Soporte para la Ingeniería de Requerimientos Orientada a Aspectos en Documentos de Gran Tamaño

Tesista
Julián Rousselot

Director
Dr. Arturo Zambrano

Codirector
Dra. Silvia Gordillo

Trabajo Final presentado para obtener el grado de Magister en Ing. de Software
Facultad de Informática - Universidad Nacional de La Plata

Enero 2016

Índice

Índice de Figuras

1. Introducción

La ingeniería de requerimientos es una tarea fundamental en el proceso de comprensión de cómo debe comportarse el sistema que está siendo construido.

Los sistemas complejos incluyen cientos o miles de requerimientos sobre múltiples incumbencias[1], tanto funcionales como no funcionales[2]. Para algunos sistemas, las fuentes de requerimientos son muchas, resultando en varios - en muchos casos extensos - documentos. En ciertos casos, los requerimientos coinciden de forma directa con las incumbencias de la aplicación, lo que les permite ser limpiamente encapsulados en diferentes módulos del sistema resultante. Sin embargo es común encontrar situaciones donde esta premisa no se cumple, tomando por ejemplo la obligación de registrar (logging) todas las acciones del sistema. La correspondiente incumbencia, el registro, no puede ser limpiamente encapsulada en un módulo, ya que afecta a muchas de las partes del sistema. Las incumbencias transversales atraviesan la estructura de requerimientos a lo largo de los diferentes documentos que componen la especificación del sistema. El seguimiento de este tipo de incumbencias a lo largo de todo el documento de requerimientos, viendo a qué requerimiento afecta, es una tarea compleja.

AORE[3] ataca la problemática de los requerimientos complejos cuando estos son difíciles o imposibles de encapsular en módulos separados. Conocida también esta práctica como **Early Aspects**, AORE modela una primera impresión de estas incumbencias transversales como aspectos (en sentido de Aspect Oriented Programming), identificando y caracterizando la influencia que cada uno tiene sobre otros requerimientos del sistema. Estos requerimientos que afectan a muchos otros se los conocen con el nombre de **Cross Cutting Concerns** (CCC)

Estos modelos permiten una mejor identificación y administración de los conflictos de los requerimientos, independientemente de la naturaleza transversal de los requerimientos. Idealmente el resultado de esta fase es obtener un modelo consistente del sistema en una etapa temprana del ciclo de desarrollo.

[1] Una tarea o incumbencia
[2] Definen restricciones adicionales con el que el sistema deberá contar
[3] Aspect-Oriented Requirements Engineering

La información sobre el modelo aspectual es almacenada por separado de la documentación en sí misma. Es decir, dado un requerimiento que es crosscutting se puede enunciar a qué otros requerimientos afecta, pero toda esta información está físicamente separada de los documentos de requerimientos.

Como consecuencia, los requerimientos que corresponden a CCC son particularmente difíciles de seguir cuando contamos con muchos requerimientos de gran extensión.

Esta dificultad en el seguimiento suele desembocar en fallas en el diseño y la implementación, cuando ciertos requerimientos CC no son considerados en todos los lugares donde deben tener efecto. Por ejemplo, en el capítulo 3 hablaremos del caso de estudio, el dominio de las tragamonedas. En dicho caso de estudio observaremos que los requerimientos del concern **Game** están atravesados por los requerimientos del concern **Error Conditions** (incumbencia transversal), es decir, diferentes momentos del juego (ingreso de billetes en la máquina, apertura de la puerta del tragamonedas, la impresora se quedó sin papel, etc.) pueden llevar al juego a diferentes condiciones de error con su correspondiente funcionalidad. No poder identificar claramente qué requerimiento del juego es atravesado por cuál condición de error desembocará en la falta de dicha funcionalidad en la etapa de diseño e implementación, dejando imposibilitado a un requerimiento del juego que informe sobre una condición anómala del mismo, sin posibilidad de notificar el error. Otro error común en la falta de detección de una incumbencia transversal que arrastra al diseño e implementación es cuando la misma incumbencia (transversal) es implementada de manera inconsistente en requerimientos distintos.

La experiencia indica que trabajar con requerimientos pertenecientes a incumbencias transversales es difícil. Sumado a esto, la falta de herramientas que hagan un seguimiento claro de las mismas produce errores costosos que son detectados de manera tardía en el el proceso de construcción de software. En el peor de los casos llegando a detectarse sólo en producción [13].

Por lo tanto, se hace evidente la necesidad de herramientas especializadas para esta tarea importante que lleva adelante un ingeniero de requerimientos.

Proveyendo una herramienta adecuada que soporte el rastreo de las incumbencias transversales **sobre los documentos de requerimientos** podría ayudar a evitar estas fallas en etapas tempranas desembocando en un

mejor diseño e implementación de un producto de software en todas las etapas del ciclo de vida de los mismos.

Este trabajo muestra la importancia de contar con una herramienta que le ayude al ingeniero de requerimientos en todo el proceso de detección y seguimiento de las incumbencias transversales en la etapa de análisis de requerimientos.

Este trabajo se desarrolla tomando como caso de estudio un dominio complejo que presenta múltiples documentos con un número considerable de CCC. En este dominio encontramos una fuerte motivación para el desarrollo de una herramienta que ayude en dicho contexto.

En los primeros capítulos se presentarán rudimentos de los paradigmas de programación, especialmente el orientado a objetos y el orientado a aspectos, se los comparará. Posteriormente se presenta una pequeña introducción a los lenguajes para la construcción de aspectos y su impacto en el desarrollo de software que involucra incumbencias transversales o crosscutting concerns (CCC).

Luego se expondrá sobre la problemática existente para aplicar el proceso de **AORE** en grandes documentos, se introducirá el caso de estudio que dio lugar al desarrollo de la herramienta. Posteriormente se presentan las necesidades y funcionalidades de una herramienta que asista al ingeniero de requerimientos durante el análisis de los mismos utilizando el paradigma de orientación a aspectos.

Por último se valida el trabajo, y se presentan las conclusiones y posibles líneas de trabajo para el futuro. En los anexos se presenta la documentación que describe la herramienta con mayor nivel de detalle.

Más concretamente el trabajo está estructurado de la siguiente manera:

- **Capítulo 2** es dedicado a los trabajos relacionados, se describen los aspectos relevantes de la ingeniería de requerimientos, la importancia de esta disciplina, sus requerimientos funcionales y no funcionales.

  Se repasan los paradigmas de programación comúnmente utilizados para el desarrollo de sistemas empresariales: La programación orientada a objetos, orientada a aspectos y una comparación entre ellos.

Se introducen los lenguajes orientados a aspectos de dominio específico y de propósito general, se desarrolla el modelado orientado a aspectos de los sistemas complejos, se discute sobre el concepto de incumbencia e incumbencias transversales, se introduce la problemática para aplicar el proceso de AORE[4] en grandes documentos.

- **Capítulo 3** refiere al caso de estudio sobre el mundo de las slot machines.

- **Capítulo 4** se repasan las herramientas comerciales existentes en el mercado y trabajos de investigación difundidos analizando qué aporte tienen en el análisis de los aspectos en la etapa temprana de requerimientos.

- **Capítulo 5** se repasan los requerimientos necesarios para una herramienta que soporte el tratamiento de las incumbencias transversales y se exponen las funcionalidades de la herramienta desarrollada en este trabajo cuyo nombre es **AORE Assistant**.

- **Capítulo 6** se valida la herramienta presentando los resultados de su aplicación al caso de estudio elegido.

- **Capítulo 7** se expresan las conclusiones y el trabajo futuro para mejorar la asistencia de la herramienta al trabajo del ingeniero de requerimientos.

[4] Aspect Oriented Requirement Specification

2. Background Ingeniería de Requerimientos y Separación de concerns

2.1. Ingeniería de Requerimientos

A través de los años se ha podido confirmar la importancia de los requerimientos en un proyecto de desarrollo de software ya que abarcan todas las actividades propias del proceso como ser la planificación con su estimación de recursos, costos y tiempo.

La especificación de requerimientos permite verificar si el producto de software cumple con los objetivos establecidos los cuales reflejan las necesidades de todos los stakeholders[5].

La ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de lo que hace a las necesidades de un producto de software orientado a los diferentes usuarios que utilizarán el mismo, por tal motivo cumple un rol fundamental en lo que hace a la producción de software ya que toma como base la definición de lo que se quiere construir, definiendo sin ambigüedades y de manera consistente las necesidades de los clientes o usuarios.

Hay muchas definiciones existentes en lo que respecta a la palabra "requerimientos", varias de ellas vienen de la IEEE[6] como por ejemplo:

"Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal". (Std 610.12-1900, IEEE: 62)

"Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo". (Std 610.12-1900, IEEE: 62)

Sommerville también define un requerimiento como: ***"Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste"***. (Sommerville, 2005: 108)

[5] Todos los interesados en el proyecto de desarrollo de software
[6] Institute of Electrical and Electronics Engineers

De las definiciones anteriores se desprende que un requerimiento es una descripción de una capacidad que debe cumplir una pieza de software derivada de la necesidad concreta de un usuario o un conjunto de este.

Los proyectos de software tienen alcances bien definidos, objetivos que contienen actividades, tienen recursos asignados ya sean monetarios o de personal, tienen un inicio y un fin. El alcance de los proyectos se define mediante requerimientos, sin una buena definición de estos no es posible hacer estimaciones realistas ni utilizar una herramienta que ayude en todo el proceso de desarrollo, en análisis, diseño, desarrollo y la arquitectura misma del software carecerán de una base firme.

2.1.1. Requerimientos Funcionales

Los requerimientos funcionales son los que describen lo que un sistema debe hacer, son las capacidades que un sistema debe ser capaz de realizar para cumplir con los objetivos del negocio en cuestión, describen las transformaciones que el sistema debe hacer de acuerdo a las entradas para tener como resultado las salidas que van a satisfacer los objetivos del negocio, están relacionados con el “qué” y no con el “cómo”, esto significa que los requerimientos funcionales deben responder a las necesidades directas que tienen los usuarios, satisfacer la razón por lo cual ellos se acercan e interactúan con el sistema.

Sin estos requerimientos funcionales no puede llevarse adelante el desarrollo de ninguna pieza de software, son la base para especificar las problemáticas que necesitan ser solucionadas por las aplicaciones.

2.1.2. Requerimientos no Funcionales

Los requerimientos no funcionales o también llamados atributos de calidad son los que responden a “cómo” el sistema debe llevar a cabo las acciones para garantizar que se cumplan los requerimientos funcionales, estos atributos de calidad son restricciones impuestas tales como la disponibilidad, seguridad, fiabilidad, mantenibilidad, son los aspectos del sistema, que en general no afectan directamente a la funcionalidad necesitada, sino que definen la calidad y las características que el sistema debe soportar.

Los requerimientos no funcionales son definidos por los distintos stakeholders con los que un sistema puede interactuar a lo largo de toda su vida, estos son los equipos técnicos, los clientes, usuarios, la organización, etc. cada uno determina un conjunto de atributos de calidad que el sistema debe cumplir.

Muchas veces para el negocio no existe una clara diferenciación entre cuáles son los requerimientos funcionales y cuáles no dado que los dos tienen importancia vital para las aplicaciones y la calidad de servicio que debe cumplirse conforme a los objetivos propuestos.

2.2. Concerns: Definición, crosscutting concerns

En el capítulo 2.1 definimos que un requerimiento es una descripción de una capacidad que debe cumplir una pieza de software derivada de la necesidad concreta de un usuario o un conjunto de este.

A la hora de construir un sistema se pasa por la etapa de elicitación, esta consiste en obtener la información necesaria para comenzar a analizar y luego diseñar el sistema, es en la etapa de análisis en donde se construye la especificación de requerimiento de software, cada requerimiento funcional representa una funcionalidad definida que deberá resolver el sistema, tiene una responsabilidad que cumplir y usualmente cada requerimiento equivale a una incumbencia(concern) que le fue asignado.

Muchas veces nos encontramos con incumbencias que no pueden ser encapsuladas limpiamente en requerimientos debido a que afectan a gran parte de los mismos, es decir, pueden atravesar todos los documentos de requerimientos. A este tipo de incumbencias transversales las llamamos CCC[7] por su naturaleza de afectar a un conjunto de requerimiento de software.

Por su naturaleza los CCC son difíciles de modelar y seguir dado que afectan a muchos documentos de requerimientos, es aquí en donde entra en juego una herramienta de software que permita hacer su localización y seguimiento a lo largo de todos los requerimientos y en todo su ciclo de vida.

2.3. Separación de concerns

2.3.1. Paradigmas de programación

Se denomina paradigma de programación a una especificación de índole tecnológico que es adoptada universalmente por una comunidad de desarrolladores, determina la visión y métodos que un programador debe utilizar a la hora de pensar las distintas soluciones que comprende un

[7] Crosscutting concerns

problema que deberá ser resuelto mediante el desarrollo de un programa.
A lo largo de toda la historia de la programación se introdujeron distintos tipos de paradigmas de programación tales como paradigma de programación imperativa, lógica, funcional, declarativa, estructurada, orientada a eventos, orientada a módulos, programación orientada a objetos y a aspectos, entre otros.

La programación imperativa trata de un conjunto de sentencias que a lo largo de la ejecución de un programa tienen la capacidad de modificar el estado del mismo, es decir, un conjunto de instrucciones que le indican a la computadora qué hacer, prácticamente todas las implementaciones de hardware de computadoras responden a este paradigma.

El paradigma de programación lógica se basa en premisas que se dan en un contexto, de acuerdo a un conjunto de reglas que se asumen como verdaderas el programa responde un valor de verdad utilizando lógica matemática. La programación funcional contiene únicamente definiciones de funciones, estas funciones no son subprogramas de un lenguaje imperativo sino matemáticas.

Ningún paradigma es capaz de resolver de manera sencilla y eficiente todos los problemas presentados en el desarrollo de aplicaciones, es importante a la hora de analizar los problemas poder elegir correctamente un paradigma que aporte una solución aceptable.

En la actualidad uno de los paradigmas más utilizados es el de la programación orientada a objetos, este tipo de paradigma trata de modelar la realidad con abstracciones de sus objetos, este paradigma será explicado posteriormente en el siguiente apartado.

2.3.2. Orientado a Objetos

En la POO[8] se modelan entidades llamadas objetos que son abstracciones de los objetos mismos de la realidad, cada uno de ellos tiene un estado, comportamiento e identidad que lo diferencia claramente de otro. En definitiva un sistema para este paradigma de programación es un conjunto de objetos que se envían mensajes entre sí.

[8] Programación Orientada a Objetos

En este paradigma se concentran conceptos fundamentales y características que lo diferencia claramente de otros paradigmas de programación, como ser:

- **Abstracción:** Consiste en modelar una entidad aislandola del contexto o del resto de los elementos que interactúan con esta entidad, haciendo énfasis en qué es lo que hace la entidad por sobre cómo hacer cumplir sus responsabilidades.

- **Encapsulamiento:** Este concepto trata del hecho de poner dentro de una unidad una serie de características propias que la distinguiran de otra unidad.

- **Ocultamiento de la información:** Trata de una decisión de diseño para proteger partes del código en unidades susceptibles a cambios

  Un objeto sabe cómo llevar a cabo sus responsabilidades pero no sabe cómo éstas son llevadas a cabo por el resto de los objetos. Esto se logra proporcionando una interfaz estable que expone las funcionalidades (susceptibles a cambios) implementadas en la unidad, preservando dar conocimiento de la implementación de dichas funcionalidades al resto de los objetos. Este mecanismo asegura que cada objeto sea responsable de modificar su estado interno sin poder acceder a modificar el estado interno de otros objetos de manera directa. Esto minimiza errores dado que cada objeto se encarga de modificar lo que tiene asignado como responsabilidad, es decir, su estado interno.

- **Polimorfismo:** Refiere a la capacidad que tienen los objetos que poseen la misma interfaz o protocolo a tener diferente comportamiento.

- **Herencia:** Es la capacidad de crear una clase a partir de otra clase existente. La subclase creada contiene atributos y métodos de la clase primaria, pudiéndole definir atributos y comportamiento nuevo a esta subclase.

Es interesante cómo este paradigma de programación propone observar la realidad y mediante un proceso de abstracción modelar los distintos objetos que compondrán la solución a partir de la comunicación entre los mismos.

Supongamos el siguiente ejemplo, un editor de figuras, las incumbencias de cada clase son claras, el paradigma trata por sobre todas las cosas de una

limpia separación de incumbencias, pero como pasa en este caso pueden quedar incumbencias transversales a varias clases:

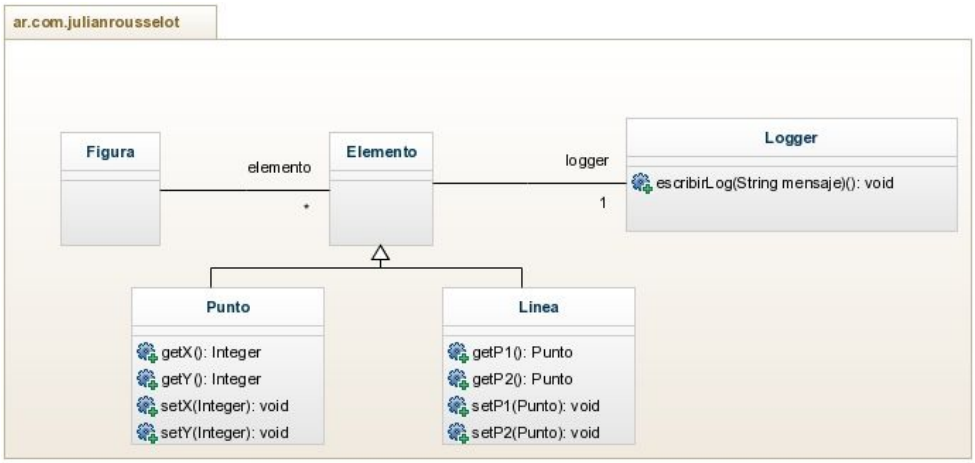


Figura 1: Diagrama de clases de editor de figuras

Cada clase tiene una responsabilidad bien definida, pero se pretende que a parte de cumplir con su incumbencia se registren los cambios de posiciones de las figuras, para ello, cada clase aparte de cumplir con su responsabilidad debe loguear un mensaje con la posición, es decir, deberán llamar al método escribirLog de la clase Logger.

Esta funcionalidad logger es transversal dado que atraviesa a todas las clases, esto trae como consecuencia una disminución de la cohesión de la clase. Retomaremos este ejemplo con más detalle en el punto **2.3.3**, mostrando cómo el paradigma orientado a aspectos soluciona este problema de las incumbencias transversales.

2.3.2.1. Ventajas del paradigma Orientado a Objetos

- **Mejora la productividad en el ciclo de desarrollo de software**, esto se debe a su naturaleza modular.

- **Mejora la mantenibilidad**, también debido a su modularidad se hacen fácil de mantener.

- **Proceso de desarrollo rápido**, el reuso de piezas de software trae aparejado una mayor velocidad en el proceso de desarrollo, la existencia de librerías con conjuntos de módulos permiten a los desarrolladores ahorrar tiempo.

- **Mejora la calidad**, debido a la velocidad en el proceso de desarrollo sumado a los bajos costos y a la reusabilidad permiten dedicarle más tiempo al proceso de verificación del código.

2.3.2.2. Desventajas del paradigma Orientado a Objetos

- **Curva de aprendizaje lenta**, algunas de las propiedades del paradigma como ser la herencia y el polimorfismo pueden llevar tiempo extra de entendimiento.

- **Gran extensión en los programas**, naturalmente tienen más líneas de código que otros paradigmas de programación como por ejemplo la programación estructurada.

- **Programas más lentos**, debido a su complejidad y naturaleza suelen ser más lentos comparados a otros paradigmas dado que necesitan más recursos

- **No apto para resolver todos los tipos de problemas**, hay problemas que se resuelven correctamente con un paradigma funcional, lógico o estructurado, si se resolvieran estos problemas con el paradigma de objetos puede dar como resultado programas poco eficientes.

Como conclusión podemos observar que este paradigma pretende mediante un proceso de abstracción encapsular las incumbencias pero suele darse en muchos casos la existencia de estas incumbencias de manera transversal a todas o varias de las clases de negocio haciendo que éstas pierdan cohesión disminuyendo la calidad de las mismas.

2.3.3. Orientado a Aspectos

Los lenguajes de programación orientados a aspectos definen una nueva unidad de programación en donde se encapsula el comportamiento que de otro modo debería corresponder a un conjunto de objetos a los que atraviesa. Esta metodología complementa al paradigma orientado a objetos.

Es un paradigma relativamente reciente, permite modularizar las aplicaciones según incumbencias logrando una mejor legibilidad y funcionamiento, se motiva en la separación de estas incumbencias.

El diseño y la evolución de los lenguajes de programación ha sido uno de los aspectos más importantes de la ciencia de la computación dado que estos

definen la manera por la cual introducimos los algoritmos que serán procesados por las computadoras.

Retomando el ejemplo del punto **2.3.2**, el editor de figuras, veíamos una funcionalidad llamada logger que atravesaba a todas las clases dado que se requería registrar los cambios de posiciones de las figuras. Esta incumbencia transversal bajaba la cohesión de los objetos en el paradigma orientado a objetos porque que esa funcionalidad no parecía estar relacionada con las responsabilidades generales del objeto.

En el paradigma orientado a aspectos se encapsula esa incumbencia transversal (logging) en una unidad nueva llamada aspecto que le brindará servicios a los objetos, en este caso el logging, dando como resultado una mejor cohesión en los objetos.

2.3.3.1. Separación de incumbencias

La programación orientada a aspectos permite encapsular en módulos diferentes conceptos que conforman un programa en lo que llamamos incumbencia, estas entidades bien definidas tratan de eliminar las dependencias entre cada uno de los módulos, de esta manera se logra eliminar la dispersión del código y las implementaciones resultan ser más entendibles y reusables.

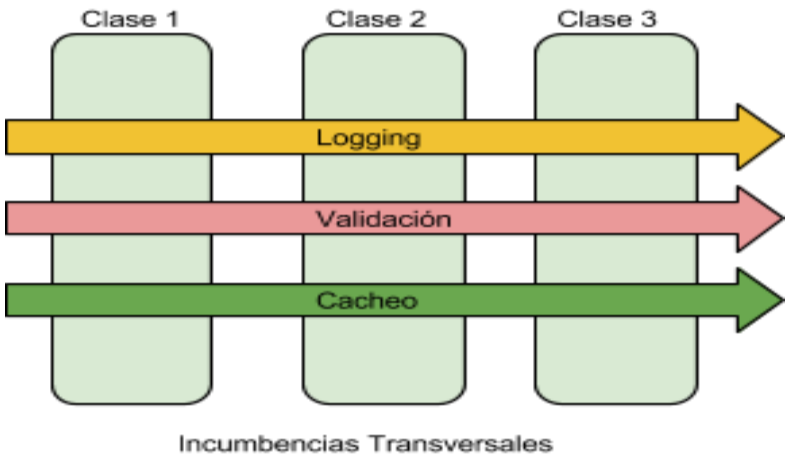


Figura 2: Incumbencias transversales en un sistema

Para comprender este paradigma es necesario introducir una serie de terminología propia que se detalla a continuación (se expresarán las siglas en inglés como comúnmente aparece en la bibliografía con su traducción entre paréntesis):

- **Aspect (Aspecto):** Cuando tenemos una funcionalidad transversal que afecta a varios módulos del sistema podemos extraerla y encapsularla dentro de lo que llamamos aspecto. Una funcionalidad común que suele modelarse como aspecto es el llamado logging o registro de eventos dado que suele afectar a varios módulos del sistema (como puede observarse en la Figura 2).

- **Join point (Punto de Unión)**: El punto de encuentro entre el mundo de los objetos y el de los aspectos está dado en principio en estos Join point, es donde conectamos por ejemplo un objeto con la ejecución de un aspecto. El código del aspecto correspondiente se ejecutará en el momento de encontrarse un join point en un objeto que lo referencie.

- **Advice (Consejo)**: Se llama advice a la implementación del aspecto, dicha funcionalidad será insertada en lo que definimos como join point, contiene el servicio que el aspecto debe brindarle a ese objeto.

- **Target (Destinatario)**: En este apartado hablamos de funcionalidades comunes que suelen encontrarse en muchos objetos, dicha funcionalidad si bien es necesaria para el objeto no atiende a su responsabilidad principal, es decir, en medida de cohesión es un aspecto negativo. Llamamos target a una clase candidata para que se le extraiga el código que si bien necesario no debería tener esa responsabilidad para encapsularla en lo que llamamos aspecto. Estos targets y sus funcionalidades transversales dieron lugar a la necesidad de los aspectos.

- **Pointcut (Puntos de Corte)**: Los advice que son definidos para cada join point se determinan mediante estos pointcut, suelen especificarse mediante patrones de nombres o expresiones regulares, también puede definirse en tiempo de ejecución.

- **Introduction (Introducción)**: Es la capacidad de añadir atributos o métodos a las clases existentes.

- **Weaving (Tejido)**: Llamamos weaving al proceso por el cual aplicamos los aspectos a los objetos target, dando como resultado los objetos y sus pointcuts asociados a un advice.

2.3.3.2. Ventajas de POA

Como beneficios del paradigma orientado a aspectos podemos nombrar:

- **Diseño modular:** Cada incumbencia se implementa en un módulo específico aparte dando como resultado más facilidad de entendimiento y mantenimiento, mejorando la trazabilidad de los módulos.

- **Mejora la evolución del sistema:** Si queremos añadir un nuevo servicio a ser implementado en un aspecto no tenemos que modificar el sistema base, solo basta con agregar el nuevo aspecto con la funcionalidad requerida, esto reduce los tiempos de respuesta a los nuevos requerimientos.

- **Reutilización**: Al estar cada incumbencia claramente separada en su correspondiente aspecto se facilita la reutilización de los mismos.

- **Bajo Acoplamiento**: Los módulos del sistema base no conocen a los aspectos, este acoplamiento bajo favorece a la reutilización.

- **Fácil de agregar aspectos:** A la hora de añadir un nuevo aspecto no hay que modificar el código base, eso reduce los costos y tiempo de implementación fundamentalmente a la hora de implementar incumbencias transversales.

2.3.3.3. Desventajas de POA

La curva de aprendizaje de un lenguaje orientado a aspectos es lenta dando como resultado que este paradigma no esté difundido e implementado ampliamente entre los programadores.

2.3.3.4. Introducción a lenguajes orientados a aspectos

Existen una serie de lenguajes que permiten encapsular las incumbencias transversales en lo que llamamos aspectos, generalmente estos lenguajes deben ser compatibles con los lenguajes base de los objetos a los cuales se les brindarán los servicios. Normalmente los lenguajes orientados a aspectos son extensiones de los orientados a objetos como por ejemplo AspectJ y AspectC, pero también pueden ser lenguajes totalmente diferentes.

Existen lenguajes orientados a aspectos llamados de dominio específico dado que tratan sobre aspectos particulares como pueden ser los que trabajan sobre la persistencia, el manejo de errores, etc. Este tipo de lenguajes no trataban con aspectos para los cuales no fueron diseñados.

Algunos lenguajes orientados a aspectos de dominio específico son:

- COOL[9]
- RIDL[10]
- AML[11]
- RG[12]

Por otro lado tenemos los lenguajes orientados a aspectos de propósito general, es decir, están diseñados para trabajar sobre cualquier clase de aspecto, correspondiendo su nivel de abstracción con el que tiene su lenguaje base. Para los desarrolladores es más fácil adoptar estos tipos de lenguajes orientados a aspectos dado que tienen una correspondencia con los lenguajes base.

AspectJ fue uno de los primeros lenguajes orientados a aspectos de propósito general, surgió como resultado de investigaciones del grupo Xerox PARC dirigido por Gregor Kiczales[13]. La semántica introducida por este lenguaje ha servido de base para los demás lenguajes orientado a aspectos que fueron surgiendo con el correr de los años.

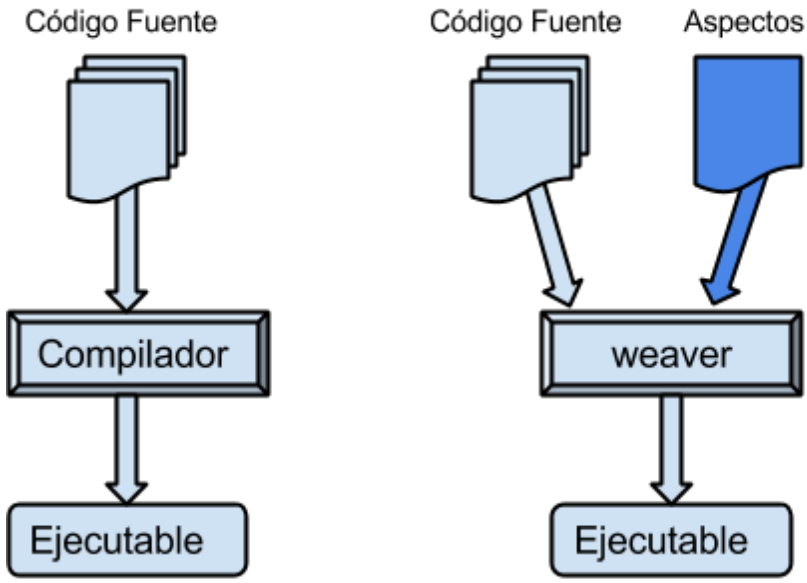
La siguiente tabla muestra la equivalencia entre los lenguajes base y su equivalente para modelar los aspectos:

Lenguaje base	Lenguaje de aspectos
Java	AspectJ / AspectWerkz
C/C++	AspectC / AspectC++
SmallTalk	AspectS / Apostle
Python	Pythius[14]

Tabla 1: Correspondencia Lenguajes base/Lenguajes orientados a aspectos

[9] COOrdination Language
[10] Remote Interaction and Data transfers Language
[11] Extensión de Matlab
[12] Lenguaje para el procesamiento de imágenes
[13] Professor at Department of Computer Science University of British Columbia
[14] Sitio Web de Pythius: http://sourceforge.net/projects/pythius/

Se complementa el lenguaje base con el lenguaje orientado a aspectos mediante el proceso de weaving (nombrado en el punto 2.3.3), es decir, combinamos los aspectos con los módulos que implementan la funcionalidad de negocio del sistema. El módulo encargado de realizar este proceso es llamado weaver, hace uso de reglas particulares para completar el proceso:



Generación de ejecutable según enfoque tradicional POA

Figura 3

En el gráfico se puede observar el desarrollo tradicional en un paradigma como el orientado a objetos y el desarrollo incorporando los aspectos.
Un lenguaje orientado a aspectos al igual que como pasa en otros paradigmas de programación consta de dos partes:

- Una especificación en donde se describen las construcciones y sintaxis del lenguaje.

- Una implementación del lenguaje que corresponde a la especificación del lenguaje declarada que mediante un proceso puede ser ejecutada por una computadora.

2.3.3.5. Aspect-oriented software development

AOSD[15] refiere a un conjunto de técnicas de desarrollo de software que soportan la modularización de las incumbencias transversales en una aplicación, desde la ingeniería de requerimientos hasta la administración de

[15] Aspect-oriented software development

procesos de negocio, pasando por el análisis, diseño, desarrollo testing e implementación completando todo el ciclo.

Busca separar y encapsular en módulos funcionalidad secundaria o de soporte de la lógica de negocio principal de una aplicación. Como se introdujo en el ciclo de desarrollo del paradigma orientado a objetos en el punto 2.3.2, el proceso tradicional consiste en descomponer el sistema en múltiples módulos de funcionalidad bien definida, suele ocurrir que existan responsabilidades que no se ajustan bien a esa funcionalidad encapsulada en esa primera descomposición, comúnmente queda en responsabilidad de los desarrolladores cubrir en módulos todas las incumbencias. **Aspect-oriented software development** pone foco en esa funcionalidad secundaria o de soporte para los objetos identificándola y separándola en una nueva unidad como puede observarse en la Figura 4.

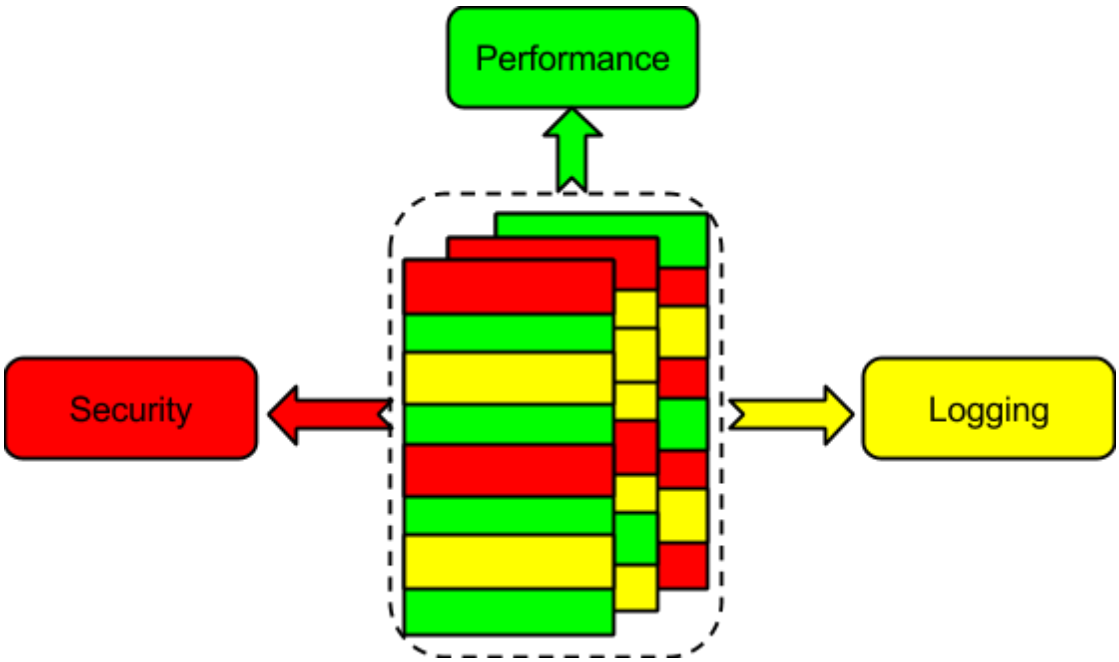


Figura 4: Implementación de módulos

2.3.3.5.1. Aspect Oriented Modelling

En la actualidad nos enfrentamos con sistemas complejos, distribuidos, de grandes dimensiones. Enfoques tradicionales de desarrollo se basaban en la descomposición del sistema en unidades de funcionalidad, en estos tipos de sistemas hay funcionalidades que no pueden ser limpiamente encapsuladas en esas unidades de comportamiento y quedan contenidas en otras. El proceso de desarrollo tradicional dejaba a criterio de los desarrolladores la programación de las unidades funcionales principales asegurándose que el

resto de las funcionalidades queden contempladas en alguna parte del código de la mejor manera posible en el lugar más apropiado.

Los desarrolladores tenían que tener en cuenta todas las funcionalidades a programar, cómo solucionar cada inconveniente o conflicto encontrado en las interacciones entre estas funcionalidades mientras se especifica el comportamiento correcto mediante la codificación de la solución.

Existen incumbencias que abarcan múltiples unidades funcionales y generalmente producen serios problemas a la hora de desarrollo y su posterior mantenimiento. La identificación y separación de las incumbencias se vuelve una tarea crítica tan importante como los documentos de requerimientos que describen el sistema.

La modelización orientada a aspectos de un sistema se concentra en la identificación, descripción y seguimiento de las incumbencias transversales y su modularización en unidades separadas de funcionamiento.

2.3.3.6. Aspect-Oriented Requirements Engineering

AORE[16] brinda soporte a la separación de incumbencias transversales en la etapa temprana de la ingeniería de requerimientos, mejora el entendimiento de la problemática que se plantea solucionar mediante el modelado de un software.

Cuando se menciona a una etapa temprana en el desarrollo de software se quiere destacar que el proceso ocurre antes de la implementación.

"Early aspects are concerns that crosscut an artifact's dominant decomposition, or base modules derived from the dominant separation-of-concerns criterion, in the early stages of the software life cycle. "Early" signifies occurring before implementation in any development iteration." [44] Baniassad.

En esta etapa los requerimientos y la etapa de diseño de software se tratan como un todo en oposición a la etapa de implementación o programación orientada a aspectos.

Un aspecto en la etapa temprana de requerimientos es una incumbencia que atraviesa los requerimientos; identificar y administrar estas incumbencias en una etapa temprana ayuda a mejorar la modularización y a detectar conflictos

[16] Aspect-Oriented Requirements Engineering

tempranamente evitando arrastrar errores a etapas posteriores en el ciclo de vida del desarrollo de software.

2.3.3.6.1. Problemática para aplicar el proceso AORE en grandes documentos

Como expusimos en el apartado 2.1, la Ingeniería de requerimientos es una tarea clave en el proceso de comprensión de cómo debe comportarse el sistema que está siendo construido. Los sistemas complejos poseen un número importante de requerimientos que contienen incumbencias tanto funcionales como no funcionales, en muchos casos los documentos de requerimientos son muy extensos, por lo general permiten ser encapsulados limpiamente en diferentes módulos del sistema resultante. Sin embargo, hay excepciones a esta regla, por ejemplo, es común tener el requerimiento de loguear todas las acciones del sistema, la correspondiente incumbencia de logueo no puede ser encapsulada limpiamente en un módulo porque atraviesa otras incumbencias del sistema (Ver Figura 2).

En los documentos de requerimientos es como si las incumbencias transversales atraviesan estos documentos y en toda la estructura de requerimientos. Para este tipo de incumbencias transversales hacer su seguimiento y saber qué requerimientos afectan es una tarea compleja.

AORE[17] aborda el problema de trabajar con requerimientos de naturaleza transversal teniendo en cuenta la dificultad para aislar las incumbencias, también conocido como aspectos en la etapa temprana trata de identificar estas incumbencias transversales desde un modelado en una etapa temprana de requerimientos.

Estos modelos permiten identificar y gestionar mejor los conflictos en los requerimientos por la naturaleza transversal de las incumbencias. El resultado de este proceso es tener un modelo consistente del sistema en una etapa temprana del ciclo de vida de desarrollo del mismo.

Realizar una AORE [18] en documentos de grandes dimensiones es una tarea difícil debido a la falta de herramientas que soportan la demarcación y el seguimiento de las incumbencias transversales de documentos de requerimientos.

La navegación de las incumbencias transversales puede sonar como una característica básica para cualquier herramienta AORE, por lo que sabemos y

[17] Aspect Oriented Requirements Engineering
[18] Aspect Oriented Requirements Engineering

hasta el momento de la realización de este trabajo, no hay soporte para dicha funcionalidad, esto convierte a la tarea del ingeniero de requerimientos en algo complejo, tendrá que manejar toda la información de un modo no modular no pudiendo determinar fácilmente las incumbencias que afectan a un requerimiento o conjunto de ellos. En nuestra experiencia la interacción de las incumbencias es una fuente difícil de detectar que puede traer consigo errores en etapas posteriores de implementación.

En los capítulos posteriores de este trabajo describiremos el caso de estudio en particular, que dio lugar a la necesidad de contar con una herramienta específica para AORE en cuyas funcionalidades exista el seguimiento de las incumbencias transversales, una herramienta destinada a ayudar al ingeniero para manejar documentos de gran tamaño con múltiples incumbencias transversales.

2.3.4. Comparación entre POA y POO

La programación orientada a objetos es el paradigma más utilizado en las aplicaciones empresariales por su forma colaborativa de llevar adelante las operaciones mediante las abstracciones de la realidad modelada en objetos.

Si bien la POO es muy eficaz a la hora de implementar la lógica de negocio de nuestra aplicación se ve limitada en el momento de incluir funcionalidad que afecta a una gran cantidad de objetos.

Siguiendo el ejemplo del editor de figuras tomado del punto **2.3.2**, teníamos objetos cuya responsabilidad principal era definir y obtener su posición en x e y pero también existía una funcionalidad secundaria para cada objeto cuyo fin era que se registre cada cambio de posición.

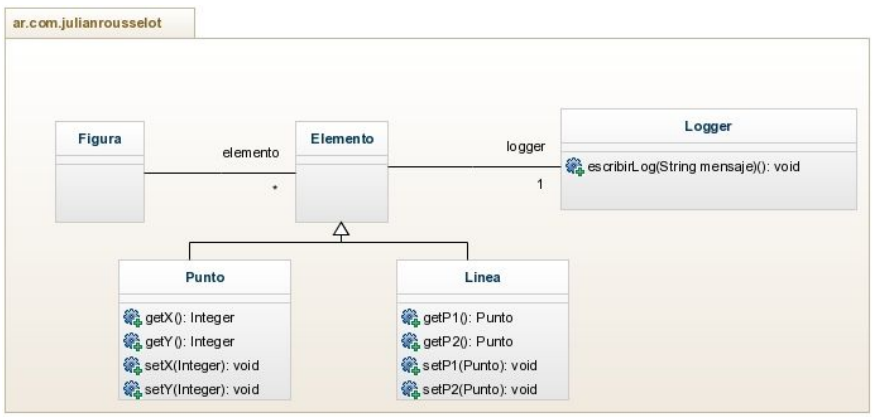


Figura 1: Diagrama de clases de editor de figuras

Si utilizamos unos patrones de colores para describir la funcionalidad de los objetos del ejemplo y su cohesión podríamos obtener el siguiente gráfico:

Objetos y sus responsabilidades

Figura 5

Si analizamos el gráfico podemos observar un conjunto de objetos con funcionalidades bien definidas según los colores de cada uno, pero también se puede apreciar una funcionalidad común a todos los objetos representada en color rojo, el logger, dicha funcionalidad si bien es necesaria en cada uno de los objetos en donde se encuentra no representa una tarea principal según la responsabilidad asignada, esta funcionalidad hace que el objeto disminuya su cohesión porque poco tiene que ver con su tarea principal, esta funcionalidad es la que llamamos en este trabajo crosscutting concern o incumbencias transversales.

La POA colabora con la POO encapsulando en aspectos esa funcionalidad que se encuentra en el conjunto de objetos sin pertenecer necesariamente a ellos. De esta manera los objetos suben su cohesión y la funcionalidad encapsulada en los aspectos es brindada como servicio a través de ellos:

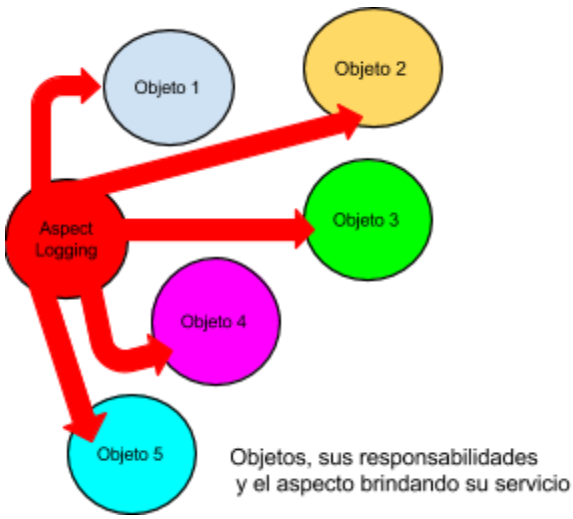


Figura 6: Representación de un conjunto de objetos y aspectos

Como conclusión vemos que la POA es un complemento de la POO manejando ciertas funcionalidades que se encuentran distribuidas en los objetos para encapsularlas y brindarlas como servicio a los mismos.

3. Caso de estudio

3.1. Introducción al dominio

El caso de estudio refiere al dominio de las tragamonedas, una experiencia [13] en donde se utilizaron aspectos para resolver la problemática planteada por el complejo dominio.

Una máquina tragamonedas es un dispositivo de juego. Cuenta con cinco carreteles que giran cuando se pulsa un botón de comienzo de juego.

Un tragamonedas incluye algunos medios para entrar dinero, dicho dinero asigna créditos al juego. El jugador apuesta una cantidad de créditos en cada jugada, el tragamonedas selecciona al azar el símbolo que se muestra para cada carretel, y paga el premio correspondiente, si lo hubiere. Los créditos pueden ser extraídos (llamado reintegro) por diferentes mecanismos tales como monedas, billetes o transferencias electrónicas.

El concepto del juego tragamoneda es desarrollado por los diseñadores del juego, las aplicaciones deben obedecer a una serie de regulaciones que controlan el hardware y el software del dispositivo.

El modo del juego es siempre, el de una máquina tragamonedas, lo que varía es la imagen alrededor de cada carretel.

Los reglamentos que se aplican a la máquina tragamonedas se pueden dividir en tres grandes grupos:

- **Regulaciones Gubernamentales:** Las regulaciones del gobierno cubren un amplio espectro de características de los dispositivos de juego: el pago, el azar, la conectividad, premios compartidos, etc Un ejemplo de estos son el Reglamento del estado de Nevada[19] en USA.

- **Normas**: Para garantizar un comportamiento adecuado de los tragamonedas, existen certificaciones y controles de calidad sobre los dispositivos.

[19] Nevada Gaming Commission. Technical Standards For Gaming Devices And On-Line Slot Systems, 2008. Available at: http://gaming.nv.gov/stats regs.htm.

El comportamiento esperado de un tragamonedas se define en los documentos llamados estándares, un ejemplo es el Estándar GLI[20].

- **Especificaciones técnicas:** Algunos requerimientos están relacionados con la conectividad del tragamonedas con los sistemas de reporte y su protocolo de comunicación. Este es el caso, por ejemplo, el protocolo G2S(Game to Server), un estándar abierto para la comunicación del tragamonedas con un backend[21].

Los requerimientos para el sistema de tragamonedas se definen en una serie de documentos(reglamentos, normas, especificaciones del protocolo, etc.) escritos por diferentes partes interesadas, con diversos intereses y orígenes. Esto resulta en un gran conjunto de documentos en donde se utilizan diferentes términos para describir un mismo objeto.

Los documentos tienen una gran extensión, pueden tener desde 90 (Regulación del estado de Nevada) a 1500 (especificación del protocolo G2S) páginas.

La gran fuente de requerimientos cuya organización y estructura está fuera del alcance del Ingeniero hace que sea difícil de reorganizar agrupando los requerimientos en incumbencias. Por otra parte, es inviable modificar los documentos originales dado que están bajo control de sus propietarios.

3.2. Incumbencias en el dominio de las tragamonedas

Varios documentos tratan con casi las mismas incumbencias, pero abordadas desde una perspectiva diferente. Si consideramos, por ejemplo, los medidores[22], que se deben actualizar a medida que transcurre el juego (desde su comienzo) son la base de la contabilidad y presentación de informes. La primera acción (actualización después de una jugada) podrá indicarse en un reglamento como el del estado de Nevada, esto significa que un elemento de una incumbencia puede ser tratada en diferentes documentos desde el punto de vista de su dueño.

Los reglamentos de los distintos estados tratan las diferentes incumbencias como ser las de medición, protocolos de comunicación, etc., a continuación se explican brevemente:

[20] Gaming Laboratories International. Gaming Devices in Casinos, 2007. Disponible en: http://www.gaminglabs.com
[21] Sistema externo de consulta
[22] Refiere a contadores dentro del sistema

- La incumbencia del juego incluye los requerimientos relacionados con los tragamonedas, contempla desde que comienza el juego en donde se apuestan créditos por parte del jugador y es recompensado de acuerdo a un resultado que se determina al azar.

- La incumbencia de los medidores refiere a que deben ser mantenidos. Miden todos los aspectos del tragamonedas para propósitos de auditoría, por ejemplo: Número de jugadas, cantidad de apuestas, totales ganados, etc.

- La incumbencia G2S abarca los requerimientos que refieren al protocolo de comunicación utilizado para monitorear remotamente a las tragamonedas, establece cuándo y cómo debe ser reportada la información. Parte de la información es guardada en los medidores.

- Condiciones de error definen cómo el tragamonedas debe comportarse en determinadas circunstancias, como por ejemplo si la puerta de la máquina está abierta, compartimento de monedas lleno, etc. Este comportamiento a menudo incluye bloquear el equipo hasta que intervenga un operador.

- Game Recall refiere a una funcionalidad de auditoría local. Cada juego debe ser almacenado con sus parámetros y resultados, esto es útil para la resolución de problemas con los jugadores, al menos las últimas diez jugadas deben ser guardadas.

- Reanudar el programa es análogo al requerimiento no funcional de persistencia. En este dominio existe cierto tipo de información que debe ser reconstituida después de reiniciar el sistema, dicha información incluye los medidores, los estados del tragamonedas, condiciones de error, colas asociadas a los protocolos de comunicación y datos del juego.

3.3. Crosscutting Conerns en el dominio de las tragamonedas

Hablando en terminología orientada a aspectos se puede decir que la incumbencia del juego es la base de las incumbencias, el resto de las incumbencias son transversales, consideremos por ejemplo la reanudación del programa de un tragamonedas, esta incumbencia es transversal a muchas otras en los requerimientos.

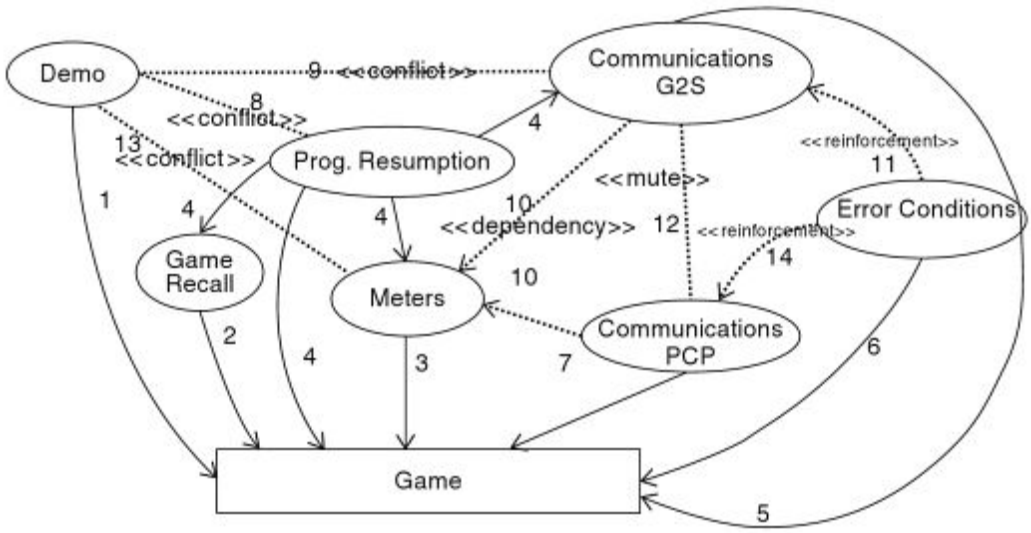


Figura 7: Interacción de concerns. Imagen obtenida del libro Aspect-Oriented Requirements Engineering De A. Zambrano, J. Fabry

En la figura 7 los óvalos con las flechas que contienen líneas continuas expresan incumbencias transversales, flechas con líneas punteadas indican la interacción entre las incumbencias, la notación expresada se utiliza para modelar las interacciones de las incumbencias transversales. Puede observarse que la incumbencia juego está representada por un rectángulo afectado por incumbencias transversales representadas por óvalos.

Las relaciones que aparecen numeradas del 1 al 7 son incumbencias transversales (flechas con líneas continuas), por cada relación hay incumbencias transversales donde uno o más requerimientos atraviesan otros. Si observamos la relación entre las incumbencias **Error Conditions** y **Game** donde el comportamiento asociado a **Error Conditions** necesita estar "tejido" en el comportamiento de **Game**. El comportamiento de la incumbencia **Game** puede arrojar un **Error Conditions** en diferentes momentos como ser cuando se ingresa un billete en la máquina, la impresora se quedó sin papel, se abrió la puerta de la tragamoneda, etc.

4. Trabajos relacionados

En este capítulo presentamos un estudio y relevamiento de las herramientas comerciales existentes en el mercado y de los trabajos de investigación difundidos como publicaciones científicas/académicas.
Evaluamos para cada herramienta la capacidad de trabajar sobre los documentos originales de requerimientos, identificando manualmente los diferentes concerns y sus interacciones. Esta capacidad fundamental para todo ingeniero de requerimientos no fue contemplada por las herramientas que analizamos.

Hay muchas herramientas para la gestión de los requerimientos de software. Un ejemplo es el **Enterprise Architect**[23], una herramienta potente para el análisis y diseño abarcando todo el ciclo de vida de desarrollo de software, a través de las etapas de análisis, modelos de diseño, pruebas y mantenimiento. Otras herramientas reconocidas son acompa[24] y Lighthouse[25], por lo general realizan las mismas tareas que el EA[26].

La siguiente es una selección de las herramientas más destacadas en el mercado en lo que hace a la gestión de requerimientos(ordenadas alfabéticamente):

Blueprint Requirements Center:[27] Soporta la definición de requerimientos, administración de requerimientos, creación de mockups para presentación y modelización visual.

Caliber[28]: Soporta administración de requerimientos y modelización visual.

codeBeamer Requirements Management[29]: Herramienta para administración de requerimientos

Enterprise Architect:[30] Provee administración de requerimientos y una herramienta visual de modelización

[23] http://www.sparxsystems.com/
[24] http://web.accompa.com/
[25] https://lighthouseapp.com/
[26] Enterprise Architecture
[27] http://www.blueprintsys.com/
[28] http://www.borland.com/
[29] http://www.intland.com/
[30] http://www.sparxsystems.com/

HP Agile Manager[31]: Provee la capacidad de administración de requerimientos y herramientas para modelado con metodologías ágiles

HP Quality Center, ALM[32]: Administración de requerimientos, administración de proyectos, pruebas y manejo de incidentes.

IBM Rational DOORS[33]: Administración de requerimientos

Innovator for Business Analysts[34]: Modelización visual y administración de requerimientos.

inteGREAT[35]: Provee construcción de requerimientos, administración de requerimientos, modelización visual

Jama[36]: Administración de requerimientos y pruebas

Jira[37]: Administración de incidentes, administración de requerimientos y proyectos

Kovair ALM Studio[38]: Administración de requerimientos y pruebas

Mingle[39]: Administración de requerimientos, de proyectos, metodologías ágiles

Polarion Requirements[40]: Provee administración de requerimientos

PTC Integrity[41]: Soporta administración de requerimientos y pruebas

Rally[42]: Administración de requerimientos y metodologías ágiles

[31] http://www.hp.com/
[32] http://www.hp.com/
[33] http://www.ibm.com/
[34] http://www.mid.de/
[35] http://www.edevtech.com/
[36] http://www.jamasoftware.com/
[37] http://www.atlassian.com/
[38] http://www.kovair.com/
[39] http://www.thoughtworks-studios.com/
[40] http://www.polarion.com/
[41] http://www.mks.com/
[42] http://www.rallydev.com/

Serena Dimensions[43]: Provee administración de requerimientos

TestTrack[44]**:** Administración y gestión de requerimientos

VersionOne[45]: Administración de requerimientos, metodologías ágiles

Visure Requirements [45]: Administración y modelado de requerimientos

Todas estas herramientas hacen un gran trabajo proporcionando la trazabilidad en la gestión de los requerimientos. Las herramientas comerciales existentes no proveen soporte para **CCC**[46], por lo tanto no dan soporte para AORE[47].

Awais Rashd en el paper titulado **TOOL SUPPORT FOR ASPECT-ORIENTED REQUIREMENTS** [46] presenta una herramienta llamada **ARCADE** para identificar aspectos en la etapa de requerimientos y su influencia en otros requerimientos, detección de conflictos entre requerimientos "aspecturales" y requerimientos simples. La herramienta se presenta en un caso de estudio que refiere a un sistema de subastas online que contempla la compra y venta de bienes por Internet, como premisa el sistema debía garantizar un mercado seguro y fiable en donde todas las partes que interactúan en la transacción deben ser solventes.

La herramienta también fue utilizada para asistir en distintos casos de reingeniería en arquitecturas que contemplaban los aspectos en el sistema, fue el caso por ejemplo de un sistema de control de tráfico aéreo, el sistema estaba especificado en varios documentos con cientos de páginas cada uno.

Otro caso de estudio en donde se utilizó la herramienta fue en un sistema para un simulador de telemetría por satélite pero a diferencia del caso anterior en este no había una especificación extensa de requerimientos, solo se contaba con los manuales de usuario y una descripción del sistema. La herramienta, según los autores del paper ha ayudado en gran medida a obtener conocimientos tempranos en lo que refiere a puntos de vista y aspectos para guiar en la reingeniería.

[43] http://www.serena.com/
[44] http://www.seapine.com/
[45] http://www.versionone.com/
[46] Cross Cutting Concern
[47] Aspect Oriented Requirement Specification

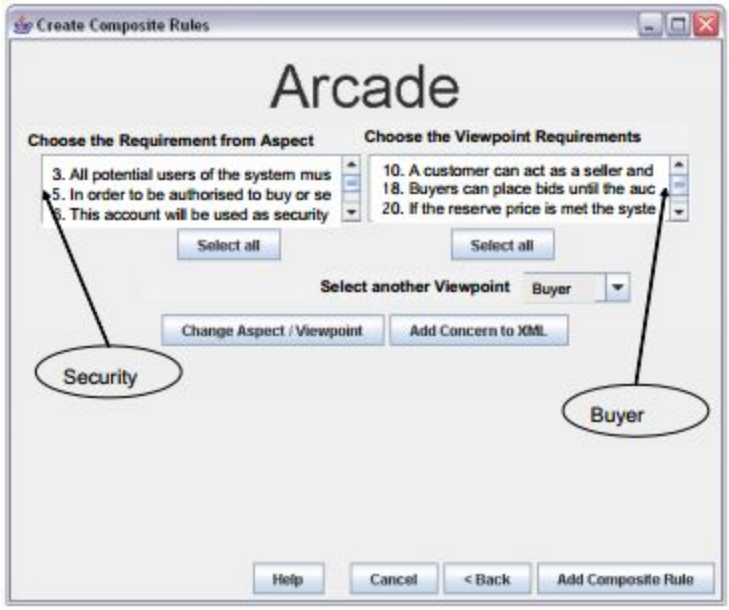


Figura 8: Vista de la herramienta Arcade

La herramienta **Lancaster Framing Tool** surgió de un trabajo del instituto de investigación de la Universidad de Lancaster [47] propone un set de herramientas para la identificación de los aspectos en la etapa de requerimientos, ayuda a modularizar las incumbencias transversales, los aportes fundamentales son:

- Ayuda a especificar las dependencias entre los requerimientos
- Analiza dependencias en búsqueda de conflictos

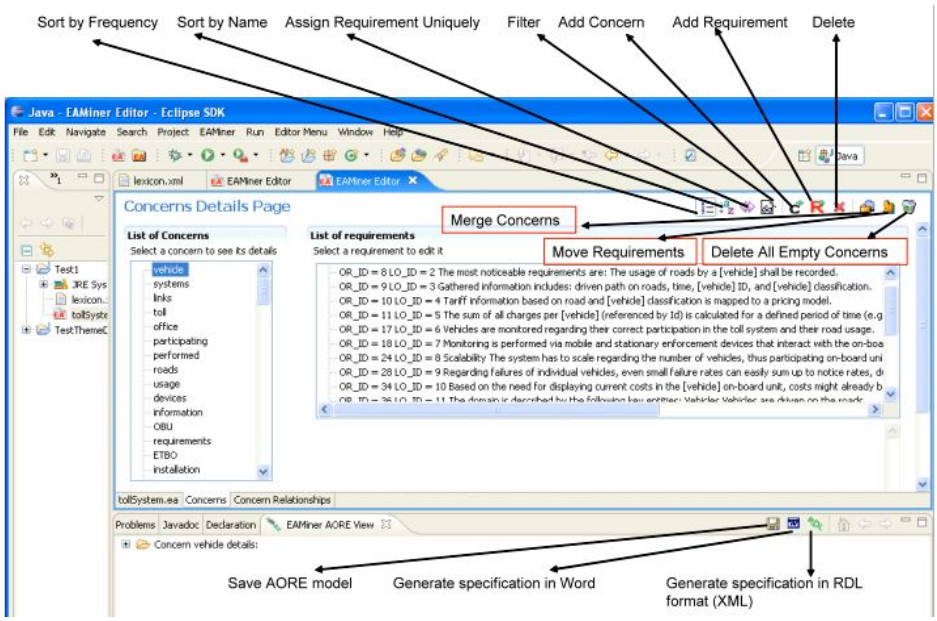


Figura 9: Vista de la herramienta Lancaster Framing Tool

4.1. Conclusión

Luego del análisis de las herramientas comerciales y trabajos de investigación podemos concluir que dichas herramientas no dan soporte para la identificación y seguimiento de las incumbencias transversales. Una característica general de estos enfoques y sus herramientas de apoyo es que exigen que los requerimientos sean introducidos manualmente, algunos trabajan con documentos XML, pero no tienen la opción de convertir los requerimientos en documentos XML con una estructura deseable, algunos enfoques solucionan este inconveniente haciendo un análisis automático del texto de los requerimientos, este enfoque tiene la complejidad derivada de trabajar con lenguaje natural.

Dada nuestra experiencia con documentos de requerimientos en un caso de magnitud industrial dudamos que sea factible un análisis automático textual que tenga un buen resultado, en cambio, creemos que los requerimientos deben ser analizados por ingenieros con conocimientos en el dominio del problema.

5. Desarrollo de una aplicación AORE para dominios complejos

En este capítulo presentamos las capacidades necesarias para una herramienta que soporte AORE en dominios con documentos de gran extensión, capacidades necesarias para el trabajo del Ingeniero de requerimientos.

5.1. Requerimientos

Considerando las características de nuestro caso de estudio, derivamos la presente lista de requisitos para una herramienta que ayude a la manipulación de incumbencias transversales en documentos de requerimientos de gran extensión.

5.1.1. Importación de Documentos

Los requerimientos para los tragamonedas son generados por instituciones externas, la única excepción es el documento de diseño del juego. El contenido de estos documentos está bajo el control de los ingenieros de cada institución y a los ingenieros de requerimientos (el juego) no se les permite la modificación de ninguno de los requerimientos correspondientes. Por lo tanto, la herramienta propuesta en este trabajo no necesita capacidades de creación o edición de documentos. En cambio, los reglamentos, normas y otros tipo de documentos se importarán en la herramienta para que puedan ser trabajados.

La mayoría de estos documentos se entregan en formato PDF o como documentos de MS Word. Ambos formatos (así como muchos otros) se pueden convertir fácilmente a HTML.

Por lo tanto, decidimos tomar como entrada de nuestros documentos **HTML** para ser trabajados en la herramienta.

5.1.2. Demarcación

Los documentos de entrada contienen los requerimientos pero no información explícita sobre cuáles son las diferentes incumbencias contenidas ellos, puede darse también que las incumbencias no coinciden una a una con los requerimientos. También podemos encontrar incumbencias transversales que afectan a varios requerimientos. Mientras el ingeniero de requerimientos

trabaja con el documento va identificando las incumbencias y dónde aplican. La herramienta debe proporcionar la capacidad de definir las incumbencias por un lado y la demarcación en el texto de los requerimientos. Cada incumbencia estará asociada a un color específico, luego deberá poder especificar a qué parte del documento pertenece cada incumbencia.

Con el texto de requerimientos demarcado con las incumbencias, el ingeniero de requerimientos podrá detectar rápidamente de manera visual las incumbencias de los requerimientos y la existencia de incumbencias transversales.

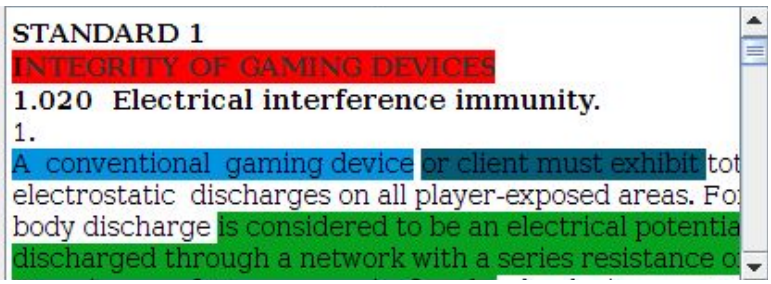


Figura 10: Demarcación de las incumbencias en los documentos de requerimientos

5.1.3. Navegación por incumbencia

Una vez que los documentos han sido procesados y la información relativa existente a las incumbencias se ha añadido, es deseable contar con capacidades de navegación que nos permiten explorar únicamente los requerimientos que pertenecen a una incumbencia específica.

Consideremos el caso de un ingeniero de requerimientos que quiere saber cómo hacer frente a las limitaciones de rendimiento contenidas en documentos como los reglamentos, normas correspondientes y otros. En este caso, es necesario poder localizar fácilmente todos los requerimientos que imponen restricciones de rendimiento o afectan el rendimiento del sistema.

5.1.4. Escalabilidad

Tener un gran volumen de documentos potencialmente grandes hace que la escalabilidad de la herramienta sea clave para que pueda manejarlos.

La presentación de esta información utilizando metáforas gráficas ayuda para la rápida comprensión de cómo se componen los documentos y dónde buscar información específica.

Por lo tanto , Se requiere que la herramienta sea capaz de proporcionar vistas de abstracción de la documentación. El usuario debe ser capaz de tener fácilmente esta visión de conjunto con los detalles específicos de los documentos. Por lo tanto, debería ser posible hacer zoom en partes específicas, revelando el texto real de los requerimientos.

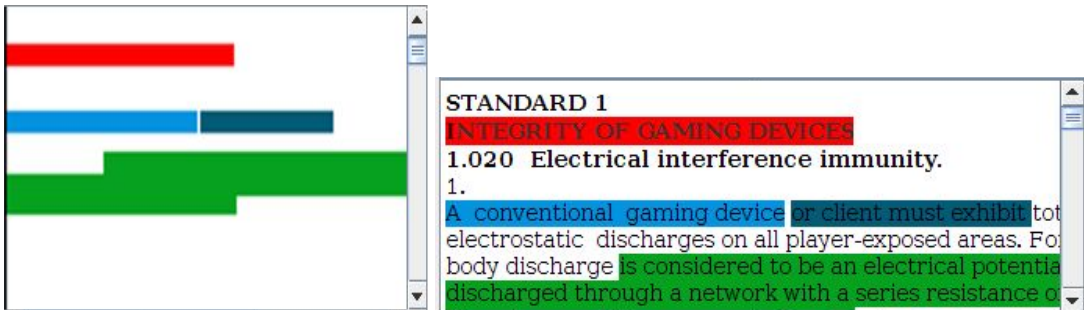


Figura 11: Vista abstracta de incumbencias y su correspondiente texto de requerimientos

5.1.5. Evolución de documentos

En el curso de los proyectos, cada documento de requerimiento pasa por varias revisiones, donde sus contenidos cambian. Aunque a un ritmo lento, tanto las normas, como reglamentos están sujetos a la aprobación de las respectivas organizaciones, no obstante evolucionan para hacer frente a los nuevos escenarios, tecnologías y otras necesidades.

Es necesario poder migrar la información de las incumbencias añadidas de una versión de un documento a una nueva revisión, con el fin de evitar duplicidades en el trabajo entre las revisiones. Para lograr este objetivo es necesario poder mantener la información relacionada con la incumbencia del

documento original, y para las nuevas versiones poder localizar las partes del documento que se mantienen sin cambios.

5.2. AORE Assistant

En esta sección se describen las funcionalidades de la herramienta **AORE** Assistant cuyo objetivo es proporcionar apoyo a los ingenieros de requerimientos para la detección y demarcación de los aspectos en una etapa temprana de requerimientos dispersos en un gran volumen de documentos de gran extensión, como los que se nombran en el caso de estudio esbozado en el capítulo 3.

El desarrollo de la herramienta fue utilizando Java Standard Edition, con componentes gráficos de swing y AWT.

La información relacionada con las incumbencias es manejada por separado y almacenada en documentos XML, especificando todas las asociaciones entre los requerimientos y incumbencias. Con esto logramos de cierta manera desacoplar los documentos de requerimientos de las demarcaciones (incumbencias), contemplando que los documentos evolucionan y cambian constantemente.

Como se mencionó, la herramienta toma los archivos HTML como entrada. La razón para hacer esto es que se quiere preservar el aspecto visual del documento original, al cual el ingeniero de requerimientos se encuentra habituado para trabajar.

Los documentos que inspiraron este trabajo por lo general se entregan como .doc (MS Word) y archivos PDF de Acrobat. Los documentos que utilizan estos formatos pueden convertirse en HTML usando herramientas existentes tales como pdftohtml (para Linux) y otras herramientas.

5.2.1. Utilización de la herramienta AORE Assistant

En esta sección se revisa brevemente el uso esperado de la herramienta. La Figura 12 muestra los pasos que se detallan en los siguientes apartados para utilizar correctamente la herramienta.

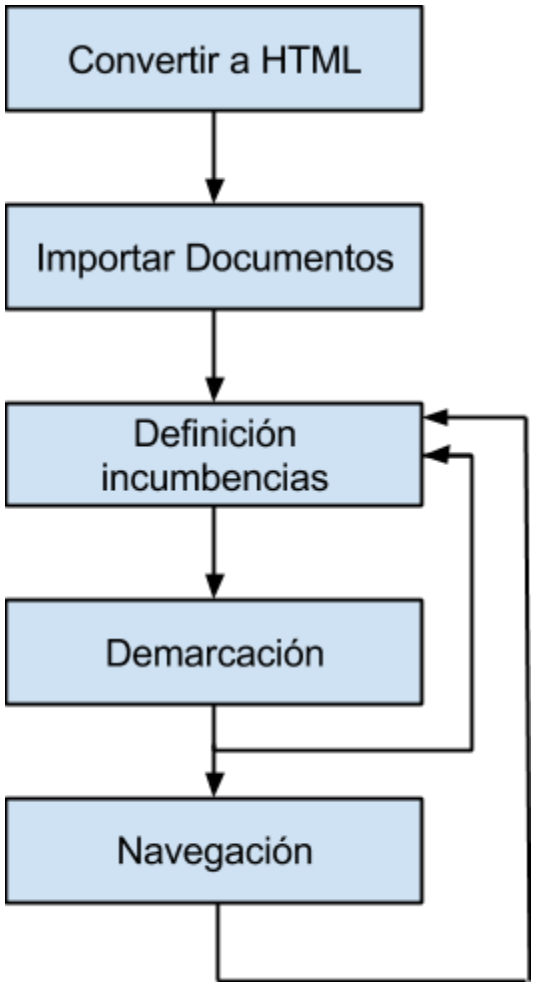


Figura 12: Pasos esperados para utilizar la herramienta AORE Assistant

5.2.2. Importar Documentos

Como se mencionó anteriormente, los Documentos de requerimientos existentes deben ser convertidos a HTML para que puedan ser importados en la herramienta AORE Assistant para ser trabajados, se respetará el formato original del documento.

Nuestra intención es que al usar AORE Assistant, el ingeniero de requerimientos perciba a nivel visual un documento prácticamente idéntico al original.

Las figuras 13, 14, 15 muestran el proceso de importación.

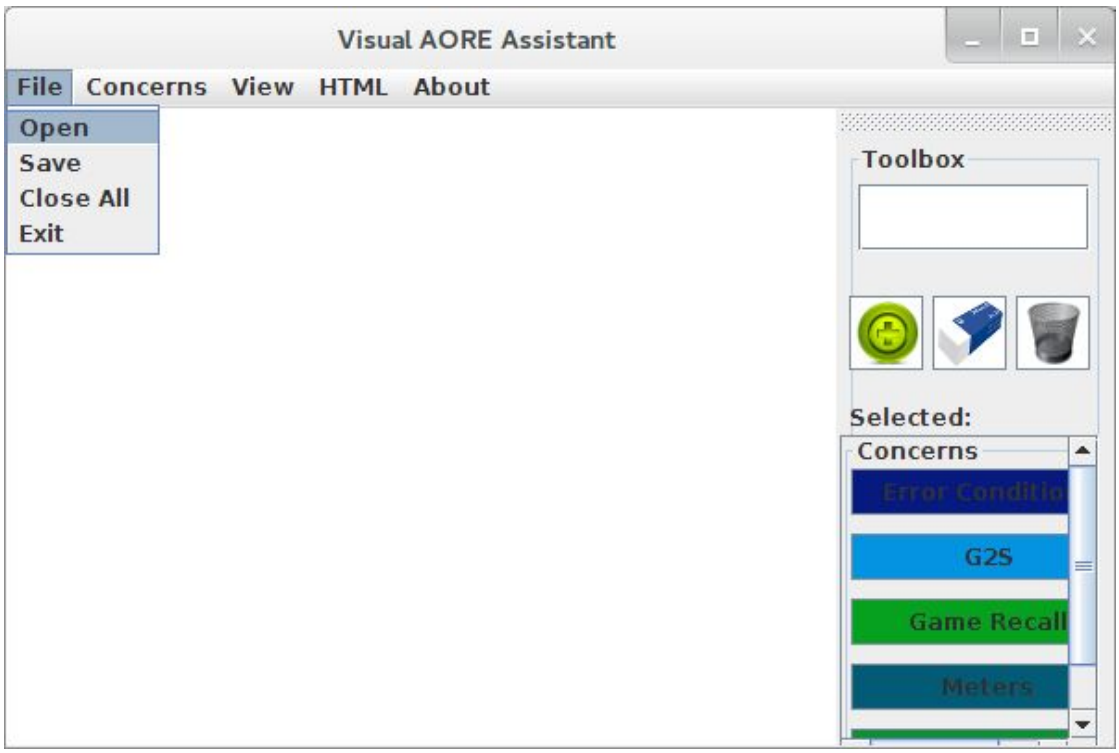


Figura 13: Importación de documento de requerimiento

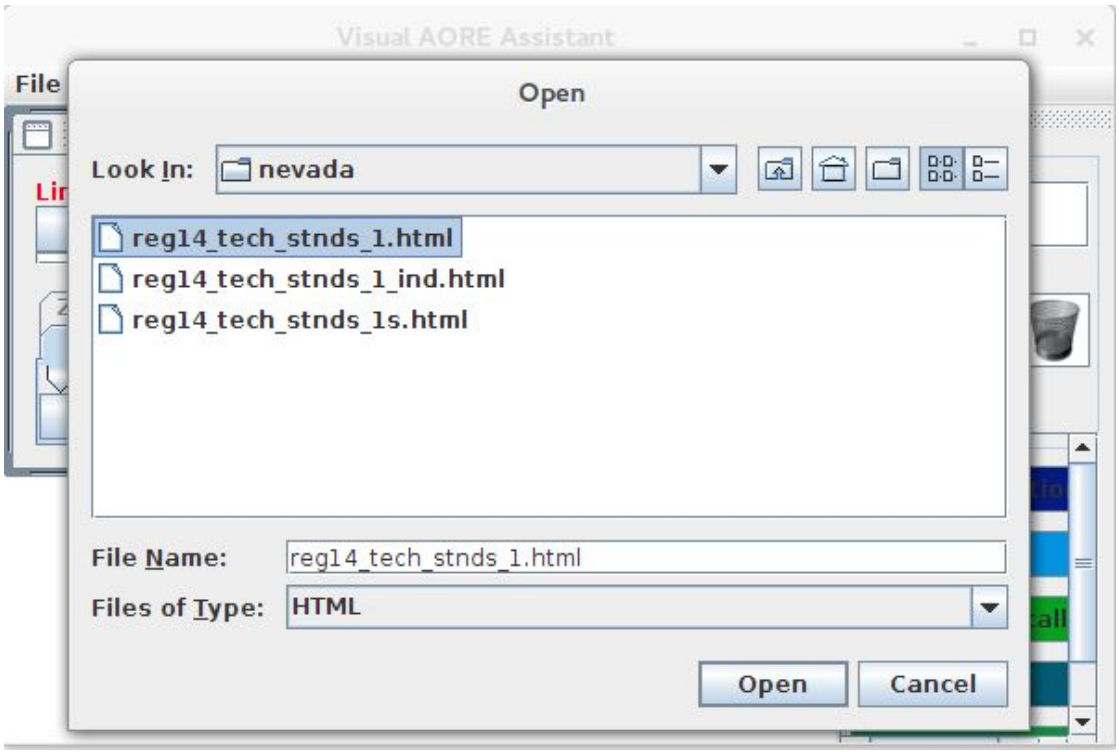


Figura 14: Importación de documento de requerimiento

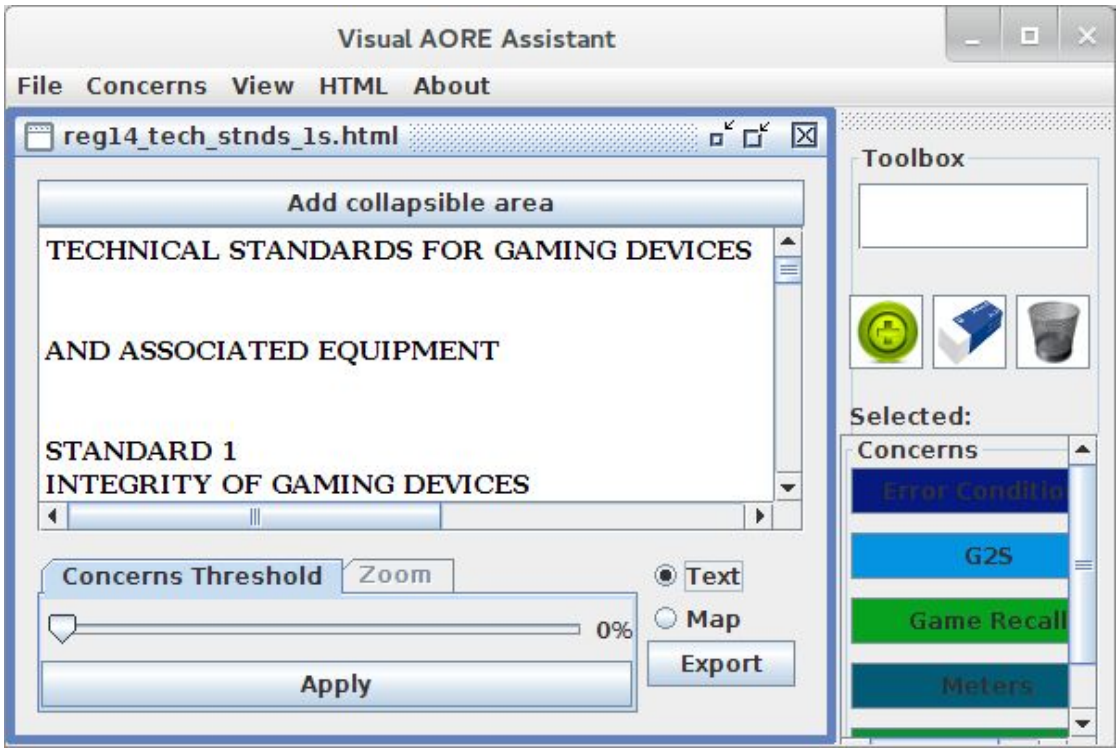


Figura 15: Importación de documento de requerimiento

5.2.3. Definición de Concerns(Incumbencias)

Antes de iniciar el proceso de demarcación, las incumbencias se deben definir en la herramienta. Esto se puede hacer mediante el botón con el signo + (más) (ver Figura. 16).

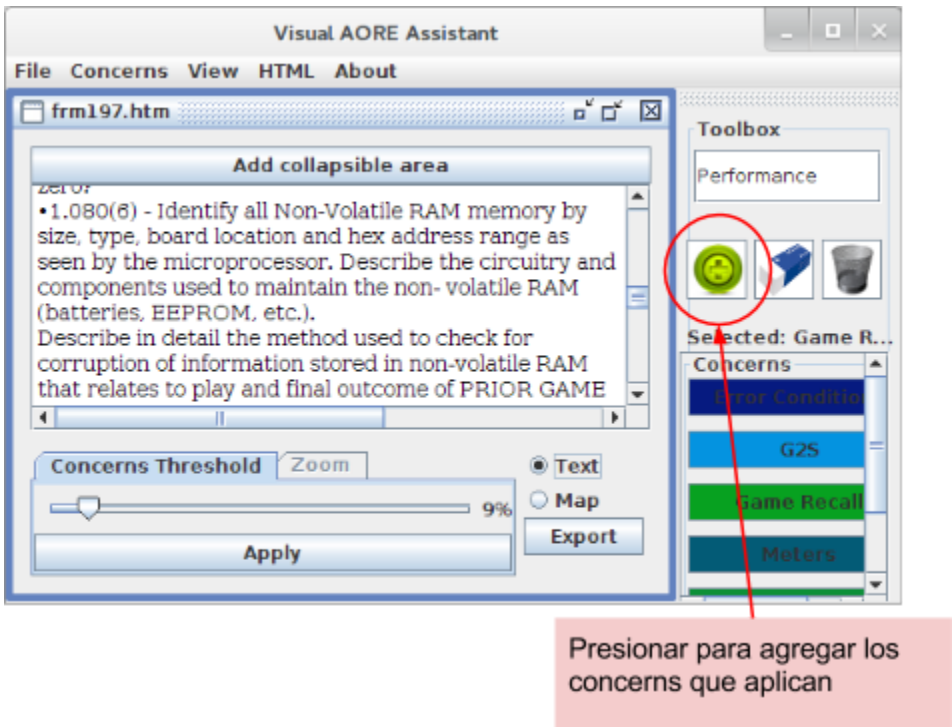


Figura 16: Agregando concerns al panel

Además de nombrar la incumbencia, es necesario elegir un color para ella. Este color es utilizado por la herramienta para representar la incumbencia en las diferentes vistas.

La definición de las incumbencias no sólo pueden hacerse en el comienzo del proceso de ingeniería, sino también en las fases posteriores.

Cada vez que el ingeniero encuentra una nueva incumbencia esta puede ser añadida a la herramienta.

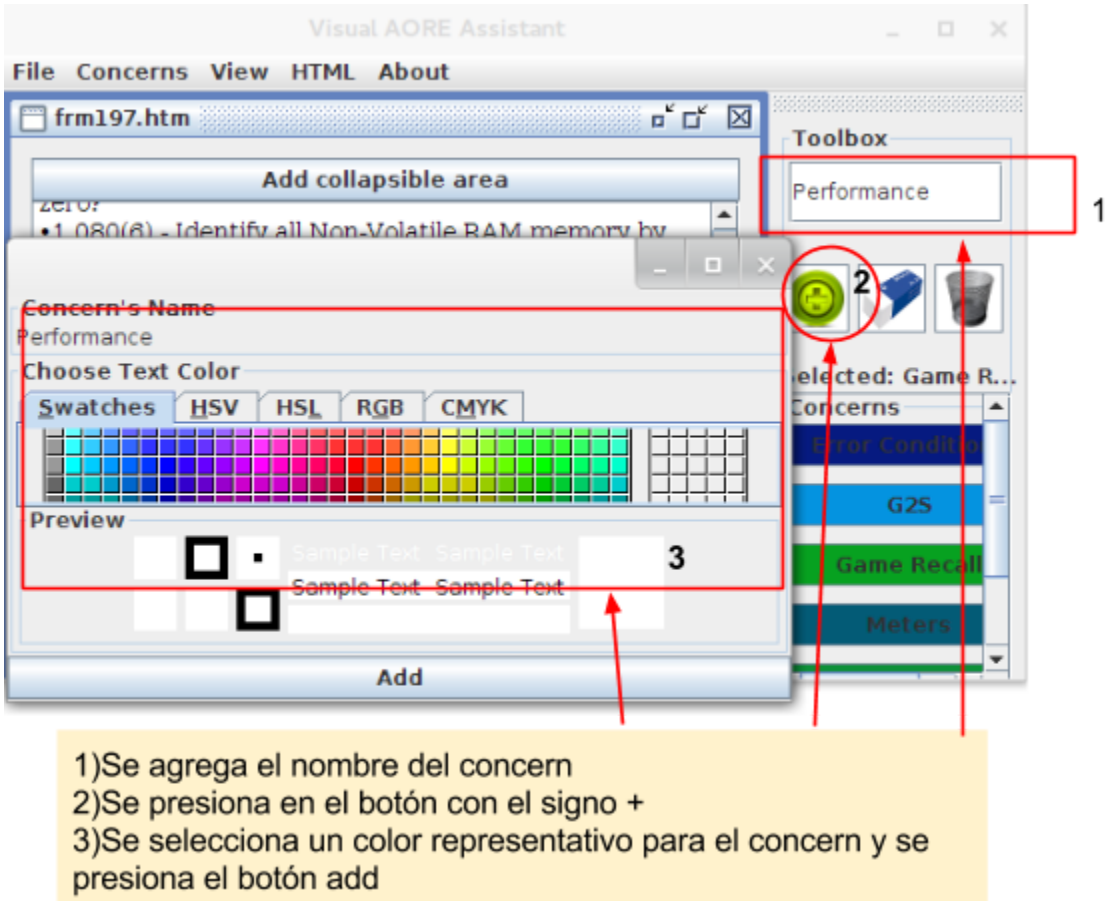


Figura 17: Agregando un concern

Una vez que los documentos son importados y las incumbencias están definidas, la herramienta se ve como en la Figura 18. Los documentos todavía no están coloreados, ya que ninguna demarcación ha sido llevada a cabo.

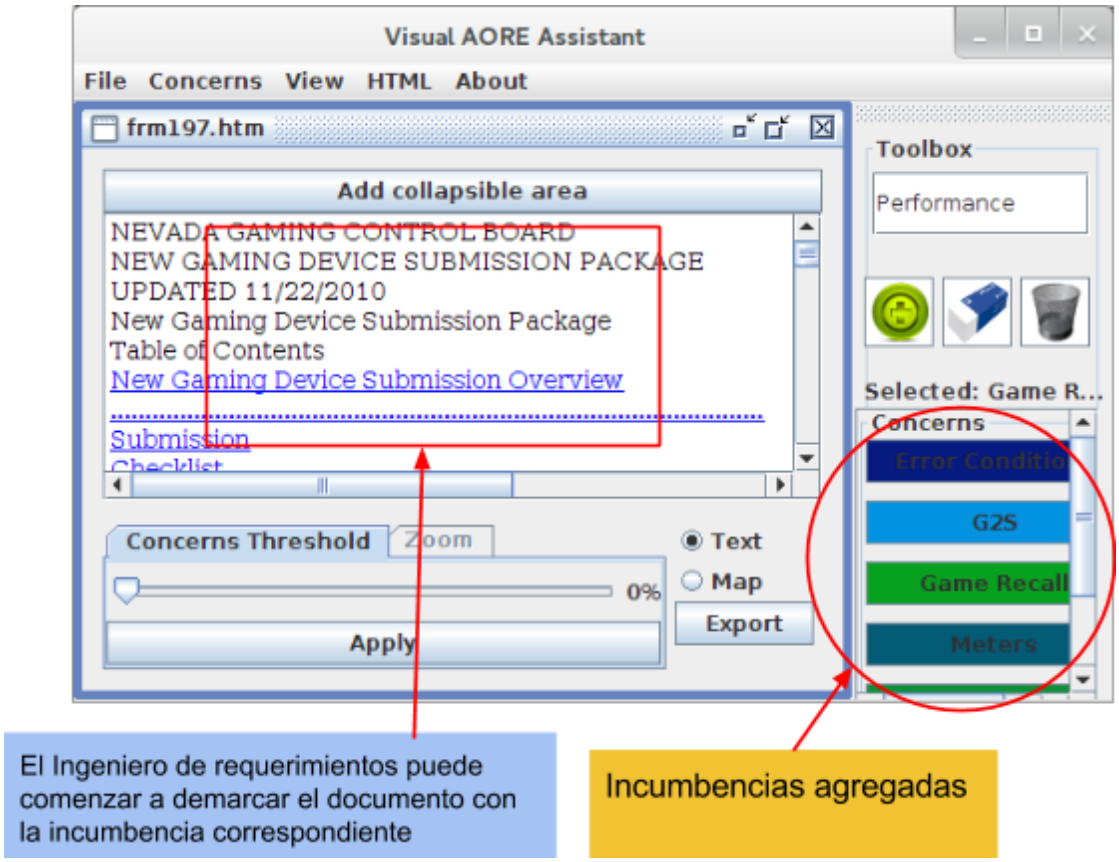


Figura 18: Herramienta lista para comenzar la demarcación

5.2.4. Demarcación

En esta fase, el ingeniero de requerimientos puede examinar los documentos y demarcar partes del texto como perteneciente a una incumbencia u otra.

Esto se hace seleccionando el texto y pulsando el botón asociado a la incumbencia deseada. Luego que el ingeniero procede a la delimitación de los documentos se verá como en la Figura 19, donde se han aplicado algunas incumbencias al texto.

Como resultado, las diferentes partes del texto de requerimientos quedan demarcadas con diferentes colores correspondientes a la incumbencia a la cual pertenecen.

La demarcación puede hacerse a nivel de palabra, es decir un párrafo puede tener asociado un conjunto de incumbencias, cada una con su color representativo.

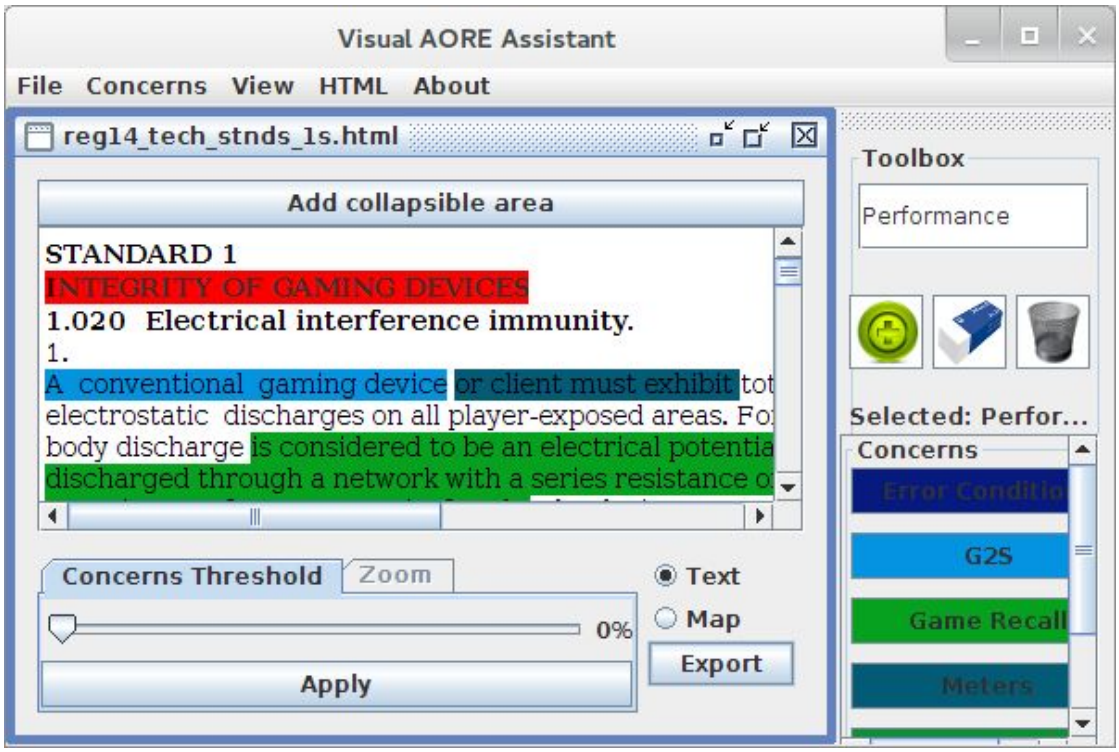


Figura 19: Demarcación de las incumbencias en los documentos de requerimientos

5.2.5. Navegación de Concerns

La herramienta presenta diferentes vistas de los documentos, una textual y una representación gráfica conteniendo colores de acuerdo con el tamaño y la posición de las diferentes incumbencias en el documento.

La vista textual se llama *vista de requerimiento*, y es la utilizada durante el proceso de demarcación (Figura. 20). En contraste, la Figura 21 muestra la representación gráfica, la llamamos *vista de mapa*, ya que funciona como un mapa para mostrar dónde cada incumbencia está presente en el documento.

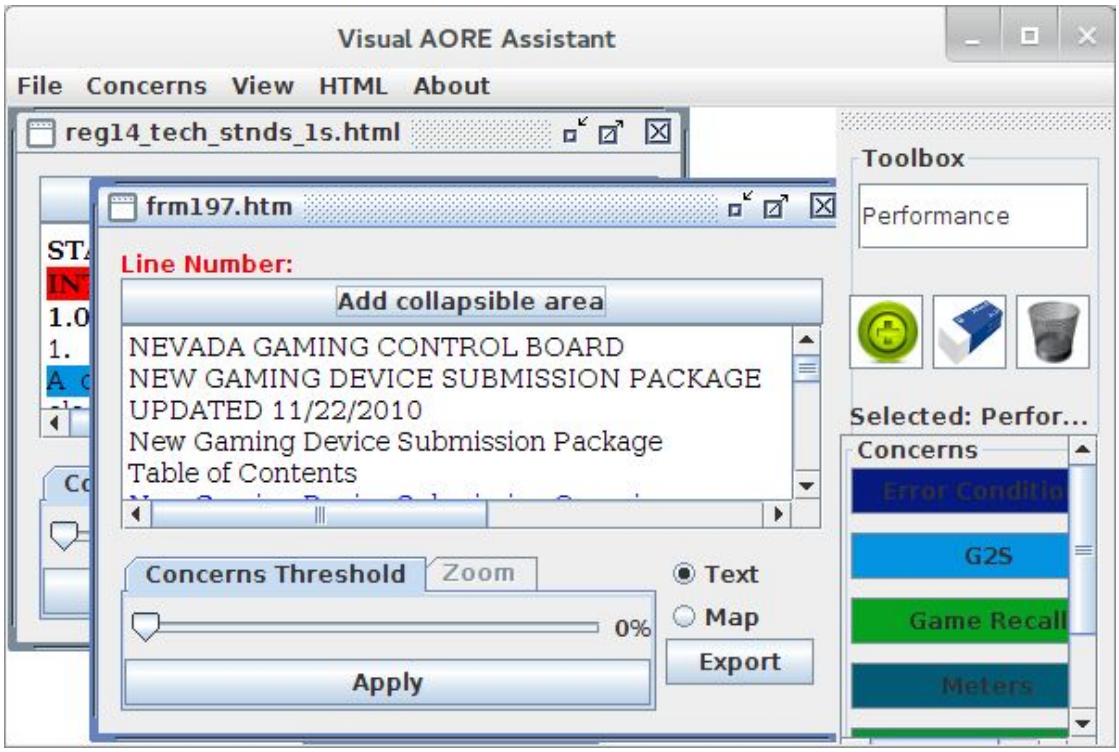


Figura 20: Vista de modo texto de documento de requerimientos

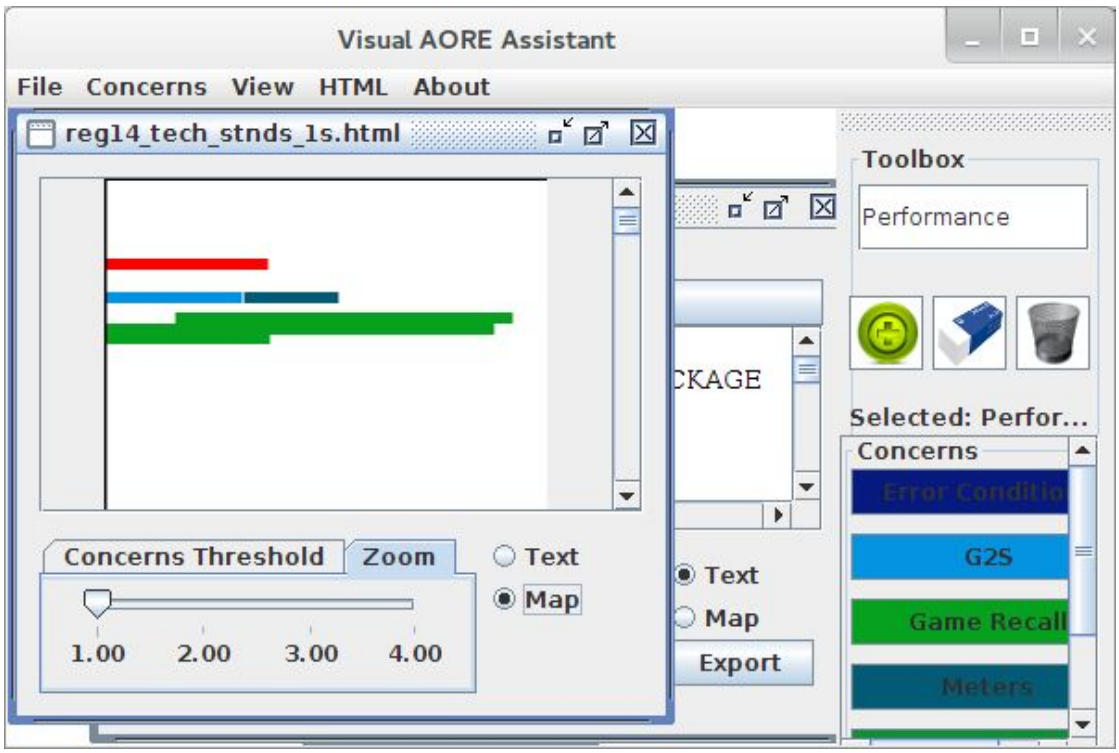


Figura 21: Vista de mapa del documento de requerimiento

En esta vista de mapa puede ser necesario la aplicación del zoom debido a la granularidad del texto en lo que refiere a las incumbencias.

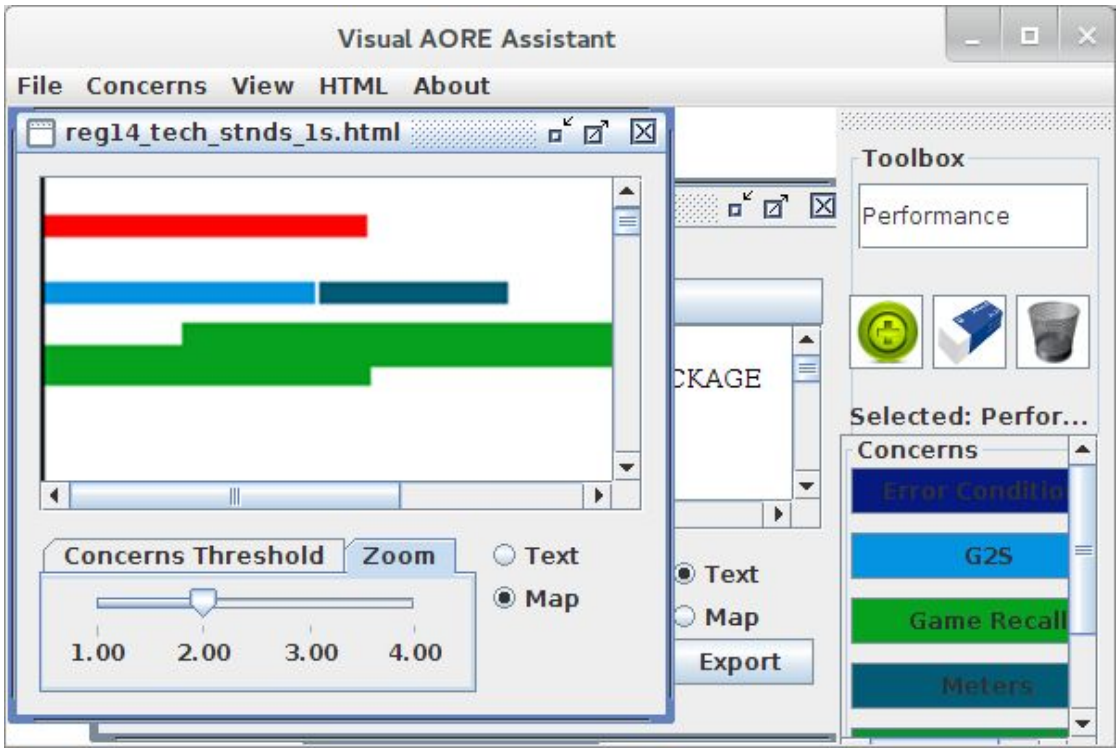


Figura 22: Vista de mapa con zoom aplicado

Se puede navegar desde la vista de mapa al texto correspondiente del documento simplemente haciendo click en la incumbencia (color correspondiente) en el punto deseado, esto hará que la aplicación muestre el texto del requerimiento, como se muestra en la figura 23:

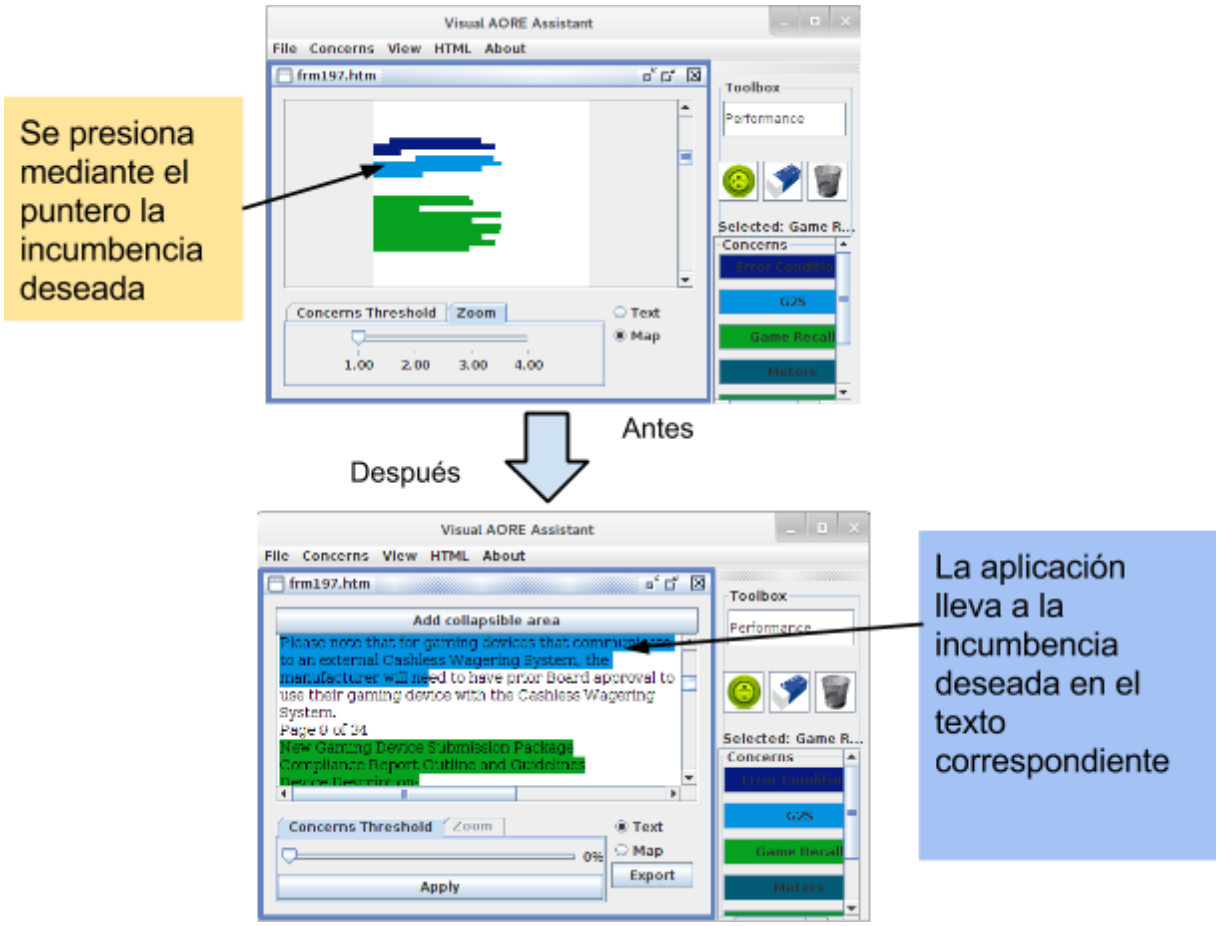


Figura 23: Navegación desde la vista de mapa a la vista textual

El control de threshold puede ser utilizado para mostrar solo las incumbencias que superen un porcentaje seleccionado de acuerdo a la cantidad y dimensión de las ocurrencias en el documento de requerimiento.

Por ejemplo, podría querer ver sólo las incumbencias que su demarcación representa el 9% del documento de requerimiento.

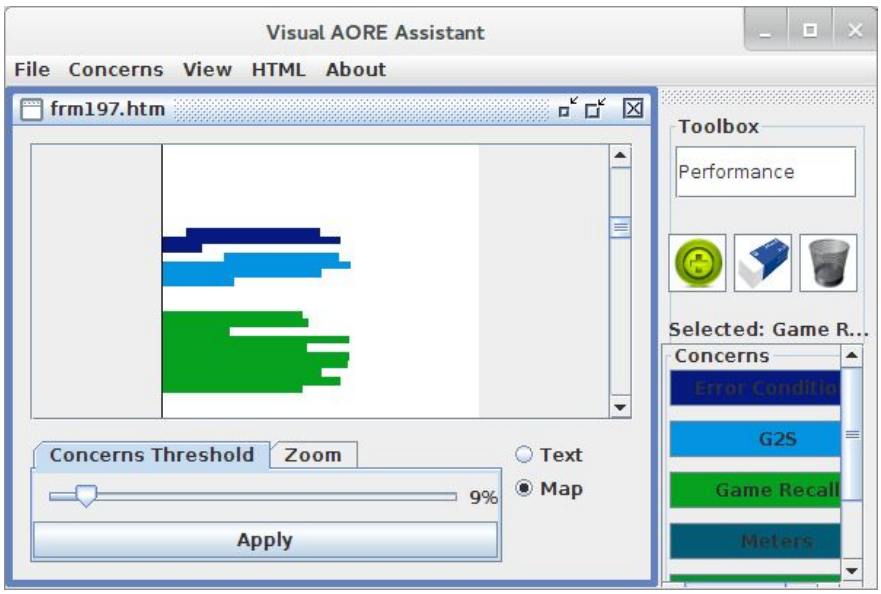


Figura 24: Mostrando el control de Threshold

5.2.6. Versionado de Documentos

Como se indicó en el apartado 3 los documentos de requerimientos van cambiando con el correr del tiempo, aunque las regulaciones en nuestro caso de estudio no suelen sufrir grandes cambios, por menores que sean, el ingeniero de requerimientos tiene que estar en conocimiento de las modificaciones que sufrió el documento sobre el cual está trabajando.

Es fundamental que el ingeniero de requerimientos pueda migrar la información referente a las incumbencias añadidas de la versión actual del documento a una nueva revisión. Para lograr este objetivo es necesario que en la herramienta pueda agregarse una nueva versión del documento y ésta detecte los cambios que fueron realizados en el mismo a fin de que el ingeniero pueda adaptar las incumbencias demarcadas al nuevo documento.

Para agregar una nueva versión del documento el ingeniero de requerimientos debe seleccionar el ítem de menú llamado "Add new version" como se indica en la figura 25

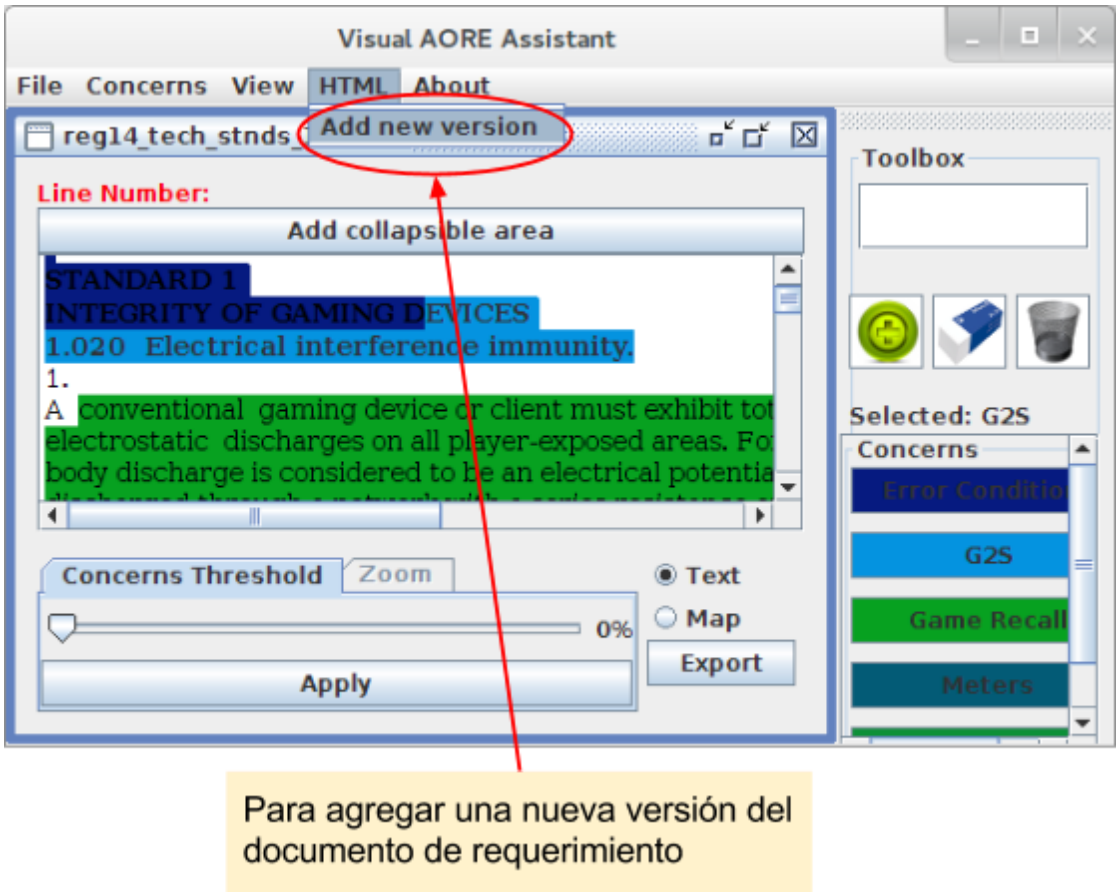


Figura 25: Agregando una nueva versión del documento

Luego seleccionar el documento, debe tener el mismo nombre que el documento actual que el ingeniero está demarcando para que el sistema lo detecte y pueda encontrar las diferencias.

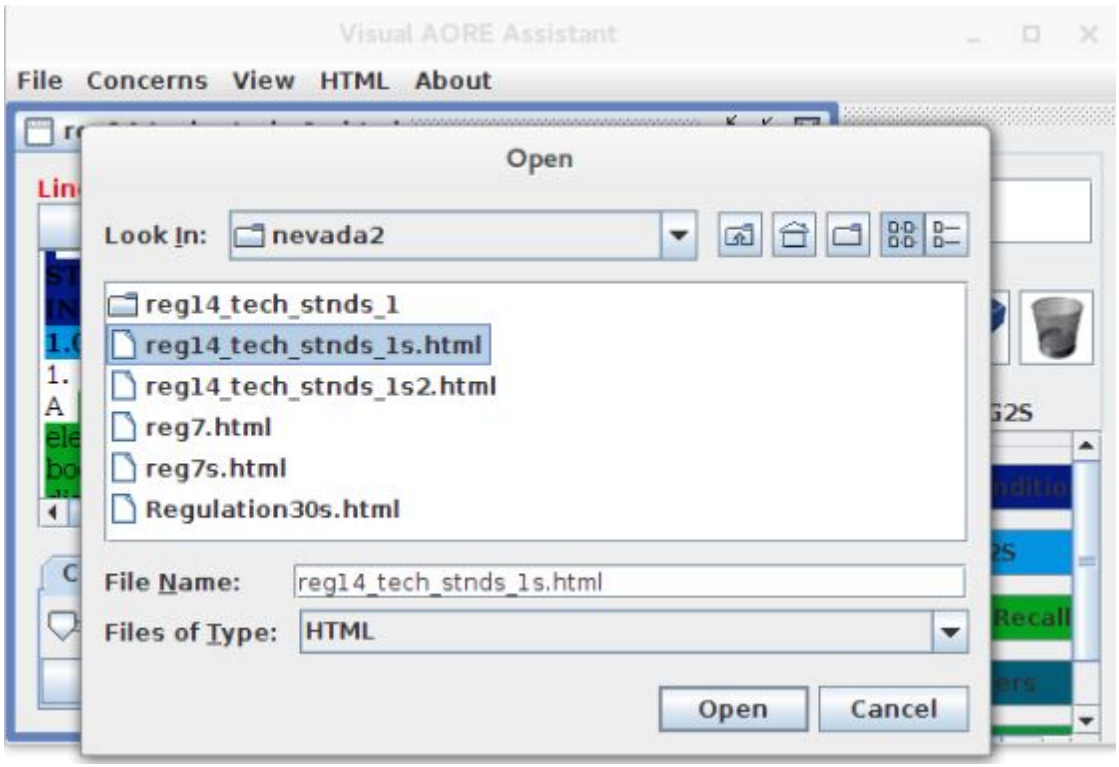


Figura 26: Ingresando la nueva versión del documento

Luego de seleccionar el documento nuevo la herramienta compara el documento actual con la nueva versión buscando las diferencias, esto lo hace en un frame especial compuesto por tres paneles, uno para el documento actual, otro para la nueva versión y por último un tercer panel que muestra las diferencias, pueden observarse en la figura 27:

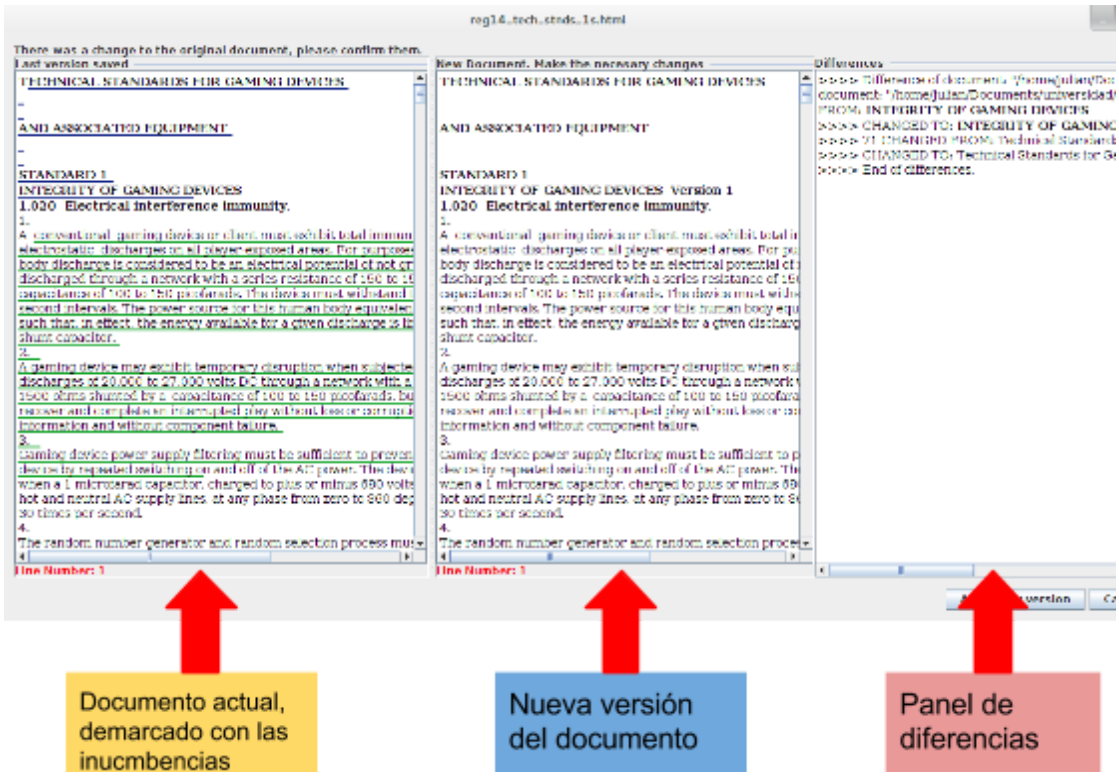


Figura 27: Frame con paneles de versiones

El frame (Figura 27) tiene información importante referida a la versión actual y la nueva del documento de requerimiento, en la parte superior se encuentra el nombre del documento.

En el primer panel, situado a la izquierda del frame se encuentra información del documento actual de requerimiento, este se encuentra demarcado con las incumbencias que el ingeniero de requerimiento marcó en su trabajo, puede observarse en la figura 28:

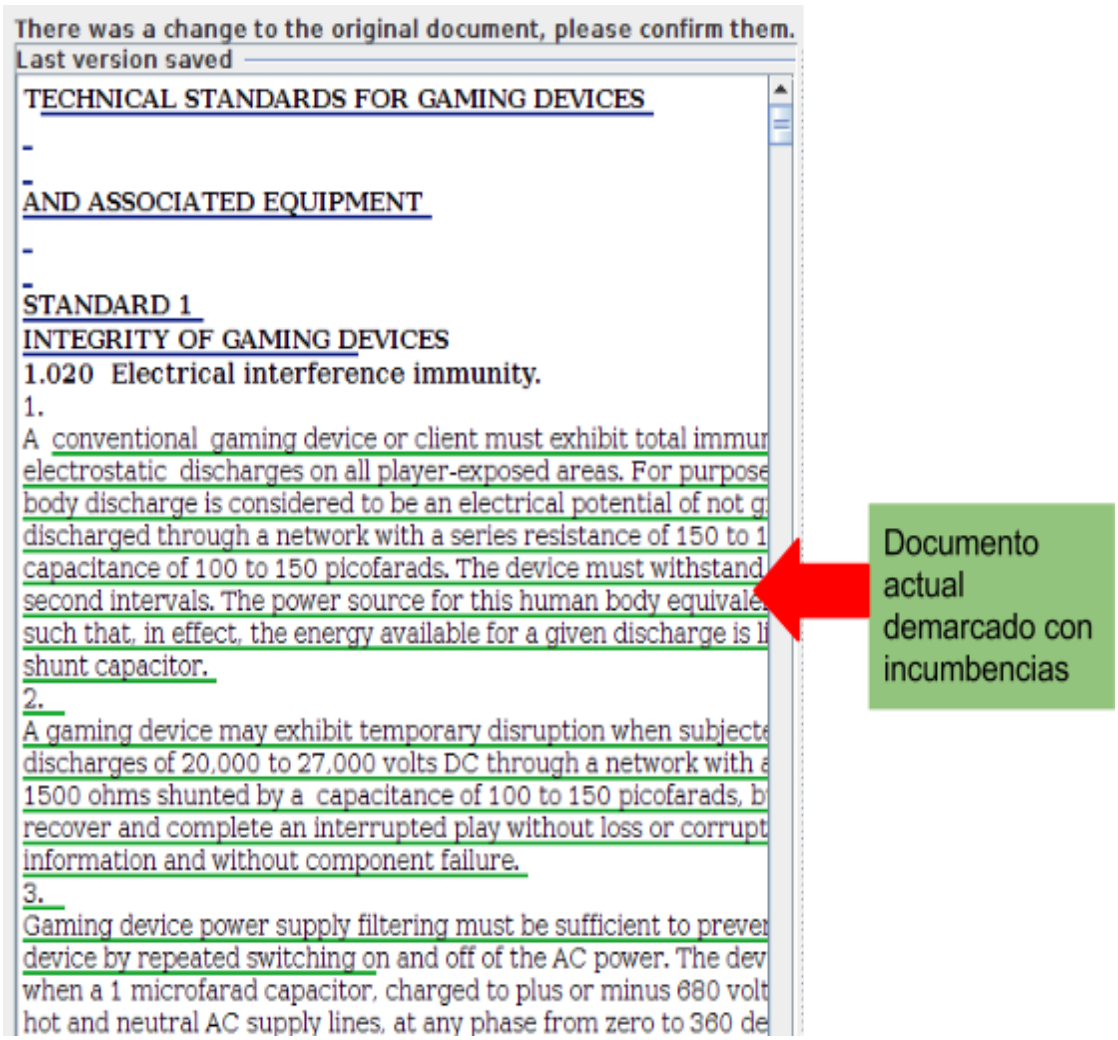


Figura 28: Documento actual demarcado con las incumbencias

En el segundo panel se encuentra el contenido de la nueva versión del documento, el ingeniero puede ver simultáneamente la versión actual y la nueva versión del documento.

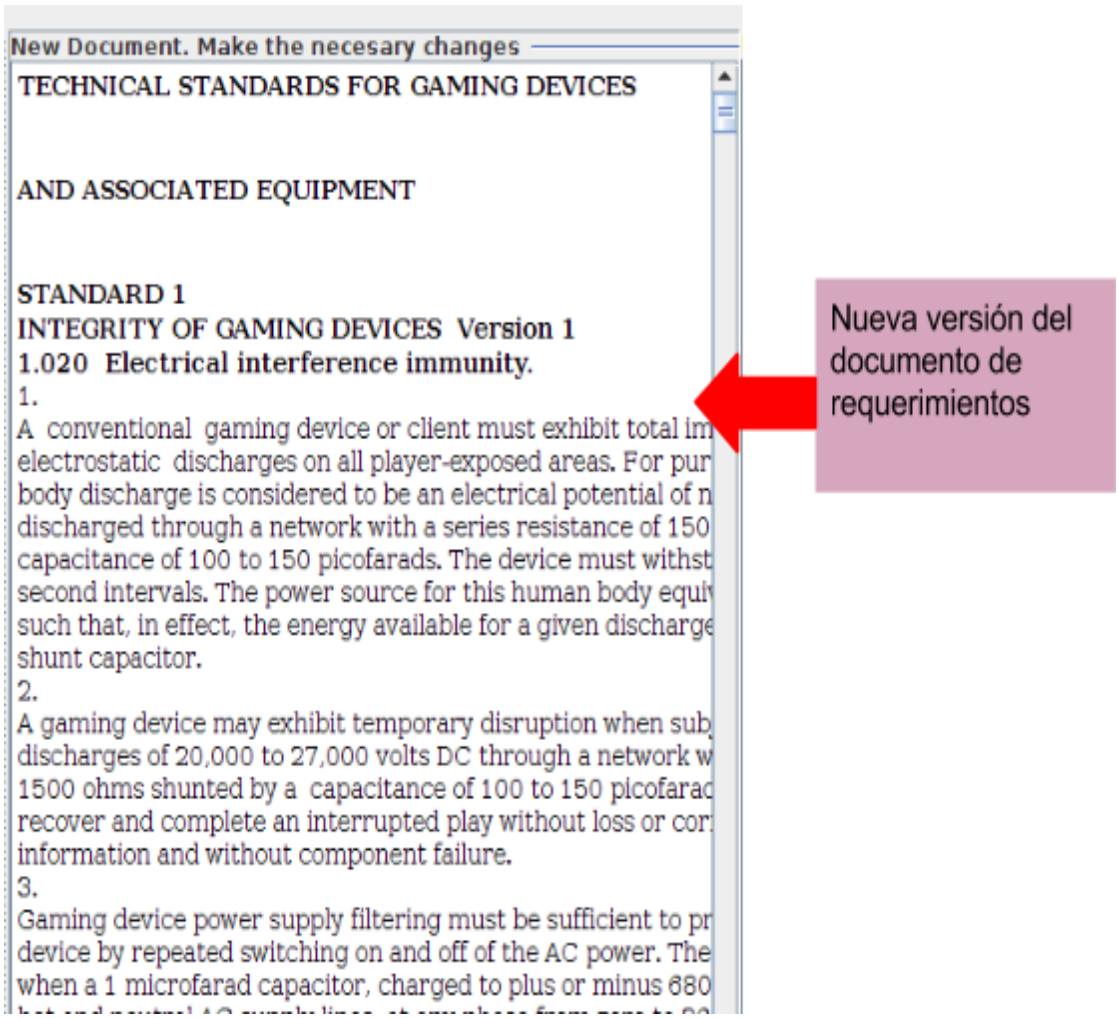


Figura 29: Vista de la nueva versión del documento

En el tercer panel del frame de versiones se encuentran especificadas las diferencias entre la versión actual y la nueva versión del documento de requerimiento, el ingeniero puede observar estas diferencias y tomar la decisión de aplicar la nueva versión.

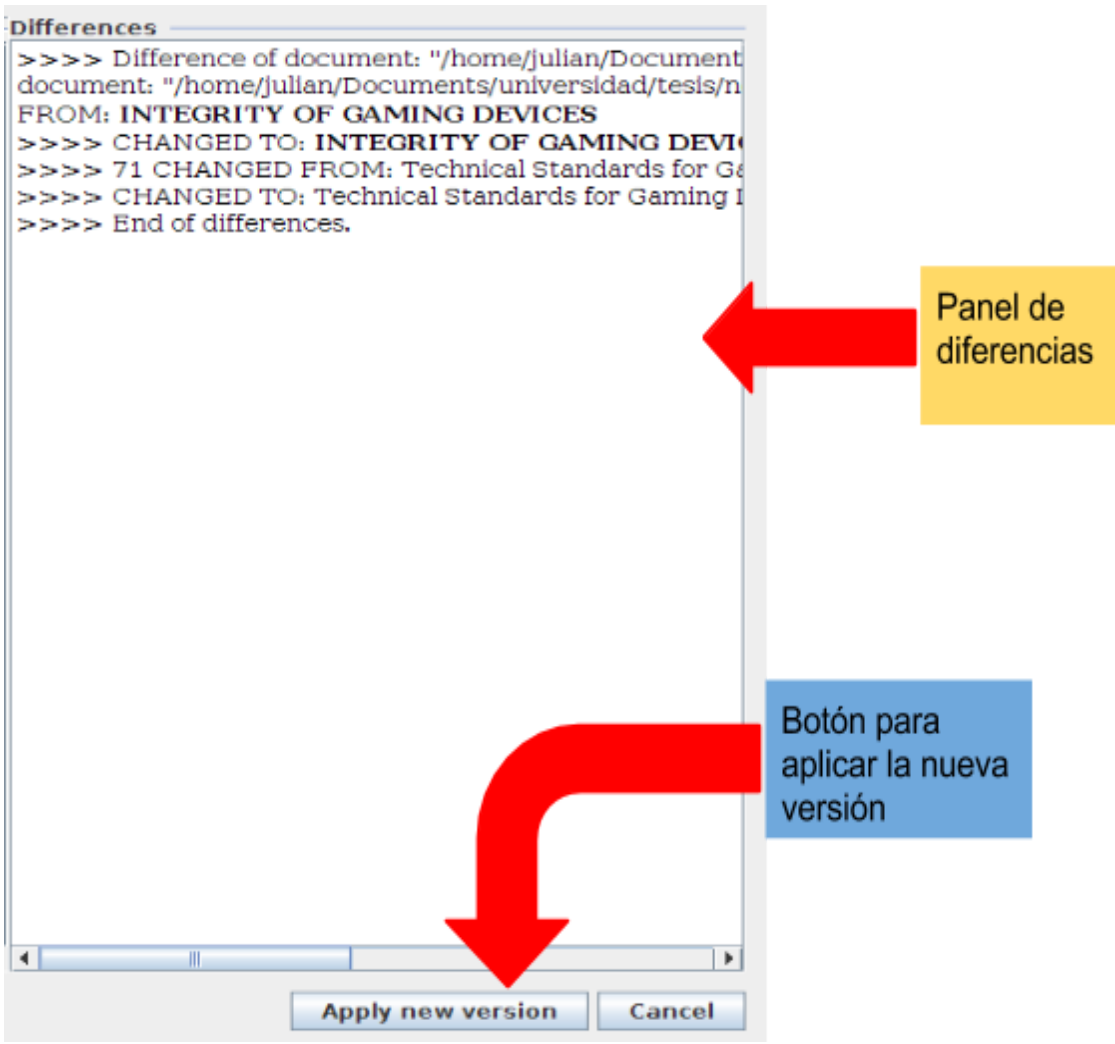


Figura 30: Vista de diferencias

El botón Apply new version copia las demarcaciones de la versión vieja a la versión nueva, luego el ingeniero de requerimientos deberá ajustar las incumbencias al nuevo documento según su criterio.

5.2.7. Vista de Requerimientos

En la *vista de requerimientos* el ingeniero puede leer el texto del documento y detectar qué parte del documento corresponde a una incumbencia, seleccionar el texto y demarcarlo. La herramienta puede guardar las demarcaciones cuando el ingeniero lo requiera para luego poder recuperarla con toda la información cargada. La herramienta soporta también la edición y cambio de color de las incumbencias.

5.2.8. Vista de Mapa

La vista de mapa que se muestra en la figura 32, presenta los documentos desde una perspectiva con gran abstracción. Cada documento está representado por colores de acuerdo a las incumbencias que contenga.
La superficie cubierta es en relación directa con la cantidad de texto dedicado a esta incumbencia en el documento original. Además, la ubicación de la superficie de color se deriva de la posición real de la incumbencia en el documento de requerimientos. Este punto de vista también resume alguna información sobre las incumbencias y lo mucho que ocupan en todos los documentos. El control deslizante en la parte inferior permite filtrar y mostrar sólo las incumbencias cuyo tamaño supera el umbral especificado para cada documento.

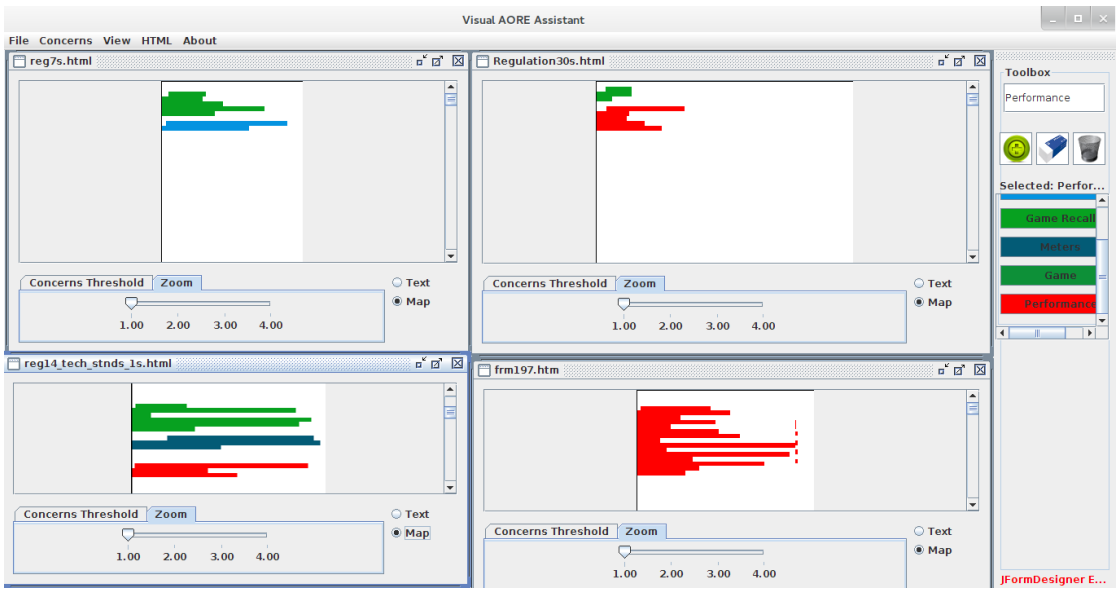


Figura 32: Vista de dominio

5.2.9. Almacenamiento de las Incumbencias

La información correspondiente a las incumbencias que afectan al documento es guardada en un archivo XML para que pueda conservada, como puede observarse en la figura 33:

Figura 33: documento XML asociado al documento de requerimiento original en donde se guardan las incumbencias aplicadas con sus correspondientes coordenadas

Una sentencia que tenga múltiples colores ayuda al ingeniero a detectar posibles interacciones entre las incumbencias, estas [12] usualmente son descuidadas y se convierten en potenciales fuentes de errores.

6. Validación

Para validar el trabajo utilizamos la herramienta en dos dominios bien definidos:

- Sistema de Gestión de Afiliados
- Dominio de las Tragamonedas

Para el sistema de gestión de afiliados contamos con 50 documentos con especificaciones de requerimientos, cada documento estaba compuesto por unas 30 páginas. Los documentos estaban relacionados con distintos órdenes gubernamentales como ser el Ministerio de Trabajo, Empleo y Seguridad Social de la Nación, la Superintendencia de Servicios de Salud, Superintendencia de Riesgos de Trabajo y el estatuto constitutivo de la organización con todas las secretarías (Gremiales, acción Social, Capacitación Cultura y Difusión, Tesorería, Administración y Finanzas) que interactúan con el sistema.

Dos ingenieros de requerimientos utilizaron **AORE Assistant** para demarcar las incumbencias de las especificaciones de requerimientos de 10 documentos. Lo que más destacaron los ingenieros de requerimientos fue la utilidad que tuvo para ellos la vista de mapa de la herramienta dado que les permitía rápidamente observar las incumbencias de manera gráfica a lo largo de los documentos haciendo que puedan comprender velozmente el impacto y a transversalidad de las mismas.

Luego que los documentos fueron demarcados en la etapa de análisis, el equipo encargado del diseño del sistema manifestó que lograron comprender el impacto y la distribución de las incumbencias muy rápidamente gracias a lo demarcado mediante **AORE Assistant**, pudieron comprender en la mitad del tiempo de lectura de los requerimientos la importancia y diversidad de las incumbencias.

Por otro lado, en el dominio de las tragamonedas nos encontramos con gran cantidad de documentos de especificación de requerimientos, alrededor de 600 con un promedio de páginas que rondaba entre las 90 y las 1500, es decir, un dominio de dimensiones industriales.

Al igual que en el sistema de gestión de afiliados, en el dominio de las tragamonedas los ingenieros de requerimientos utilizaron **AORE Assistant**

para demarcar las incumbencias en las especificaciones de requerimientos, tomadas del total de documentos (regulaciones, estándares, especificaciones de protocolos y otros). En este proceso los ingenieros manifestaron que fue de gran utilizad la vista de mapa que les proveyó **AORE Assistant** de las incumbencias y la trazabilidad entre estas y el texto de los requerimientos.

En lo que duró el proceso de análisis de los extensos documentos muchos de ellos fueron modificados teniendo que adaptar la demarcación de las incumbencias al nuevo documento, para esta circunstancia les pareció muy útil el versionado de los documentos en donde se contempla su modificabilidad permitiendo adaptar las demarcaciones (incumbencias).

El equipo de diseño que recibió los documentos demarcados con **AORE Assistant** manifestó que las demarcaciones y su navegabilidad les fue de gran utilidad a la hora de comprender el análisis para diseñar los módulos del negocio.

A pesar que todavía la herramienta se encuentra en un estado de beta, pudo proveer un mecanismo útil de demarcación y navegación de los documentos extensos para el ingeniero de requerimientos en lo que hace a las incumbencias transversales, tanto para el sistema de gestión de afiliados como para el de las tragamonedas.

La herramienta será publicada para que puedan sumarse a la validación ingenieros de requerimientos que habitualmente trabajan con documentos de gran extensión con la complejidad de encontrarse en los mismos incumbencias transversales, de esta manera esperamos obtener más información que nos permita arribar a una conclusión válida sobre el impacto de la herramienta en el trabajo cotidiano con los requerimientos.

7. Conclusiones y trabajo futuro

En el trabajo se presentó un dominio en donde existe un gran número de requerimientos de gran extensión que contienen incumbencias transversales en sus requerimientos funcionales y no funcionales. A partir de las necesidades que imponen realizar análisis de requerimientos sobre el mismo, hemos derivado un conjunto de requerimientos para construir una herramienta destinada a asistir al ingeniero de requerimientos cuando este se enfrenta a escenarios con numerosos requerimientos e incumbencias transversales.

A partir de estos requerimientos se desarrolló la herramienta **AORE Assistan que incluye toda la funcionalidad necesaria para operar en escenarios como el descripto. Por ejemplo:** importación y renderizado de documentos de requerimientos, demarcación de las incumbencias, visualización modo texto y gráfica de alto nivel de abstracción de las incumbencias y gran facilidad para navegarlas.

Utilizando esta herramienta se han analizado los requerimientos del dominio de las tragamonedas. Validando de esta manera la utilidad de **AORE ASSISTANT**. Los resultados de estas evaluaciones han sido publicados en el paper titulado Supporting Aspect Oriented Requirements Engineering for Large Documents EJS, 10(1) 38-52(2011).

Estos resultados demuestra la aplicabilidad de la herramienta.

Por ejemplo, a partir del uso de la herramienta ha sido posible documentar exitosamente todos los crosscuting concerns que afectan a cada requerimiento del sistema. De esta manera se facilita la etapa de diseño, ya que se dispone de toda la información necesaria para modelar cada una de las piezas del software en desarrollo. Por lo tanto, se evitan errores de omisión en la aplicación de las incumbencias transversales.

La herramienta también contempla la evolución de los documentos de requerimientos. Hemos validado esta funcionalidad procesando nuevas versiones de los documentos y migrando exitosamente la información de demarcación existente.

Esto es una característica fundamental considerando que la herramienta está pensada para trabajar sobre grandes documentos. Por lo tanto, no es

aceptable que se pierda la información de demarcación de concerns ya generada.

Considerando el grado de madurez de la herramienta listamos algunas las líneas de trabajo futuro:

- Proveer funcionalidad avanzada para la localización y seguimiento de las incumbencias transversales, dar soporte y documentar las interacciones dentro de **AORE Assistant**. Las interacciones entre concerns como reinforcement o mutex proveen información valiosa para el diseño e implementación. Extendiendo **AORE Assistant** es posible dar soporte a tales interacciones, cuya existencia tiene un alto impacto en el ciclo de desarrollo de software [48]

- Análisis de nuevos casos de estudio: Para completar la herramienta es necesario comprobar su funcionamiento en diferentes dominios a fin de generalizar las soluciones propuestas.

- Generación de informes y reportes: Además de proveer las distintas vistas analizamos la posibilidad de generar reportes que permitan tener una visión resumida respecto de los requerimientos y su transversalidad que permitan a los ingenieros encargados del modelado realizar de manera más efectiva su trabajo.

- Nuevas visualizaciones: En concordancia con el punto anterior, es posible utilizar diferentes tipos de visualizaciones que ayuden a tener una perspectiva de más alto nivel respecto de la complejidad del software a desarrollar. Para este tipo de análisis pueden utilizar herramientas de visualización de software.

8. Referencias

[1] Accompa. http://www.accompa.com.

[2] Enterprise architect. http://www.sparxsystems.com/.

[3] Lighthouse. http://www.artifactsoftware.com/products/Requirements-Management.html.

[4] E. Baniassad and S. Clarke. Theme: An approach for aspect-oriented analysis and design. In ICSE ’04: Proceedings of the 26th International Conference on Software Engineering, pages 158–167, Washington, DC, USA, 2004. IEEE Computer Society.

[5] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In AOSD ’07: Proceedings of the 6th international conference on Aspect-oriented software development, pages 36–48, New York, NY, USA, 2007. ACM.

[6] S. Clarke and E. Baniassad. Aspect-Oriented Analysis and Design. The Theme Approach. Object Technology Series. Addison-Wesley, Boston, USA, 2005.

[7] Gaming Laboratories International. Gaming Devices in Casinos, 2007. Available at: http://www.gaminglabs.com/.

[8] Gaming Standard Association. Game to Server (G2S) Protocol Specification, 2008. Available at: http://www.gamingstandards.com/.

[9] A. Moreira, A. Rashid, and J. Araujo. Multi-dimensional separation of concerns in requirements engineering. In Proc. 13th IEEE International Conference on Requirements Engineering, pages 285–296, 29 Aug.–2 Sept. 2005.

[10] Nevada Gaming Commission. Technical Standards For Gaming Devices And On-Line Slot Systems, 2008. Available at: http://gaming.nv.gov/stats regs.htm.

[11] A. Rashid and A. Moreira. Domain models are NOT aspect free. In ACM/IEEE 9th International Conference on Model Driven Engineering

Languages and Systems (MODELS06), volume 4199 of Lecture Notes in Computer Science, pages 155–169. Springer Verlag, October 2006.

[12] F. Sanen, E. Truyen, B. D. Win, W. Joosen, N. Loughran, G. Coulson, A. Rashid, A. Nedos, A. Jackson, and S. Clarke. Study on interaction issues. Technical Report AOSD-Europe Deliverable D44, AOSD-Europe-KUL-7, Katholieke Universiteit Leuven, 28 February 2006 2006. Supporting AORE for Large Documents 52

[13] A. Zambrano, J. Fabry, G. Jacobson, and S. Gordillo. Expressing aspectual interactions in requirements engineering: experiences in the slot machine domain. In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC 2010), pages 2161–2168. ACM Press, 2010.

[14] ASPECT ORIENTED PROGRAMMING, A CRITICAL ANALYSIS OF A NEW PROGRAMMING PARADIGM, T.J. Highley, Michael Lack, Perry Myers

[15] Salgado, Pablo Daniel. Monitoreo de la interacción entre objetos usando orientación a aspectos. Tesis de Ingeniería en Informática. Universidad Argentina de la Empresa. Octubre de 2006. www.aise-research.com.ar

[16] Craig Larman, UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda Edición. Prentice Hall, 2002. ISBN: 84-205-3438-2

[17] AspectJ Home Page: http://www.parc.xerox.com/aop/aspectj/

[18] Kai Böllert. Aspect-Oriented Programming. Case Study: System Management Application. Graduation thesis, Fachhochschule Flensburg, 1998
.
[19] K. Boellert. On Weaving Aspect. Position papers in the European Conference on Object-Oriented Programming Workshop, 1999.

[20] C. Videira Lopez, G. Kiczales . Recent Developments in AspectJ . Position paper at the ECOOP'98 workshop on Aspect Oriented Programming, 1998.

[21] J. Suzuki, Y. Yamamoto. Extending UML with Aspect: Aspect Support in the Design Phase. 3 er Aspect-Oriented Programming (AOP) Workshop at ECOOP'99.

[22] K. Mehner, A. Wagner, An Assesment of Aspect Language Design,K University of Paderborn. First International Symposium on Generative and Component-Based Software Engineering (GSCE' 99) Young Researchers Workshop Abstracts

[23] Q. Tran and L. Chung, "NFR-Assistant: Tool Support for Achieving Quality," Application-Specific Systems and Soft. Eng. & Tech.,1999, IEEE Computer Society, pp. 284.

[24] A. Sampaio et al., "EA-Miner: a Tool for Automating Aspect-Oriented Requirements Identification," ASE, 2005. A. Sampaio et. al., "Mining Aspects in Requirements," Early Aspects, Workshop (at AOSD), Chicago, USA, 2005.

[25] A. Rashid et al., "Modularisation and Composition of Aspectual Requirements," AOSD 2003, ACM, pp. 11-20.

[26] Y. Yu et al. "From Goals to Aspects: Discovering Aspects from Requirements Goal Models," Proc. RE 2004, IEEE CS, pp. 38-47

[27] Awais Rashid Aspect-Oriented Requirements Engineering:[2008] An Introduction

[28] Sampaio, R. Chitchyan, A. Rashid, P. Rayson, [2005] "EAMiner: a Tool for Automating Aspect-Oriented Requirements Identification", International Conference on Automated Software Engineering (ASE), ACM, pp. 353-355

[29] E. Baniassad, S. Clarke, [2004] "Theme: An Approach for Aspect-Oriented Analysis and Design", International Conference on Software Engineering (ICSE), IEEE CS, pp. 158-167.

[30] Rashid, A. Moreira, J. Araujo, [2003] "Modularisation and Composition of Aspectual Requirements", International Conference on Aspect-Oriented Software Development (AOSD), ACM, pp. 11-20.

[31] Ana Moreira, Awais Rashid, João Araújo Multi-Dimensional Separation of Concerns in Requirements Engineering

[32] J.L. Herrero, M. Sánchez y F. Sánchez. "Changing UML metamodel in order to represent separation of concerns". ECOOP'00.

[33] S. Clarke. “Composition of Object-Oriented Software Design Models”. Doctor of Philosophy in Computer Applications thesis. January, 2001, Dublin City University.

[34] Joseph D. Gradecki, Nicholas Lesiecki, “Mastering AspectJ. Aspect-Oriented Programming in Java”. Ed. Wiley, 2003

[35] Erik Hilsdale, Jim Hugunin, Wes Isberg, Gregor Kiczales, Mik Kersten, “Aspect Oriented Programming with AspectJ”, 2002.

[36] Xerox Corporation, “The AspectJ tm Programming Guide”, 2002. http://www.eclipse.org/aspectj/doc/released/progguide/index.html

[37] I. Brito, A. Moreira, Advanced Separation of Concerns for Requirements Engineering, VIII Jornadas de Ingeniería del Software e Base de Datos, JISBD 2003, Alicante, Espanha, November 2003.

[38] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, 2000.

[39] FOWLER, Martin. UML distilled a brief guide to the standard object modeling language. 2nd ed. 2000. Boston: Addison Wesley. 185 p. ISBN 020165783X.

[40] LARMAN, Craig Y GARCÍA MOLINA, Jesús, rev.. UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2a ed. 2003. Madrid: Pearson Educación. xxv, 590 p. ISBN: 9788420534381

[41] WIRFS-BROCK, Rebecca ; MCKEAN, Alan. Object Design: Roles, Responsibilities, and Collaborations. 2003. Boston: b c2003. xxiii, 390 p. ISBN: 9780201379433

[42] Design Patterns: Elements of Reusable Object-Oriented Software (ISBN 0-201-63361-2)

[43] What is an Aspect in Aspect-oriented Requirements Engineering?, Hermann Kaindl, Vienna University of Technology, Gusshausstr

[44] Baniassad; E., Clements, P.C., Araujo, J., Moreira, A., Rashid, A., Tekinerdogan, B., 2006. Discovering early aspects, IEEE Software, 23(1): 61–70

[45] Visure Requirements Tool http://www.visuresolutions.com/

[46] Awais Rashd, TOOL SUPPORT FOR ASPECT-ORIENTED REQUIREMENTS, 2006

[47] Lancaster Framing Tool http://www.aosd-europe.net/

[48] Arturo Zambrano, Addressing Aspect Interactions in an Industrial Setting: Experiences, Problems and Solutions

9. Anexos

9.1. AORE Assistant

9.1.1. Descripción de la tecnología utilizada

El sistema AORE Assistant fue desarrollado con tecnología Java Standard Edition, se utilizó swing y AWT para la parte visual de la aplicación utilizando componentes como JPanel, JFrame, JDesktopPane, JFileChooser, JLabel, JMenu, JMenuBar, JMenuItem, JOptionPane, JPanel, JScrollPane, JTextField, JTextPane, JToolBar, ScrollPaneConstants, SwingConstants, TitledBorder, FileNameExtensionFilter y Highlighter entre otros.

La creación de las imágenes a partir del texto de los requerimientos fue implementada por medio de la librería de java Graphics2D.

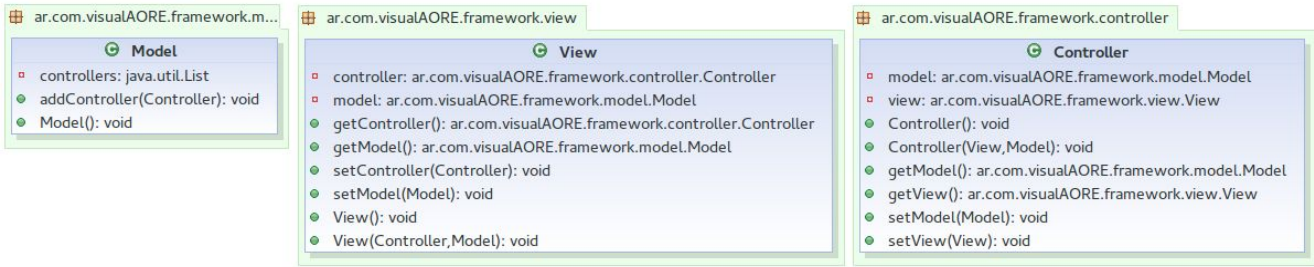
Como entorno de desarrollo se utilizo Eclipse Kepler con el plugin Architexa para el modelado UML de las clases y diagramas de secuencia.

La aplicación completa se encuentra en el CD adjunto, contiene el código fuente y todas las librerías necesarias para poder ejecutarla.

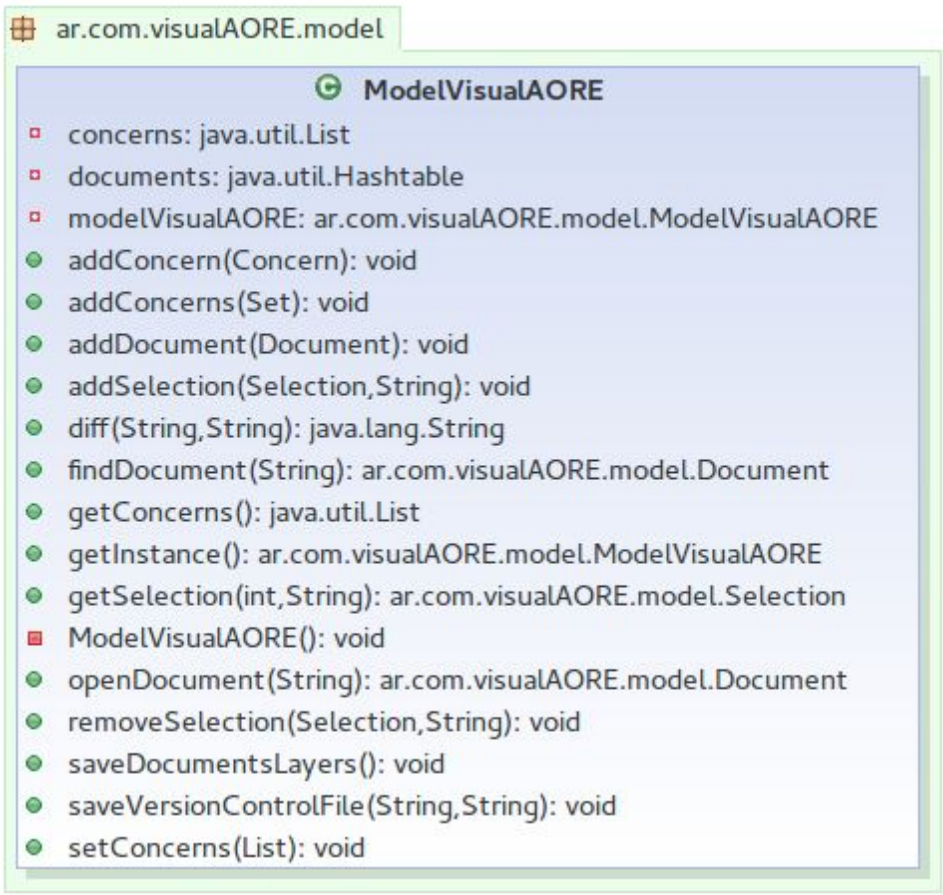
9.1.2. Diagrama de Clases

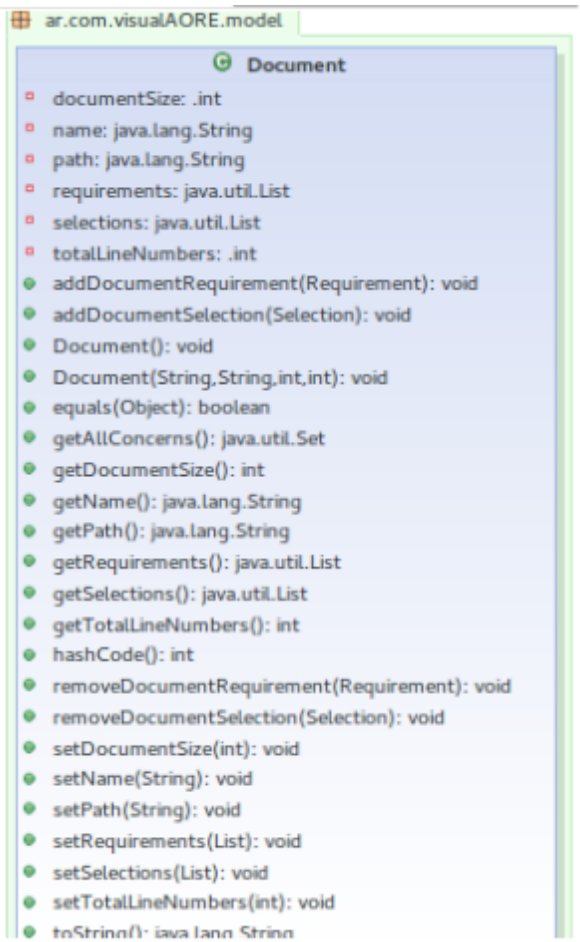
Aclaración: El diagrama de clases completo con sus relaciones se encuentra en el CD adjunto.

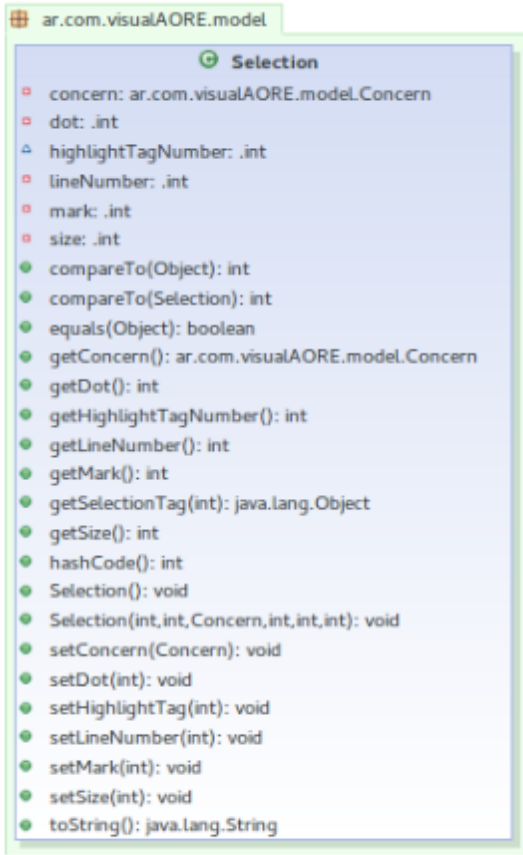
9.1.2.1. Framework del Modelo, Vista, Controlador del sistema



9.1.2.2. Modelo




ar.com.visualAORE.model
Document
documentSize: .int
name: java.lang.String
path: java.lang.String
requirements: java.util.List
selections: java.util.List
totalLineNumbers: .int
addDocumentRequirement(Requirement): void
addDocumentSelection(Selection): void
Document(): void
Document(String,String,int,int): void
equals(Object): boolean
getAllConcerns(): java.util.Set
getDocumentSize(): int
getName(): java.lang.String
getPath(): java.lang.String
getRequirements(): java.util.List
getSelections(): java.util.List
getTotalLineNumbers(): int
hashCode(): int
removeDocumentRequirement(Requirement): void
removeDocumentSelection(Selection): void
setDocumentSize(int): void
setName(String): void
setPath(String): void
setRequirements(List): void
setSelections(List): void
setTotalLineNumbers(int): void
toString(): java.lang.String

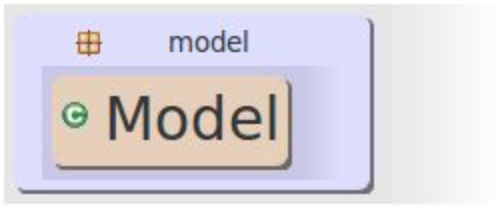

ar.com.visualAORE.model
Selection
concern: ar.com.visualAORE.model.Concern
dot: .int
highlightTagNumber: .int
lineNumber: .int
mark: .int
size: .int
compareTo(Object): int
compareTo(Selection): int
equals(Object): boolean
getConcern(): ar.com.visualAORE.model.Concern
getDot(): int
getHighlightTagNumber(): int
getLineNumber(): int
getMark(): int
getSelectionTag(int): java.lang.Object
getSize(): int
hashCode(): int
Selection(): void
Selection(int,int,Concern,int,int,int): void
setConcern(Concern): void
setDot(int): void
setHighlightTag(int): void
setLineNumber(int): void
setMark(int): void
setSize(int): void
toString(): java.lang.String


ar.com.visualAORE.model
Concern
color: java.awt.Color
cwStyle: javax.swing.text.Style
name: java.lang.String
Concern(): void
Concern(String,Style,Color): void
equals(Object): boolean
getColor(): java.awt.Color
getName(): java.lang.String
hashCode(): int
toString(): java.lang.String

9.1.3. Composición del Sistema

9.1.3.1. Paquetes

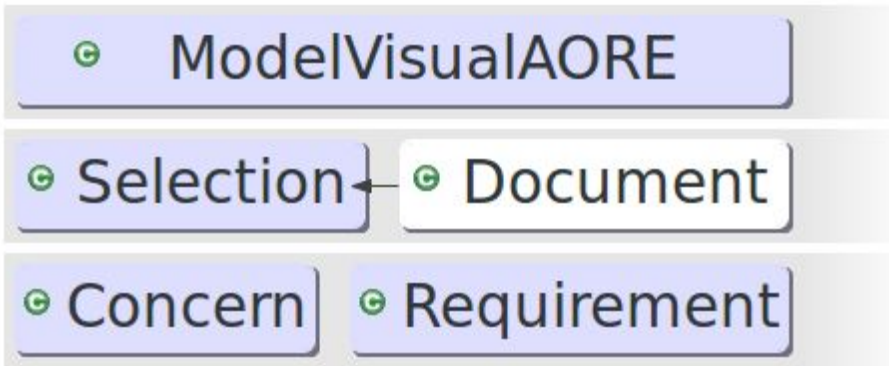
ar . com . visualAORE . framework .



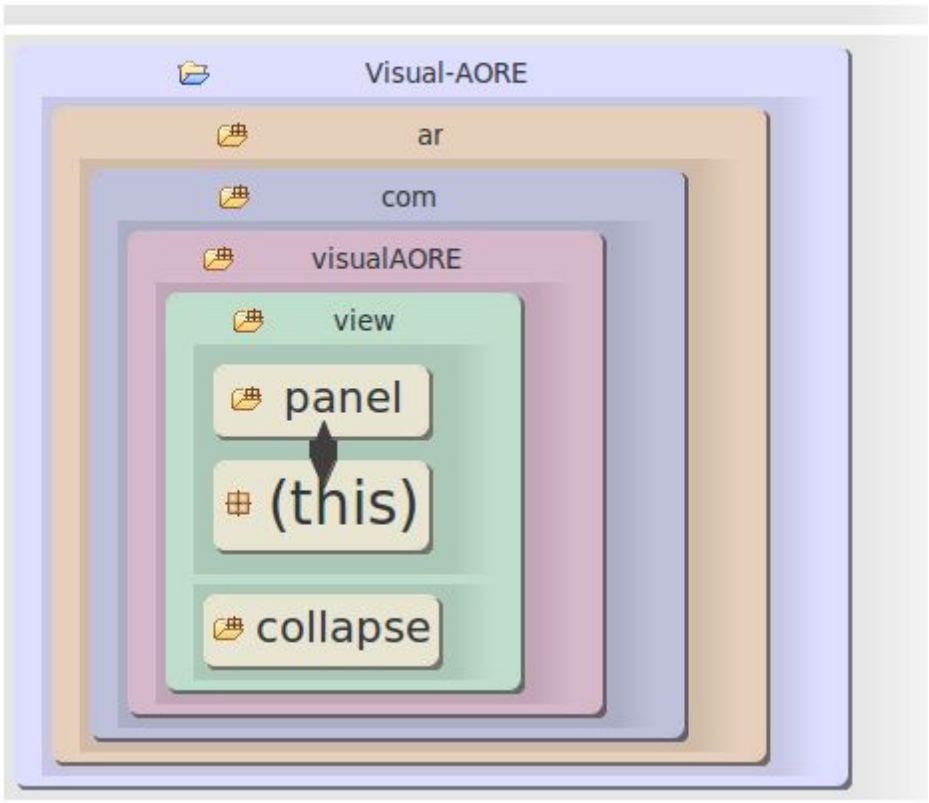
ar . com . visualAORE . framework .

ar . com . visualAORE . framework .

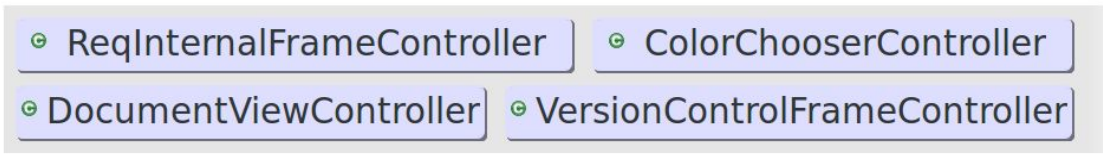
ar . com . visualAORE . model .



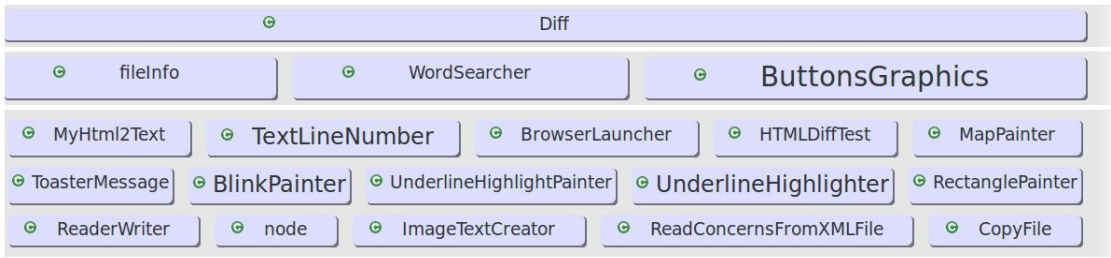
ar . com . visualAORE . view .



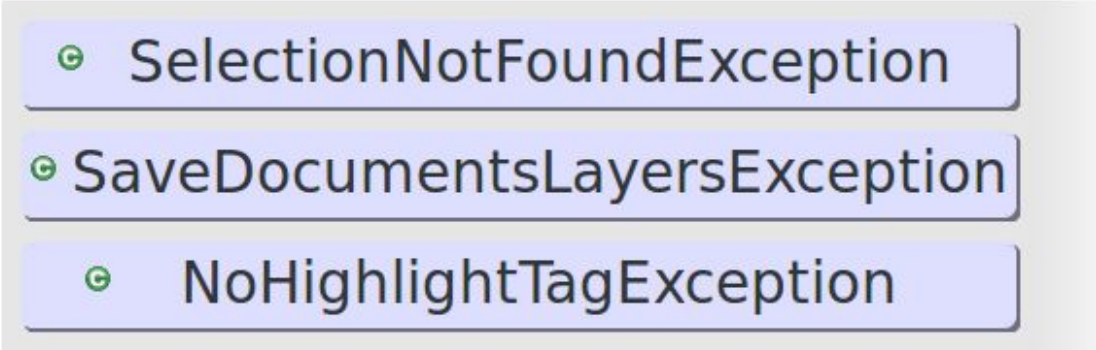
ar . com . visualAORE . controller .



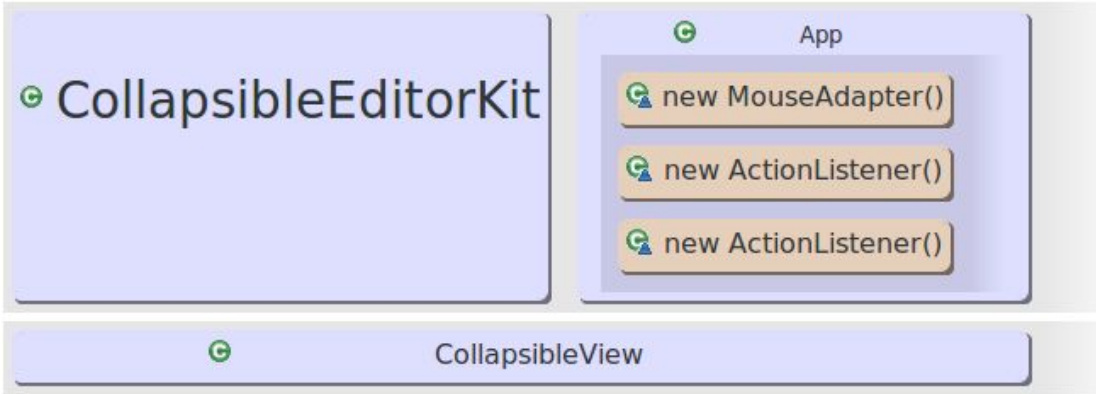
ar . com . visualAORE . util .



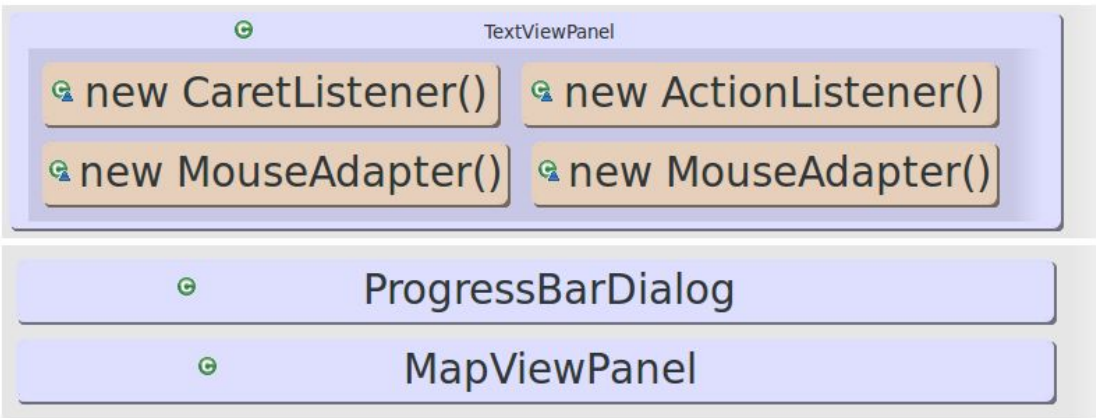
ar . com . visualAORE . exceptions .



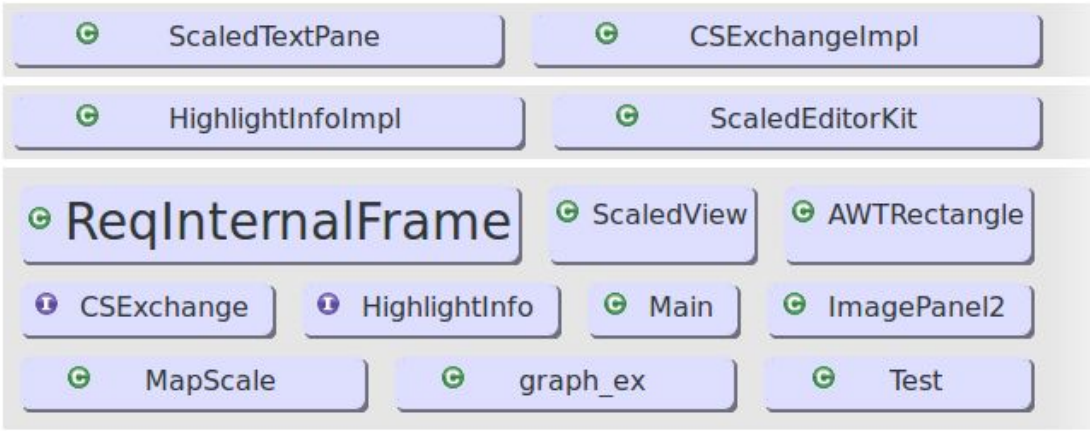
ar . com . visualAORE . view . collapse .



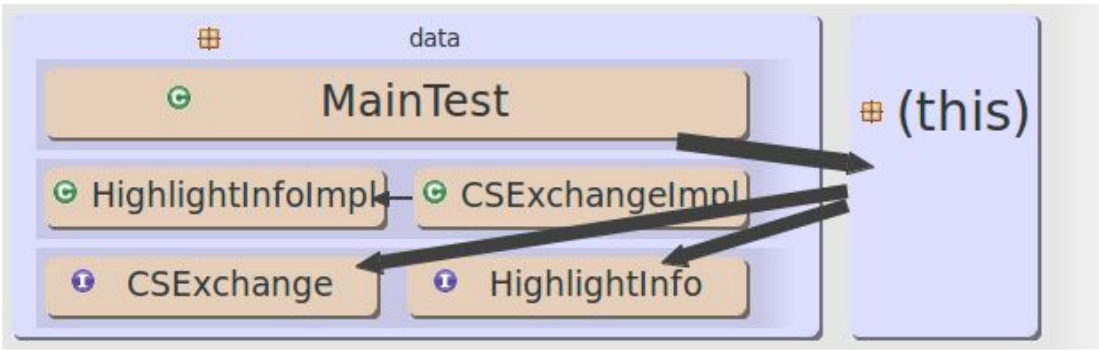
ar . com . visualAORE . view . panel .



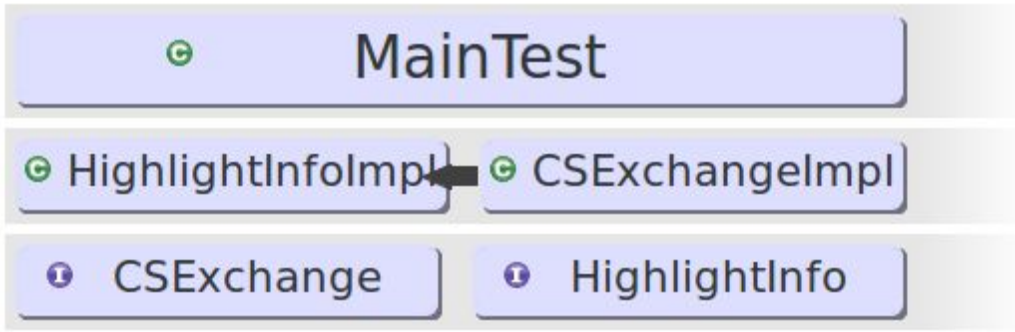
main .



sl . com . sl .

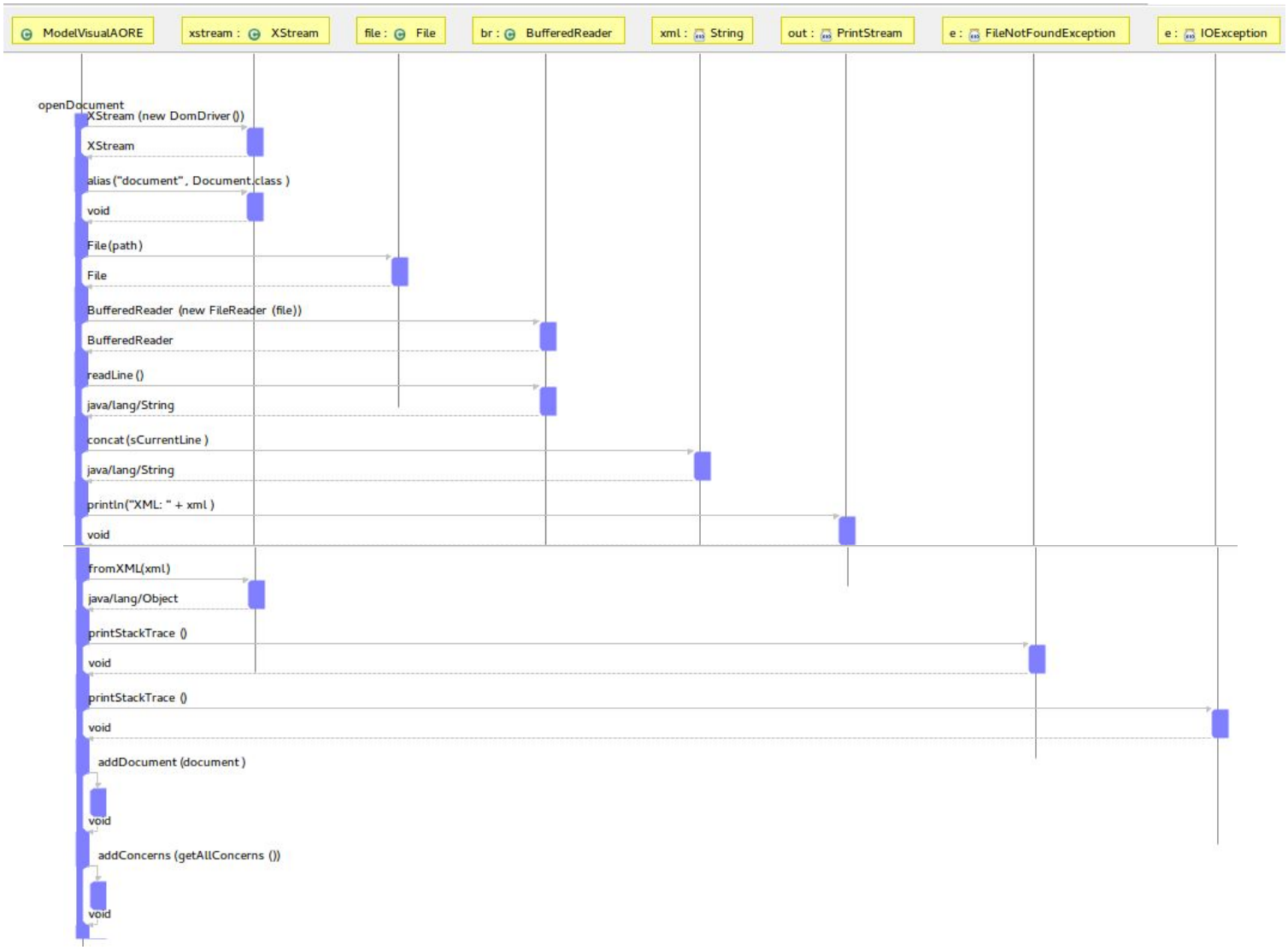


sl . com . sl . data .

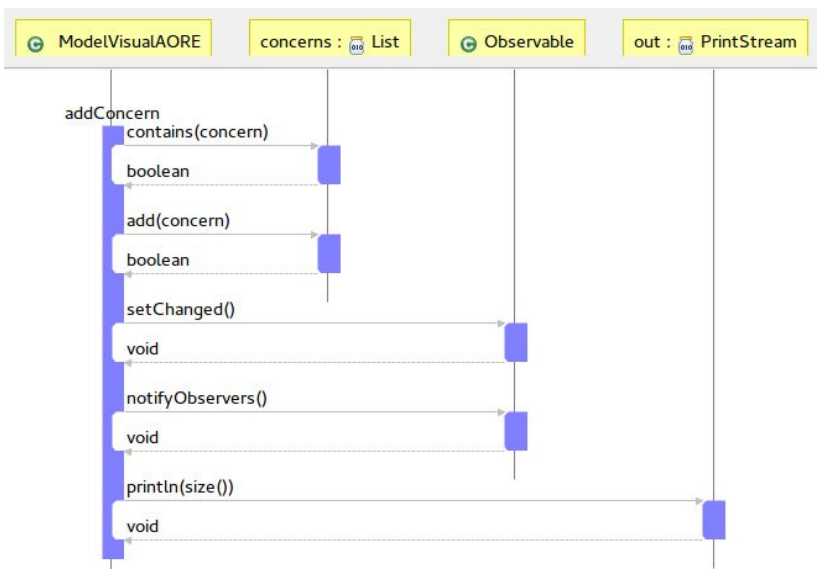


9.1.4. Diagramas de Secuencia

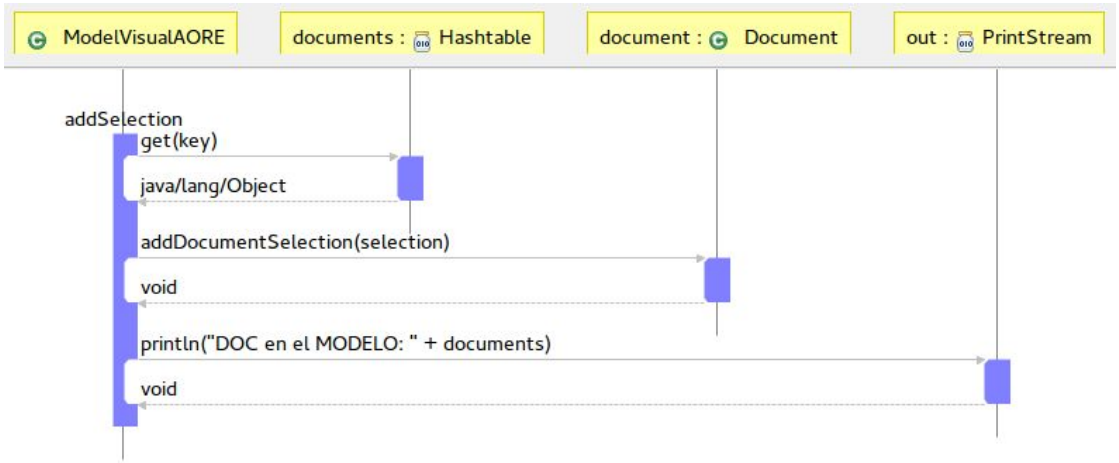
9.1.4.1. Abrir Documento



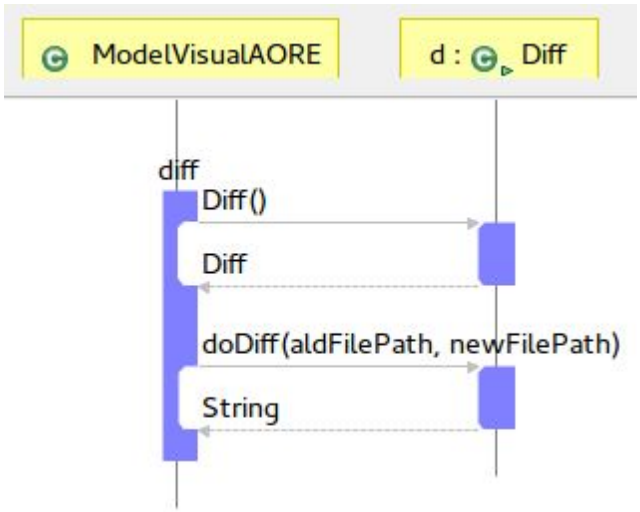
9.1.4.2. Agregar Concern



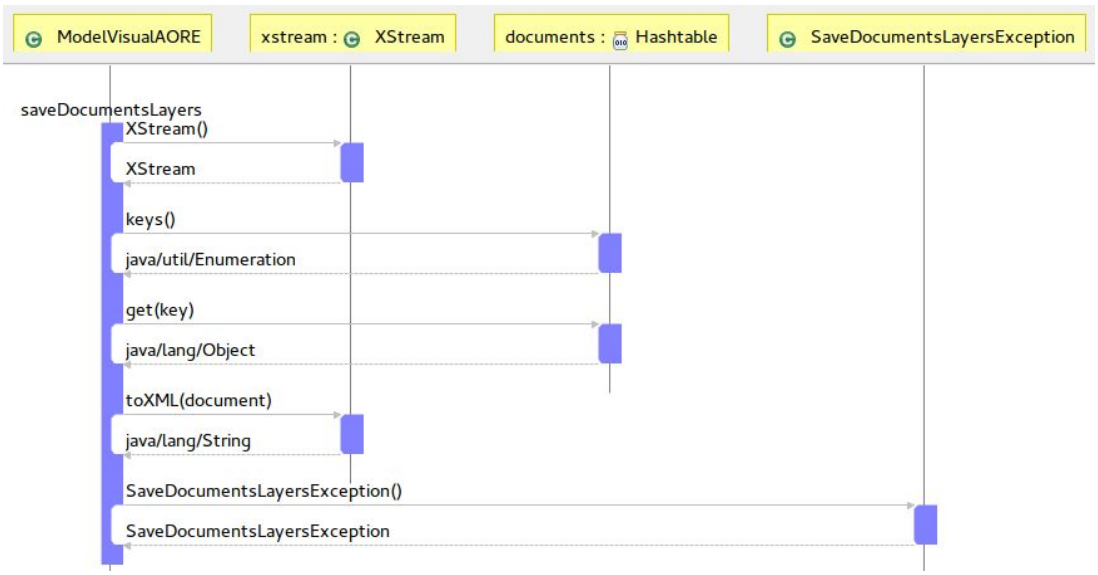
9.1.4.3. Agregar Selección



9.1.4.4. Diff



9.1.4.5. Guardar documento con la información de los concerns



9.1.5. XML

El siguiente xml es un ejemplo del que se guardada asociado al requerimiento original, contiene las coordenadas de las demarcaciones y qué concern es aplicado:

```
<document>
 <requirements/>
 <selections>
  <ar.com.visualAORE.model.Selection>
   <highlightTagNumber>0</highlightTagNumber>
   <dot>4</dot>
   <mark>42</mark>
   <concern>
    <name>G2S</name>
    <color>
     <red>4</red>
     <green>147</green>
     <blue>224</blue>
     <alpha>255</alpha>
    </color>
   </concern>
   <size>38</size>
   <lineNumber>1</lineNumber>
  </ar.com.visualAORE.model.Selection>
  <ar.com.visualAORE.model.Selection>
   <highlightTagNumber>1</highlightTagNumber>
   <dot>6348</dot>
   <mark>6686</mark>
   <concern>
    <name>Game Recall</name>
    <color>
     <red>7</red>
     <green>161</green>
     <blue>32</blue>
     <alpha>255</alpha>
    </color>
   </concern>
   <size>338</size>
   <lineNumber>138</lineNumber>
  </ar.com.visualAORE.model.Selection>
  <ar.com.visualAORE.model.Selection>
   <highlightTagNumber>2</highlightTagNumber>
   <dot>6850</dot>
   <mark>7075</mark>
```

```
      <concern>
        <name>Meters</name>
        <color>
          <red>3</red>
          <green>91</green>
          <blue>118</blue>
          <alpha>255</alpha>
        </color>
      </concern>
      <size>225</size>
      <lineNumber>143</lineNumber>
    </ar.com.visualAORE.model.Selection>
    <ar.com.visualAORE.model.Selection>
      <highlightTagNumber>3</highlightTagNumber>
      <dot>7418</dot>
      <mark>7605</mark>
      <concern>
        <name>Performance</name>
        <color>
          <red>255</red>
          <green>0</green>
          <blue>0</blue>
          <alpha>255</alpha>
        </color>
      </concern>
      <size>187</size>
      <lineNumber>150</lineNumber>
    </ar.com.visualAORE.model.Selection>
  </selections>
  <name>reg14_tech_stnds_1s.html</name>

<path>/home/julian/Documents/universidad/tesis/nevada/reg14_tech_stnds_1s.html<
/path>
  <documentSize>84241</documentSize>
  <totalLineNumbers>1</totalLineNumbers>
</document>
```