en cuenta que las **herramientas automáticas** están lejos de ser infalibles y tienen ciertas limitaciones, pudiendo dar falsos positivos (considerar como error algo que no lo es) o no detectar algunos errores que el usuario debe revisar manualmente.

A continuación se describirán los validadores automáticos de accesibilidad más utilizados.

TAW. **TAW** son las siglas de **Test de Accesibilidad Web**. El Test de Accesibilidad Web fue desarrollado por la Fundación CTIC, sede del W3C en España y está disponible desde abril de 2001. Es el primer validador automático de accesibilidad web en español. Dispone de una versión online y de otra descargable que permite trabajar sin conexión a Internet, siendo ambas versiones gratuitas. Además existe una extensión para Firefox.

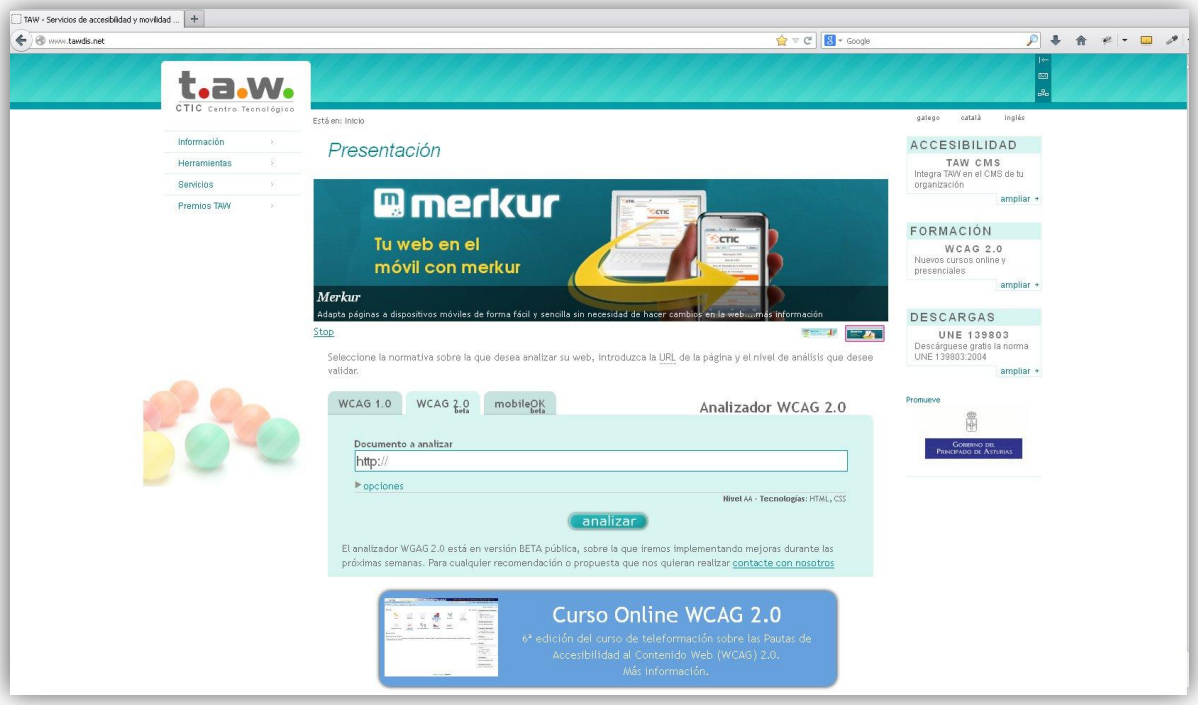

Figura 6.13 Sitio web TAW [TAW 2013]

TAW es una herramienta para el análisis de la **accesibilidad** de sitios web, alcanzando de una forma integral y global a todos los elementos y páginas que lo componen.

El objetivo de **TAW** es comprobar el nivel de accesibilidad alcanzado en el diseño y desarrollo de páginas web con el fin de permitir el acceso a todas las personas independientemente de sus características diferenciadoras.

TAW presenta los resultados de una manera muy visual, numerando los errores según el nivel de prioridad y marcando con interrogantes los puntos a revisar a mano.

HERA. **HERA** es una herramienta para revisar la accesibilidad de las páginas web de acuerdo con las recomendaciones de las Directrices de Accesibilidad para el Contenido Web 1.0 (WCAG 1.0). Es una herramienta que pertenece al Sidar, Seminario Internacional para la Discapacidad y Acceso a la Red, también de España.

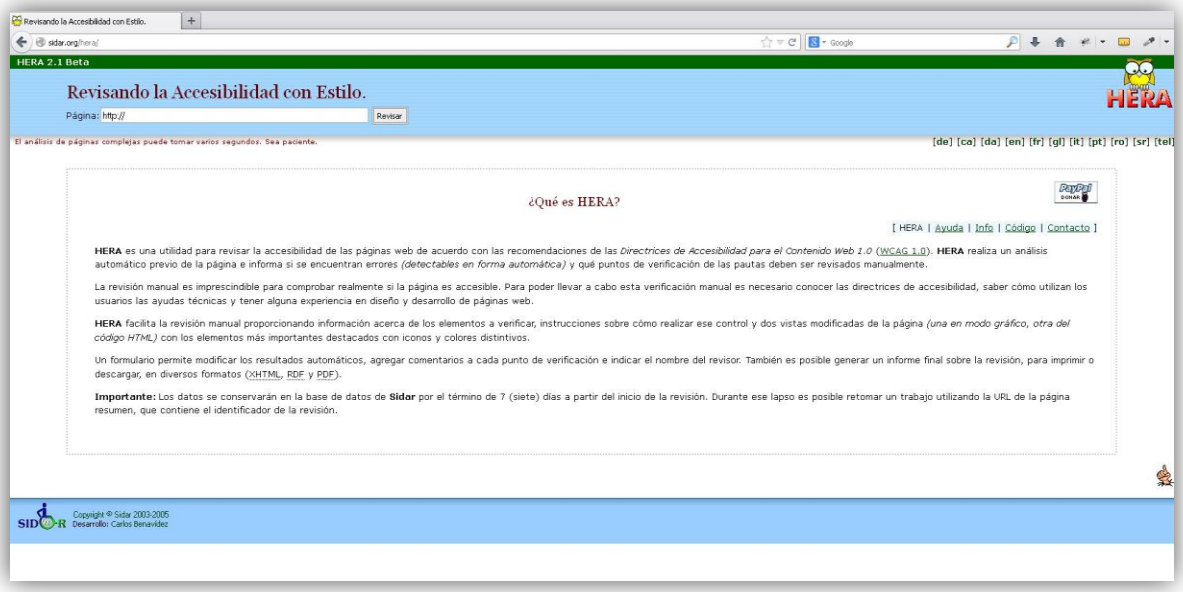

Figura 6.14 Sitio web HERA [HERA 2013]

HERA realiza un análisis automático previo de la página e informa si se encuentran errores (detectables en forma automática) y qué puntos de verificación de las pautas deben ser revisados manualmente.

La revisión manual es imprescindible para comprobar realmente si la página es accesible. Para poder llevar a cabo esta verificación manual es necesario conocer las directrices de accesibilidad, saber cómo utilizan los usuarios las ayudas técnicas y tener alguna experiencia en diseño y desarrollo de páginas web.

HERA facilita la revisión manual proporcionando información acerca de los elementos a verificar, instrucciones sobre cómo realizar ese control y dos vistas modificadas de la página (una en modo gráfico, otra del código HTML) con los elementos más importantes destacados con iconos y colores distintivos.

Un formulario permite modificar los resultados automáticos, agregar comentarios a cada punto de verificación e indicar el nombre del revisor. También es posible generar un informe final sobre la revisión, para imprimir o descargar, en diversos formatos (XHTML, RDF y PDF).

eXaminator Afines de 2005, un emprendimiento en conjunto del sitio Accesible.com.ar y el experto argentino Carlos Benavidez permitió el surgimiento de **eXaminator**.

Este es un validador en línea, de uso muy simple, que ofrece la característica de dar un “puntaje de accesibilidad”, además de un informe analítico que nos informa de los errores cometidos y nos muestra el camino para solucionarlos.

El algoritmo de eXaminator analiza los diversos elementos y atributos en el código de una página web y cuenta el numero de situaciones (errores y buenas prácticas) posibles de inferir automáticamente, teniendo como base las Pautas de Accesibilidad al contenido en la Web 1.0 (WCGA).

Ofrece reportes en XHTML y PDF. También ofrece una versión plug in para Firefox.

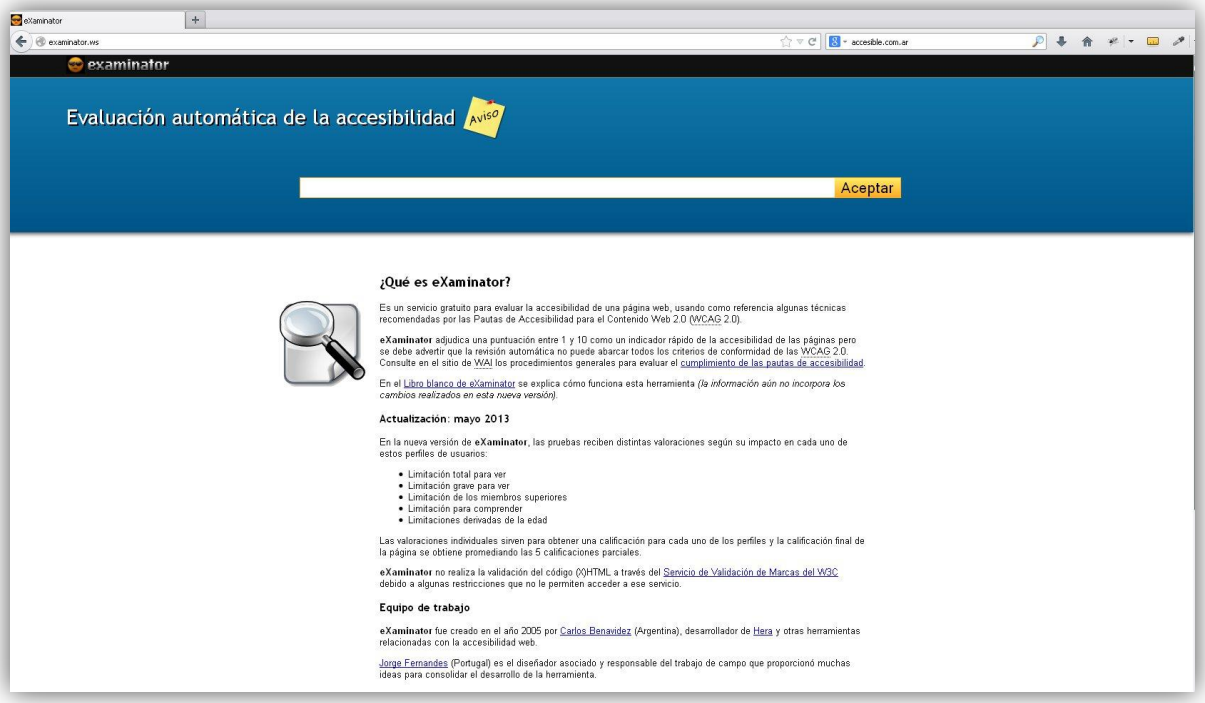

Figura 6.15 Sitio web eXaminator [eXaminato 2013]

6.8 Herramientas de evaluación aplicada a TICC

La evaluación de accesibilidad para la aplicación web TICC se realizo siguiendo las recomendaciones definidas en el punto 10 de la Guía breve para crear sitios accesibles. La cual consiste en realizar los siguientes pasos:

1. Evaluación de la gramática del lenguaje XHTML
2. Evaluación de la gramática de las hojas de estilo CSS.
3. Revisión de los puntos de comprobación y las pautas de WCAG 1.0

En las siguientes secciones se describirá en detalle cada uno de los pasos y su aplicación en TICC.

6.8.1 Evaluación del Lenguaje XHTML

En primer lugar se realizo la revisión de la sintaxis y semántica del lenguaje empleado en la creación de las páginas web de la aplicación TICC. Cada una de las páginas, que conforman la aplicación TICC, fue revisada y validada a través del **Validador (X)HTML** provisto por la **W3C.**

Para cada una de las páginas se mostrara el resultado de la evaluación, por los cambios efectuados a partir de los errores encontrados y por último el resultado final de la evaluación de la página modificada.

Página Subir archivo. Al realizar la revisión de esta página con la herramienta de evaluación se obtuvo el siguiente resultado que se ve en la figura 6.16.

En este caso no fue necesario efectuar cambios en la página ya que la misma cuenta con una correcta sintaxis del lenguaje XHTML.

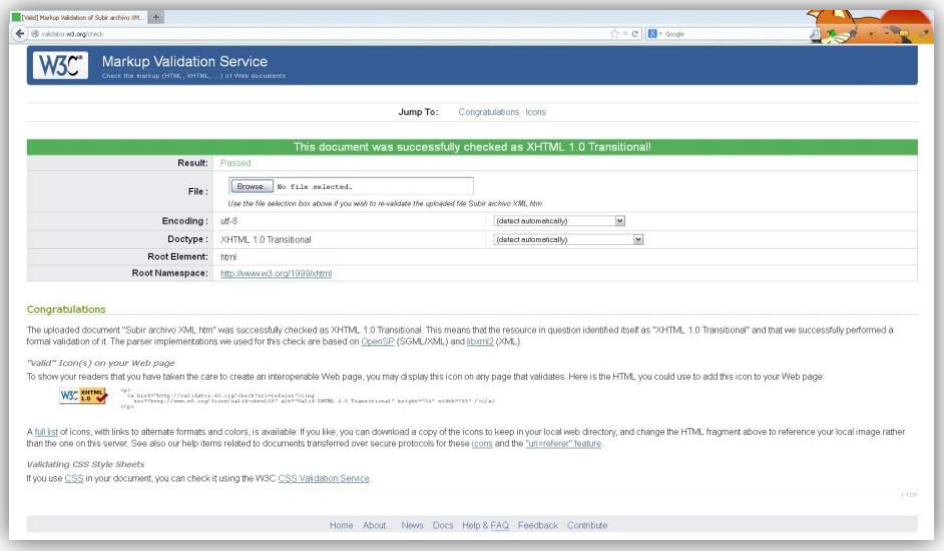

Figura 6.16 Resultado de la revisión de la página Subir archivo

Página Definición de estados. Al realizar la revisión de esta página con la herramienta de evaluación se obtuvo el siguiente resultado que se ve en la figura 6.17.

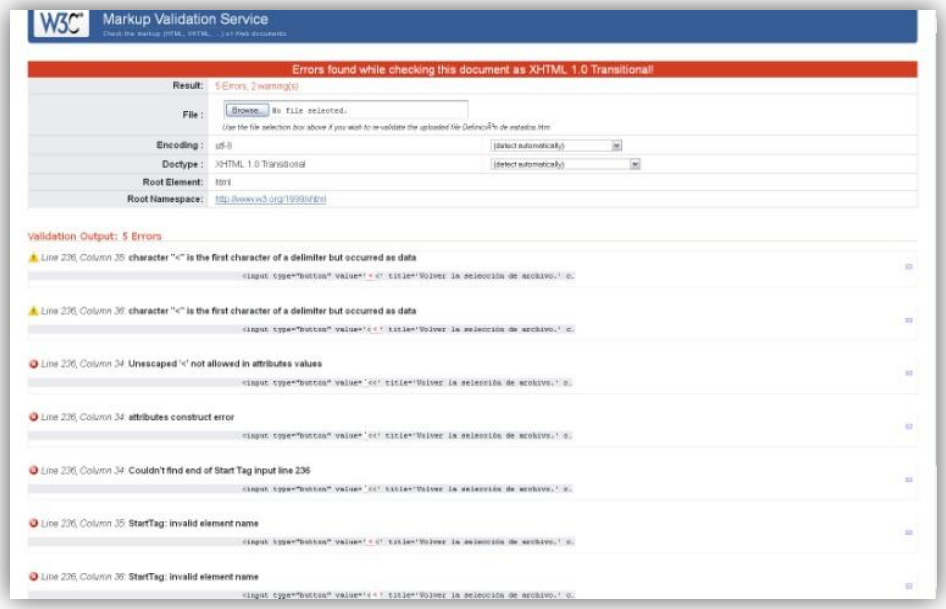

Figura 6.17 Resultado de la revisión de la página Definición de estados

Como se puede ver en la figura, se encontraron 5 errores y dos advertencias. Las mismas indicaban un mal uso del carácter “<”, ya que este es el primer carácter de un delimitador, pero fue utilizado como dato.

Para solucionar estos errores, la herramienta proponer modificar el carácter “<” por su correspondiente valor escapado "<".

Dado que este carácter es usado en casi todas las páginas de la aplicación, se modifico todas sus ocurrencias en los archivos de internacionalización por su valor escapado (ver figura 6.18).

```
state.groups.load.file = &lt;&lt;
concerns.table.graph.definition = &lt;&lt;
concerns.table.previous.result =&lt;&lt;
concerns.charting.results.final.result.back = &lt;&lt;
```

Figura 6.18 Escapado del caracter ">" y "<"

Luego, al volver al validar la página modificada se logro pasar la revisión, como se muestra en la siguiente imagen:

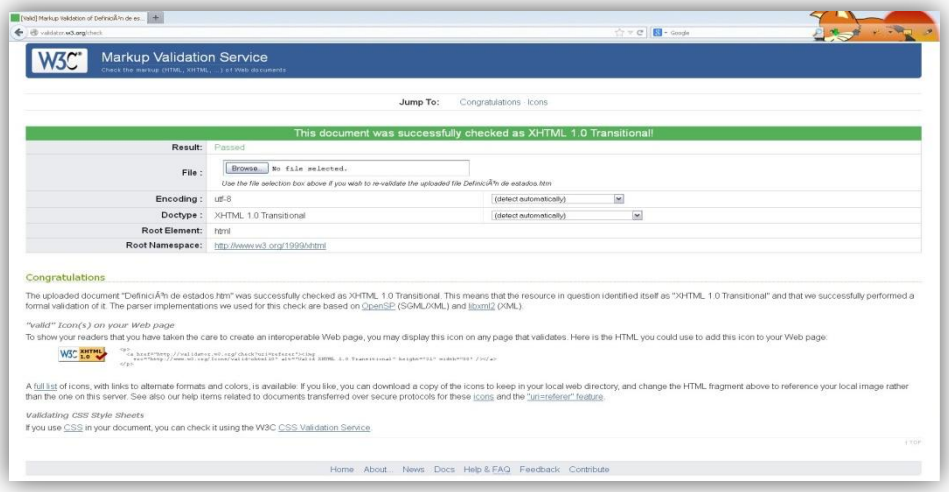

Figura 6.19 Resultado de la revisión de la página modificada

Página Conteo de referencias. Al realizar la revisión de esta página con la herramienta de evaluación se obtuvo un resultado satisfactorio, como se muestra en la siguiente imagen:

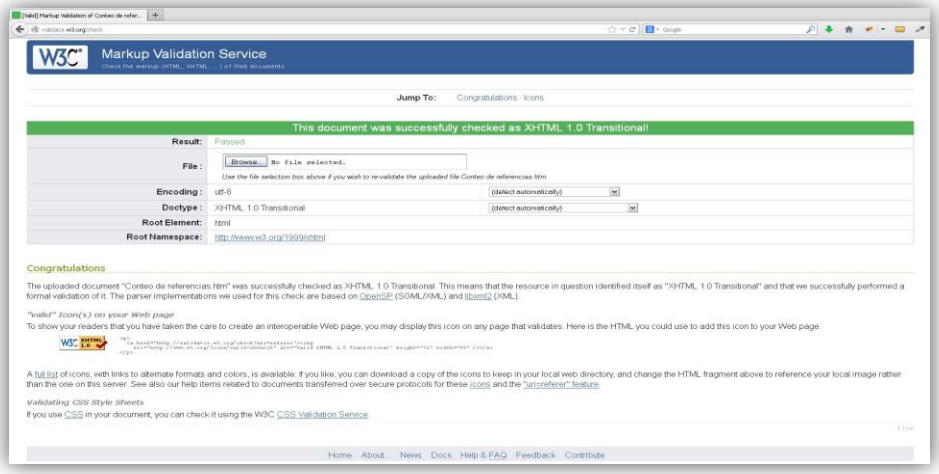

Figura 6.20 Evaluación de la página Connteo de Referencias

Página Orden Final. La evaluación de la página a través de la herramienta de evaluación arrojo los siguientes resultados que se ven en la figura 6.21.

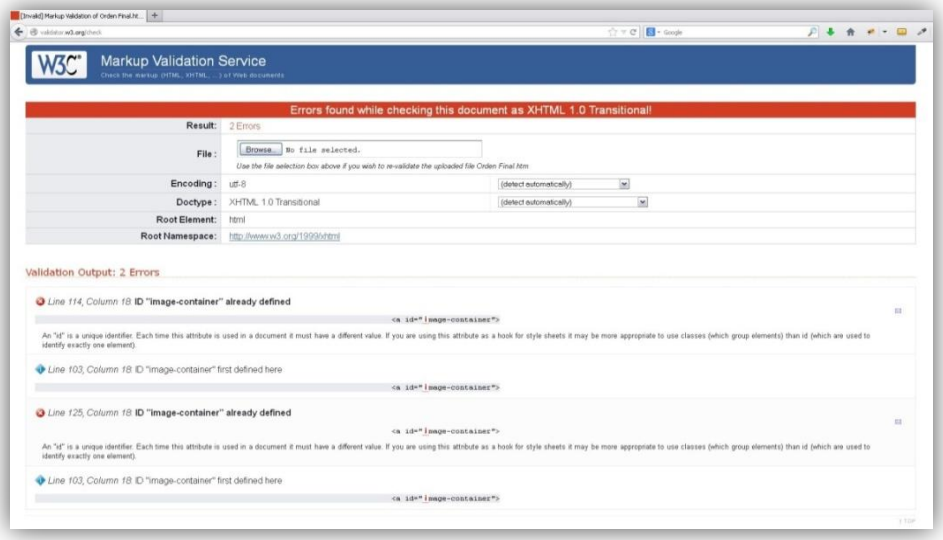

Figura 6.21 Resultado de la revisión de la página Conteo de referencias

El error marcado por la herramienta, fue el uso inadecuado del atributo id de las etiquetas HTML. El atributo id permite definir un identificador único. Por lo tanto cada vez que este atributo se utiliza en un documento debe tener un valor diferente.

Para solucionar los errores encontrados, se asocio a cada elemento de la página un identificador único (ver figura 6.22).

Finalmente se realizo una segunda evaluación de la página modificada para verificar los cambios realizados, obteniéndose los resultados de la imagen 6.23.

```
<c:forEach items="${finalResults}" var="finalState">
  <tr>
        <td id="state_${finalState.name}">${finalState.name}</td>
        <td class="number-cell">${finalState.groupNameReferences}</td>
        <td class="number-cell">${finalState.average}</td>
        <td class="number-cell" id="state_${finalState.idState}">
              <a id="image-container-${finalState.idState}"> [1]
                    <img alt="<spring:message code='concerns.table.actions.view.details.alt'/>"
src="<c:url value="/resources/image/view-detail.png"/>"/>
              </a>
        </td>
  </tr>
</c:forEach>
```

Figura 6.22 Definición de un identificador único para los elementos

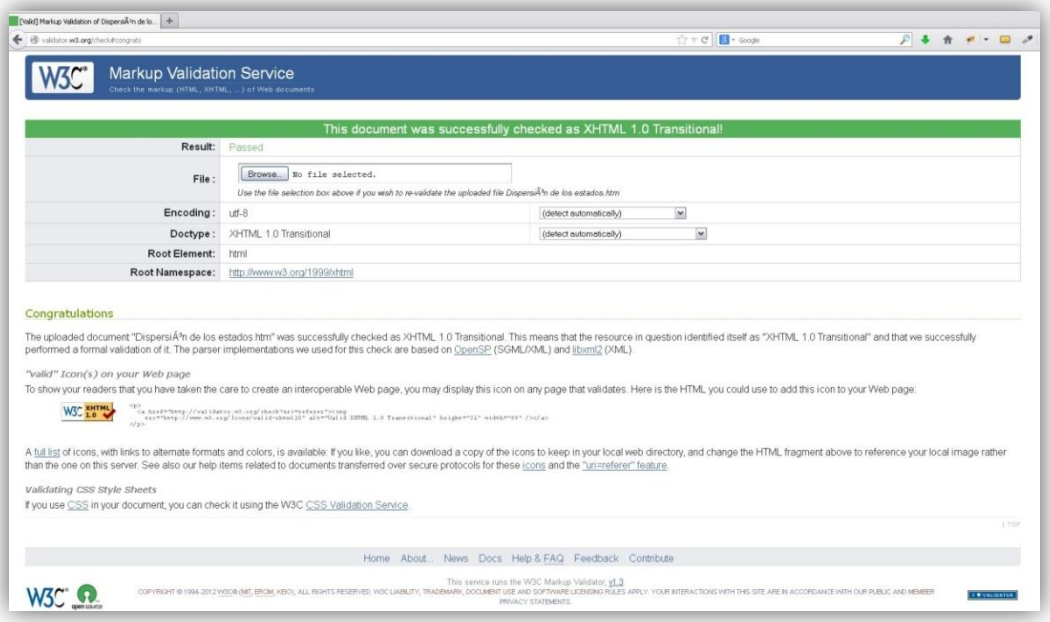

Figura 6.23 Resultado de la página de dispersión de estados

Página Dispersión de los estados. Al realizar la evaluación de la página que muestra la dispersión de los estados, se obtuvo un resultado satisfactorio de la misma, como así lo indica la siguiente: imagen:

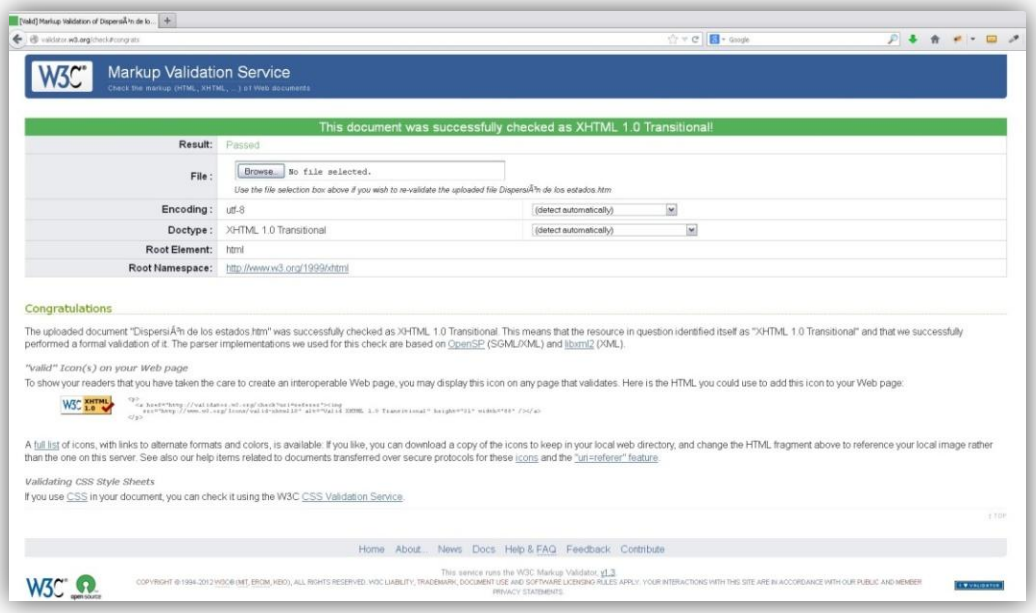

Figura 6.24 Evaluación del gráfico de dispersión

Página Descripción del gráfico de dispersión. Por último se realizo la evaluación de la página que describe el gráfico de dispersión, obteniendo una validación satisfactoria, como se puede observar en la siguiente imagen:

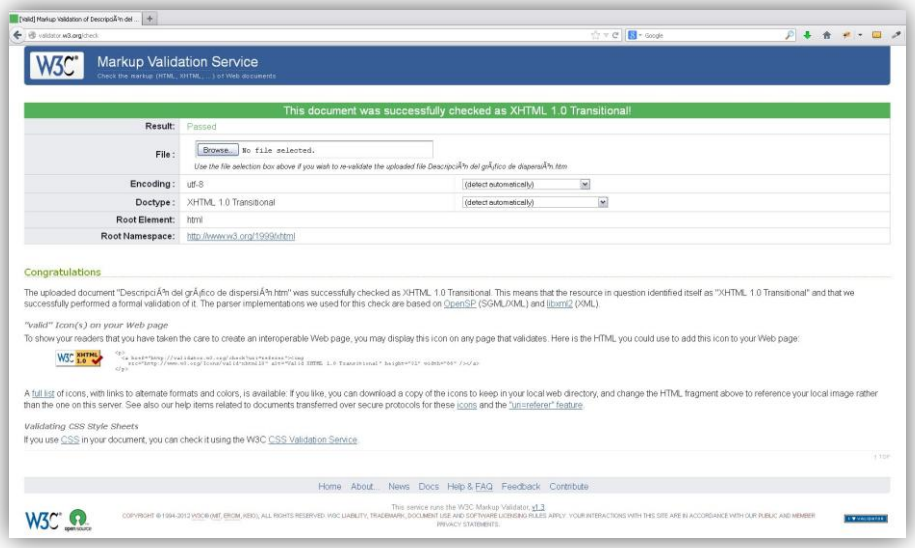

Figura 6.25 Resultado de la revisión de la página Descripción del gráfico de dispersión

Al lograr que todas las páginas de la aplicación TICC pasen la revisión de la sintaxis y semántica satisfactoriamente, se procedió a incorporar en el pie de las páginas el icono que indica que se ha realizado una validación correcta del lenguaje de las páginas:

```
<p>
  <a href="http://validator.w3.org/check?uri=referer">
        <img src="http://www.w3.org/Icons/valid-xhtml10-blue" alt="Valid XHTML 1.0!"
        height="31" width="88" />
  </a>
</p>
```

Figura 6.26 Código para incorporar el logo de la validación XHTML

6.8.2 Evaluación del lenguaje CSS

Una vez finalizada la revisión de la sintaxis y semántica del lenguaje empleado en las páginas de la aplicación, se prosiguió con la revisión y validación del lenguaje CSS utilizado. Para ello se utilizo la herramienta automática *Validador de CSS* provisto por W3C.

La evaluación del lenguaje CSS de la hoja de estilo de la aplicación arrojo como resultado un error y 25 advertencias. En la siguiente imagen se muestra los resultados obtenidos:

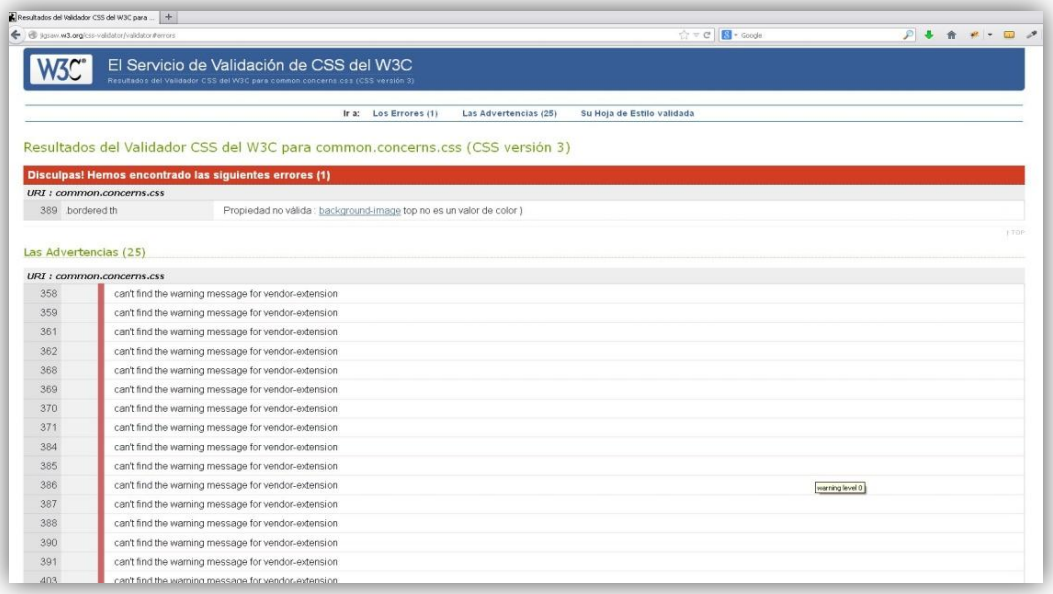

Figura 6.27 Evaluación de la hoja de estilo

Esta herramienta de validación genera una nueva hoja de estilo, con una sintaxis correcta del lenguaje CSS. Por lo tanto se incorporo el cambio sugerido por la misma en la hoja de estilos de la aplicación TICC. Finalmente se realizo una segunda evaluación de la hoja de estilo, dando como resultado una correcta validación. En la siguiente imagen se muestra el resultado obtenido.

Figura 6.28 Evaluación de la hoja de estilos modificada

Dado que la hoja de estilo de la aplicación TICC fue correctamente evaluada, se incorporo el icono que indica dicha situación.

```
<p>
  <a href=http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px" src="http://jigsaw.w3.org/css-
alidator/images/vcss-blue" alt="¡CSS Válido!" />
  </a>
</p>
```

Figura 6.29 Código para incorporar el logo de CSS válido

6.8.3 Evaluación del nivel de accesibilidad

Finalmente se efectuó la evaluación de la aplicación TICC siguiendo el método de los puntos de verificación de las "Pautas de Accesibilidad para el Contenido de la Web. WCAG1": Prioridad 1. Los puntos de verificación de prioridad 1, son aquellos puntos que un desarrollador web tiene que cumplir, ya que de otra manera ciertos grupos de usuarios no podrían acceder a la información del sitio web.

Para llevar a cabo la evaluación de la accesibilidad se utilizo la versión ejecutable de la herramienta automática TAW (Test de Accesibilidad de la Web), la cual permite evaluar la accesibilidad respecto a los estándares WCAG. TAW realiza la evaluación del código respecto a los puntos de verificación de cada uno de los niveles de prioridad, y obtiene el número de errores, tanto automáticos como manuales, que existen en la página.

En este punto, cabe destacar que todas las herramientas de este tipo, facilitan la detección de problemas de accesibilidad, y por tanto, el diseño de sitios web accesibles. No obstante, es imprescindible la participación de un experto en accesibilidad que complemente la herramienta en aquellos puntos que una máquina no alcanza a decidir, como por ejemplo, cuestiones relacionadas con la semántica.

De las distintas versiones existentes de TAW (TAW 3 Web, TAW 3 Descargable, TAW 3 Webstart y TAW3 en un clic) se eligió la versión **TAW3 descargable** (ver figura 6.30).

En el **informe TAW** se puede observar la página analizada, en la cual se insertan los iconos de alerta sobre los problemas de accesibilidad encontrados. Estos iconos representan los tres niveles de prioridad y pueden ser:

1) Prioridad 1 (color rojo). El texto alternativo es "1.automático". 
2) Prioridad 2 (color naranja). El texto alternativo es "2.automático". 
3) Prioridad 3 (color verde). El texto alternativo es "3.automático" 

Estos problemas son los denominados automáticos, aquellos en los que la herramienta tiene la certeza de que incumplen las pautas (por ejemplo, una imagen sin texto alternativo).

Asimismo, también pueden aparecer los siguientes iconos que indican los problemas manuales:

1) Prioridad 1 (color rojo). El texto alternativo es "1.manual". 
2) Prioridad 2 (color naranja). El texto alternativo es "2.manual". 
3) Prioridad 3 (color verde). El texto alternativo es "3.manual". 

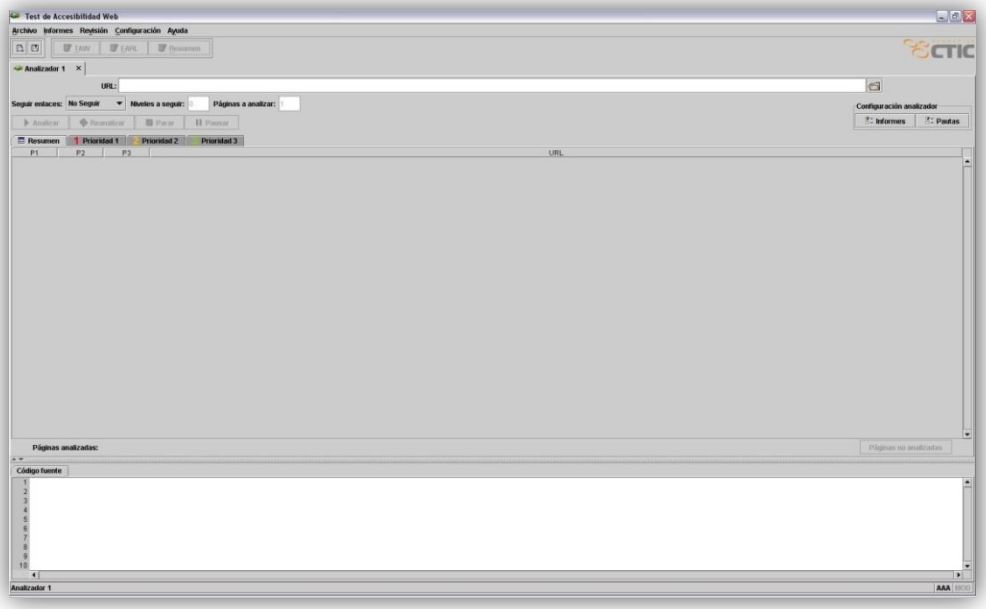

Figura 6.30 TAW3 descargable [TAW3 2013]

El nivel de prioridad es el mismo, pero se trata de problemas o advertencias que necesitan ser revisados por el desarrollador. Se refiere a problemas de accesibilidad bajo ciertas condiciones que se deben comprobar (por ejemplo, la necesidad de una descripción larga para las imágenes).

A continuación se describirá los resultados obtenidos, por cada una de las páginas de la aplicación TICC. El nivel de prioridad que se busco satisfacer es el del nivel 1.

Página Subir archivo. Al evaluar la página que nos permite subir el archivo XML, se obtuvieron los resultados que se ven en la figura 6.31.

Se puede observar que el nivel de prioridad 1, se satisface. La validación recomienda realizar validaciones manuales con respecto a las hojas de estilo.

Página Definición de estados. Con la evaluación de la página, en donde se definen los estados y sus elementos, el nivel de prioridad 1 solo arrojo recomendaciones para realizar evaluaciones manuales. No encontrándose errores en ese nivel, como lo ilustra la siguiente imagen 6.32.

Página Conteo de referencias. La página de conteo de referencias, también cumplió con el nivel de prioridad 1. En la evaluación se obtuvieron los siguientes resultados que se ven en la figura 6.33. En esta, se puede observar, que no se encontraron errores de nivel de prioridad 1, pero si recomendaciones para una evaluación manual de las hojas de estilos y las tabla utilizada.

Página Orden Final. Al efectuarse la evaluación de la página de los resultados finales, se obtuvieron los siguientes resultados que se ven en la figura 6.34. Esta página también cumple con el nivel de prioridad 1. La evaluación recomienda realizar evaluaciones manuales sobre las hojas de estilos y la tabla utilizada.

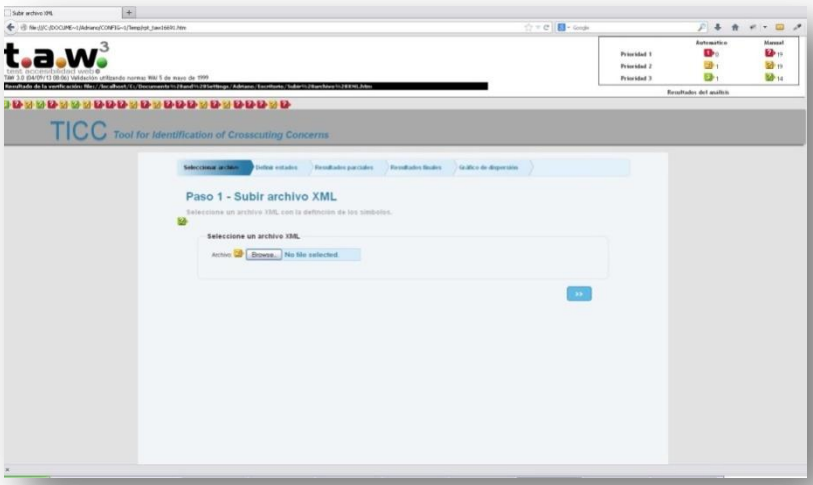

Figura 6.31 Validación de Subir Archivo

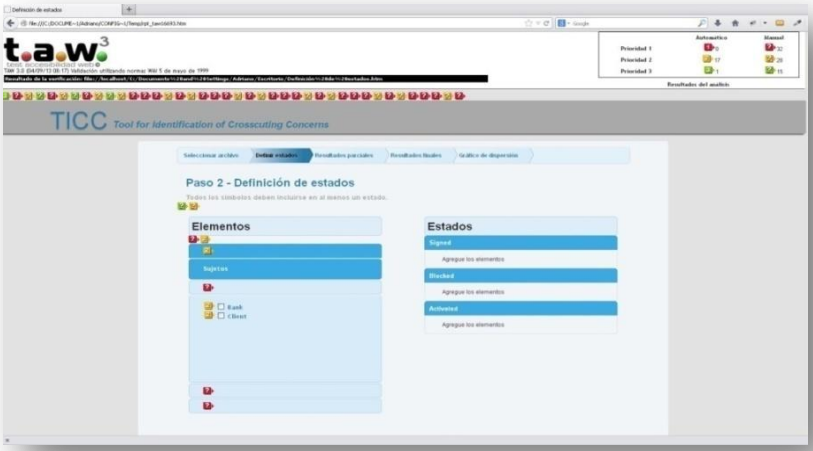

Figura 6.32 Validación de definición de estados

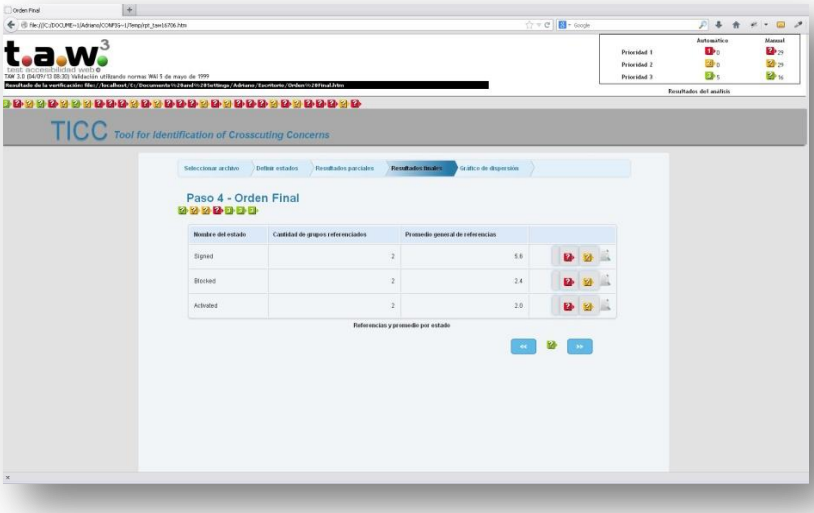

Figura 6.33 Validación de conteo de referencias

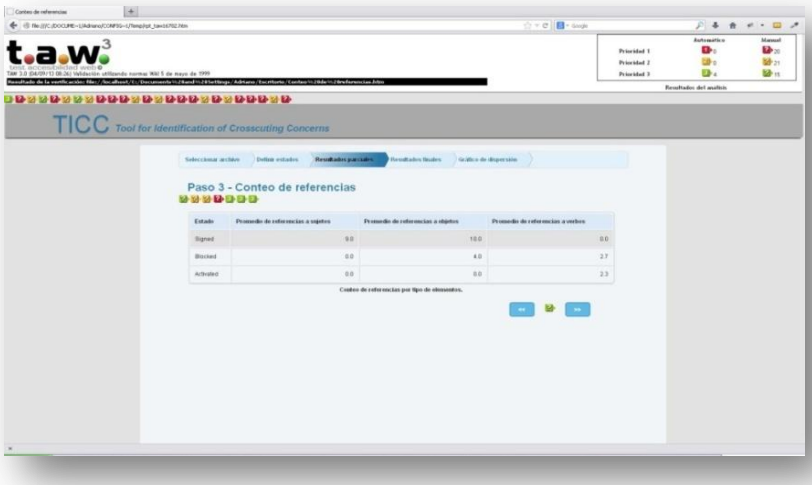

Figura 6.34 Validación del Orden Final

Página Dispersión de los estados. La evaluación de la página que muestra la dispersión de los estados, también obtuvo un nivel de prioridad 1 sin errores, como se puede observar en la siguiente imagen.

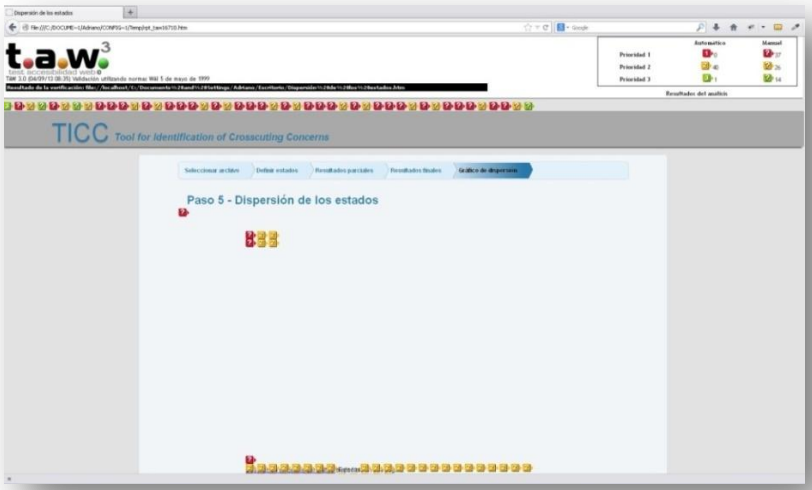

Figura 6.35 Validación de Dispersión de los estados

También se encontraron advertencias para realizar de forma manual sobre las hojas de estilos.

Página Descripción del gráfico de dispersión. Por último la página que describe al gráfico de dispersión, también satisface el nivel de prioridad de nivel 1, como lo muestra la Figura 6.36.

Se puede observar que no se encontraron errores de prioridad 1, pero si recomendaciones para realizar una validación manual de las hojas de estilos.

Figura 6.36 Validación de Descripción del gráfico de dispersión

6.9 Conclusión

En primer lugar se ha intentado dar una justificación de porque es importante y necesario tener en cuenta la accesibilidad cuando se desarrolla una aplicación Web. La accesibilidad de la web no es solo un derecho de las personas con discapacidad sino que, al mismo tiempo, beneficia a muchos tipos de usuarios con diversidad cultural, de idioma, de tecnología de conexión, etc.

La aplicación de las recomendaciones descriptas en la “Guía breve para crear sitios web accesibles” [W3C 2013a], permitió obtener un sitio web que cumple con las pautas de prioridad 1 del documento “Pautas de Accesibilidad para el Contenido de la Web. WCAG1”.

El análisis automático permitió confirmar algunos aspectos fundamentales de accesibilidad recogidos en las pautas WAI aunque hay que ser conscientes de que muchos de estos puntos de verificación sólo podrán analizarse con la intervención humana, puesto que ésta puede ayudar a asegurar la claridad del lenguaje y a facilitar su comprensión. La mayoría de los errores automáticos hallados correspondían a errores que habría que comprobar de forma manual para ratificar si afectan o no la accesibilidad.

Capítulo 7

7. Conclusiones

En muchas ocasiones los profesionales de informática no logramos abstraernos de los conocimientos técnicos y pensamos únicamente en la mejor forma de resolver el problema, cuando en muchos casos la respuesta está dada por el dominio de la información y la relación directa con los requerimientos de software. En este trabajo quedó evidenciada la cantidad de información que puede obtenerse del dominio utilizando una herramienta tan simple como LEL, cuya construcción no consideramos compleja desde el momento en que está basada en lenguaje natural y ayuda a que las partes que integran un proyecto de software pueden llegar a un entendimiento óptimo.

La estrategia implementada por la herramienta TICC comienza con una descripción del contexto de la aplicación en lenguaje natural. Esta descripción debe realizarse en lenguaje LEL (Léxico Extendido por el lenguaje) a través de la aplicación C&L. Esta tarea puede ser realizada por cualquier interesado (stakeholder) sin ninguna clase de experiencia y con poco esfuerzo.

Luego de construido el LEL, la herramienta TICC realiza los cálculos de forma automática y obtiene los cálculos finales. Al visualizar los resultados, obtenidos con una configuración dada, eventualmente se pueden realizar cambios en la definición de los estados y repetir los cálculos para determinar que símbolos son candidatos a ser características transversales.

Vale la pena destacar, que la estrategia implementada en TICC puede ser aplicada casi en cualquier etapa del desarrollo de software, incluidas las etapas de requerimientos y codificación. El elemento clave que permite utilizar la herramienta en cualquier etapa es el LEL, ya que es el que sintetiza el conocimiento de la aplicación.

Los casos de estudios ejemplificados en la tesis ilustran como la herramienta aplica la estrategia de identificación de características transversales de una forma eficiente. Esta utilidad radica principalmente en la automatización de la estrategia de identificación de características transversales y en la simplicidad que presenta TICC para ser utilizado.

El análisis de la usabilidad y accesibilidad de la herramienta TICC permite que usuario con distintos roles y puntos de vista puedan usar la herramienta disminuyendo la probabilidades de cometer errores al realizar los cálculos así como también la repetición de los cálculos al agrupar de formas distintas los símbolos del LEL.

8. Referencias

[Ackoff 1974] Ackoff, R., "Redesigning The Future", Wiley, 1974.
[Ajax 2012] Ajax API, http://api.jquery.com/jQuery.ajax/, accedido 2012.
[Almentero 2009] Kinder Almentero E., "Re-engenharia do software C&L para plataforma Lua-Kepler utilizando princípios de transparência" dissertação de Mestrado PUC Rio, Rio de Janeiro, 8 abril de 2009.
[Antonelli 1999] Antonelli, L., Oliveros, A., Rossi, G., "Baseline Mentor, An Application that Derives CRC Cards from Lexicon and Scenario", XXVIII JAIIO, II Workshop Iberoamericano en Ingeniería de Requerimientos, WER'99, Buenos Aires, Argentina, Septiembre 9 y 10, 1999.
[Antonelli 2011] Antonelli L., "Identificación temprana de características transversales en el lenguaje de aplicación capturado con el Léxico Extendido del Lenguaje", Tesis de doctorado 2012.
[Arango 1989] Arango, G., "Domain Analysis: From Art Form to Engineering Discipline", Fifth International Workshop on Software Specification and Design. Pp 152-159. ACM Press. 1989.
[Baniassad 2004] Baniassad E., Clarke S., "Finding Aspects In Requirements with Theme/Doc", Documento del Early-Aspects Workshop held as part of AOSD, March 2004.
[Baniassad 2006] Baniassad E., Clements P.C., Araujo J., Moreira A., Rashid A., Tekinerdogan B., "Discovering early aspects, In: IEEE software", ISSN:0740-7459, Volume 23, Issue 1, January, pp 61 – 70, 2006.
[Boehm 1997] Boehm B. W., "Software Engineering", Computer society Press, IEEE, 1997.
[Bounour 2006] Bounour N., Ghoul S., Atil F., "A comparativeclassification of aspect mining approaches", Journal of Computer Science Volume 2, Number 4, ISSN 1549-3636, 2005 Science publication, pp 322 - 325, 2006.
[Breitman 2003] Breitman K.K., Leite J.C.S.P., "Ontology as a Requirements Engineering Product", In: Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE), IEEE Computer Society, Monterey Bay, California, USA, ISBN 0-7695-1980-6, 2003.
[Brooks 1995] Brooks F., "The Mythical Man-Month: Essays on Software Engineering", Addison-Wesley Professional, 2 edition, 1995.
[CSSVS 2013] CSS Validation Service de W3C, http://jigsaw.w3.org/css-validator/, accedido 2013.
[C&L 2009] Herramienta C&L, http://pes.inf.puc-rio.br/cel/, accedido 2012.
[Demeter 1995] Página del grupo Demeter,http://www.ccs.neu.edu/research/demeter/, accedido 2012.
[Dijkstra 1976] E. W. Dijkstra, "A Discipline of Programming", Prentice Hall, 1976
[eXaminator2013] Herramienta eXaminator, http://examinator.ws/, accedido 2013.
[Felicíssimo 2004] Felicíssimo C. H., Leite J.C.S.P., Breitman K. K., Fernandes da Silva L., "C&L: Um Ambiente para Edição e Visualização de Cenários e Léxicos", in sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software Brasília, Brasil, October, pp 43 – 48, 2004.
[Ferré 2001] Xavier Ferré, Natalia Juristo, Helmut Windl, Larry Constantine, "Usability basics for software developers", IEEE Software, January/February 2001. p. 22-29.
[Gil 2000] Gil D., Figueroa D. A., Oliveros A., "Producción del LEL en un Dominio Técnico. Informe de un caso.", Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brasil, 2000.
[Google 2012] Google API, https://developers.google.com/chart/, accedido 2012.
[Gruber 1993] Gruber T.R., "A translation approach to portable ontology specifications", Knowledge acquisition journal, volume 5, June 1993, pp 199 – 220, 1993.
[HERA 2013] Herramienta HERA, http://sidar.org/hera/, accedido 2013.
[ISO 1998b] ISO 9241-11, "Ergonomic requirements for office work with visual display terminals", ISO, 1998.
[ISO 1998a] ISO/IEC 9126-1, "ISO Standard 9126: Software Engineering – Product Quality", ISO, 1998.
[Jackson 1999] Jackson, M., "Problem Analysis and Structure", AT&T Research, Florham Park NJ, USA, y consultor independiente, London, England, 1999
[Java 1997] Lenguaje Java, http://www.oracle.com/technetwork/java/index.html, accedido 2012.
[JavaScript 2012] Javascript Tutorial, http://www.w3schools.com/js/, accedido 2012.
[Jetty 2012] Servidor Web Jetty, http://www.eclipse.org/jetty/, accedido 2012.
[JnLua 2012] Framework JnLua, https://code.google.com/p/jnlua/, accedido 2012.
[Jquery 2012] Librería Jquery, http://jquery.com/download/, accedido 2012.

[Kahlua 2012] Framework Kahlua, https://github.com/krka/kahlua2, accedido 2012.
[Kiczales 2001] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm y W. Griswold, "An Overview of AspectJ", ECOOP 2001.
[Leite 1997] Leite, J.C., Rossi, G., et al.: "Enhancing a Requirements Baseline with Scenarios". Proceedings of RE 97', IEEE Third International Requirements Engineering Symposium, IEEE Computer Society Press, 1997, pp 44-53.
[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://www-di.inf.puc-rio.br/~julio/teses.htm, Facultad de informática, Universidad Nacional de La Plata, Argentina, Noviembre, 2001.
[Lua 2012] Lenguaje Lua, http://www.lua.org/home.html, accedido 2012.
[Luaj 2012] Framework Luaj, http://luaj.org/luaj/README.html, accedido 2011.
[LuaJava 2011] Framework LuaJava, http://www.keplerproject.org/luajava/, accedido 2011.
[LuaRocks 2011] Framework LuaRocks, available http://luarocks.org/, accedido 2011.
[Mahoney 2007] Mahoney M., Elrad T., "Generating code from scenario and state based models to address crosscutting concerns", In: proceedings of the sixth international workshop on scenarios and state machines (SCESM'07), IEEE, 2007.
[Mahoney 2005] Mahoney M., "Modeling Crosscutting Concerns in Reactive Systems with Aspect-Orientation", In: Models 2005, Doctoral Symposium, 2005.
[Maven 2012] Framework Maven, http://maven.apache.org/, accedido 2012.
[MVS 2013] Markup Validation Service de W3C, http://validator.w3.org/, accedido 2013.
[Nielsen 1995] Jakob Nielsen, "Usability Heuristics for User Interface Design", http://www.nngroup.com/articles/ten-usability-heuristics/, publicado Enero 1995.
[Nielsen 1999] Jakob Nielsen, "Designing Web Usability", Peachpit Press, 1999.
[Nielsen 2012] Jakob Nielsen, "Introduction to Usability", http://www.nngroup.com/articles/usability-101-introduction-to-usability/ publicado Enero 2012.
[Nielsen 1995b] Jakob Nielsen, "How to Conduct a Heuristic Evaluation" http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/, publicado Enero 1995.
[Hommy 1995] Hommy Jose, Rosario Noguera, Guillermo Montilla, "Heurística de Nielsen extendida para la evaluación de las interfaces del software educativo", http://65.54.113.26/Publication/12011455, publicado 1995.
[Rago 2009] Rago A., Marcos C., "Análisis Semántico para la Identificación de Aspectos", Jornadas Chilenas de Computación, 2009.
[Rashid 2003] Rashid A., Moreira A., Araújo J., "Modularisation and composition of aspectual requirements", 2nd international conference on Aspect-oriented software development, ISBN:1-58113-660-9, Boston, Massachusetts, pp 11 – 20, 2003.
[Redish 2000] Redish, J. C., "What Is Information Design?", Technical Communication, Volume 47, Number 2, Society for Technical Communication, Mayo 2000.
[Sampaio 2005] Sampaio A., Loughran N., Rashid A., Rayson P.,"Mining Aspects in Requirements", Workshop on Early Aspects (held with AOSD 2005) Illinois, Chicago, USA, 2005.
[Sampaio 2005b] Sampaio A., Chitchyan R., Rashid A., Rayson P., "EA-Miner: A Tool for Automating Aspect-Oriented Requirements Identification" Proc. Int'l Conf. Automated Software Eng. (ASE 05), ACM Press, pp. 352–355, 2005.
[Shneiderman 1998] B. Shneiderman, "Designing the User Interface: Strategies for Effective Human-Computer Interaction", Addison-Wesley, 1998.
[Spring 2012] Framework Spring, http://spring.io, accedido 2011.
[TAW3 2013] Herramienta TAW Descargable http://www.tawdis.net/tools/accesibilidad/desktop/?lang=es, accedido 2013.
[Van Den Berg 2005] Van Den Berg K., Conejero J.M., "A Conceptual Formalization of Crosscutting in AOSD", Desarrollo de Software Orientado a Aspectos, Granada, España, 2005.
[W3C 1996] W3C Standars Guide http://www.w3c.es/Divulgacion/GuiasBreves/Estandares
[W3C 2013b] W3C. "Guía breve para crear sitios web accesibles". http://www.w3.org/WAI/References/QuickTips/qt.es.htm, accedido 2013
[W3C 2013a] World Wide Web Consortium (W3C), http://www.w3.org , accedido 2013.
[WAI 2013] Web Accessibility Initiative, http://www.w3.org/WAI/, accedido 2013.
[WCAG 1.0] WCGA, "Pautas de Accesibilidad al Contenido en la Web 1.0". http://www.w3.org/TR/WCAG10/, accedido 2013.

[WCAG 2.0] WCGA, "Pautas de Accesibilidad al Contenido en la Web 2.0". http://www.w3.org/TR/WCAG20/, accedido 2013.
[Wikipedia 2013a] Wikipedia, "Usability". http://en.wikipedia.org/wiki/Usability , accedido 2013.
[Wikipedia 2013b] Wikipedia, "Accesibilidad". http://es.wikipedia.org/wiki/Accesibilidad, accedido 2013.
[Wikipedia 2013c] Wikipedia, "Accesibilidad web". http://es.wikipedia.org/wiki/Accesibilidad_web, accedido 2013.