Identificación de Atributos de Calidad en Requerimientos

Tesis entregada para el grado de
Ingeniero de Sistemas
en la Facultad de Ciencias Exactas

Por
Francisco Andrés Bertoni
Sebastián Villanueva

Bajo la supervisión de la
Dra. Claudia Marcos
Dr. Andrés Díaz Pace

Universidad Nacional del Centro
de la Provincia de Buenos Aires
Tandil, Argentina
Noviembre 2010

Índice de contenidos

Capítulo I - Introducción 13

I.1. Problemática 13

I.2. Atributos de calidad y aspectos tempranos 14

I.3. Enfoque propuesto 15

I.4. Organización del informe 17

Capítulo II - Conceptos relacionados 19

II.1. Arquitectura de Software 19

II.1.1. Definición de Arquitectura de Software 19

II.1.2. Ejemplo de una arquitectura 21

II.2. Atributos de calidad 22

II.2.1. Principales atributos de calidad 23

II.2.1.1. Disponibilidad 23

II.2.1.2. Performance 24

II.2.1.3. Seguridad 24

II.2.1.4. Usabilidad 24

II.2.1.5. Modificabilidad 25

II.2.1.6. Testeabilidad 25

II.2.2. Atributos de calidad en una arquitectura 26

II.3. Aspectos tempranos y atributos de calidad 26

II.4. Especificación de requerimientos 27

II.4.1. Casos de uso 28

II.4.2. Escenarios de calidad 29

Capítulo III - Trabajos Relacionados 33

III.1. Enfoques y técnicas existentes 33

III.1.1. Métodos de elicitación 33

III.1.1.1. Elicitación de requerimientos con casos de uso 34

Metamodelo 34

Quality Model 36

III.1.1.2. Atributos de calidad “crosscutting” en la ingeniera de requerimientos 38
III.1.2. Métodos de detección semiautomáticos 41
III.1.2.1. Clasificador NFR 41
Etapa de entrenamientos (Training phase) 42
Etapa de clasificación (Classification phase) 43
III.1.2.2. Técnicas de clasificación para requerimientos informales 43
III.1.2.3. Identificación de QARs en especificaciones textuales. Enfoque semi-supervisado de aprendizaje 45
III.2. Conclusiones 47
Capítulo IV - Identificación de atributos de calidad 49
IV.1. Enfoque propuesto 49
IV.2. Tokens Generation 51
IV.2.1. Input Processor 52
IV.2.2. Tokens Filter 54
IV.2.2.1. Filtro Lower Case 55
IV.2.2.2. Filtro Stop Words 56
IV.2.2.3. Filtro Stemming 56
IV.2.2.4. Filtro Ocurrencias 57
IV.2.2.5. Filtro Pesos 58
IV.3. Token Analysis 59
IV.3.1. QAT Identification 60
IV.3.1.1. Maps Identification 61
IV.3.1.1.1. Descripción algorítmica 61
IV.3.1.1.2. Descripción matemática 63
IV.3.1.1.3. Ejemplo 64
IV.3.1.2. Maps Combination 66
IV.3.2. Word-QA Association 67
IV.3.2.1. Ontología 68
IV.3.2.1.1. Componentes de una ontología 69
IV.3.2.1.2. Ontología para QAs 70
IV.3.2.2. Word-Instance Association. 71
IV.3.2.3. Instance-QA Map Association 73

Capítulo V - Evaluación de la técnica propuesta 75
V.1. Métricas a utilizar 75
V.1.1. Falsos y verdaderos, positivos y negativos 76
V.2. Casos de estudio 77
V.2.1. HWS (Health Watcher System) 78
V.2.1.1. Casos de Uso 78
V.2.1.2. Atributos de calidad del sistema HWS 80
Usability 83
Availability 84
Performance 85
Security 85
Scalability/ Distribution 86
Persistency/ Data consistency 87
V.2.1.3. Aspectos tempranos identificados 87
V.2.1.4. Análisis con la técnica propuesta 90
V.2.1.5. Métricas 97
V.2.1.5.1. Métricas para K=0 97
V.2.1.5.2. Métricas para K=0,5 99
V.2.1.5.3. Métricas para K=1 100
V.2.1.5.4. Resumen de las métricas 100
V.2.1.6. Tiempo de ejecución 101
V.2.2. Sistema CRS (Course Registration System) 102
V.2.2.1. Casos de uso 103
V.2.2.2. Atributos de calidad del sistema CRS 105
V.2.2.3. Aspectos tempranos identificados 105
V.2.2.4. Análisis con la técnica propuesta 107
V.2.2.5. Métricas 112
V.2.2.5.1. Métricas para K=0 113
V.2.2.5.2. Métricas para K=0,5 114
V.2.2.5.3. Métricas para K=1 115
V.2.2.5.4. Resumen de los resultados 116

V.2.2.6. Tiempos de ejecución 116
V.3. Conclusiones de los casos de estudio 117
Capítulo VI - Conclusiones 119
VI.1. Ventajas y desventajas 119
VI.2. Trabajos futuros 121
ANEXO I 123
Definición de “mapa” 123
Suma de dos mapas 123
División de un mapa por un número real 123
Multiplicación de un mapa por un número real 123
ANEXO II. Especificaciones de casos de uso completas 125
HWS (Health Watcher System) 125
Sistema CRS (Course Registration System) 138
Bibliografía 148

Índice de Figuras

Figura I.1 - Diagrama simplificado del proceso 15
Figura II.1 - Arquitectura propuesta para el problema de KWIC 21
Figura II.2 - Escenario de calidad 29
Figura II.3 - Conjunto de valores para escenarios generales del atributo disponibilidad 31
Figura II.4 - Escenario concreto del atributo disponibilidad 31
Figura III.1 - Metamodelo propuesto 35
Figura III.2 - Modelo de calidad para "eficiencia" 36
Figura III.3 - Creación de un modelo basado en la experiencia 37
Figura III.4 - Fragmento del template de requerimientos resultante 38
Figura III.5 - Modelo de requerimientos para atributos de calidad 38
Figura III.6 - Template propuesto para atributos de calidad 40
Figura III.7 - La medida de respuesta atraviesa los requerimientos funcionales 41
Figura III.8 - Proceso de clasificación de NFRs (QARs) 42
Figura III.9 - Principales términos indicadores por atributo de calidad 43
Figura III.10 - Área de aplicación del sistema propuesto 44
Figura III.11 - Semi-suppervised approach 46
Figura IV.1 - Diagrama de actividades principales 50
Figura IV.2 - Etapa 1: Tokens Generation 51
Figura IV.3 - Actividad Input Processor 52
Figura IV.4 - Actividad Tokens Filter 54
Figura IV.5 - Etapa Token Analysis 59
Figura IV.6 - Actividad QAT Identification 60
Figura IV.7 - Sub-actividad Maps Identification 61
Figura IV.8 - Sub-actividad Maps Combination 66
Figura IV.9 - Sub-actividad Word-QA Association 68
Figura IV.10 - Ontología para QAs 70
Figura IV.11 - Una única instancia se corresponde con la palabra "user" 72
Figura IV.12 - Dos instancias, de diferente tipo, se corresponden con la palabra "system" 72
Figura V.1 - Arquitectura propuesta por Zhang y otros (arquitectura 1) 82
Figura V.2 - Arquitectura propuesta por Pinto y otros (arquitectura 2) 82
Figura V.3 - Módulos “GUI” y “Data Formatting” de la arquitectura 1 (Zhang) 83
Figura V.4 - Módulos User GUI y Data Formatting de la arquitectura 2 (Pinto) 84
Figura V.5 - Módulo "Replication" de la arquitectura 1 84
Figura V.6 - Módulo "Concurrency" de la arquitectura 2 85
Figura V.7 - Módulo "Authentication" de la arquitectura 1 85
Figura V.8 - Módulo "Security" presente en la arquitectura 2 86
Figura V.9 - Módulo "Distribution", presente en la arquitectura 1 86
Figura V.10 - Módulo "Consistency", presente en la arquitectura 2 87
Figura V.11 - Diagrama de Casos de Uso y Aspectos Tempranos 89

Figura V.12 - Captura QA MINER. EA: Data Formatting (K=0.0) 92
Figura V.13 - Captura QA MINER. EA: Persistency (K=0,5) 93
Figura V.14 - Captura QA MINER. EA: Consistency (K=1) 94
Figura V.15 - Captura QA MINER. EA: Distribution (K=0,5) 95
Figura V.16 - Captura QA MINER. EA: Security (K=0) 96
Figura V.17 - Captura QA MINER. EA: Error Handling (K=0.0) 97
Figura V.18 - Resultados para K=0 (HWS) 98
Figura V.19 - Resultados para K=0.5 (HWS) 99
Figura V.20 - Resultados para K=1 (HWS) 100
Figura V.21 - Resultados para K=0 (CRS) 113
Figura V.22 - Resultados para K=0.5 (CRS) 114
Figura V.23 - Resultados para K=1 (CRS) 115

Índice de Tablas

Tabla IV.1 - Fracción de use case 53
Tabla IV.2 - Fracción de early aspect 53
Tabla IV.3 - Tokens generados luego del análisis de la entrada 53
Tabla IV.4 - Estado de los tokens luego del filtro LowerCase 55
Tabla IV.5 - Lista de tokens pos filtro Stop Words 56
Tabla IV.6 - Estado de los tokens luego del filtro Stemming 57
Tabla IV.7 - Lista de tokens luego de aplicar el filtro Ocurrencias 58
Tabla IV.8 - Ponderación de las secciones de los casos de uso 58
Tabla IV.9 - Estado final de los tokens, luego del filtro Pesos 59
Tabla IV.10 - Lista de dos tokens 64
Tabla V.1 - Breve descripción del caso de uso "Query information" 78
Tabla V.2 - Breve descripción del caso de uso "Complaint specification" 79
Tabla V.3 - Breve descripción del caso de uso "Login" 79
Tabla V.4 - Breve descripción del caso de uso "Register Tables" 79
Tabla V.5 - Breve descripción del caso de uso "Update complaint" 79
Tabla V.6 - Breve descripción del caso de uso "Register new employee" 79
Tabla V.7 - Breve descripción del caso de uso "Update employee" 80
Tabla V.8 - Breve descripción del caso de uso "Update health unit" 80
Tabla V.9 - Change logged employee 80
Tabla V.10 - Atributos de calidad del HWS 81
Tabla V.11 - Ponderación de las secciones de los casos de uso 90
Tabla V.12 - Resultados para el aspecto "Data Formatting" 91
Tabla V.13 - Resultados para el aspecto "Persistency" 92
Tabla V.14 - Resultados para el aspecto "Consistency" 93
Tabla V.15 - Resultados para el aspecto "Distribution" 94
Tabla V.16 - Resultados para el aspecto "Security" 95
Tabla V.17 - Resultados para el aspecto "Error Hadling" 96
Tabla V.18 Valores de QVP. QFP, QFN y QV para HWS (K=0) 98
Tabla V.19 - Valores de QVP. QFP, QFN y QV para HWS (K=0.5) 99
Tabla V.20 - Valores de QVP. QFP, QFN y QV para HWS (K=1) 100
Tabla V.21 - Valores de Precision y Recall para los distintos valores de K 101
Tabla V.22 - Tiempos de ejecución por aspecto temprano (HWS) 101
Tabla V.23 - Breve descripción del caso de uso "Login" 103
Tabla V.24 - Breve descripción del caso de uso "View Report Card" 103
Tabla V.25 - Breve descripción del caso de uso "Register for Courses" 103
Tabla V.26 - Breve descripción del caso de uso "Select Courses to Teach" 103
Tabla V.27 - Breve descripción del caso de uso "Submit Grades" 104

Tabla V.28 - Breve descripción del caso de uso "Maintain Professor Information" 104
Tabla V.29 Breve descripción del caso de uso "Maintain Student Information" 104
Tabla V.30 - Breve descripción del caso de uso "Close Registration" 104
Tabla V.31 - Atributos de calidad del sistema CRS 105
Tabla V.32 - Crosscut de Aspectos en CRS 107
Tabla V.33 - Resultados para el aspecto "Persistency" 108
Tabla V.34 - Resultados para el aspecto "Entitlement" 108
Tabla V.35 - Resultados para el aspecto "External Interface" 109
Tabla V.36 - Resultados para el aspecto "Security" 110
Tabla V.37 - Resultados para el aspecto "Usability" 110
Tabla V.38 - Resultados para el aspecto "Data Validation" 111
Tabla V.39 - Resultados para el aspecto "Performance" 112
Tabla V.40 - Valores de QVP, QFP, QFN y QV para CRS (K=0) 113
Tabla V.41 - Valores de QVP, QFP, QFN y QV para CRS (K=0.5) 114
Tabla V.42 - QVP, QFP, QFN y QV para CRS, con K=1 115
Tabla V.43 - Valores de Precision y Recall para los distintos valores de K 116
Tabla V.44 - Tiempos de ejecución para el caso de estudio CRS 116

Capítulo I - Introducción

Durante mucho tiempo los sistemas de software han sido construidos basándose en requerimientos funcionales. Implícitamente, existía la idea de que éstos eran suficientes para construir un sistema de software exitoso. En este contexto, la *Arquitectura de Software* ha emergido como un eslabón importante en el proceso de desarrollo de software. La arquitectura de software de un programa o sistema computarizado es la estructura o estructuras del sistema, que involucra elementos de software y las propiedades externamente visibles de esos elementos [1].

En el desarrollo de los sistemas actuales, la arquitectura de software constituye el primer paso hacia el diseño de un sistema que tiene un conjunto de propiedades deseadas, tales como: Performance, Disponibilidad, Portabilidad, Seguridad, Usabilidad, etc. Estas propiedades son requerimientos adicionales del sistema [2] que hacen referencia a características o restricciones que éste debe satisfacer, y complementan los requerimientos funcionales del mismo. Estas características o atributos se conocen con el nombre de "atributos de calidad" (*Quality Attributes, QAs*) [3].

Existen evidencias de la relación entre QAs correctamente identificados y el éxito o fracaso de un sistema [4]. Por ejemplo, las consecuencias de una especificación equivocada de performance pueden incluir: relaciones dañadas con el cliente, fallas en el negocio, pérdida de ingresos, aumento del costo del proyecto debido a los costos de sumar recursos adicionales para resolver la falla, entre otras. En este contexto, la correcta identificación y comprensión de los QAs de un sistema es comúnmente señalada como un factor clave de éxito en la construcción de software de calidad [5].

I.1. Problemática

Dado el rol que los QAs juegan en el diseño de la arquitectura, la elicitación de QAs es importante para tomarlos en consideración desde las primeras etapas del ciclo de vida de un sistema.

Por lo general, los QAs de un sistema provienen de distintas fuentes, como por ejemplo los objetivos de negocio, las entrevistas con los stakeholders y/o las especificaciones de requerimientos. En este último caso, cuando se elicitan los requerimientos funcionales (por ejemplo, en casos de uso), muchos stakeholders introducen aspectos de calidad relacionados con funcionalidades especificas del sistema. Algunos investigadores han señalado que, en determinadas ocasiones, ciertos QAs están "ocultos" entre los requerimientos que especifican la funcionalidad y, por ello, podrían ser ignorados por los analistas [6].

A pesar de que podría parecer una tarea sencilla, en la práctica la identificación de QAs en casos de uso puede resultar una tarea dificultosa para el analista, insumiendo gran cantidad de tiempo y esfuerzo. Esto se debe a que, en general, los requerimientos funcionales están expresados en forma de texto plano. Esta dificultad se acrecienta en especificaciones de requerimientos de gran tamaño, debido a la dificultad del analista para analizar texto no estructurado.

Por esta razón, es de utilidad el desarrollo de herramientas semi-automáticas que asistan al analista en la identificación de QAs en especificaciones de requerimientos, presentando un conjunto de QAs candidatos extraídos de los casos de uso. De esta manera, se reduciría el tiempo y el esfuerzo invertido en realizar esta tarea de forma manual.

I.2. Atributos de calidad y aspectos tempranos

Los aspectos tempranos (EA) son intereses (*concerns*) que se encuentran mezclados y diseminados en los requerimientos y/o en los artefactos arquitectónicos del sistema. En la actualidad, existen diversas técnicas y herramientas que tienen como objetivo identificar estos aspectos tempranos.

Muchos EAs del sistema se relacionan con atributos de calidad del mismo. Por ejemplo, los aspectos tempranos *GUI (Graphic User Interface)* o *Authentication* que usualmente se detectan en las especificaciones de requerimientos, se pueden relacionar de forma directa con los atributos de calidad *Usabilidad y Seguridad*, respectivamente. Tanto el descubrimiento de los aspectos tempranos como de los atributos de calidad tiene una significativa importancia a la hora de diseñar la arquitectura. En particular, el descubrimiento de los primeros puede proporcionar información importante para identificar atributos de calidad. Es decir, dado un aspecto temprano puede ser posible realizar un razonamiento acerca de los potenciales atributos de calidad al que éste hace referencia.

En general, el problema de identificación y/o clasificación de QAs se ha estudiado a través de varios enfoques. La mayoría de éstos utilizan estrategias de elicitación de requerimientos o aplican herramientas semi-automáticas, generalmente basadas en técnicas de NLP (procesamiento del lenguaje natural) o IR (recuperación de información). Pocas propuestas tienen en cuenta la información aportada por los aspectos tempranos para la identificación de QAs. Por un lado, en los métodos de elicitación, resulta difícil detectar los aspectos tempranos del sistema como para que, a la vez, estos puedan ser utilizados para la creación de templates, checklists o cuestionarios, en esta misma etapa. Por otro lado, las herramientas semi-automáticas existentes se centran en identificar directamente los QAs, sin utilizar información de los aspectos tempranos.

I.3. Enfoque propuesto

El enfoque que se presenta en este trabajo tiene como objetivo identificar un conjunto de atributos de calidad candidatos, presentes en los casos de uso de un sistema, utilizando como soporte la información que aportan los aspectos tempranos detectados previamente. Se considera que sería beneficioso identificar primero los EAs del sistema, ya que podrían aportar información relevante en la identificación de los QAs presentes en la especificación de requerimientos.

Básicamente, a partir de los casos de uso de un sistema y los aspectos detectados sobre los mismos, se realiza un análisis para identificar los QAs candidatos. Para ello, se utiliza una ontología [7], definida especialmente en este trabajo (Figura I.1). La ontología definida representa el dominio de atributos de calidad y escenarios de calidad. La técnica propuesta utiliza esta ontología para consultar el grado de relación de las palabras que conforman la entrada con los diferentes atributos de calidad.

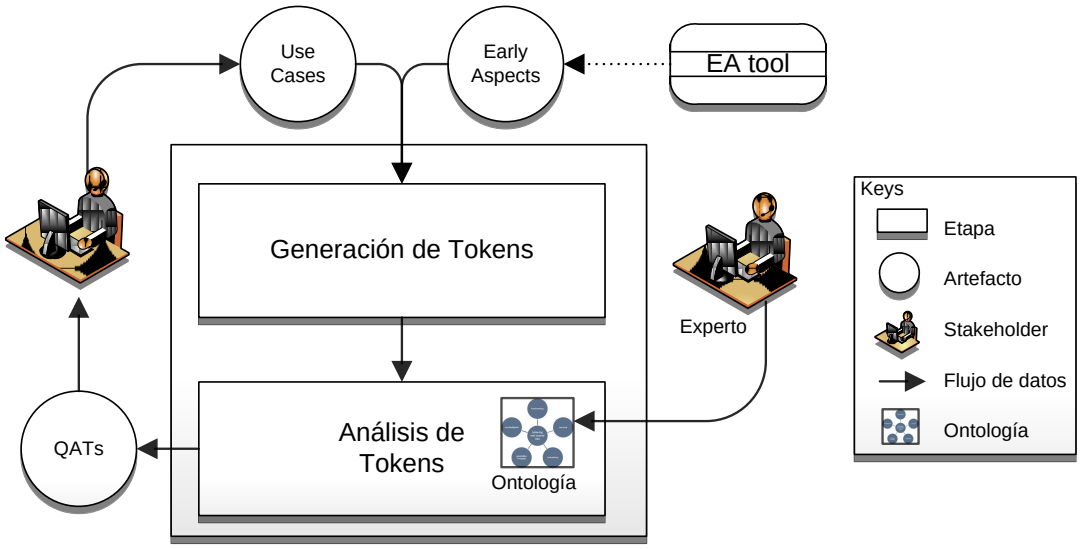

Figura I.1 - Diagrama simplificado del proceso

Los actores principales son: el analista, que es el encargado de definir los casos de uso del sistema y, además, quien se beneficia de la salida del proceso; y el experto en el dominio, cuya responsabilidad es desarrollar la ontología para que pueda ser utilizada durante la etapa de análisis.

Las entradas del proceso son los casos de uso y el aspecto temprano que los relaciona. Se asume que el aspecto temprano ha sido previamente identificado por medio de algún tool de detección de aspectos tempranos. En este trabajo, se utilizó para este propósito la herramienta *Aspect Extractor Tool* [8], la cual identifica un conjunto de aspectos tempranos candidatos a partir de una especificación de requerimientos.

El proceso, básicamente, consta de 2 etapas. La primera etapa es *Generación de Tokens,* en donde se analiza la información de entrada para representarla en un formato uniforme y clasificarla. La segunda etapa es *Análisis de Tokens* en donde se realiza el análisis de los tokens generados por la etapa anterior para calcular, con ayuda de la ontología, el grado de relación de cada uno de ellos con los diferentes atributos de calidad y, así, generar la salida.

Para representar la salida, se introduce el concepto de Quality Attribute Theme (QAT), que está formado por un subconjunto de los casos de uso y un aspecto temprano, que en forma conjunta harán referencia a uno o varios atributos de calidad candidatos. Como resultado, se busca producir una lista priorizada de QATs que resuma los concerns y atributos de calidad candidatos de una especificación de requerimientos.

Como soporte a este proceso se ha desarrollado una herramienta, denominada *QA Miner*, la cual es un *plugin* de Eclipse. Esta herramienta permite al analista seleccionar como entrada el aspecto temprano que desea analizar, junto a los casos de uso que los relaciona. Mediante *QA Miner*, el analista podrá contar con información acerca de los atributos de calidad identificados para el conjunto de datos de entrada. Es decir, para cada aspecto temprano del sistema, se muestra un ranking de porcentajes, con un atributo de calidad cada uno, indicando el grado de relación entre ese aspecto temprano y los distintos atributos de calidad presentes en la ontología. De esta manera, el analista podrá seleccionar los atributos de calidad destacados, para tenerlos en cuenta en las etapas siguientes del ciclo de desarrollo.

Adicionalmente, se llevó a cabo una evaluación de la técnica propuesta utilizándola en distintas especificaciones de sistemas. Particularmente, la evaluación de *QA Miner* se efectuó sobre dos casos de estudio. El primer caso de estudio se denomina HWS (Health Watcher System, *Sistema de Salud Vigía*) [9]. El mismo está compuesto por nueve casos de uso, con un total aproximado de 2300 palabras. El segundo caso de estudio se denomina CRS (Course Registration System, *Sistema de Registro de Cursos*) [10] y [11] . El mismo está compuesto por ocho casos de uso, con un total aproximado de 3900 palabras. El análisis realizado con Aspect Extractor Tool identificó 6 aspectos candidatos para el primer caso de estudio y 7 para el segundo.

Para efectuar la evaluación, se utilizaron métricas originarias del área de *Recuperación de Información* (IR) [12]. Particularmente, se tomaron las métricas de *Precisión* y *Recall*. Además, para el caso de estudio HWS, se analizaron documentos de arquitecturas propuestas, contando así con información precisa de los QAs del sistema.

I.4. Organización del informe

El resto del informe se encuentra organizado en 5 capítulos.

En el *Capítulo II* se presentará una explicación de los conceptos claves del Desarrollo de Software, enfatizando en Arquitecturas de Sistemas, Atributos de Calidad y Aspectos Tempranos. Adicionalmente, se introducirán conceptos relacionados con la especificación de requerimientos significativos en este trabajo, como Casos de Uso y Escenarios de Calidad.

En el *Capítulo III* se llevará a cabo el análisis y la comparación de trabajos de investigación actuales en el campo de la identificación de QAs en especificaciones de requerimientos.

En el *Capítulo IV* se presentará en detalle la técnica propuesta para la identificación de atributos de calidad en especificaciones de requerimientos a partir de un conjunto de aspectos tempranos del mismo. Además, se describirá la implementación de la herramienta *QA Miner*.

En el *Capítulo V* se describirán los dos casos de estudio, los cuales serán analizados con ayuda de la herramienta desarrollada, para obtener las métricas correspondientes al enfoque propuesto.

En el *Capítulo VI* se presentarán las conclusiones del trabajo y se discutirán las posibles mejoras de esta propuesta.

Capítulo II - Conceptos relacionados

En este capítulo se presentarán los conceptos que se utilizarán a lo largo del presente trabajo. Se introducirán los conceptos de “Arquitectura de software”, “Atributo de calidad”, “Escenario” y “Early Aspect” (Aspecto Temprano). Además, se introducirán algunas técnicas de especificaciones de requerimientos, centrándose en el concepto de “Caso de Uso” y su relación con los aspectos tempranos y atributos de calidad.

II.1. Arquitectura de Software

Durante mucho tiempo los diseñadores de software han construido sistemas basados mayormente en requerimientos funcionales. De esta manera, el diseñador debía basar su diseño en estos requerimientos, para luego crear el sistema a partir del mismo. A pesar del feedback que podría haber entre el diseñador y el analista, existía implícitamente la idea que los requerimientos funcionales eran suficientes para construir un diseño satisfactorio.

En este contexto, la *Arquitectura de Software* ha emergido como un eslabón crucial en el proceso de diseño, especialmente para el desarrollo exitoso de sistemas de software complejos [13].

La visión arquitectónica de un sistema es abstracta, concentrándose en el comportamiento e interacción de componentes, prestando poca atención a detalles de implementación, algoritmos o representación de datos. La arquitectura de software constituye el primer paso hacia el diseño de un sistema que tiene un conjunto de propiedades deseadas.

II.1.1. Definición de Arquitectura de Software

En la actualidad no hay ninguna definición de arquitectura de software que esté unánimemente respaldada por la totalidad de la comunidad. En general, las definiciones entremezclan el trabajo dinámico de estipulación de la arquitectura dentro del proceso de ingeniería o el diseño (su lugar en el ciclo de vida), la configuración o topología estática de sistemas de software contemplada desde un elevado nivel de abstracción y la caracterización de la disciplina que se ocupa de uno de esos dos asuntos, o de ambos. [14].

Una definición reconocida es la de Clements [15]:

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, el comportamiento de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema.

Sin embargo se debe mencionar que, algunos años después, el mismo Clements, junto con Bass, Kazman y Northrop, definen a la arquitectura de software de la siguiente forma [1]:

La arquitectura de software de un programa o sistema computarizado es la estructura o estructuras del sistema, que involucra elementos de software y las propiedades externamente visibles de esos elementos.

Por lo tanto, a pesar de la gran cantidad de definiciones del campo de la arquitectura de software, existe en general acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, compuesta por elementos de software, junto con sus propiedades y relaciones entre ellos. Estas cuestiones estructurales se vinculan con el diseño, pues la arquitectura de software es una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema. Este diseño ocurre a un nivel más abstracto que el de los algoritmos y las estructuras de datos. En este trabajo se tomará como referencia la última definición mostrada. Por ello vale la pena profundizar en algunas implicaciones de tal definición.

La primera implicancia es que la arquitectura define *elementos de software*. Es decir que la arquitectura incorpora información acerca de cómo esos elementos se relacionan entre ellos. Esto último implica que la arquitectura omite cierta información acerca de elementos que no pertenecen a la interacción. Se puede ver a la arquitectura como una abstracción de un sistema que suprime ciertos detalles que no afectan a cómo los elementos son usados, se relacionan o interactúan con otros elementos.

La segunda implicancia es que la definición deja bien en claro que los sistemas poseen más de una estructura y que ninguna de estas puede adjudicarse ser “la arquitectura” del sistema.

La tercera implicancia es que la definición implica que todo sistema de software tiene una arquitectura, porque todo sistema puede ser descripto como un conjunto de componentes y relaciones entre estos componentes.

Finalmente, la cuarta implicancia es que el comportamiento de cada componente es parte de la arquitectura en tanto y en cuanto este comportamiento pueda ser observado o discernido desde el punto de vista de otro elemento. Ese comportamiento es lo que permite a los elementos interactuar entre ellos, lo que es parte de la arquitectura.

II.1.2. Ejemplo de una arquitectura

A continuación se muestra un ejemplo de una arquitectura simple, utilizando una vista de componentes y conectores. El mismo es un ejemplo de una arquitectura de software para el problema de KWIC (Key Word in Context). En su artículo de 1972, Parnas propuso el siguiente problema [16]:

El KWIC [Kew Word in Context] acepta un conjunto ordenado de líneas, donde cada línea es un conjunto ordenado de palabras, y cada palabra es un conjunto ordenado de caracteres. Toda línea debe ser "desplazada circularmente'' varias veces, quitando la primera palabra y añadiéndola al final de la línea. El sistema de KWIC tiene como salida un listado, ordenado alfabéticamente, de todos los posibles desplazamientos circulares de todas las líneas.

En la Figura II.1 se muestra una arquitectura propuesta como solución para el problema citado anteriormente. La misma se basa en el estilo de "Pipe and Filters" [17]. En este esquema, a los elementos se los denomina filtros (filters), mientras que a los conectores se los denominan tuberías (pipes).

Cada filtro realiza una tarea específica, procesando los datos de entrada y emitiéndolos por su salida. El control es distribuido: cada filtro puede procesar los datos apenas le llegan, independientemente de lo que estén haciendo los otros filtros. Idealmente los filtros no comparten entre sí variables o estados internos. Cada filtro tiene una entrada, por donde le llegan datos, y una salida donde escribe datos producidos. La comunicación entre los filtros es restringida, sólo se realiza a través de las tuberías. Éstas conectan la salida de los filtros con las entradas de otros, transmitiendo los datos de una punta a la otra.

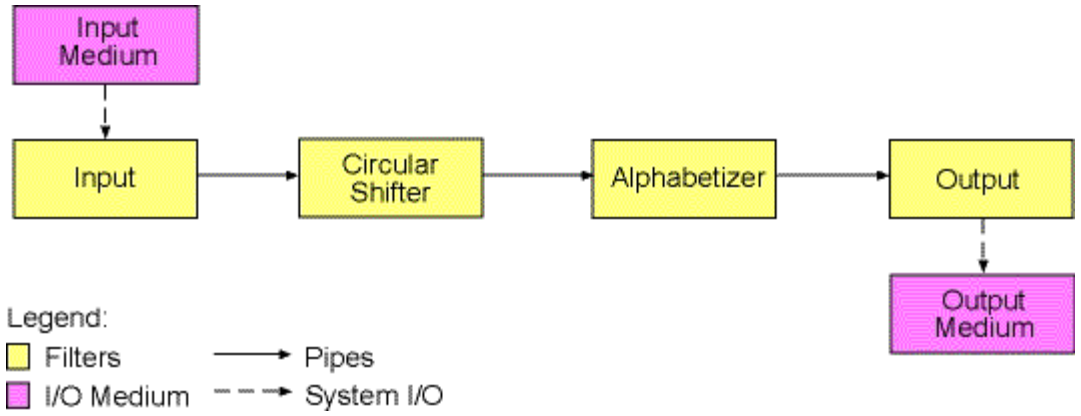

Figura II.1 - Arquitectura propuesta para el problema de KWIC

En la arquitectura propuesta, se muestran cuatros filtros: "Input", "Circular Shifter", "Alphabetizer" y "Output".

El filtro “Input” parsea la entrada y escribe las líneas parseadas en su salida. Una tubería conecta la salida de este filtro con la entrada del filtro “Circular Shifter”. De esta forma, las líneas parseadas por el filtro “Input” sirven como entrada del segundo filtro. Éste procesa las líneas de entrada produciendo desplazamientos circulares sobre esas líneas. Las nuevas líneas desplazadas son escritas en la salida del filtro, en donde nuevamente se encuentra una tubería que conecta la salida del filtro con el filtro “Alphabetizer”. De esta manera, las líneas producidas por el filtro “Circular Shifter” sirven como entrada para el filtro “Alphabetizer”. Éste ordena las líneas alfabéticamente, escribiéndolas en su salida. Análogamente, una tubería conecta este filtro al filtro “Output”, que mediante otra tubería escribe los resultados finales en la salida estándar.

II.2. Atributos de calidad

La calidad de software se define como el grado en el cual éste posee una combinación deseada de atributos, tales como: Performance, Disponibilidad, Portabilidad, Seguridad, Usabilidad, etc. [18]. Estos atributos son requerimientos adicionales del sistema [2] que hacen referencia a características o restricciones que éste debe satisfacer, y complementan los requerimientos funcionales del mismo. Estas características o atributos se conocen con el nombre de “atributos de calidad” [3].

La arquitectura de un sistema de software es determinante para que éste posea una combinación deseada o requerida de atributos de calidad. La arquitectura es la que exhibe o inhibe los atributos de calidad de un sistema, pues una arquitectura representa las decisiones más tempranas del diseño, que impactarán de manera decisiva sobre las etapas de desarrollo posteriores. De esta manera, es posible hacer predicciones sobre cuáles atributos de calidad, y cuáles no, poseerá el sistema de software mediante la evaluación de su arquitectura.

En términos generales, Bass y otros autores [1] establecen una clasificación de los atributos de calidad en dos categorías:

- Observables en tiempo de ejecución: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución (por ejemplo disponibilidad, performance, seguridad, etc.)
- No observables en tiempo de ejecución: aquellos atributos que se establecen durante el desarrollo del sistema (por ejemplo modificabilidad, reusabilidad, portabilidad, escalabilidad, etc.).

La “funcionalidad” es la habilidad de un sistema para realizar el trabajo para el que fue pensado. Los atributos de calidad de un sistema son independientes de la funcionalidad del mismo [1]. Si esto no fuera así, una funcionalidad seleccionada dictaría los niveles de seguridad, modificabilidad, disponibilidad, etc. de un sistema. Esto no significa que cualquier nivel de

cualquier atributo de calidad puede ser alcanzado para cualquier funcionalidad. Manipular gráficos complejos o realizar complejos cálculos matemáticos, por ejemplo, ciertamente tienen un impacto negativo sobre la performance. Sin embargo, lo que sí es posible es que, bajo determinada funcionalidad, se puedan tomar ciertas decisiones que determinen un nivel relativo de calidad.

En general, es en la funcionalidad en donde se pone el foco de atención durante las distintas etapas de desarrollo, dejando a los atributos de calidad, con suerte, en un segundo plano. De esta manera, la mayoría de los enfoques tienen en cuenta a los atributos de calidad separadamente de la funcionalidad, dejando su integración para las etapas finales del ciclo de desarrollo [3]. Sin embargo, los atributos de calidad deben ser considerados en todas las etapas del proceso, ya que ninguno es completamente dependiente de una sola etapa [1].

Particularmente, la arquitectura es fundamental para la realización de varios atributos de calidad de un sistema. Es la primera etapa del ciclo de desarrollo en donde las estructuras deben ser diseñadas satisfaciendo a los atributos requeridos, y en donde estos pueden ser incluso testeados. Es por esta razón, que el tener un conocimiento exacto de los atributos de calidad de un sistema en las etapas tempranas de desarrollo es fundamental para lograr un sistema exitoso [19].

En este trabajo se denominará QAR (Quality Attribute Requirement) a un requerimiento de un atributo de calidad [20]. En la bibliografía los QARs también son conocidos como “Non Functional Requirements”, o NFRs. Sin embargo, se adopta como convención la primera denominación.

II.2.1. Principales atributos de calidad

A continuación se describen algunos atributos de calidad que se mencionarán luego a lo largo del trabajo. Estos son disponibilidad (availability), performance (performance), seguridad (security), usabilidad (usability), modificabilidad (modifiability) y testeabilidad (testeability).

II.2.1.1. Disponibilidad

La disponibilidad se relaciona con las fallas o interrupciones que pueden ocurrir en un sistema y sus consecuencias asociadas. Una falla del sistema ocurre cuando el mismo no puede brindar un servicio consistente con sus especificaciones. Dicha falla puede ser observada por usuarios del sistema - ya sean humanos u otros sistemas.

Algunas de las áreas concernientes son: identificar cómo la falla del sistema es detectada, cuán frecuentemente ésta puede ocurrir, qué sucede cuando la falla ocurre, cuánto tiempo el

sistema está fuera de operaciones luego de que ésta haya ocurrido, cuántas fallas pueden ser prevenidas o que clases de notificaciones se deben producir cuando una falla ocurre.

De esta manera, la disponibilidad está fuertemente relacionada con la confiabilidad, es decir, con la habilidad del sistema de mantenerse operativo todo el tiempo.

II.2.1.2. Performance

Performance se relaciona con el tiempo. Ocurren eventos (por ejemplo, interrupciones, mensajes, pedidos de usuarios o simplemente el paso del tiempo), y el sistema debe responder a ellos. Hay una gran variedad de caracterizaciones de eventos que arriban y sus respuestas, pero básicamente la performance tiene que ver con cuánto tiempo le lleva al sistema responder cuando un evento ocurre.

La performance a nivel arquitectónico se puede entender, modelar y analizar, observando: la distribución y la tasa de llegada de los servicios de pedidos, los tiempos de procesamiento, el tamaño de las colas, y la latencia. Se puede simular la performance de un sistema mediante la construcción de un modelo de encolado estocástico, basado en escenarios de carga de trabajo anticipado.

II.2.1.3. Seguridad

La seguridad es una medida de la capacidad del sistema para resistir el uso no autorizado sin dejar de ofrecer sus servicios a los usuarios legítimos. Un intento de violación de la seguridad se denomina “ataque” y puede adoptar diversas formas. Puede ser un intento no autorizado para acceder a datos o servicios o para modificar datos, o podría ser el intento de negar servicios a los usuarios legítimos.

Los ataques pueden ser de una amplia variedad, como el robo de dinero por transferencia electrónica, modificación de datos sensibles, robo de números de tarjetas de crédito, destrucción de los archivos en los sistemas informáticos o ataques de denegación de servicio ocasionados por gusanos o virus.

II.2.1.4. Usabilidad

La usabilidad concierne a la facilidad que se le brinda al usuario para completar una tarea deseada y al soporte de usuario que el sistema provee. Puede ser descompuesto en las siguientes áreas:

- Aprender las características del sistema.
- Usar el sistema eficientemente.
- Minimizar el impacto de los errores.
- Adaptar el sistema a las necesidades del usuario.
- Aumentar la confianza y satisfacción del usuario.

II.2.1.5. Modificabilidad

La modificabilidad de un sistema es la habilidad para hacer cambios en forma rápida con un costo razonable. En general, esta capacidad deriva directamente de la arquitectura ya que este atributo es, casi por completo, función de la localización de cada cambio. Hacer un cambio disperso en el sistema es, frecuentemente, más costoso que hacer un cambio en un solo componente y que los restantes permanezcan sin cambios.

La regla anterior tiene excepciones. Un solo componente que es excesivamente grande y complejo, puede ser más costoso de modificar que cinco componentes más simples. Sin embargo, también es fácil imaginar un cambio global, simple y sistemático: por ejemplo, modificar el valor de una constante que aparece en todos lados. De esta manera, se considera como principio general que los cambios localizados son los mejores [1].

Dado que la arquitectura define los componentes y las responsabilidades de cada uno, también define las circunstancias bajo las cuales cada componente tendrá que cambiar. Una arquitectura clasifica los posibles cambios en tres categorías, de acuerdo a si el cambio se realizará sobre un solo componente, sobre más de un componente, o algo más drástico como el cambio del estilo de la arquitectura.

II.2.1.6. Testeabilidad

La testeabilidad del software se refiere a la facilidad en la cual éste puede ser sujeto a pruebas (generalmente basadas en la ejecución) para demostrar sus defectos. Al menos el 40% del costo de desarrollo de sistemas bien diseñados es absorbido por las pruebas. Si el arquitecto de software puede reducir este coste, la recompensa es significativa.

En particular, la capacidad de prueba se refiere a la probabilidad, en el supuesto de que el software tiene al menos un defecto, de que se produzca una falla en la ejecución de la siguiente prueba.

Para que un sistema sea testeable, debe ser posible controlar el estado y la entrada de cada componente interno y después observar su salida.

II.2.2. Atributos de calidad en una arquitectura

Siguiendo con el ejemplo de la arquitectura mostrada en la Figura II.1, se puede analizar en qué medida este diseño impacta en los atributos de calidad mencionados.

En primera medida, se aprecia que ese diseño posee un alto nivel de modificabilidad. Cada filtro posee una funcionalidad específica, por lo que a la hora de realizar una modificación a la funcionalidad actual es sencillo ubicar el lugar indicado. Al ser cada módulo independiente del otro, el cambio en uno no afecta al resto. También es sencillo agregar, modificar o eliminar módulos sin afectar al resto de sistema.

El hecho de que dos o más módulos puedan realizar su tarea simultáneamente tiene un impacto positivo sobre la performance, al agregar concurrencia.

Sin embargo, como atributos de calidad en los cuales la arquitectura impacta negativamente, se debe mencionar que el diseño dificulta la interacción con el usuario, resultando en un decremento del atributo de usabilidad. Como los filtros no tienen estado, se debe transmitir toda la información de un filtro a otro. Si es un volumen grande de información, se produce un alto overhead, que impacta negativamente sobre la performance.

Como se ve, ningún atributo de calidad se logra en aislamiento. Generalmente, satisfacer un atributo dado tiene un impacto negativo sobre otro atributo.

II.3. Aspectos tempranos y atributos de calidad

Se define un “concern” como “cualquier asunto de interés en un sistema de software” [21]. El desarrollo de software tiene que tratar con un gran número de concerns. Algunos de estos están relacionados al producto (el software) que debe ser creado, como la funcionalidad y la performance. Otros concerns están relacionados con el proceso de desarrollo en sí mismo, como los tiempos y costos del desarrollo.

Mientras que, por medio de técnicas convencionales de modularización u orientación a objetos, un tipo de concern puede ser encapsulado dentro de artefactos como módulos, clases y

operaciones a nivel de diseño o implementación, esto mismo no es posible para otro tipo de concern. Estos cortan transversalmente (crosscut) el diseño o implementación de unos cuantos o incluso muchos artefactos y por lo tanto son llamados crosscutting concerns [21].

En el contexto de la Ingeniería de Requerimientos (*Requirements Engineering*, RE) los concerns de interés central son los requerimientos. Un requerimiento es un tipo especial de concern. Una especificación de requerimientos bien escrita está caracterizada por el hecho de que cada declaración de requerimientos no mezcla varios requerimientos sino que representa exactamente un requerimiento, por ejemplo, solamente un concern.

Sin embargo, esto no siempre sucede en la práctica, donde ocurre que ciertos concerns se encuentran presentes en más de un requerimiento. Los requerimientos que cortan transversalmente a otros son denominados *requerimientos crosscutting* (crosscutting requirements) [22] o aspectos tempranos (early aspects). De esta manera, se define a un "aspecto temprano" como un concern que atraviesa el diseño de un sistema [23], y se manifiesta generalmente en las especificaciones de requerimientos u otros documentos preliminares producidos en el análisis de requerimientos.

La Ingeniería de Requerimientos orientada a Aspectos (Aspect-oriented requirement Engineering, AORE) se enfoca en la identificación, especificación y representación de los aspectos tempranos. Ejemplos de estos últimos incluyen a concerns que se identifican con persistencia, distribución, seguridad o restricciones de tiempo real.

Muchos aspectos tempranos se corresponden con QARs de alto nivel como seguridad, performance, portabilidad y usabilidad [24]. De hecho, el descubrimiento tanto de aspectos tempranos como de atributos de calidad tiene una significativa importancia a la hora de diseñar la arquitectura. Además, la correcta identificación de aspectos tempranos es significativa no sólo para propósitos de diseño arquitectónico, sino para que puedan ser evaluados y modelados en la etapa de diseño y codificación del sistema, evitando tener que ser "minados" y refactorizados en etapas posteriores.

En consecuencia, dada la similitud entre aspectos tempranos y atributos de calidad, el descubrimiento de los primeros puede proporcionar pistas para identificar QARs. De esta manera, dado un aspecto temprano puede ser posible realizar un razonamiento acerca del potencial atributo de calidad al que éste hace referencia.

II.4. Especificación de requerimientos

Tanto los requerimientos funcionales como los atributos de calidad (QA) son capturados en un documento denominado "Especificación de Requerimientos de Software" (Software Requirements Specification), o SRS [25].

El SRS es una completa descripción del comportamiento de un sistema a ser desarrollado. Principalmente, está conformado por un conjunto de casos de uso, que especifican la funcionalidad del sistema y como éste responde a distintos estímulos externos. Idealmente, el SRS también incluye una especificación de los requerimientos de atributos de calidad, como así también un prototipo del sistema a ser desarrollado. Sin embargo, esto último rara vez se cumple ó, en caso contrario, los QARs no son especificados de una manera formal mediante escenarios de calidad, sino que son expresados de una manera informal utilizando lenguaje natural.

De hecho, en un estudio realizado [24], en base a 15 especificaciones de requerimientos, se manifestó una escasez de menciones a QARs. Esto puede indicar que los desarrolladores fallan a la hora de analizar la importancia de los QARs, o que falsamente asumen que éstos son sobreentendidos y aceptados por todos los stakeholders.

En general, los documentos son organizados en términos de funcionalidad, poniendo muy poco énfasis en los QARs. De esta manera, éstos quedan diseminados o esparcidos en varios documentos. Como se mencionó antes, esto conlleva a que los atributos de calidad no sean tenidos en cuenta en las primeras etapas del desarrollo, originando diseños arquitectónicos que no cumplen con los requerimientos solicitados y productos de software que no satisfacen las necesidades del cliente.

Más aún, la identificación de los QARs no es una tarea sencilla y puede consumir gran cantidad de tiempo y esfuerzo. En general, esta identificación es realizada “ad-hoc”, analizando varios documentos que describen al sistema. Como los QARs pueden estar esparcidos entre muchos de estos documentos, se corre el peligro de malinterpretarlos fácilmente o, directamente, ignorarlos por completo.

II.4.1. Casos de uso

En ingeniería del software, un caso de uso es una técnica para la captura de requisitos de un nuevo sistema o una actualización de software [26]. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario ó con otro sistema para conseguir un objetivo específico. Normalmente, en los casos de usos se evita el empleo de términos técnicos, prefiriendo en su lugar un lenguaje más cercano al usuario final. En ocasiones, se utiliza a usuarios sin experiencia junto a los analistas para el desarrollo de casos de uso.

Los casos de uso se convirtieron en una de las prácticas más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos. También, como se mencionó anteriormente, representan la espina dorsal de los SRS y es en donde generalmente se pone el mayor énfasis, dejando en un segundo plano la especificación de los requerimientos de los atributos de calidad.

Los casos de uso evitan típicamente los términos técnicos, prefiriendo la lengua del usuario final o del experto del campo del saber al que se va a aplicar. Los casos de uso son a

menudo elaborados en colaboración por los analistas de requerimientos y los clientes. Cada caso de uso se centra en describir cómo alcanzar una única meta o tarea de negocio.

Desde una perspectiva tradicional de la ingeniería de software, un caso de uso describe una característica del sistema. Para la mayoría de proyectos de software, esto significa que quizás a veces es necesario especificar diez o centenares de casos de uso para definir completamente el nuevo sistema. El grado de la formalidad de un proyecto particular del software y de la etapa del proyecto influenciará el nivel del detalle requerido en cada caso de uso. Los casos de uso pretenden ser herramientas simples para describir el comportamiento del software o de los sistemas. Un caso de uso contiene una descripción textual de todas las maneras que los actores previstos podrían trabajar con el software o el sistema. Los casos de uso no describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican cómo se implementará. Simplemente muestran los pasos que el actor sigue para realizar una tarea.

Un caso de uso debe:

- tener un nivel apropiado de detalle;
- describir una tarea del negocio que sirva a una meta de negocio;
- ser bastante sencillo como para que un desarrollador lo elabore en un único lanzamiento.

II.4.2. Escenarios de calidad

Los escenarios de calidad son una forma de representar, de una manera formal, los QARs de un sistema [19]. Un escenario consiste de seis partes [1] (Figura II.2):

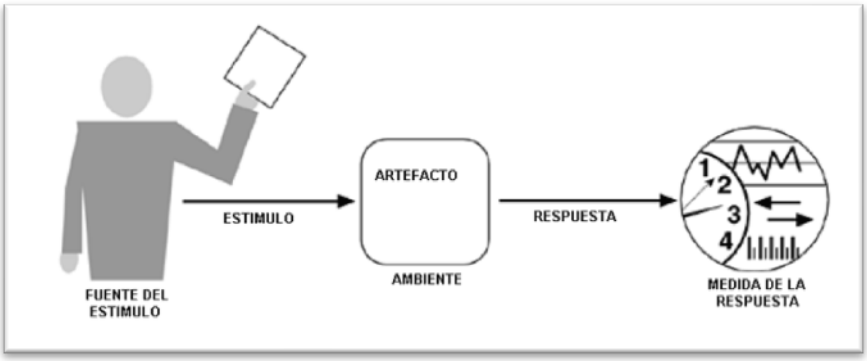

Figura II.2 - Escenario de calidad

1. Fuente del estímulo (Concrete Source). Alguna entidad (un humano, un sistema de computación u otro actor) que genera un estímulo.

2. Estímulo (Concrete Stimulus). Una condición que necesita ser considerada cuando arriba al sistema.

3. Ambiente (Concrete Enviroment). El estímulo ocurre bajo ciertas condiciones. El sistema puede estar sobrecargado o puede estar ejecutándose normalmente cuando el estímulo ocurre, o alguna otra condición puede ser verdadera.

4. Artefacto (Concrete Artifact). Algún artefacto es estimulado. Este puede ser el sistema completo o alguna parte de él.

5. Respuesta (Concrete Response). La respuesta es una actividad llevada a cabo luego del arribo del estímulo.

6. Medida de respuesta (Concrete Response Measure). Cuando la respuesta ocurre, ésta debería ser medida de alguna manera así el requerimiento puede ser testeado.

Se distingue entre escenarios generales y concretos de calidad. Escenarios generales son aquellos que son independientes del sistema y pueden, potencialmente, pertenecer a cualquier sistema. Los escenarios concretos son específicos de un sistema en particular que está bajo consideración. La caracterización de atributos de calidad puede ser hecha como una colección de escenarios generales, sin embargo, para traducir la caracterización de los atributos a requerimientos para un sistema específico, los escenarios generales deben transformarse en específicos.

Un escenario general para el atributo de calidad de disponibilidad, por ejemplo, se describe en la Figura II.3. Se muestran sus seis partes, indicando el conjunto de valores que pueden tomar. El escenario general se forma al elegir, para cada parte, uno de estos posibles valores.

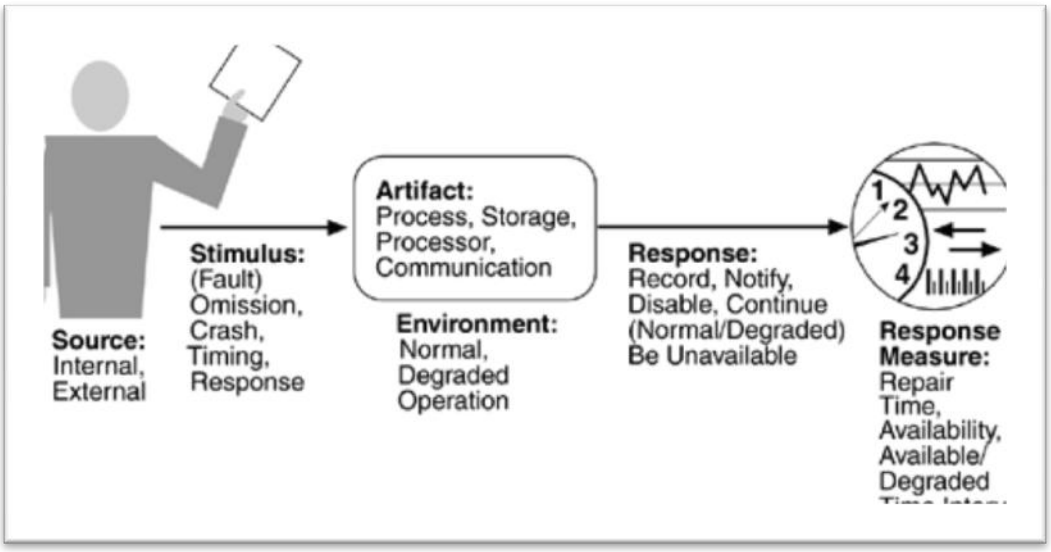

Figura II.3 - Conjunto de valores para escenarios generales del atributo disponibilidad

En la Figura II.4 se muestra un ejemplo de un escenario concreto de disponibilidad, derivado del escenario general mostrado en la Figura II.3. Esto se logra haciendo a cada parte del escenario general específica del sistema en consideración. En este caso, el escenario representado es "An unanticipated external message is received by a process during normal operation. The process informs the operator of the receipt of the message and continues to operate with no downtime."

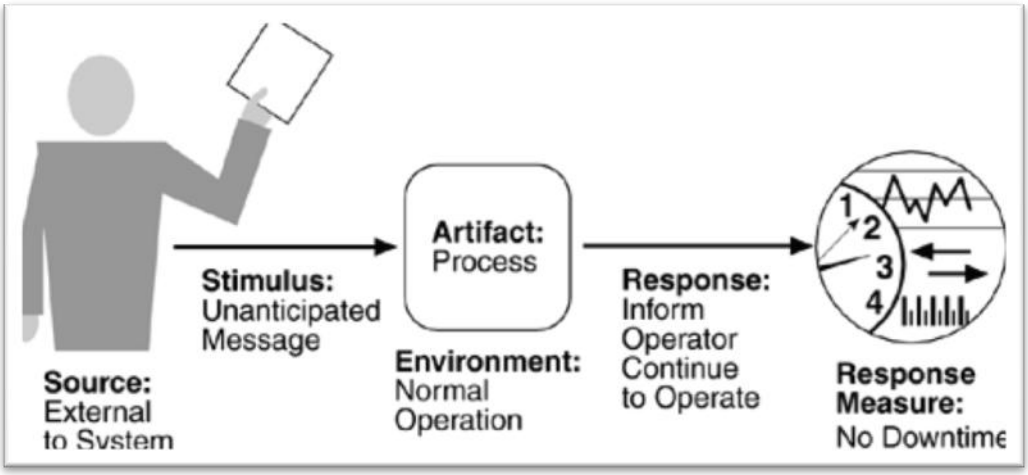

Figura II.4 - Escenario concreto del atributo disponibilidad

Una colección de escenarios concretos puede ser usada como los QARs de un sistema. La información contenida en el conjunto de escenarios indica qué atributos de calidad deben estar presentes en la arquitectura de un sistema de software.

En general, las definiciones de un atributo de calidad no son operacionales. No tiene significado decir que un sistema será, por ejemplo, modificable, ya que todos los sistemas son modificables respecto a un conjunto de cambios y no modificable con respecto a otro conjunto. Los escenarios brindan una solución a esta cuestión al hacer posible su testeo a un nivel arquitectural.

Capítulo III - Trabajos Relacionados

En este capítulo se realizará una reseña de técnicas (o enfoques) existentes para interiorizar y comprender las ideas exploradas por distintos investigadores en el área de identificación de atributos de calidad. Éste no pretende ser un estudio exhaustivo acerca del conjunto de técnicas actuales, sino más bien, mencionar las que sean más representativas de las líneas seguidas.

III.1. Enfoques y técnicas existentes

En general, el problema de identificación y/o clasificación de QARs es abordado por los investigadores desde dos perspectivas distintas [24]. La primera perspectiva se basa en métodos de elicitación de requerimientos, proveyendo soporte para la identificación de QARs a medida que estos son descriptos, negociados o explicados con los stakeholders. La segunda perspectiva incluye métodos de detección para la extracción semi-automática o manual de QARs a partir de una variedad de documentos existentes.

III.1.1. Métodos de elicitación

Los métodos de elicitación basan su enfoque en la provisión de cuestionarios, checklists o plantillas para indagar a los stakeholders acerca de los atributos de calidad involucrados en el sistema. Los QARs son identificados a medida que los documentos de especificaciones de requerimientos están siendo creados. Este hecho brinda la posibilidad de que los QARs puedan ser negociados con los stakeholders en las etapas tempranas de desarrollo.

Por ejemplo, supóngase que un cliente pretende un sistema para el control de ATMs de su banco. Durante la elicitación de requerimientos quizás el stakeholder haga hincapié en la seguridad del sistema, ya que esta es su única o principal preocupación. Sin embargo, en un sistema de esas características influyen otros atributos de calidad como usabilidad, disponibilidad o performance, los cuales no fueron considerados por el cliente. De esta manera, el uso de checklists o cuestionarios durante la elicitación harían razonar al stakeholder sobre aspectos del sistema que no había tenido en cuenta en un principio. Además, en el momento en que se hacen evidentes se pueden negociar con el cliente, analizando la influencia de cada uno sobre el otro y el resultado final que se pretende.

III.1.1.1. Elicitación de requerimientos con casos de uso

En Dörr y otros [27] se presenta un enfoque tendiente a lograr un conjunto de QARs que sea mínimo y completo, y en donde éstos sean medibles y trazables. “Mínimo” se refiere a que solo los QARs relevantes son detectados para no restringir a la arquitectura prematuramente. “Completo” indica que todos los QARs de los stakeholders son identificados. “Medibles” significan que una métrica es dada para poder verificar que el sistema final satisface el QAR detectado. Por último, “trazable” se refiere a que el enfoque propuesto también provee la lógica efectuada por la cual el QAR en cuestión fue identificado. Es decir, informar en que documento/s o parte/s de los mismos se identificó la información que luego fue utilizada por la técnica para identificar ese QAR.

En la técnica propuesta se propone un enfoque para especificar requerimientos relacionados con el atributo de calidad de “Eficiencia”(*efficiency*), en concordancia con los casos de uso y, si está disponible, la arquitectura del sistema. A pesar de estar acotada a ese atributo, la técnica se podría ampliar para cubrir otros, ya que se basa en características generales de todos los atributos de calidad.

Los principales conceptos del enfoque están representados en un metamodelo que luego debe ser instanciado en un modelo de calidad (quality model).

Metamodelo

El metamodelo de la Figura III.1 muestra los conceptos principales del enfoque.

Las principales características y conceptos del mismo son:

- El metamodelo define a un atributo de calidad (QA) como una característica no funcional de un sistema (S*ystem*), tarea del usuario (U*ser Task*), tarea del sistema (S*ystem Task*) u organización (*organization*). La distinción entre diferentes tipos de atributos de calidad (QA) es importante para el proceso de elicitación. Cada atributo es elicitado de diferente forma. Además, un atributo puede ser refinado en más atributos.
- Un sistema puede ser refinado en varios subsistemas. A su vez, éste se encuentra restringido por requerimientos arquitectónicos (architectural requirements).
- Se distinguen dos tipos de tareas: tareas del usuario y tareas del sistema. Las primeras son tareas que el usuario tiene que realizar. Éstas están soportadas por el sistema, pero el usuario se encuentra involucrado. Las tareas del sistema son tareas que el sistema debe realizar, pero ningún usuario está involucrado en la elaboración de las mismas.

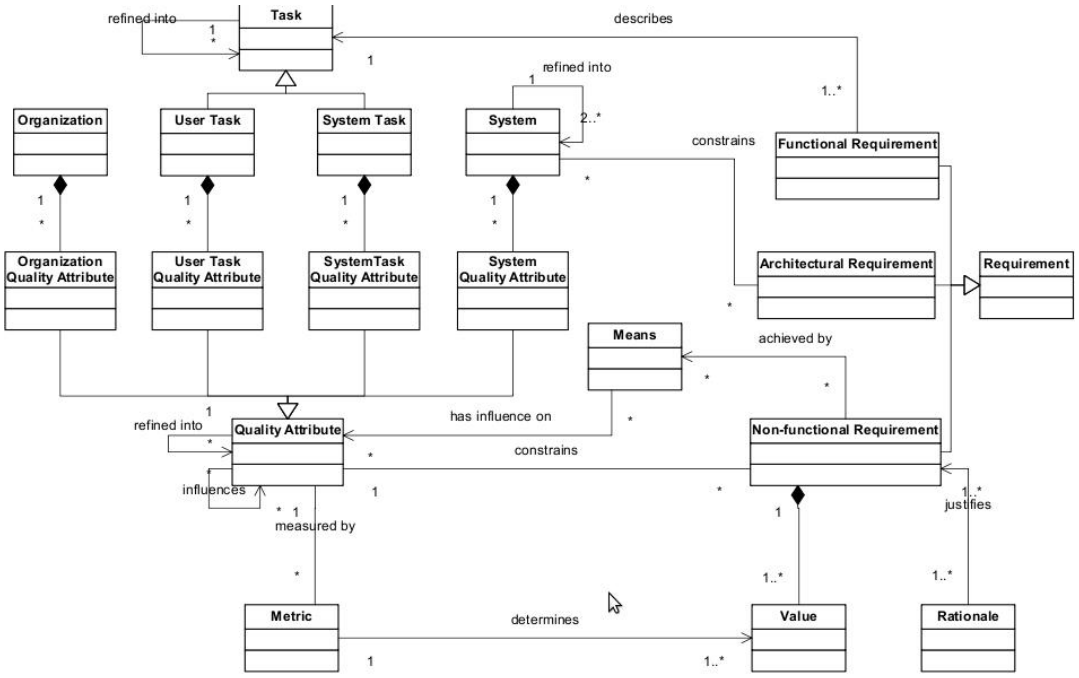

Figura III.1 - Metamodelo propuesto

- Un requerimiento no funcional (*non functional requirement*, NFR) describe un cierto valor para un QA que debe alcanzarse en un proyecto específico. Un NFR limita a un atributo de calidad mediante la determinación de un valor para una métrica (*metric*) asociada con el atributo de calidad. Por ejemplo, el NFR "The database of our new system shall handle 1000 queries per second." limita el atributo de calidad "workload of database ". El valor se determina basándose en métrica "Number of jobs per time unit ". Para cada uno de los NFRs, una justificación (*rationale*) justifica su existencia (por ejemplo, "the user will be unsatisfied if it takes more than 2 seconds to display alarm message ").

- El modelo distingue entre refinamientos orientados a problemas (*problem-oriented refinement,* que serían refinamientos de NFRs en base a restricciones de QAs) y refinamientos orientados a soluciones (*solucion-oriented refinments*). El último se materializa a través de significados (*means*). Un significado es utilizado para satisfacer cierto conjunto de NFRs. En muchas situaciones un significado describe una solución arquitectónica que satisface cierto atributo de calidad (por ejemplo, "balanceo de carga" es utilizada para satisfacer un conjunto de atributos de calidad relacionados con el atributo "distribución de la carga del trabajo").

Quality Model

Un modelo de calidad (*Quality Model*) se crea instanciando partes del metamodelo. Describe refinamientos de QAs de alto nivel en QAs, métricas y significados más específicos. La idea del modelo es refinar QAs en QAs más medibles; por ejemplo, QAs en los cuales una métrica puede ser asociada. A su vez describe relaciones entre distintos QAs, capturando experiencias de proyectos previos.

La Figura III.2 muestra un ejemplo de un modelo de calidad para el atributo de calidad de eficiencia. Los QAs están representados por rectángulos blancos. Los rectángulos grises son *significados* que tienen influencia en los QAs relacionados y los óvalos son métricas para medir los QAs relacionados.

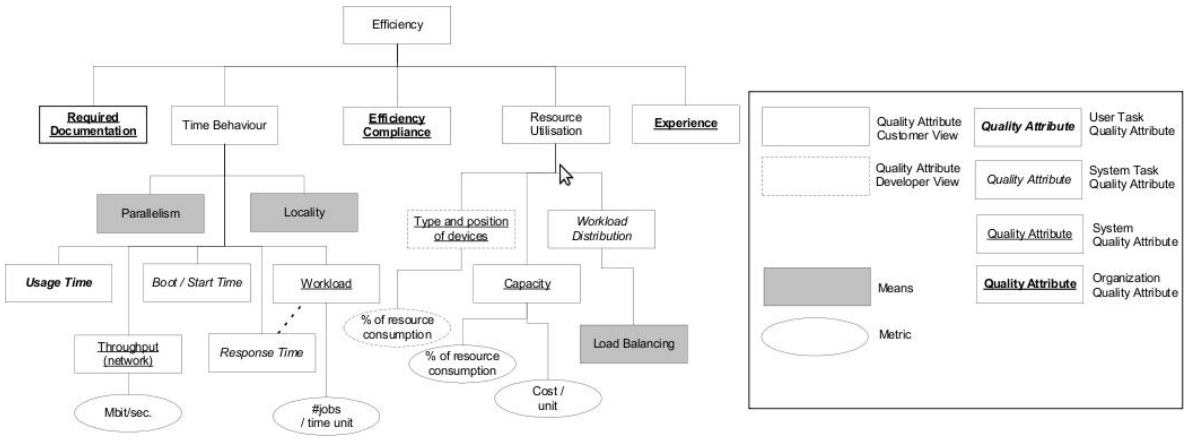

Figura III.2 - Modelo de calidad para "eficiencia"

El enfoque propuesto en este trabajo presenta un modelo de calidad que puede ser utilizado sin modificaciones por una compañía. Las razones para ello pueden ser falta de dinero o de tiempo. Sin embargo, se recomienda adaptar el modelo al contexto de cada empresa y proyecto.

La Figura III.3 muestra el proceso de adaptar el modelo de calidad en un modelo particular para la empresa. El modelo adaptado (*Experience Based Quality Model*) es utilizado como entrada para la actividad “Derive Facilities”, que elabora los checklists y templates para documentar los NFRs. En este caso, éstos estarán asociados con el atributo de calidad para el cual el modelo fue creado.

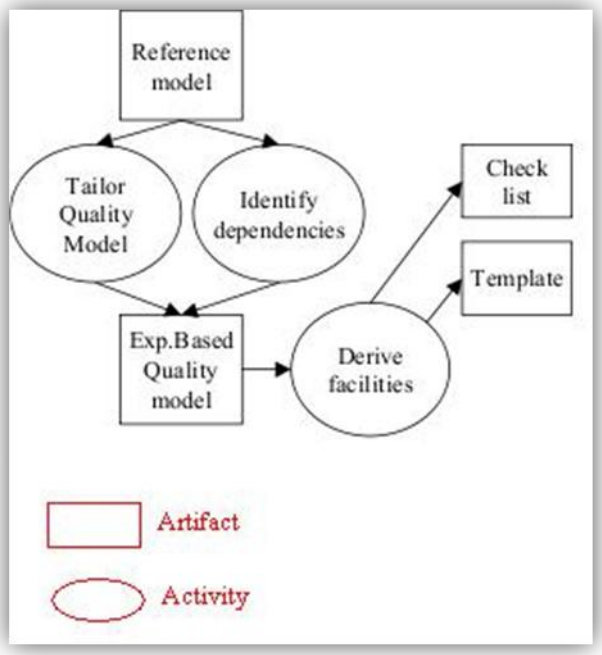

Figura III.3 - Creación de un modelo basado en la experiencia

La estructura de los checklists está dada por la jerarquía del modelo de calidad. QAs generales (por ejemplo, *time behavior*) son un medio para estructurar el checklist, mientras que QAs menos generales (por ejemplo, *usage time)* son directamente utilizados para elicitar los NFRs que los restringen. El tipo de QA influencia en la manera en que las preguntas en el checklist son formuladas:

- Los *Organizations QAs* son usados para la iniciación de los checklists que se enfocan en aspectos generales.
- Los QAs de tareas del usuario son utilizados para iterar sobre los casos de uso (por ejemplo, primero el caso de uso 1, luego el 2 y así sucesivamente).
- Los QAs de tareas del sistema son utilizados para iterar sobre los pasos de un caso de uso.
- Los QAs del sistema son utilizados para iterar sobre los distintos subsistemas del sistema (por ejemplo, primero las bases de datos, luego las redes, etc.).

La estructura del template también está influenciada por el modelo de calidad. Los NFRs restringiendo los distintos atributos de calidad son denotados en diferentes partes del template. Por ejemplo, los NFRs que restringen QAs organizacionales (*Organizations QAs*) son documentados en una sección de requerimientos organizacionales; los NFRs que restringen tareas del usuario son adjuntados junto con los diagramas de casos de uso; etc. La Figura III.4 muestra un fragmento del template resultante.

1. Organizational requirements
   1.1. Process requirements
   1.2. Stakeholder requirements
2. Task descriptions
   2.1. UC diagram
   2.2. Textual UC description
3. Task overspanning requirements
   3.1. Textual description of Task overspanning NFR´s

Figura III.4 - Fragmento del template de requerimientos resultante

Tanto los templates como los checklists son luego utilizados durante el proceso de elicitación, como soporte para la identificación, descripción y análisis de los requerimientos no funcionales.

III.1.1.2. Atributos de calidad "crosscutting" en la ingeniera de requerimientos

El trabajo propuesto por Moreira y otros [28] se focaliza en los atributos de calidad que atraviesan distintas funcionalidades descriptas en los requerimientos. Es por ello que se propone un modelo para identificar y especificar QARs que atraviesen los requerimientos.

El modelo de proceso propuesto es compatible con UML y está compuesto por tres etapas principales: *identificar* (identify), *especificar* (specify) e *integrar* (integrate).

La Figura III.5 muestra las principales actividades del modelo.

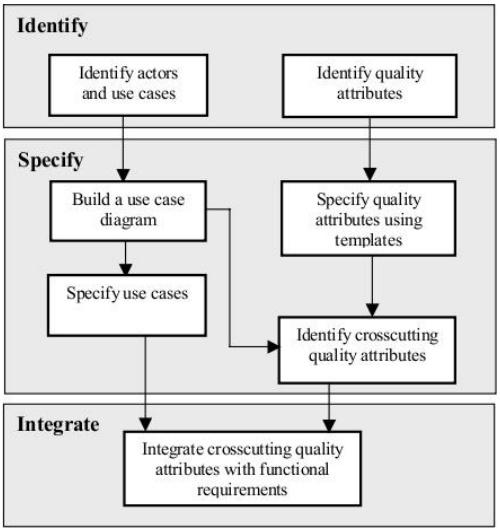

Figura III.5 - Modelo de requerimientos para atributos de calidad

La primera etapa consiste en identificar todos los requerimientos de un sistema y seleccionar, de éstos, los atributos de la calidad relevantes al dominio de aplicación y a los stakeholders. A su vez, se identifican los casos de uso y los actores del sistema.

La segunda etapa se divide en dos partes principales: (1) especificar los requerimientos funcionales utilizando un enfoque basado en casos de uso, (2) describir los atributos de calidad utilizando plantillas especiales e identificar aquellos que atraviesan los requerimientos funcionales (es decir, los atributos de calidad transversales).

Especificar los requerimientos funcionales basados en casos de uso implica primeramente representar los casos de uso en diagramas de casos de uso. Esto permite visualizar las relaciones entre los casos de uso, particularmente los eventos compartidos al comienzo de los mismos. Una vez realizado lo anterior se especifican los casos de uso a través de escenarios, como los descriptos en la sección II.4.2.

El modelo seleccionado para especificar atributos de calidad está influenciado por los enfoques de Chung y otros [29] y Malan y Bredmeyer [30] (Figura III.6). Para identificar el carácter transversal de algunos de los atributos de calidad se tiene en cuenta la información contenida en las filas Dónde (*Where*) y Requerimientos (*Requeriments*). Si un atributo de calidad atraviesa varios requerimientos y modelos, se considera transversal o *crosscuting*.

Name	The name of the quality attribute
Description	Executive description
Focus	A quality attribute can affect the system (i.e. the end product) or the development process
Source	Source of information (e.g. stakeholders, documents)
Decomposition	Quality attributes can be decomposed into simpler ones. When all (sub) quality attributes are needed to achieve the quality attribute, we have an AND relationship. If not all the sub quality attributes are necessary to achieve the quality attribute, we have an OR relationship
Priority	Expresses the importance of the quality attribute for the stakeholders. A priority can be MAX, HIGH, LOW and MIN
Obligation	Can be optional or mandatory
Influence	Activities of the software process affected by the quality attribute
Where	List of the actors influenced by the quality attribute and also a list of models (e.g. use cases and sequence diagrams) requiring the quality attribute
Requirements	Requirements describing the quality attribute
Contribution	Represents how the quality attribute affects other quality attributes. This contribution can be positive (+) or negative (-)

Figura III.6 - Template propuesto para atributos de calidad

La tercera actividad propone una serie de modelos para representar la integración entre los atributos de calidad transversales y los requerimientos. Básicamente, se utilizan diagramas de UML para mostrar esa integración, como por ejemplo casos de uso con nuevo estereotipo (<<stereotype>>). La Figura III.7 muestra un ejemplo de lo anterior. Allí se ve como se creó un caso de uso con el estereotipo <<Response Time>>, que representa una medida de respuesta del atributo performance, y el diagrama de secuencia derivado del mismo. Las flechas y rectángulos grises indican una restricción, manifestando que “SingleToll” tiene que responder en t2 – t1 unidades de tiempo.

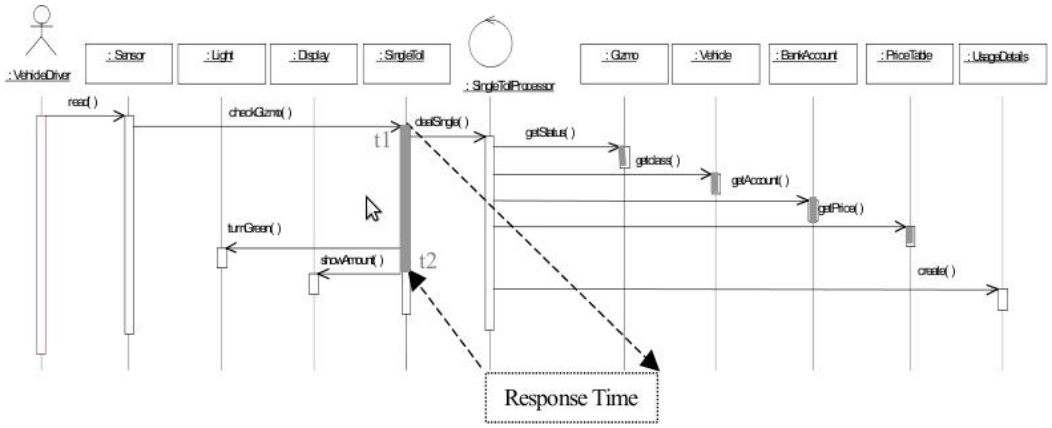

Figura III.7 - La medida de respuesta atraviesa los requerimientos funcionales

Con el método anterior se pretende lograr una separación de concerns, identificando y poniendo de manifiesto los atributos de calidad que atraviesan la funcionalidad.

III.1.2. Métodos de detección semiautomáticos

Otros enfoques se basan en herramientas semi-automáticas que extraen QARs a partir de una variedad de documentos creados en la etapa de requerimientos. En estos casos se suelen utilizar técnicas de recuperación de la información (Information Retrieval, IR) o procesamiento natural del lenguaje (Natural Language Process, NLP) [6].

III.1.2.1. Clasificador NFR

En [24] el problema de detección y clasificación de QARs es abordado desde una perspectiva de clasificación supervisada. Este enfoque utiliza métodos de recuperación de la información para encontrar e identificar QARs. El método supone que los diferentes tipos de QARs se caracterizan por el uso de palabras claves, relativamente distintas, a las que denominan "términos indicadores". Cuando los términos indicadores se aprenden de un tipo específico de QAR, pueden ser utilizados para detectar requerimientos, oraciones, o frases relacionadas con ese tipo. El proceso, que se representa en la Figura III.8, incluye las dos fases principales de *Training (Entrenamiento)* y *Classification* (*Clasificación*). Una fase adicional, marcada en el diagrama como *Iterative Training Phase* refina los términos indicadores previamente clasificados al proveer feedback indicando la correctitud de éstos resultados.

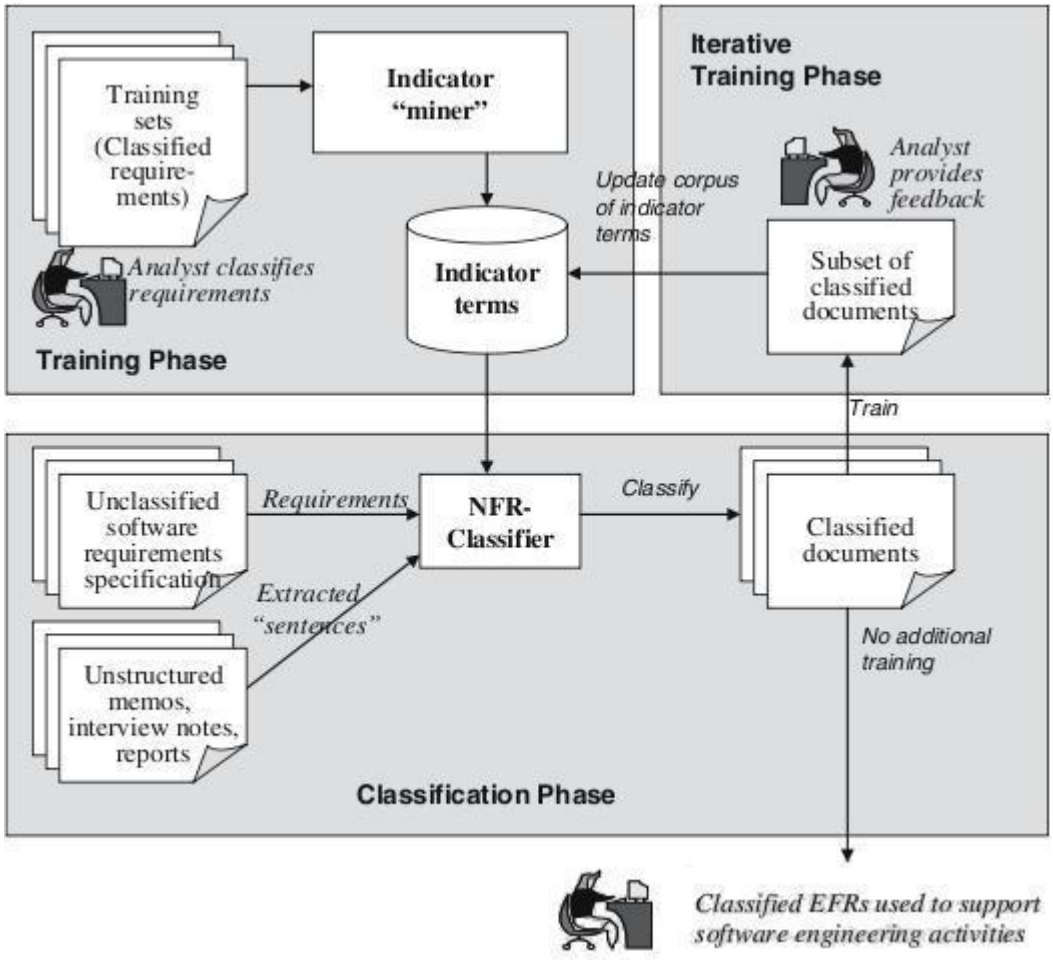

Figura III.8 - Proceso de clasificación de NFRs (QARs)

Etapa de entrenamiento (Training phase)

Durante la primera etapa, un conjunto de *términos indicadores (*indicator terms*)* es identificado para cada categoría de atributo de calidad. Este paso supone la existencia de un conjunto de requerimientos correctamente pre-clasificados que se pueden utilizar para el entrenamiento. Los requerimientos del conjunto de entrenamiento son utilizados para calcular el peso probabilístico de cada término indicador potencial, con respecto a cada tipo de atributo de calidad. Este peso es una medida de cuán fuerte ese término indicador representa a un atributo de calidad. Por ejemplo, términos como “autenticar” o ''acceso”, que se mencionan con frecuencia en requerimientos de seguridad y con poca frecuencia en otros tipos de requerimientos, representan términos indicadores fuertes del atributo de seguridad. A su vez, otros términos tales como "garantizar," que se mencionan menos con frecuencia en los requerimientos de seguridad o se encuentran en varios tipos distintos de requerimientos diferentes, representa un indicador mucho más débil.

Como se ve en Figura III.8 un grupo de requerimientos pre-clasificados (*classified requirements*) son la entrada de la fase de entrenamiento. A partir de estos requerimientos se "minan" los términos indicadores representativos de cada atributo de calidad. De esta manera, se forma un conjunto de términos indicadores, cada uno con su peso, para cada atributo de calidad.

Rank	Availability	Legal	Look and feel	Maintainability
1	avail	compli	appear	updat
2	achiev	regul	interfac	mainten
3	dai	standard	profession	releas
4	time	sarban	appeal	new
5	hour	oxlei	colour	chang
6	pm	php	look	dure
7	year	pear	simul	promot
8	technic	legal	product	product
9	downtim	law	compli	addit
10	long	estimat	scheme	everi

Figura III.9 - Principales términos indicadores por atributo de calidad

La Figura III.9 muestra un ejemplo de los resultados obtenidos luego de esta etapa. La misma muestra los 10 términos indicadores con mayor peso para cuatro atributos de calidad. Esta prueba se hizo en base a 15 especificaciones de requerimientos creadas por estudiantes de la Universidad de DePaul [24].

Etapa de clasificación (Classification phase)

Una vez que los términos indicadores son extraídos y ponderados pueden ser utilizados, en una segunda etapa, para clasificar nuevos requerimientos. Un valor de probabilidad, que representa justamente la probabilidad de que el nuevo requerimiento pertenezca a un tipo determinado de atributo de calidad, se calcula en función de la aparición de términos indicadores de ese atributo en el requerimiento. Éste se clasifica de acuerdo a un tipo determinado de atributo de calidad si contiene varios términos indicadores representativos de ese tipo. Si un requerimiento recibe un valor que supera un cierto umbral para un atributo de calidad, el requerimiento es clasificado como perteneciente a ese atributo. Si el requerimiento no supera el umbral para ningún atributo de calidad, se etiqueta a ese requerimiento como funcional.

III.1.2.2. Técnicas de clasificación para requerimientos informales

En el trabajo realizado por Park et al. [31] se propone un sistema de soporte para el analista que permite la clasificación de requerimientos bajo distintas categorías. La motivación de

este enfoque es el hecho que los proyectos de gran escala cuentan con un gran número de requerimientos, generalmente provenientes de fuentes distribuidas. Estos requerimientos en general son escritos en lenguaje natural, sin una adecuada organización o formalidad. Es por ello que se hace necesaria una correcta clasificación de los mismos, para su adecuado manejo en las etapas tempranas de desarrollo.

Se presenta un enfoque automático de clasificación de requerimientos en distintas categorías. El mismo utiliza técnicas de procesamiento del lenguaje natural (Natural Language Process, NLP). Estás categorías pueden ser, por ejemplo, costo, prioridad, tiempo de desarrollo, criticidad, atributo de calidad o cualquier otra en la que se haya entrenado a la misma.

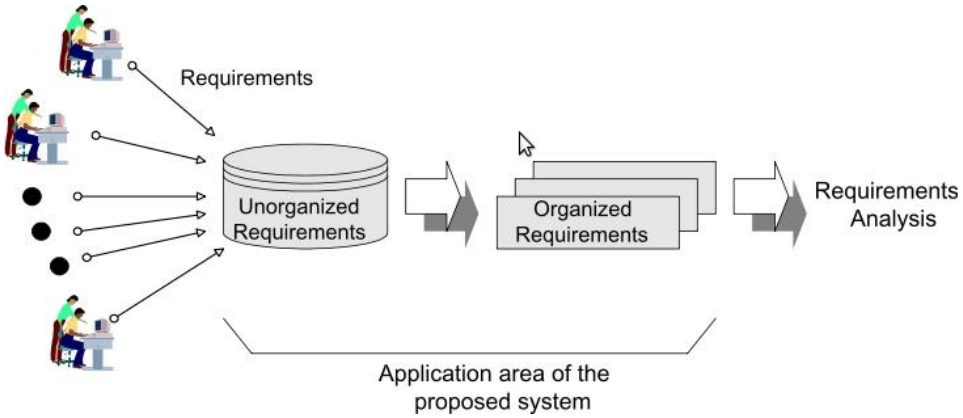

Figura III.10 - Área de aplicación del sistema propuesto

En la Figura III.10 se muestra el área de aplicación del sistema propuesto. Varios requerimientos, provenientes de distintas fuentes, son ingresados al sistema. Éstos forman un conjunto de requerimientos desorganizados. Este grupo es la entrada de la técnica propuesta que, luego de su aplicación, devuelve un conjunto de requerimientos organizados según criterios predefinidos y listos para un análisis posterior.

El proceso seguido para la clasificación de requerimientos consta de tres etapas: *pre-procesamiento*, *construcción de un conjunto de "oraciones-centrales" para el entrenamiento de cada categoría* y *aprendizaje del clasificador*.

- Pre-procesamiento: en esta etapa se extraen las sentencias de un grupo de requerimientos preseleccionados. Por cada sentencia se separan las palabras y se realiza un etiquetado POS [32].

- *Construcción de un conjunto de "oraciones-centrales" para el entrenamiento de cada categoría:* se basa en el hecho de que un analista puede realizar una lista de *topic words*. Éstas son palabras representativas de cada categoría o tópico por el que se quiere clasificar los requerimientos. Mediante esta lista de *topic words* se crea una lista de *key*

words. Estas *key words* son palabras, provenientes de los requerimientos utilizados en el pre-procesamiento, altamente relacionadas con las *topic words.* Estos dos conjuntos de palabras son utilizados para obtener un conjunto de *oraciones centrales* para cada categoría. Este conjunto está formado por las oraciones más representativas de cada categoría y contienen una o más *key words* o *topic words.*

- *Aprendizaje del clasificador:* las oraciones centrales obtenidas en el paso anterior sirven como base para el entrenamiento de un clasificador bayesiano.

III.1.2.3. Identificación de QARs en especificaciones textuales. Enfoque semi-supervisado de aprendizaje

En [6] se propone un enfoque de categorización de texto semi-supervisado para la identificación automática y clasificación de requerimientos no funcionales. Un pequeño número de requerimientos permiten el aprendizaje inicial de un clasificador de NFRs, que sucesivamente puede identificar el tipo de requerimientos adicionales en un proceso iterativo. El objetivo de la técnica es la integración de un sistema de recomendación para asistir a los analistas de requerimientos y diseñadores de software en el proceso de diseño arquitectónico.

En este trabajo, se presenta un enfoque semi-supervisado para la detección y clasificación automática de NFRs destinado a la utilización de requerimientos no clasificados para reducir el número de ejemplos necesarios para el aprendizaje. Para la clasificación de texto, los datos no etiquetados (requerimientos sin etiquetas de clase asignadas) se utilizan junto con una pequeña cantidad de datos etiquetados (requerimientos con etiquetas de clase asignada) para producir una mejora considerable en la exactitud del aprendizaje, tomando ventaja de la co-ocurrencia natural de las palabras en los textos. Los clasificadores resultantes pueden ser utilizados para el análisis de requerimientos en proyectos de software en el que los analistas son capaces de proporcionar un conjunto inicial de requerimientos categorizados previamente, por ejemplo, durante las entrevistas o utilizando cualquier otro método alternativo. En este escenario, los analistas recibirán sugerencias acerca de una posible clasificación de los demás requerimientos y podrán aportar información para refinar la clasificación en un proceso iterativo.

En la Figura III.11 se observa un diagrama del proceso seguido en este enfoque.

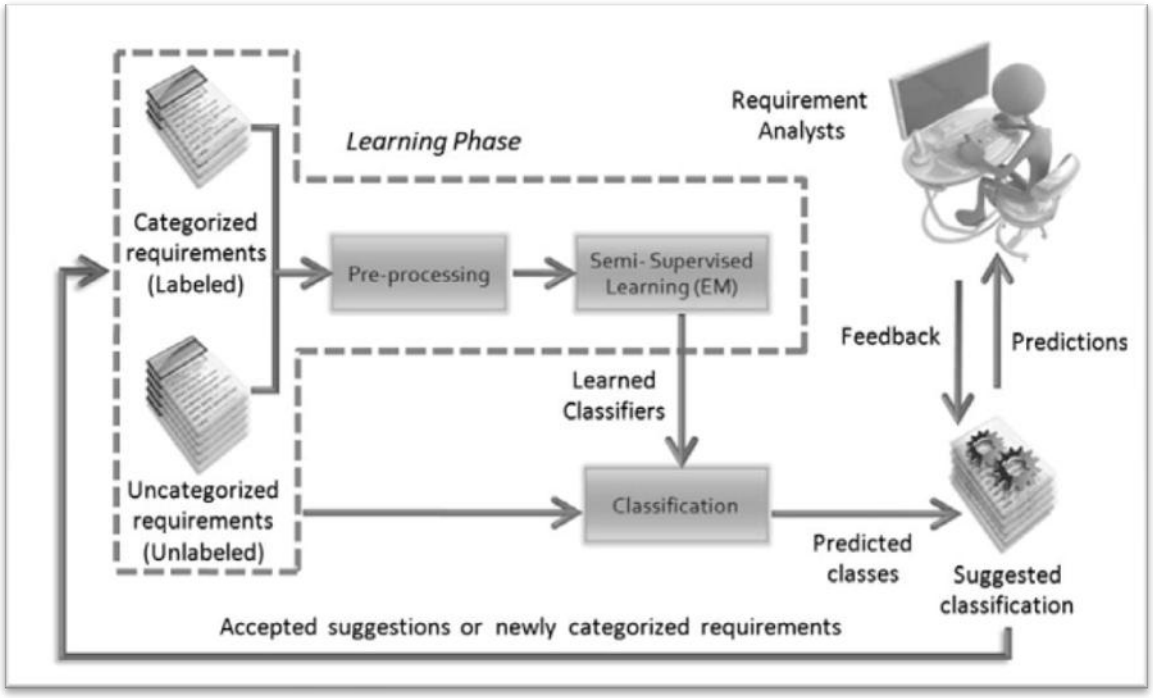

Figura III.11 - Semi-suppervised approach

En principio, durante la etapa de Learning (Aprendizaje) se utilizan requerimientos clasificados en conjunto con no clasificados para entrenar el clasificador, utilizando un algoritmo semi-supervisado. En esta etapa se realizan dos actividades principales: Pre-processing y Semi-Supervised Learning.

Durante el pre-procesamiento, Pre-proccessing, se utiliza uno de los métodos más comunes en el campo de recuperación de información (IR) para obtener representaciones de documentos, el modelo de espacio vectorial (VSM). En este modelo, cada documento es identificado por una característica en el espacio vectorial, en el cual cada dimensión corresponde a un término distinto con un valor numérico o peso indicando su importancia. Previo a esto, se realizan varios pasos para transformar los requerimientos textuales en vectores de acuerdo al modelo VSM. Algunos de estos pasos son la normalización textual (modificar letras mayúsculas, dígitos, guiones y marcas de puntuación), limpieza de stop-words y stemming.

La clasificación parcialmente supervisada implica que no hay necesidad de una supervisión completa, lo que reduce considerablemente el esfuerzo requerido para el etiquetado manual. Una de las posibles estrategias para la supervisión parcial, comúnmente conocido como el aprendizaje semi-supervisado, consiste en aprender de los ejemplos etiquetados y sin etiquetar o documentos en el caso de la categorización de textos. Esta estrategia también se conoce como aprendizaje LU (L es por *labeled* – etiquetado - y U de *unlabeled* – sin etiquetar). Los algoritmos de aprendizaje "LU" se basan en un pequeño conjunto de ejemplos etiquetados pertenecientes a cada categoría y

en un conjunto mucho mayor de ejemplos no etiquetados que es utilizado para mejorar el aprendizaje.

El algoritmo de Maximización de la Expectativa (EM) es un algoritmo iterativo para la estimación de máxima verosimilitud en problemas con datos incompletos. Consta de dos pasos, el paso de "Expectación" (o paso-E) y el paso de "Maximización" (o paso-M). Básicamente, el primer paso rellena los datos que faltan sobre la base de la estimación actual de los parámetros y el segundo paso re-estima los parámetros maximizando la probabilidad. Los documentos no etiquetados pueden considerarse incompletos debido a la falta de etiquetas de clase. Los parámetros que se encuentran en el paso-M a su vez se utilizan para iniciar otro paso-E, y el proceso se repite hasta que EM converge a un mínimo local, cuando los parámetros del modelo se estabilizan.

Los resultados experimentales obtenidos en [6] demuestran la viabilidad de utilizar el aprendizaje semi-supervisado como un método para la detección de NFRs. La evidencia empírica muestra que la semi-supervisión requiere menos esfuerzo humano en el etiquetado de los requerimientos que los métodos totalmente supervisados y se puede mejorar sobre la base de los comentarios de los analistas, una vez integrado en un sistema de apoyo para la gestión de requerimientos. Además, este enfoque no se basa en la existencia de requerimientos previamente categorizados (por ejemplo, pertenecientes a proyectos anteriores), ya que pueden introducir ruido en los textos debido a las variaciones en el vocabulario empleado por los equipos de captura de requerimientos. Por el contrario, la clasificación semi-supervisada requiere una pequeña cantidad de requerimientos etiquetados del proyecto para comenzar el aprendizaje.

III.2. Conclusiones

Se han analizado distintas técnicas para la identificación y clasificación de QARs en etapas tempranas del desarrollo. La mayoría de estos enfoques utilizan estrategias de elicitación de requerimientos o aplican herramientas semi-automáticas, generalmente basadas en técnicas de NLP o IR.

Se observa que, en general, muy pocas de las propuestas utilizan explícitamente información de aspectos tempranos para la identificación de QARs, aunque la propuesta de Moreira [28] menciona la existencia de intereses (o aspectos tempranos) que cortan transversalmente los requerimientos.

En los métodos de elicitación es difícil detectar los aspectos tempranos del sistema como para que estos puedan, en esta misma etapa, ser utilizados para la creación de templates, checklists o cuestionarios.

Las herramientas semi-automáticas analizadas se centran en identificar directamente los QAs, sin utilizar información de los aspectos tempranos. Sin embargo, identificar primero los EA

del sistema sería beneficioso, ya que agregaría información relevante, o pistas, para descubrir QAs ocultos en las especificaciones de requerimientos. Algunos enfoques, como la propuesta de Cleland-Huang [24], poseen una técnica que identifica indistintamente aspectos tempranos y QARs dada la similitud entre ambos. Sin embargo, no se vinculan los aspectos tempranos como precursores de QARs.

En la actualidad existen herramientas semi-automáticas que “minan” aspectos tempranos a partir de una variedad de documentos creados en las etapas tempranas de desarrollo. Es por ello que se ve la necesidad de investigar si, a partir de la información que aportan estos aspectos tempranos detectados, es posible descubrir QARs candidatos del sistema.

Capítulo IV - Identificación de atributos de calidad

En este capítulo se describe en detalle la técnica de identificación de atributos de calidad en especificaciones de requerimientos. Esta técnica ofrece una asistencia extra al analista en la identificación de los atributos de calidad del sistema en etapas tempranas del desarrollo.

La misma parte de un número de aspectos tempranos previamente detectados con una herramienta externa. Cada uno de estos aspectos atraviesa o relaciona un subconjunto del total de casos de uso del sistema. A partir del análisis de las palabras de un aspecto temprano, y del subconjunto de casos de uso que éste relaciona, se intenta identificar el o los atributos de calidad al que aquellos dos hacen referencia.

IV.1. Enfoque propuesto

En este trabajo se ha definido un proceso por medio del cual es posible identificar el grado de relación que tiene un conjunto de early aspects (aspectos tempranos) y casos de uso con los atributos de calidad del sistema, tal como se muestra en la Figura IV.1. Este proceso utiliza como entrada una lista de casos de uso y una lista de early aspects. A partir de esta entrada, se analiza un early aspect por vez junto a los casos de uso relacionados, agregando a cada conjunto <early aspect, casos de uso relacionados> información de los atributos de calidad dominantes para el conjunto en cuestión.

Básicamente, el proceso está constituido por dos etapas. La primera etapa, denominada *Tokens Generation* (Generación de Tokens), realiza un procesamiento sobre los artefactos de entrada (use cases y early aspect) para extraer de los mismos la lista de tokens que los conforman y luego, mediante un filtrado, establecer cuáles son relevantes para la etapa de análisis.

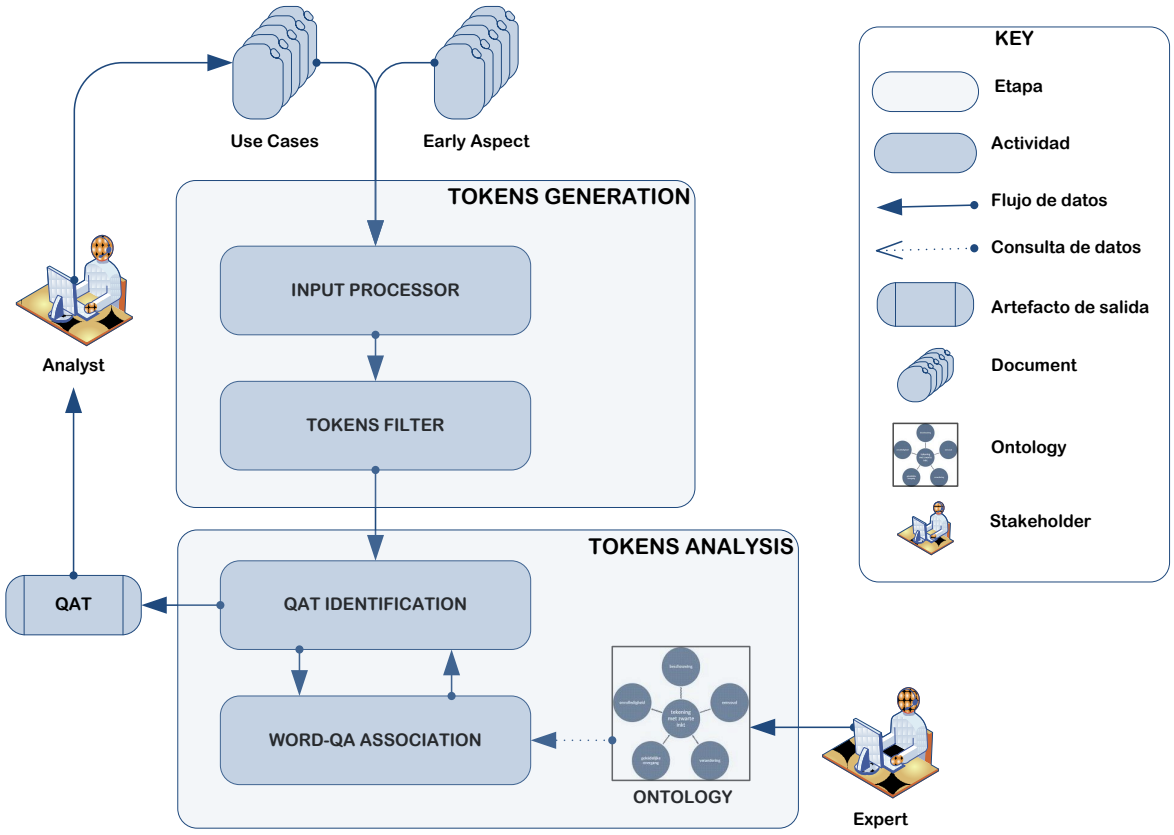

Figura IV.1 - Diagrama de actividades principales

En la segunda etapa, *Tokens Analysis* (Análisis de Tokens), se calcula el porcentaje de pertenencia de cada elemento de entrada con los diferentes atributos de calidad del sistema. Posteriormente, esa información es combinada para obtener el atributo de calidad dominante para el conjunto de entrada. De esta manera, la salida de la técnica es un *Quality Attribute Theme* (QAT). El mismo está formado por un conjunto de casos de uso relacionados con un early aspect. Adicionalmente, un QAT contiene uno o más atributos de calidad al que esos casos de uso y el early aspect hacen referencia.

Para soportar dicho proceso se ha implementado una herramienta, denominada *QA Miner,* como una extensión - plugin - del entorno de desarrollo integrado (IDE) Eclipse [33]. *QA Miner* permite al analista seleccionar como entrada el early aspect que desea analizar, junto a los casos de uso que los relacionan. A partir del resultado obtenido a través de la ejecución de *QA Miner*, el analista podrá contar con la información acerca de los atributos de calidad identificados, además de tener la posibilidad de observar cómo cada token, extraído de la entrada, influye en la formación del QAT.

A continuación se describirán las dos etapas en detalle.

IV.2. Tokens Generation

El conjunto de actividades correspondientes a esta etapa, tiene como finalidad llevar a cabo un análisis léxico y sintáctico sobre los textos definidos como entrada para la técnica propuesta. Esta entrada es una lista de casos de uso y el early aspect que los relaciona. Existen herramientas que generan esta información, o sobre las cuales se puede adaptar la salida para este propósito. Una de las herramientas utilizada para el desarrollo de este enfoque fue Aspect Extractor Tool [8], la cual, a través de un análisis semántico de los casos de uso, identifica los early aspects ocultos en la especificación [34]. La salida producida por esta herramienta es la lista de early aspects detectados, los casos de uso analizados y las relaciones entre ellos.

La especificación textual de los casos de uso respeta el estándar establecido por Rational [35]. De igual modo, la especificación de los early aspects sigue el formato definido en Aspect Extractor Tool, el cuál es semejante al de los casos de uso. Con toda esta información, mediante la técnica propuesta, se confeccionan las listas de tokens que serán suministradas a la próxima etapa.

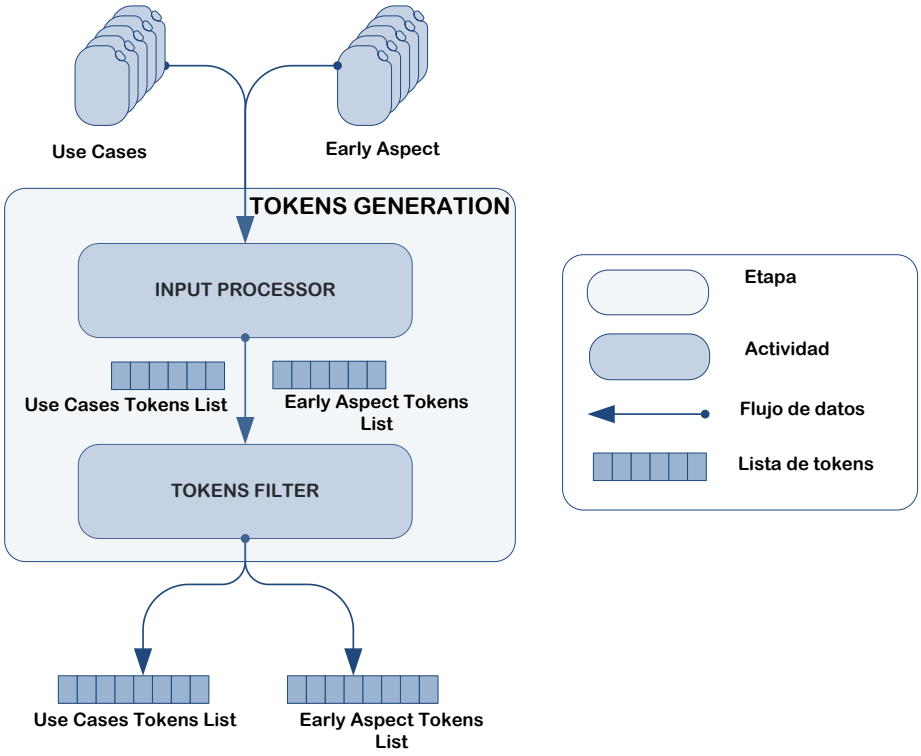

Figura IV.2 - Etapa 1: Tokens Generation

Para el análisis de la entrada, se han definido dos actividades secuenciales. En primer lugar, un procesamiento de los datos de entrada para representarlos en un formato común denominado *Input Processor*. En segundo lugar, la actividad *Tokens Filter* realiza un filtrado sobre los mismos para identificar qué información es relevante para la próxima etapa. En estas

actividades, se logrará determinar un subconjunto del conjunto de palabras que forman la especificación de los casos de uso y la definición de los early aspects, las cuales serán relevantes para la identificación de los atributos de calidad involucrados.

A continuación se describen en detalle cada una de las actividades que se desarrollan durante esta etapa.

IV.2.1. Input Processor

El objetivo de esta actividad es procesar la información de entrada para llevarla a una representación interna uniforme. Este procesamiento es realizado a través de la separación de la información de las entidades de entrada (especificaciones de los use cases y del early aspect) en tokens. Se define un token como una unidad básica de texto que puede ser enriquecida con diferentes pares <atributos, valor>, como por ejemplo: <peso: 1>, <ocurrencias: 4>, etc. Estos tokens representan al conjunto de palabras de entrada.

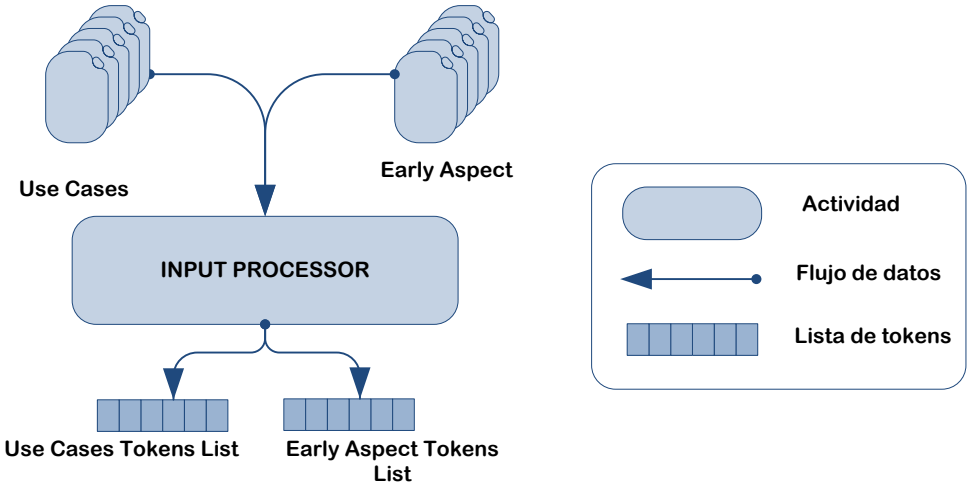

Figura IV.3 - Actividad Input Processor

El resultado de este procesamiento es una lista de tokens para las palabras obtenidas de los casos de uso y otra lista de tokens para las palabras del aspecto temprano en cuestión. Al realizar la división de la entrada en tokens, se agregan los siguientes atributos para cada uno de ellos:

- Id: identificador único del token.
- Tipo: este atributo representa el tipo de documento del cual se extrajo el token (en este punto será early aspect o use case).

- Sección: se registra la sección a la cual pertenece el token. Para los casos de uso, será alguna de las secciones establecidas por el template de Rational (nombre, descripción, flujo básico, actor, etc.) mientras que para el early aspect será alguna de las partes que conforman un aspecto temprano según lo establecido en la herramienta Aspect Extractor Tool[1].

Por ejemplo, considérense la fracción de la especificación de un caso de uso y considérense la fracción de un early aspect, mostradas en la Tabla IV.1 y en Tabla IV.2, respectivamente. Tomando estas especificaciones como entrada, el resultado de aplicar el procesamiento descripto anteriormente generaría como resultado la lista tokens descripta en la Tabla IV.3.

Nombre	Register User
Descripción	Record user information in the system

Tabla IV.1 - Fracción de use case

Nombre	Distribution
<verbo, objeto directo>	<sent, server> , <sending, data>

Tabla IV.2 - Fracción de early aspect

Palabra	**Atributos**
Register	<id,1>, <tipo, use case >,<sección, name>
User	<id,2>, <tipo, use case >,<sección, name >
Record	<id,3>, <tipo, use case >,<sección, description>
user	<id,4>, <tipo, use case >,<sección, description>
information	<id,5>, <tipo, use case >,<sección, description>
in	<id,6>, <tipo, use case >,<sección, description>
the	<id,7>, <tipo, use case >,<sección, description>
system	<id,8>, <tipo, use case >,<sección, description>
Distribution	<id,9>, <tipo, early aspect>,<sección, name>
sent	<id,10>, <tipo, early aspect>,<sección, pair>
server	<id,11>, <tipo, early aspect >,<sección, pair>
sending	<id,12>, <tipo, early aspect >,<sección, pair>
data	<id,13>, <tipo, early aspect >,<sección, pair>

Tabla IV.3 - Tokens generados luego del análisis de la entrada

[1] La herramienta Aspect Extractor Tool representa un early aspect mediante un nombre y un conjunto de pares formados por un objeto directo y un verbo, o simplemente un verbo.

IV.2.2. Tokens Filter

Como se mencionó en la sección anterior, el filtrado de los tokens es otra de las tareas que se realiza sobre la información de entrada. En este caso, se ejecutan una serie de filtros sobre los tokens de las listas de use cases y del early aspect.

El objetivo de esta actividad es filtrar la información proveniente de la etapa anterior, *Tokens Generation*, eliminando información irrelevante y dejando sólo aquella que es útil al análisis. Además, los filtros enriquecen cada token con diferentes atributos (por ejemplo peso, ocurrencias, etc.) que serán concluyentes para la etapa de análisis.

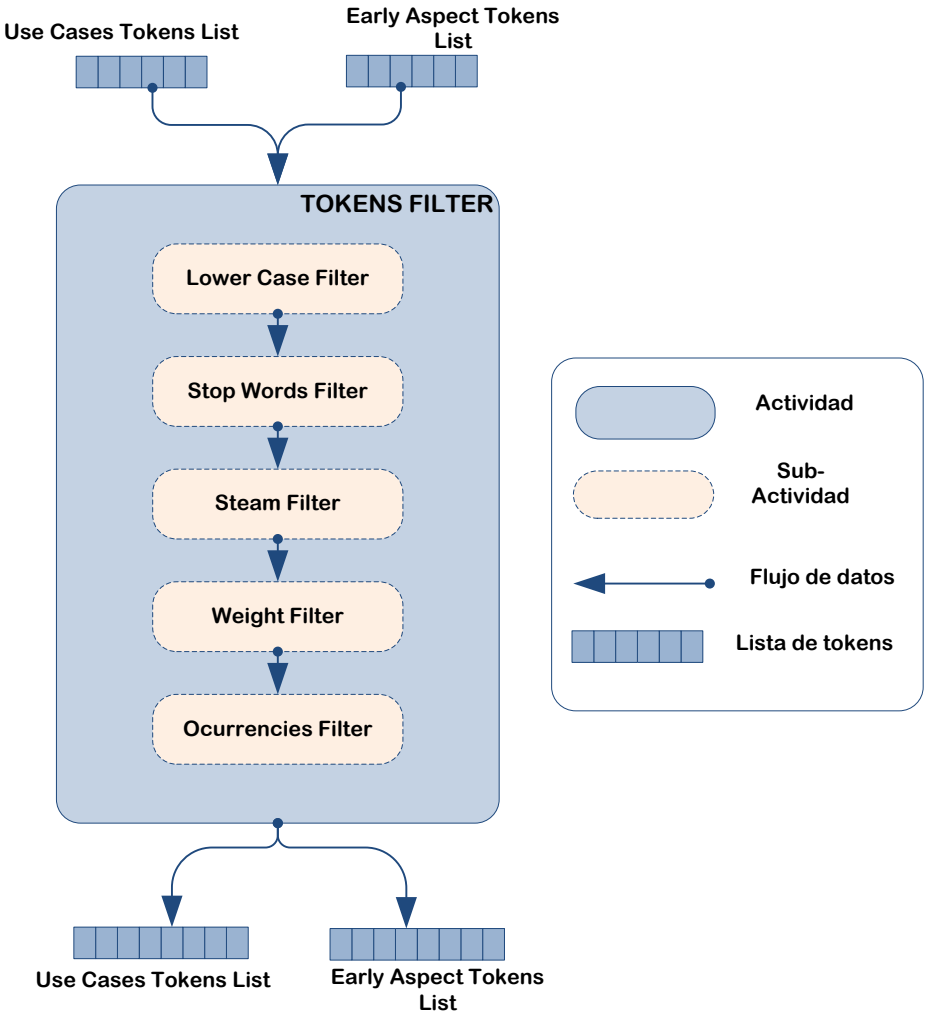

Figura IV.4 - Actividad Tokens Filter

Se considera que un filtro es una unidad de procesamiento que realiza una modificación (enriquece, refina o transforma) sobre los datos de entrada y los copia a la salida para que otro filtro trabaje sobre los mismos datos. Mediante los filtros se pueden realizar transformaciones independientes sobre el flujo de datos.

El patrón de arquitectura Pipes & Filters [17] provee una estructura para procesar flujos de datos. Cada paso de procesamiento se encapsula en un filtro y es independiente del resto. Los datos se transmiten usando los pipes entre filtros adyacentes y mediante la combinación de éstos últimos se pueden conseguir diferentes salidas.

En este caso, la entrada para los filtros es una lista de tokens, por lo que el filtro ejecutará acciones sobre los tokens de la lista (modificar la palabra, agregar atributos), para luego devolver la lista de tokens modificada.

Para esta actividad, hemos definido una serie de filtros que realizan transformaciones sobre los tokens de las listas. Los mismos fueron definidos de forma tal que puedan ser reutilizados y combinados para darle un formato común a cualquier otro texto que represente información relevante durante el proceso.

IV.2.2.1. Filtro Lower Case

Este filtro es el encargado de pasar todos los caracteres de las palabras de los tokens a minúscula. De esta forma, se reconocería que, por ejemplo, “server” y “Server” son la misma palabra. En caso de que se desee que la herramienta sea sensible a mayúsculas/minúsculas bastaría con desconectar este filtro de la secuencia de procesamiento.

Continuando con los tokens generados en la Tabla IV.3, sólo se modifican los caracteres en mayúscula de las palabras de los tokens. En la Tabla IV.4 se observa la lista de tokens luego de la transformación aplicada por este filtro.

Palabra	Atributos
register	<id,1>, <tipo, use case >,<sección,name>
user	<id,2>, <tipo, use case >,<sección, name >
record	<id,3>, <tipo, use case >,<sección,description>
user	<id,4>, <tipo, use case >,<sección, description>
information	<id,5>, <tipo, use case >,<sección, description>
in	<id,6>, <tipo, use case >,<sección, description>
the	<id,7>, <tipo, use case >,<sección, description>
system	<id,8>, <tipo, use case >,<sección, description>
distribution	<id,9>, <tipo, early aspect>,<sección, name>
sent	<id,10>, <tipo, early aspect>,<sección, pair>
server	<id,11>, <tipo, early aspect >,<sección, pair>
sending	<id,12>, <tipo, early aspect >,<sección, pair>
data	<id,13>, <tipo, early aspect >,<sección, pair>

Tabla IV.4 - Estado de los tokens luego del filtro Lower Case

IV.2.2.2. Filtro Stop Words

Este filtro elimina las denominadas stop-words, que son palabras que, desde el punto de vista no lingüístico, no contienen información relevante. Algunas de las stop words más comunes son los artículos, las preposiciones, etc. (por ejemplo: a, an, in, the, he, she, them).

Por lo tanto, este filtro elimina de la lista de tokens aquellos en los cuales su palabra aparezca en la lista de stop-words dada. El analista/desarrollador tiene la posibilidad de incluir en esta lista las palabras, que para el dominio en el cuál este analizando, no representen información valiosa para el análisis. De igual modo, se puede modificar la lista quitando aquellas palabras que en el domino resulten relevantes y el desarrollador desee que sean procesadas. Este es uno de los puntos de configuración que presenta *QA Miner*.

La Tabla IV.5 muestra el estado de los tokens de la Tabla IV.4 luego de haberse aplicado este filtro.

Palabra	Atributos
register	<id,1>, <tipo, use case >,<sección,name>
user	<id,2>, <tipo, use case >,<sección, name >
record	<id,3>, <tipo, use case >,<sección,description>
user	<id,4>, <tipo, use case >,<sección, description>
information	<id,5>, <tipo, use case >,<sección, description>
system	<id,8>, <tipo, use case >,<sección, description>
distribution	<id,9>, <tipo, early aspect>,<sección, name>
sent	<id,10>, <tipo, early aspect>,<sección, pair>
server	<id,11>, <tipo, early aspect >,<sección, pair>
sending	<id,12>, <tipo, early aspect >,<sección, pair>
data	<id,13>, <tipo, early aspect >,<sección, pair>

Tabla IV.5 - Lista de tokens pos filtro Stop Words

IV.2.2.3. Filtro Stemming

Diferentes palabras pueden tener el mismo significado desde el punto de vista semántico, pero no tener exactamente la misma secuencia de caracteres. Un ejemplo de esto pueden ser las palabras “aprender”, “aprenden” y “aprendió”, que, desde el enfoque en este trabajo, tienen el mismo significado. Por ello, se decidió utilizar la técnica stemming [36], que es el proceso de transformar una palabra en su raíz (stem). Para el ejemplo anterior, todas esas palabras estarían identificadas por su raíz, es decir, “aprend”. Para realizar el stemming se utiliza uno de los algoritmos más populares, el algoritmo de Porter [37].

Continuando con el ejemplo observado en la Tabla IV.5, se modifican las palabras de los tokens, llevando cada una de ellas a su raíz. En la Tabla IV.6 se muestra el estado de los tokens luego de que se aplicó el filtro de stemming.

Palabra	Atributos
regist	<id,1>, <tipo, use case >,<sección,name>
user	<id,2>, <tipo, use case >,<sección, name >
record	<id,3>, <tipo, use case >,<sección,description>
user	<id,4>, <tipo, use case >,<sección, description>
inform	<id,5>, <tipo, use case >,<sección, description>
system	<id,8>, <tipo, use case >,<sección, description>
distribution	<id,9>, <tipo, early aspect>,<sección, name>
sent	<id,10>, <tipo, early aspect>,<sección, pair>
server	<id,11>, <tipo, early aspect >,<sección, pair>
sending	<id,12>, <tipo, early aspect >,<sección, pair>
data	<id,13>, <tipo, early aspect >,<sección, pair>

Tabla IV.6 - Estado de los tokens luego del filtro Stemming

IV.2.2.4. Filtro Ocurrencias

Este filtro elimina tokens duplicados en la lista de entrada y enriquece cada token con el número de ocurrencias. Se agrega el siguiente atributo al token:

- Ocurrencias: número de ocurrencias del token en la lista. Para los tokens obtenidos de los casos de uso se considera un token duplicado en caso de que el token sea la misma palabra y además aparezca en la misma sección. Esta particularidad se debe a que luego los tokens son ponderados y debe respetarse la sección en la cual aparece a la hora de asignarle su peso. Por lo tanto, existirán tokens con la misma palabra pero diferente sección.

Igualmente, para continuar con el ejemplo, por simplicidad sólo se tendrá en cuenta la palabra del token para que se considere repetido. De esta forma, se observa en la Tabla IV.7 el nuevo atributo que se agrega a cada token.

Palabra	Atributos
regist	<id,1>, <tipo, use case >,<sección, name>, <ocurrencias, 1>
user	<id,2>, <tipo, use case >,<sección, name>, <ocurrencias, 2>
record	<id,3>, <tipo, use case >,<sección,description>, <ocurrencias, 1>
inform	<id,4>, <tipo, use case >,<sección, description>, <ocurrencias, 1>
system	<id,5>, <tipo, use case >,<sección, description>, <ocurrencias, 1>
distribution	<id,9>, <tipo, early aspect>,<sección, name>
sent	<id,10>, <tipo, early aspect>,<sección, pair>
server	<id,11>, <tipo, early aspect >,<sección, pair>
sending	<id,12>, <tipo, early aspect >,<sección, pair>
data	<id,13>, <tipo, early aspect >,<sección, pair>

Tabla IV.7 - Lista de tokens luego de aplicar el filtro Ocurrencias

IV.2.2.5. Filtro Pesos

Este filtro es aplicado para enriquecer los tokens de la lista de tokens de casos de uso, asignando un peso a cada token. Este peso depende de la sección donde aparece la palabra. En la técnica propuesta las palabras tienen distinta importancia según la sección del caso de uso en la que aparezcan. Por ejemplo, se asigna un peso alto a una palabra perteneciente a los flujos alternativos o requerimientos especiales ya que se presume que en ellos se encuentren QAs, mientras que se asigna un peso menor a las secciones actor o prioridad.

Hemos realizado un estudio de la importancia que tiene cada una de las componentes del template que conforman los casos de uso para nuestra problemática, y por defecto se asignan los pesos mostrados en la Tabla IV.8 de acuerdo a la importancia de los textos y palabras que conforman las diferentes secciones. Sin embargo, el desarrollador podría modificar este peso dependiendo de las características de la definición de sus casos de uso. *QA Miner* soporta esta parametrización a través de un archivo de configuración donde se establecen estos pesos.

Considerando los pesos por defecto mostrados en la Tabla IV.8, la lista de tokens final quedaría tal como se muestra en la Tabla IV.9.

Sección	Peso
<Nombre>, <Descripción>,<Prioridad>,<Actor>	1
<Flujo Básico>	2
<Flujo Alternativo>,<Trigger><Requerimientos especiales>,<Precondiciones>,<Postcondiciones>	3

Tabla IV.8 - Ponderación de las secciones de los casos de uso

Palabra	Atributos
regist	<id,1>, <tipo, use case >,<sección, name>, <ocurrencias, 1>,<peso, 5>
user	<id,2>, <tipo, use case >,<sección, name>, <ocurrencias, 2>,<peso, 5>
record	<id,3>, <tipo, use case >,<sección, description>, <ocurrencias, 1>,<peso, 4>
inform	<id,4>, <tipo, use case >,<sección, description>, <ocurrencias, 1>,<peso, 4>
system	<id,5>, <tipo, use case >,<sección, description>, <ocurrencias, 1>,<peso, 4>

Tabla IV.9 - Estado final de los tokens, luego del filtro Pesos

Para el early aspect no es necesario aplicar el filtro de peso debido a que se considera que todas las palabras dentro del early aspect tienen la misma importancia. Sin embargo, el caso de uso está dividido en secciones, y se considera que hay secciones que deben ser ponderadas sobre otras.

IV.3. Token Analysis

En la etapa *Token Analysis* se encuentran las tareas requeridas para identificar los atributos de calidad a partir de las listas de tokens extraídos de la información de entrada. En este punto, la entrada ha sido analizada y se han construido dos listas de tokens. Además, la palabra de cada token ha sido pasada a minúscula, llevada a su raíz (stem), se ha ponderado según su importancia en el contexto y se ha dejado constancia del número de ocurrencias.

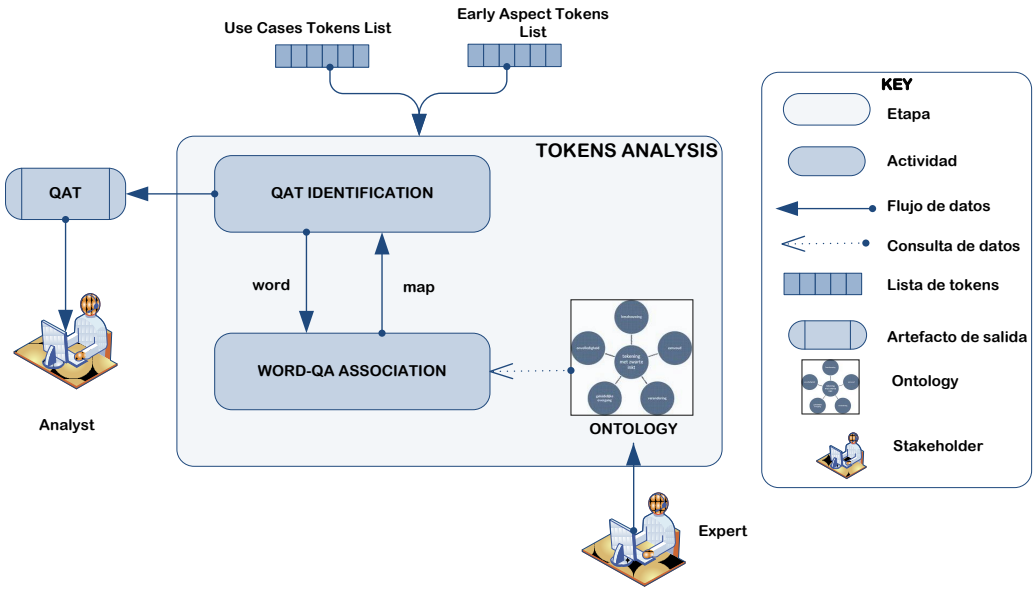

Figura IV.5 - Etapa Token Analysis

Esta etapa (Figura IV.5) se compone de dos actividades principales *QAT Identification* y *Word-QA Association*. La actividad *QAT Identification* se nutre de la información suministrada por la actividad *Tokens-QA Association*, que a su vez se nutre de información de una ontología (Ontology) para su ejecución.

El objetivo de esta etapa es identificar el o los atributos de calidad al que las listas de entrada hacen referencia, para así, finalmente, formar el Quality Attribute Theme.

IV.3.1. QAT Identification

Esta actividad tiene como objetivo asignar un conjunto de atributos de calidad, con un porcentaje asociado a cada uno, al conjunto de la entrada suministrada por la etapa *Token Generation* (Figura IV.6). Como se mencionó anteriormente, la entrada de esta actividad son dos listas de tokens: una extraída de los use cases y otra del early aspect. Cada uno de estos tokens posee diferentes atributos, como sección, peso, ocurrencias, etc. que serán determinantes en la identificación del QAT.

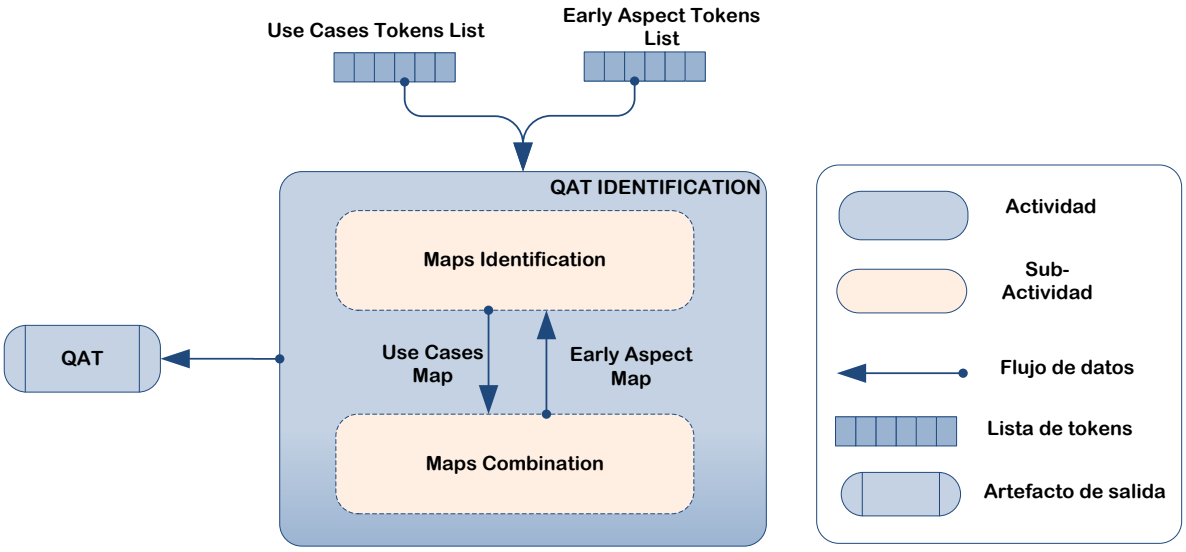

Figura IV.6 - Actividad QAT Identification

La actividad *QAT Identification* está compuesta por dos sub-actividades: *Maps Identification* y *Maps Combination*. La primera, calcula un mapa a partir de cada una de las listas de entrada, mientras que la segunda combina estos dos mapas en un mapa final. Ambas sub-actividades se detallan a continuación.

Para poder llevar a cabo la actividad *QAT Identification* es necesario obtener información mediante la invocación a la actividad *Word-QA Association (*

Figura IV.5*)*. Esta última actividad relaciona un token con un conjunto de pares <atributo de calidad, porcentaje>, indicando, para los distintos atributos de calidad, el porcentaje de asociación de ese token con el mismo. A este conjunto de pares se lo ha denominado con el término "mapa" y además se han definido las operaciones de *suma de dos mapas*, *división de un mapa por un número real* y *multiplicación de un mapa por un número real.* [2]

IV.3.1.1. Maps Identification

Esta sub-actividad tiene como objetivo calcular un mapa de atributos de calidad y porcentajes para cada una de las listas de entrada (Use Case Map y Early Aspect Map). Estas listas son la lista de tokens extraídos de los casos de uso y la lista de tokens extraídos del early aspect.

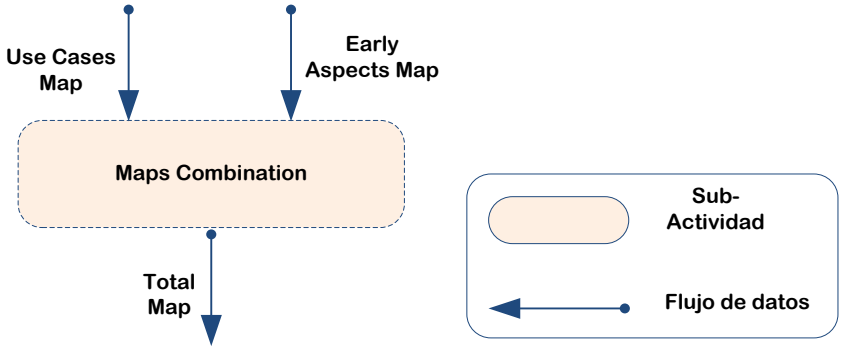

Figura IV.7 - Sub-actividad Maps Identification

A continuación, se muestra el algoritmo que sigue esta sub-actividad para calcular el mapa a partir de los tokens extraídos de una lista. Luego, se muestra el mismo razonamiento, pero expresado en forma matemática y definiendo ecuaciones formales. Finalmente, se presenta un ejemplo del funcionamiento de esta sub-actividad.

IV.3.1.1.1. Descripción algorítmica

La idea subyacente de esta sub-actividad es realizar una iteración sobre cada uno de los tokens de la lista de entrada (use cases list o early aspect list). Por cada uno de los mismos se calcula el porcentaje de asociación del atributo "palabra" con los distintos atributos de calidad. Los resultados parciales de cada token se van acumulando y, cuando finalmente concluye la iteración, se calcula el promedio entre todos los tokens.

[2] En el Anexo I se encuentra una definición formal del concepto "mapa" y de cada una de las operaciones mencionadas anteriormente, que se utilizarán a lo largo de este trabajo.

Para un mejor entendimiento, a continuación se muestra un fragmento de pseudocódigo correspondiente a la sub-actividad “Maps Identification”.

```
1: Mapa <QA,porcentaje> porcentaje_total_map = crear mapa con todos los
atributos de calidad e igualar porcentajes a 0;

2: cantidad_de_palabras = 0;

3: por c/ token de la lista de tokens

4: inicio loop

5: palabra = token.palabra

6:     Mapa<QA,porcentaje>     porcentaje_palabra_map     =     word-qa
association(palabra);

7: n = token.ocurrencias * token.peso;

8: porcentaje_palabra_map = porcentaje_palabra_map * n;

9: cantidad_de_palabras = cantidad_de_palabras + n

10: porcentaje_total_map = porcentaje_total_map + porcentaje_palabra_map

11: final loop

12: porcentaje_total_map = porcentaje_total / cantidad_de_palabras
```

En la primera línea, se declara un mapa de pares <atributo de calidad, porcentaje>. A este mapa se le agregan como pares todos los nombres de los atributos de calidad con los que se trabajará (performance, disponibilidad, modificabilidad, etc.), con los porcentajes de cada uno inicializados en 0.

En la línea 2, se declara la variable *cantidad_de_palabras* en 0. Esta variable cuenta la cantidad de palabras analizadas de los tokens de la lista, siempre y cuando el número de ocurrencias de un token y su peso sean igual a 1. En caso contrario, esta variable aumenta en la magnitud del producto entre el número de ocurrencias del token y el peso del mismo (valor de la variable n). Esta información se extrae de los atributos “peso” y “ocurrencias” pertenecientes a cada token.

En la línea 3, se inicializa una iteración por cada uno de los tokens de una lista de tokens. Por cada uno de éstos se recupera el atributo “palabra”, como se muestra en la línea 5, y se le asocia un conjunto de pares <atributo de calidad, porcentaje>, como se observa en la línea 6. Como se mencionó anteriormente, esta asociación entre la “palabra” del token y un conjunto de atributos de calidad es realizada por la tarea “Word-QA Association”, en adelante denominada *WQA*, que toma como entrada un token (aunque solamente utiliza la palabra del mismo).

En la línea 7 se declara la variable n, que es un número que representa el producto entre los atributos “peso” y “ocurrencias” que posee el token analizado en esa iteración.

En la línea 8 cada porcentaje de cada atributo de calidad del mapa al que se le asocia una palabra se multiplica por el valor de n, que, como se mencionó anteriormente, representa el número de ocurrencias de ese token y el peso del mismo. Se supone que el número de ocurrencias es mayor o igual a uno (si no fuera así el token no aparecería en la lista). El peso también se supone mayor a cero, aunque un peso igual a cero no afectaría al algoritmo, ya que en este caso se interpretaría como que el token se encontraba en una sección que no debe ser tenida en cuenta para el análisis.

Los porcentajes de cada palabra, para cada atributo de calidad, se suman y almacenan en el mapa *porcentaje_total*.

La línea 9 indica que la cantidad de palabras analizadas se va almacenando en la variable *cantidad_de_palabras.* Ya que cada valor del mapa es multiplicado n veces, la cantidad de palabras no aumenta necesariamente en una unidad por cada iteración, sino que aumenta n unidades.

En la línea 10 se suman los resultados parciales de `porcentaje_palabra_map` a los totales acumulados en `porcentaje_total_map`.

Una vez terminadas las iteraciones, en la línea 12, se divide cada porcentaje de cada atributo de calidad del mapa *porcentaje_total_map* por el valor de la variable cantidad_de_palabras.

IV.3.1.1.2. Descripción matemática

La metodología de la sub-actividad *Maps Identification* también puede ser detallada matemáticamente. De esta manera, sea UCL[] un arreglo compuesto por tokens extraídos, por ejemplo, de los casos de uso.

Como se ha mencionado, cada token tiene varias propiedades, y en las siguientes ecuaciones se denomina como *token*.palabra al conjunto de caracteres del token que representan la palabra, *token*.ocurrencias a la propiedad que indica el número de ocurrencias de ese token y *token*.peso al valor o peso que se le asigna al mismo.

Además, sea i un número entero, UCL[i] se refiere al token en la posición i del arreglo UCL. Por último, se define como “WQA” las siglas que denominan a la tarea “Word-QA Association”. Entonces se calcula el mapa porcentaje_total_map de la siguiente manera:

$$porcentaje\_total\_map = \frac{\sum_{i=1}^{n} WQA\left(UCL_{[i]}.palabra\right) * UCL_{[i]}.ocurrencias * UCL_{[i]}.peso}{\sum_{i=1}^{n} UCL_{[i]}.ocurrencias * UCL_{[i]}.peso}$$

Esta ecuación muestra lo mismo que el fragmento de pseudocódigo explicado en la subsección anterior. Por cada palabra de cada token se invoca la actividad WQA y los resultados obtenidos son multiplicados por el peso y las ocurrencias del token. A su vez, el producto entre estos dos valores se va sumando para cada token, como se muestra en el dividendo de la ecuación.

Para la lista de tokens extraídos del early aspect se modifica ligeramente la ecuación anterior. En este caso, el peso del token siempre es igual a 1, ya que, como se explicó anteriormente, el peso de los tokens sólo se aplica a los extraídos de los casos de uso y depende de la sección en donde se encuentren. De esta manera, para el caso de la lista de tokens extraídos del early aspect, la ecuación anterior se puede ver como:

$$porcentaje\_total\_map = \frac{\sum_{i=1}^{n} WQA\left(EAL_{[i]}.palabra\right) * EAL_{[i]}.ocurrencias}{\sum_{i=1}^{n} EAL_{[i]}.ocurrencias}$$

IV.3.1.1.3. Ejemplo

A continuación, se verá un ejemplo de lo explicado anteriormente. En el mismo se supone una lista de sólo dos tokens extraídos de los casos de uso (Tabla IV.10). El primero es un token con la palabra “fast”, tiene un peso de 1 y un número de ocurrencias también de 1. El segundo token de la lista posee la palabra “minute”, con un valor de 2 para el número de ocurrencias y un valor de 3 para el peso.

Palabra	Atributos
fast	<id,1>, <tipo, use case>,<sección,description>,<ocurrencias,1>,<peso,1>
minute	<id,2>, <tipo, use case >,<sección, name>, <ocurrencias,2>, <peso,3>

Tabla IV.10 - Lista de dos tokens

A fin de que se aprecie mejor el ejemplo, las palabras de los tokens aparecen sin la transformación generada por el filtro de stemming.

Supóngase que se tiene sólo los atributos de calidad Modifiability, Performance y Availability. Considerando el pseudocódigo presentado en la subsección IV.3.1.1.1, la línea 1 inicializa un mapa con estos tres atributos en los pares, cada uno con valores en 0. En la línea 2 se inicializa el contador de palabras en 0. De esta manera, después de las inicializaciones, las variables quedan en el siguiente estado:

porcentaje_total_map = (<Modifiability,0>; < Availability*,0>; <Performance,0>).*

cantidad_de_palabras = 0;

En este punto comienza la iteración sobre los tokens de lista. El primer token contiene la palabra “fast”. Supóngase que la actividad *WQA* relaciona, para esa palabra, el siguiente mapa:

WQA(“fast”) = (<Modifiability,0.25>; < Availability*,0.0>; <Performance,0.75>)*

Este resultado significa que la actividad asigna a la palabra “fast” una relación de 0.25 o 25% con el atributo de calidad “Modifiability”, un 0% con “Availability” y un 75% con el atributo “Performance”.

La línea 7 indica que la variable n es igual al producto entre el peso del token multiplicado por el número de ocurrencias. Por lo que el valor de n es igual a 1 en la primera iteración.

Los valores de *porcentaje_total_map* no varían luego del producto de este por n, y la variable cantidad_de_palabras aumenta en 1, ya que este es el valor de n.

De esta manera, el estado de las variables luego de la primera iteración es el siguiente:

porcentaje_total_map = (<Modifiability,0.25>;< Availability*,0.0>;<Performance,0.75>).*

cantidad_de_palabras = 1;

Distinto es el caso del segundo token, que posee la palabra “minute”, con un peso de 3 y un número de ocurrencias igual a 2.

Supóngase que en este caso el mapa asociado para la palabra posee un valor de 0.25 para Modifiability, 0.50 para Availability y 0.25 para Performance. Estos valores deben ser multiplicados por n=6 (ya que este es el valor del producto entre el peso y el número de ocurrencias). Además, la cantidad de palabras también aumenta 6 unidades durante la segunda iteración. De esta manera, el estado de las variables luego de la segunda iteración es el siguiente:

porcentaje_total_map = (<Modifiability,1.75>;< Availability*,3.0>;<Performance,2.25>).*

cantidad_de_palabras = 7;

Luego de esta iteración no hay más tokens, por los que los valores del mapa *porcentaje_total_map* deben ser divididos por *cantidad_de_palabras*, dando como resultado el siguiente mapa:

porcentaje_total_map = (<Modifiability,0.25>;< Availability*,0.43>;<Performance,0.32>)*

La interpretación de este resultado es que la lista de tokens se relaciona en un 25% con Modifiability, un 43% con Availability y un 32 % con Performance.

Como se puede notar, el peso y el número de ocurrencias del segundo token, que en su conjunto son seis veces mayores que el primero, hacen inclinar la balanza hacia Availability, que es el atributo de calidad con el que más se relaciona la palabra “minute”.

IV.3.1.2. Maps Combination

En la sub-actividad precedente se calcularon dos mapas, uno proveniente de la lista de tokens extraídos de los use cases y otro a partir de la lista de tokens extraídos del early aspect. De esta forma, la función de esta sub-actividad es combinar estos dos mapas para formar el mapa resultante que será parte del QAT.

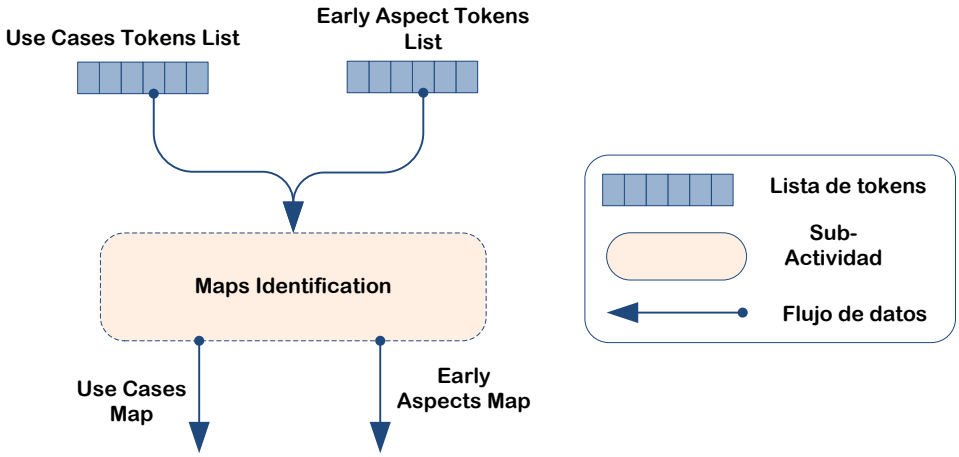

Figura IV.8 - Sub-actividad Maps Combination

La solución propuesta en este trabajo combina los resultados parciales de ambos mapas a partir de un factor K. Este factor es un número real entre 0 y 1 e indica la importancia que se le debe dar a cada uno de los mapas de entrada. A su vez, *QA Miner* permite que el analista sea capaz de ingresar este valor según sus preferencias.

Cada valor de cada atributo de calidad del mapa de la lista de tokens del caso de uso (use cases map) se multiplica por K, mientras que cada valor de la lista del mapa de tokens del early aspect (early aspect map) se multiplica por 1-K. Luego, para cada atributo de calidad de cada mapa, se suman los valores y se almacenan en el mapa resultante.

Supóngase, por ejemplo, que el mapa A= (<Modifiability,0.25>; < Availability,0.25>; <Performance,0.50>) es el relacionado con la lista de los casos de uso y el mapa B= (<Modifiability,0.10>; <Availability,0.30>; <Performance,0.60>) es el relacionado con el early aspect. Eligiendo un valor de K=0.4 se tendría que multiplicar cada valor de A por 0.4 y cada valor de B por 1 - 0.4 = 0.6, para luego sumar, para cada atributo de calidad, los valores de ambas listas. Esto daría como resultado un mapa C= (<Modifiability ,0.16>; <Availability, 0.28>; <Performance,0.56>).

Nótese que eligiendo un K=1 el mapa resultante es el mismo que el mapa que se relaciona con los casos de uso (en el caso anterior el mapa resultante, C, sería igual al A). Sin embargo, con un K=0, el mapa resultante es el mismo que el mapa identificado a partir de la lista de tokens del early aspect. Un valor de K=0,5 implica que cada mapa aporta la misma proporción al mapa resultante.

Para describir está sub-actividad de forma matemática supóngase a useCaseMap como el mapa calculado a partir de los casos de uso y earlyAspectMap como el mapa calculado a partir del early aspect. Entonces el mapa resultante, totalMap, es igual a

$$totalMap = K * \text{useCaseMap} + (1 - K) * \text{earlyAspectMap}$$

donde K es un número real entre 0 y 1.

IV.3.2. Word-QA Association

Esta actividad relaciona una secuencia de caracteres que representa una palabra con un mapa de atributos de calidad y porcentajes. Cada porcentaje indica el grado de asociación o pertenencia que tiene esa palabra con cada atributo de calidad.

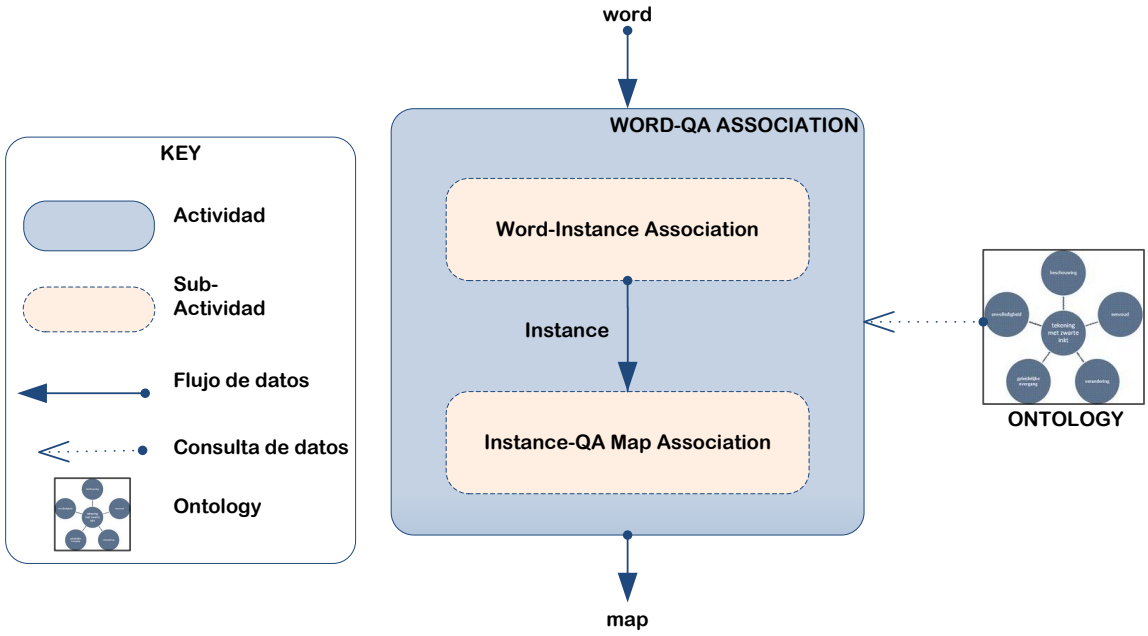

Figura IV.9 - Sub-actividad Word-QA Association

Básicamente, esta actividad consta de dos sub-actividades. La primera es averiguar a qué parte del escenario de calidad corresponde la palabra. Es decir, descubrir si corresponde a una "Fuente de estímulo", a un "Estímulo" o a alguna otra. Este comportamiento esta agrupado en la sub-actividad *Word-Instance Association*. Una vez reconocida la instancia a la que se corresponde la palabra, se averigua cuántos y cuáles escenarios se relacionan con esa instancia. A partir de esto último, y de calcular los porcentajes en que ese grupo de escenarios se relacionan con los atributos de calidad, se obtiene un mapa de atributos de calidad y porcentajes. Este comportamiento se agrupa en la sub-actividad *Instance-QA Map Association*.

Ambas sub-actividades consultan las instancias de una ontología definida para el dominio en cuestión. Esta ontología se considera que ha sido modelada por un experto. Es por ello que a continuación ésta es detallada primero, para luego describir las sub-actividades que componen a esta actividad.

IV.3.2.1. Ontología

La literatura contiene varias definiciones de ontologías, muchas veces contradictorias entre ellas. En este trabajo se acuerda en definir a una ontología de la siguiente manera:

"Una ontología es un modelo de datos que describe conceptos (también llamados clases) en un dominio del discurso, propiedades de cada concepto que describen las diversas características y atributos del concepto, y restricciones sobre esas propiedades." [7].

Un dominio es un área de temática específica o de conocimiento, tal como medicina, fabricación de herramientas, bienes inmuebles, reparación automovilística, gestión financiera, etc. [38]. Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos. También, codifican el conocimiento de un dominio específico y el conocimiento que extiende de los dominios.

Un dominio específico es la parte del mundo que se quiere modelar. Representa el significado aplicado a los términos usados en la construcción de la ontología. Una ontología es la descripción de los conceptos que forman parte del dominio según un punto de vista. Un sistema sólo conoce lo que puede representar en algún lenguaje, por lo tanto, todo lo que no se exprese en la ontología no será conocido para el sistema que use la ontología.

IV.3.2.1.1. Componentes de una ontología

Las ontologías cuentan con los siguientes componentes que sirven para representar el conocimiento de algún dominio [39]. Los principales son:

1. **Conceptos o Clases:** generalmente, las clases son las ideas básicas que se intentan formalizar. Suelen ser el componente principal de cualquier ontología. Por ejemplo, en una ontología de vinos, la clase “Vino” representa todos los vinos. Una clase también puede tener sub-clases que representan conceptos que son más específicos que las superclases. Por ejemplo, subclases de la clase “Vino” podrían ser “Tinto”, “Blanco”, “Rosado”, etc.

2. **Instancias:** las instancias son representaciones de objetos determinados de un concepto. Siguiendo el ejemplo anterior, vinos específicos son instancias de la clase “Vino”. Una botella de New Age que se vende en el supermercado es una instancia de la clase “Vino”.

3. **Propiedades o Slots:** los slots describen propiedades de clases o instancias. En el ejemplo anterior “New Age” es la marca del vino. Por lo tanto, esa instancia posee un slot (propiedad), que poseen todas las instancias de la clase “Vino”, que podría denominarse “marca”, y cuyo valor es la cadena de caracteres “New Age”. El valor de una propiedad puede ser una cadena de caracteres, un número, otra instancia, etc. En el caso de que fuera otra instancia se forma lo que se denomina una relación.

IV.3.2.1.2. Ontología para QAs

En la Figura IV.10 se muestra un diagrama con la ontología que servirá de soporte de las sub-actividades siguientes. Su construcción se basa principalmente en conceptos, definiciones, y relaciones extraídas del libro “Software Architecture in Practice” [1] y explicadas en el Capítulo II.

Esta ontología, definida para el enfoque propuesto, representa el dominio de atributos y escenarios de calidad sobre el cual se está investigando. La técnica propuesta se basa en esta ontología para consultar el grado de relación de las palabras de cada token con los atributos de calidad.

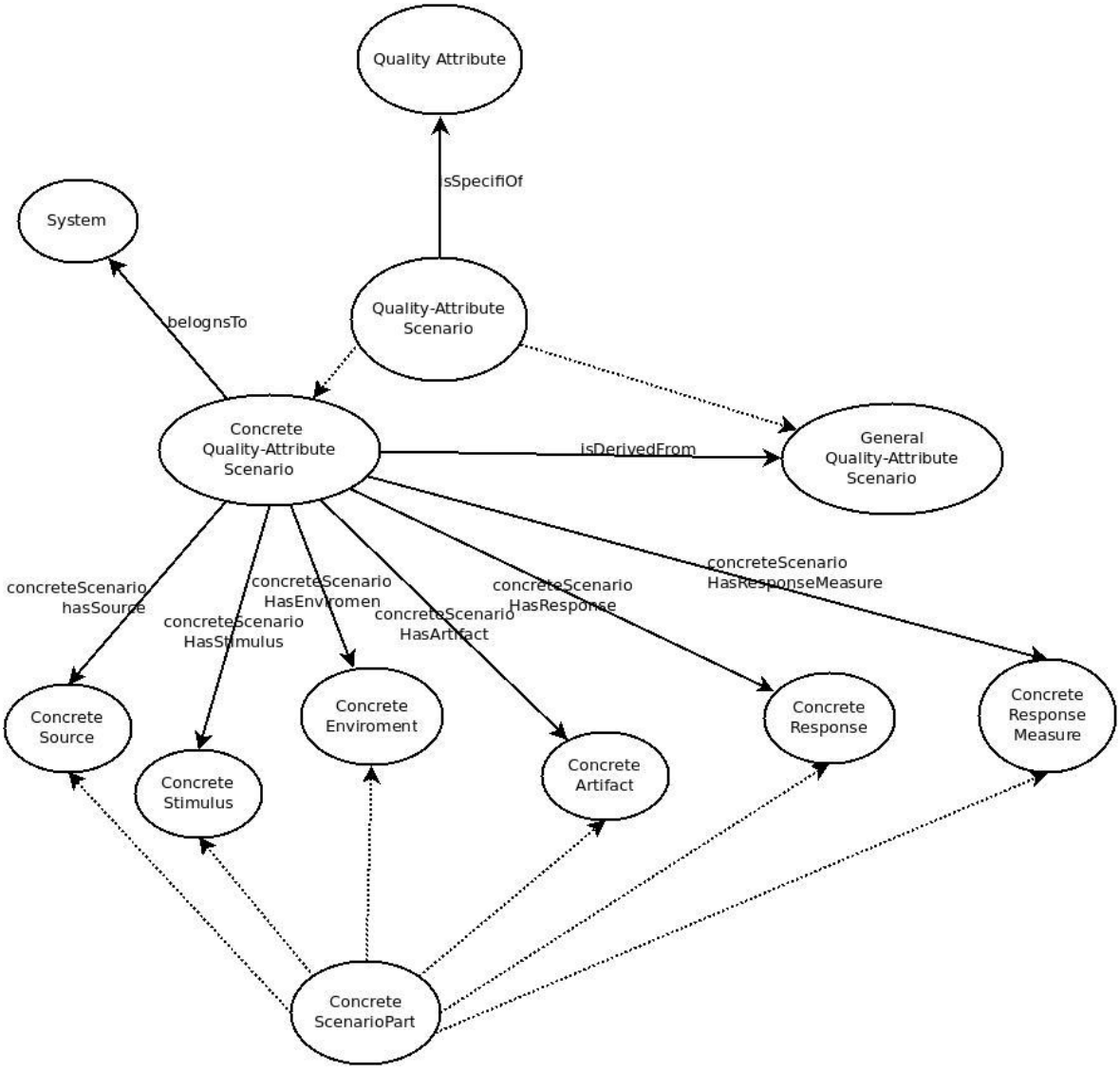

Figura IV.10 - Ontología para QAs

Se asume que las instancias de la ontología están creadas por un experto en el dominio. De esta manera, lo que la ontología incluye es la realidad del domino que se desea modelar, permitiendo un entendimiento común de todos los conceptos.

Como se puede apreciar en la Figura IV.10, aparecen los conceptos de *Quality-Attribute Scenario*, que representa a los escenarios de calidad, y *Concrete Quality-Attribute Scenario* y *General Quality-Attribute Scenario* que representan a los escenarios concretos y generales, respectivamente.

Los atributos de calidad se representan mediante el concepto *Quality Attribute* y se relacionan con el concepto de *Quality-Attribute Scenario* mediante la propiedad *isSpecificOf*.

A su vez, las partes de un escenario se ven reflejadas en los conceptos *Concrete Source*, *Concrete Stimulus*, *Concrete Enviroment*, *Concrete Artifact*, *Concrete Response* y *Concrete Response Measure*. Todos estos son conceptos hijos del concepto *Concrete Scenario Part* y se relacionan con el concepto de *Quality-Attribute Scenario*.

IV.3.2.2. Word-Instance *Association.*

En esta sub-actividad se busca identificar la correspondencia entre una palabra y alguna instancia de los conceptos de ConcreteSource (Fuente), ConcreteStimulus (Estímulo), ConcreteEnviroment (Ambiente), ConcreteArtifact (Artefacto), ConcreteResponse (Respuesta) y ConcreteResponseMeasure (Medida de Respuesta) de la ontología. Básicamente, a partir de una palabra dada se intenta responder a la pregunta, ¿la palabra es una “Fuente”, un “Estímulo”, un “Ambiente”, un “Artefacto”, una “Respuesta” o una “Medida de Respuesta”?

Esta tarea no es trivial. El caso ideal es que exista una sola instancia, de alguna de esas seis, que se corresponda con la palabra.

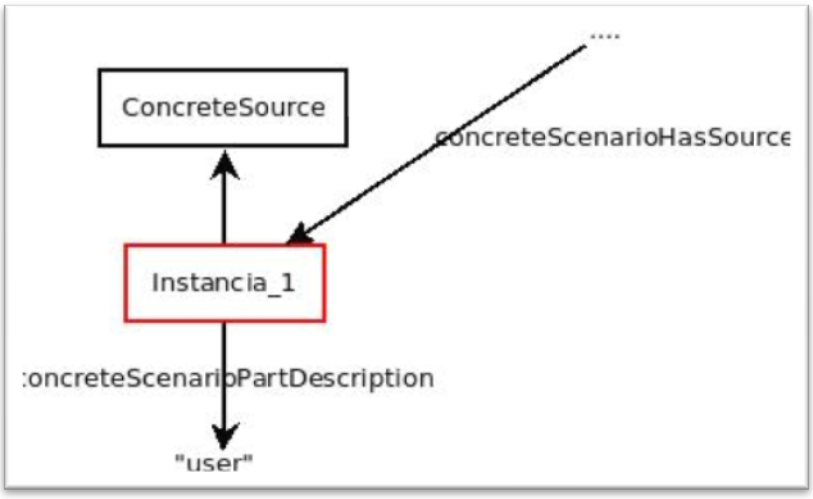

Figura IV.11 - Una única instancia se corresponde con la palabra "user"

La Figura IV.11 muestra lo afirmado anteriormente. En ella se observa en el recuadro superior una clase ("ConcreteSource"), mientras que en el recuadro inferior se representa la instancia ("Instancia_1"). A su vez, la instancia posee una propiedad (de tipo "cadena de caracteres") que se denomina "concreteScenarioPartDescription" y cuyo valor, en este caso, es "user". Uno o más escenario se pueden relacionar con esta instancia mediante la propiedad "concreteScenariohasSource".

Se pretende identificar la palabra "user". En este caso hay una sola instancia en la ontología, cuyo identificador único de instancia es "Instancia_1", de tipo ConcreteSource, que mediante la propiedad "concreteScenarioPartDescription" posee el valor "user", por lo que se retorna esta instancia.

Distinto es el caso en el que la misma palabra pueda representar, simultáneamente, distintas partes de un escenario (Figura IV.12).

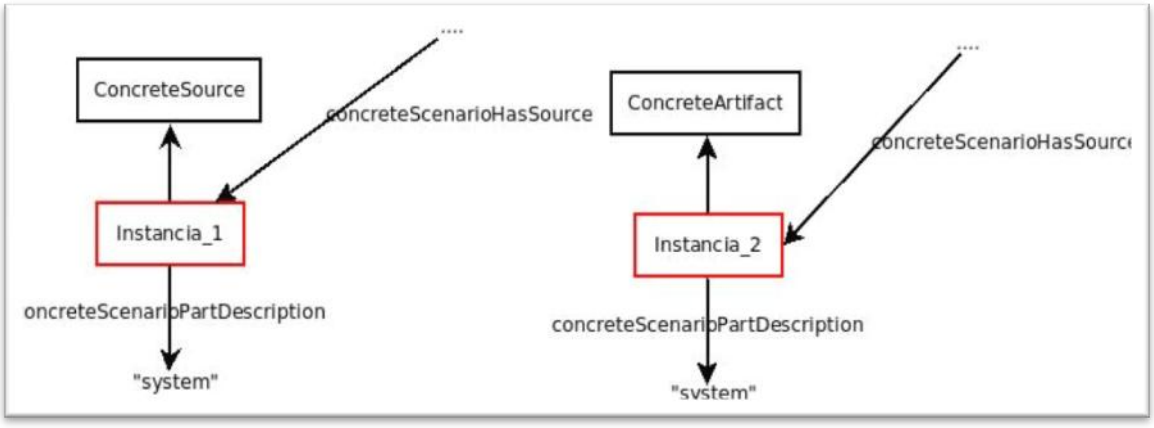

Figura IV.12 - Dos instancias, de diferente tipo, se corresponden con la palabra "system"

En este caso la palabra “system” podría ser un ConcreteSource o un ConcreteArtifact. Lo que lleva a la cuestión de decidir entre ambas opciones. Para ello se contabiliza con cuántos escenarios se relacionan cada una de las instancias. Será elegida como significado semántico de la palabra la instancia que se relacione con un mayor número de escenarios. En este último caso, si la “instancia_1” se relaciona con 3 escenarios y la “instancia_2” con 6, se retorna la “instancia_2”, tomándose como que la palabra “system” es un ConcreteArtifact. En el caso de que sean iguales se elige, indistintamente, cualquiera de las instancias.

En este punto cabe aclarar que no existen dos instancias que se relacionen con la misma palabra (mediante la propiedad “concreteScenraioPartDescription”) y que sea del mismo tipo (ConcreteScource, ConcreteArtifact, etc.) Es decir, por ejemplo, no puede haber dos instancias en las que ambas sean estímulos y además se relacionen con la palabra “developer”. Si esto ocurriera, se pasan todas las relaciones de una instancia a la otra, y se elimina la primera. Este es un proceso que se realiza al crear la ontología. Es por ello que en la descripción anterior, con devolver la instancia ya se asegura que es única para esa parte de escenario.

IV.3.2.3. Instance-QA Map Association

Una vez que se ha identificado la instancia de alguna parte de un escenario que se corresponde con una palabra, se dispone a obtener los porcentajes de los distintos atributos de calidad que se relacionan con esa instancia. Los atributos de calidad, junto con los porcentajes asociados, se agrupan en un mapa.

Para calcular estos porcentajes, primero se recuperan todos los escenarios que se relacionan con la instancia encontrada en el paso anterior. Cada uno de estos escenarios se relaciona con un atributo de calidad mediante la propiedad “isSpecificOf”. De esta manera, para cada atributo de calidad, se devuelven los porcentajes en que estos escenarios se relacionan con cada uno de ellos.

Por ejemplo, supóngase que como entrada se obtuvo el término "latency". Después del paso anterior, se encuentra que esa palabra se corresponde con la “Instancia_48” que es de tipo ResponseMeasure. A su vez, 40 escenarios se relacionan con la “Instancia_48” mediante la propiedad “scenarioHasResponseMeasure”. De esos 40 escenarios, 30 de ellos se relacionan con el atributo de calidad "Performance", 8 con " Availability " y 2 con "Modifiability". De esta manera, es posible concluir que ese concepto, “latency”, se relaciona en un 75% de certeza con "Performance", 20% con " Availability " y 5% con "Modifiability". En este caso, esta sub-actividad relacionaría la palabra “latency” con <Performance 0.75, Availability 0.2, Modifiability 0.05>.

Capítulo V - Evaluación de la técnica propuesta

En este capítulo se realiza un análisis de la técnica propuesta para la identificación de atributos de calidad. Esta evaluación será llevada a cabo con dos casos de estudios reales, los cuales serán explicados brevemente y se proveerá su especificación, como así también los aspectos tempranos detectados.

V.1. Métricas a utilizar

Antes de comenzar con el análisis de cada caso de estudio en particular, en esta sección se establecen métricas para los resultados obtenidos a través de la herramienta *QA Miner*. Se proponen dos tipos de métricas: el lapso de tiempo para efectuar la identificación, y métricas provenientes de la rama de *Recuperación de Información*. Estas últimas se pueden entender desde dos puntos de vista: como un escenario de recuperación de información o como una tarea de clasificación estadística. Estos son escenarios análogos, y serán explicados para cada métrica.

Las métricas utilizadas para evaluar el desempeño de los algoritmos son *Precision* y *Recall*. *Precision* puede ser interpretada como una métrica de exactitud o fidelidad, mientras que *Recall* es una métrica de completitud.

Desde el punto de vista de la recuperación de información: una *Precision* perfecta sería 1.0, y significa que cada resultado recuperado por una búsqueda fue relevante (pero no dice nada acerca de si todos los documentos relevantes fueron recuperados). En contraste, un *Recall* perfecto con 1.0, significa que todos los documentos relevantes fueron recuperados por la búsqueda (pero no dice nada acerca de cuantos documentos irrelevantes también fueron recuperados). Para realizar el análisis, se utilizan los conceptos de *Relevant Documents* y *Documents Retrieved.* El primero representa la cantidad de documentos relevantes que hay sobre una búsqueda dada, y el segundo concepto representa aquellos documentos que fueron recuperados en una búsqueda.

$$Precision = \frac{|\{Relevant\ Documents\} \cap \{Documents\ Retrieved\}|}{|\{Documents\ Retrieved\}|}$$

$$Recall = \frac{|\{Relevant\ Documents\} \cap \{Documents\ Retrieved\}|}{|\{Relevant\ Documents\}|}$$

Análogamente, desde el punto de vista de la clasificación estadística se definen los conceptos de *True Positives*, *True Negatives*, *False Positives y False Negatives*. *True Positives* representa el número de ítems identificados como pertenecientes a la clase que efectivamente pertenecen a esa clase. *True Negatives* representa el número de ítems no etiquetados como pertenecientes a la clase que efectivamente no pertenecen a esa clase. *False Positives* representa el número de ítems etiquetados como pertenecientes a la clase que efectivamente no pertenecen a esa clase. Por último, *False Negatives* representa el número de ítems no etiquetados como pertenecientes a la clase que efectivamente pertenecen a esa clase. En este contexto, *Precision* de una clase es el número de *True Positives* dividido por el número total de elementos etiquetados como pertenecientes a la clase. *Recall* en este contexto es definido como el número de *True Positives* dividido el número total de elementos que actualmente pertenecen a la clase. Las formulas de Precision y Recall son las siguientes:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

V.1.1. Falsos y verdaderos, positivos y negativos

Para nuestro análisis, se utilizarán las siguientes definiciones, basándonos en las establecidas previamente:

QA Verdaderos Positivos (QVP)	Son aquellos QAs **identificados** por *QA Miner*, que **son realmente** QAs del sistema.
QA Verdaderos Negativos(QVN)	Son aquellos QAs **no identificados** por *QA Miner*, que **no son realmente** QAs del sistema.
QA Falsos Positivos (QFP)	Son aquellos QAs **identificados** por *QA Miner*, que **no son realmente** QAs del sistema o que, desde un punto de vista semántico, no se corresponde con el EA de origen.
QA Falsos Negativos (QFN)	Son aquellos QAs **no identificados** por *QA Miner*, que **son realmente** QAs del sistema.
QA Verdaderos (QV)	Son QAs que **son realmente** atributos de calidad del sistema. $QV = QVP + QFN$

Los QAs en los casos de estudio se corresponden con la definición de la siguiente manera: los QVP serian “los QAs correctos” y los QFP serían los “QAs incorrectos y mal identificados”. La cantidad total de los QV se estableció realizando inspecciones manuales de las especificaciones.

También en algunos casos se analizaron arquitecturas propuestas por distintos autores, [40] y [9], para así extraer de las mismas los atributos de calidad que los diseñadores tuvieron en cuenta a la hora de modelarlas. Estos valores se determinaron para cada caso de estudio. Notar que los QVN no son considerados porque no tiene sentido en este tipo de evaluación.

Utilizando la nomenclatura utilizada anteriormente, las formulas para calcular estas métricas se reescriben de esta manera:

$$Precision = \frac{QVP}{QVP + QFP}$$

$$Recall = \frac{QVP}{QVP + QFN}$$

Se debe resaltar que según nuestra opinión, en el caso particular de la identificación de atributos de calidad presentada en este trabajo, el *Recall* es la métrica más importante de las dos. En general, los trabajos relacionados también se enfocan principalmente en esta métrica, ya que los costos generados por la omisión de un QA relevante pueden llegar a ser muy costosos. Sin embargo, cabe destacar que la técnica basa su identificación de QAs en aspectos previamente detectados por otra herramienta, con lo cual los QA identificados están limitados por la salida de dicha herramienta. Es decir, que para hallar todos los QAs del sistema (y así lograr un *recall* igual a 1.0) todos los QAs deberían estar relacionados con un aspecto, cosa que no tiene por qué cumplirse obligatoriamente en todos los casos.

V.2. Casos de estudio

Se tratará individualmente cada uno de los casos de estudio que fueron seleccionados para la evaluación. En primera instancia, se explicará brevemente el sistema en cuestión, además de mostrar los casos de uso que conforman sus requerimientos. En segunda instancia, se indicarán los aspectos tempranos detectados utilizando la herramienta Aspect Extractor Tool [8]. También, se mostrará por cada aspecto temprano, la lista de casos de uso que éste atraviesa. En tercera instancia, se realizará una serie de pruebas, que constituyen la parte central de la evaluación. Por cada aspecto temprano, se correrá la herramienta *QA Miner* para identificar un conjunto de QAs. Luego, este conjunto se comparará con los atributos de calidad del sistema. A partir de esta comparación se calcularán ciertas métricas que serán de utilidad para evaluar la técnica propuesta.

V.2.1. HWS (Health Watcher System)

El objetivo del Sistema de Salud Vigía (HWS) [41] consiste en recoger y gestionar las quejas y notificaciones relacionadas con la salud pública. El sistema también se utiliza para notificar a las personas información importante con respecto al Sistema de Salud.

Un ciudadano accede al sistema a través de Internet para hacer su denuncia o solicitar información sobre los servicios de salud. En el caso de que se haya hecho una queja, ésta se registrará en el sistema y será dirigida a un departamento específico. Este departamento será capaz de manejar la demanda en una manera apropiada y devolver una respuesta luego de que el reclamo haya sido tratado. Esta respuesta se registrará en el sistema y estará disponible para ser consultada.

Con el desarrollo del sistema de salud de vigía, el Sistema Público de Salud pretende mejorar considerablemente:

- El control de quejas (registro y notificaciones).
- La calidad del servicio en relación a la diseminación de la información; por ejemplo: las campañas de vacunación, prevención de enfermedades, guías de salud, obtener certificados de nacimiento y defunción y detalles de las aplicaciones para licencias sanitarias.

V.2.1.1. Casos de Uso

El presente caso de estudio está formado por nueve casos de uso. Debido a la extensión de los mismos, estos se pueden consultar en el Anexo II. Sin embargo, a continuación se menciona a cada uno, aportando una breve descripción de los mismos. Adicionalmente, se deja constancia de la cantidad de palabras que conforman cada caso de uso.

Nombre	[FR01] Query information
Descripción	This use case allows a citizen to perform queries.
Cantidad de palabras	776

Tabla V.1 - Breve descripción del caso de uso "Query information"

Nombre	[FR02] Complaint Specification
Descripción	This use case allows a citizen to register complaints. Complaints can be animal complaint, food complaint, special complaint, animal complaint or special complaint.
Cantidad de palabras	525

Tabla V.2 - Breve descripción del caso de uso "Complaint specification"

Nombre	[FR10] Login
Descripción	This use case allows an employee to have access to restricted operations on the Health-Watcher System
Cantidad de palabras	117

Tabla V.3 - Breve descripción del caso de uso "Login"

Nombre	[FR11] Register Tables
Descripción	This use case allows the registration of system tables. The following operations are possible: insert, update, delete, search and print.
Cantidad de palabras	116

Tabla V.4 - Breve descripción del caso de uso "Register Tables"

Nombre	[FR12] Update complaint
Descripción	This use case allows the state of a complaint to be updated.
Cantidad de palabras	280

Tabla V.5 - Breve descripción del caso de uso "Update complaint"

Nombre	[FR13] Register new employee
Descripción	This use case allows new employees to be registered on the system.
Cantidad de palabras	185

Tabla V.6 - Breve descripción del caso de uso "Register new employee"

Nombre	[FR14] Update employee
Descripción	This use case allows of the employee's data to be updated on the system.
Cantidad de palabras	117

Tabla V.7 - Breve descripción del caso de uso "Update employee"

Nombre	[FR15] Update health unit
Descripción	This use case allows the health unit's data to be updated.
Cantidad de palabras	245

Tabla V.8 - Breve descripción del caso de uso "Update health unit"

Nombre	[FR16] Change logged employee
Descripción	This use case allows the currently logged employee to be changed.
Cantidad de palabras	59

Tabla V.9 - Change logged employee

V.2.1.2. Atributos de calidad del sistema HWS

La Tabla V.10 muestra los atributos de calidad del sistema HWS, junto con una breve descripción de cómo estos se deberían manifestar mediante escenarios. Esta tabla fue armada por nosotros, teniendo en cuenta las especificaciones de requerimientos del sistema HWS detalladas en [41]. A su vez, se tuvieron en cuenta los trabajos de Zhang y otros [9] (en adelante denominada "arquitectura 1") y de Pinto y otros [40] (en adelante denominado "arquitectura 2"). En ambos trabajos se proponen diseños arquitectónicos basados en aspectos (*aspectual architectures*) para el sistema HWS. Los módulos de estas arquitecturas tienen ciertas responsabilidades o implementan mecanismos arquitectónicos relacionados con ciertos QAs. En base a ello, se puede deducir qué atributos de calidad han sido considerados en cada arquitectura del HWS.

Atributo de calidad	Descripción
Usability	The system should have an easy to use GUI, as any person who has access to the internet should be able to use the system. The system should have an on-line HELP to be consulted by any person that uses it.
Availability	The system should be available 24 hours a day, 7 days a week. The nature of the system not being a critical system, the system might stay off until any fault is fixed. A serve or a processor can fail, usually in various ways, and must be built reliable using internal redundancy so that the register service remains available.
Performance	The system must be capable to handle 20 simultaneous users. The response time must not exceed 5 seconds.
Scalability / Distribution	HW system would be used by citizens from different area, and it must be designed to cater for variations The system should be capable of running on separate machines.
Security	To have access to the complaint registration features, access must be allowed by the access control sub-system. (Authentication) The system should store confidential information using some kind of encryption.
Persistence/ Data consistency	Data is usually constructed and shared by citizen distributed in different locations. Therefore a major issue is the preserve the consistency of data in the concurrency and failure.

Tabla V.10 - Atributos de calidad del HWS

En la Figura V.1 y la Figura V.2 se muestran las vistas arquitectónicas de componentes y conectores (incluyendo aspectos) propuestas por los dos trabajos antes mencionados.

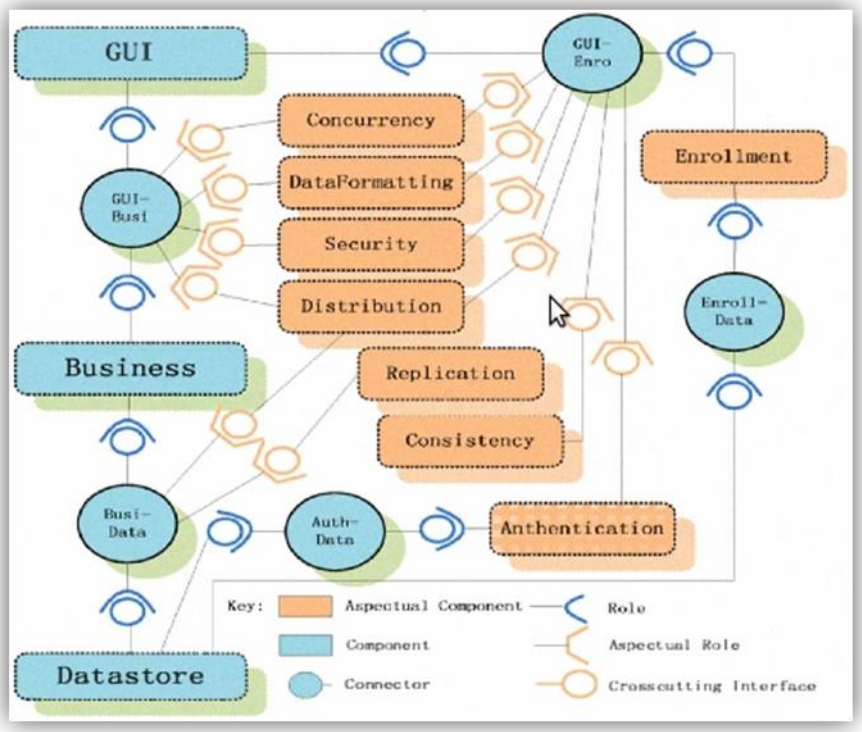

Figura V.1 - Arquitectura propuesta por Zhang y otros (arquitectura 1)

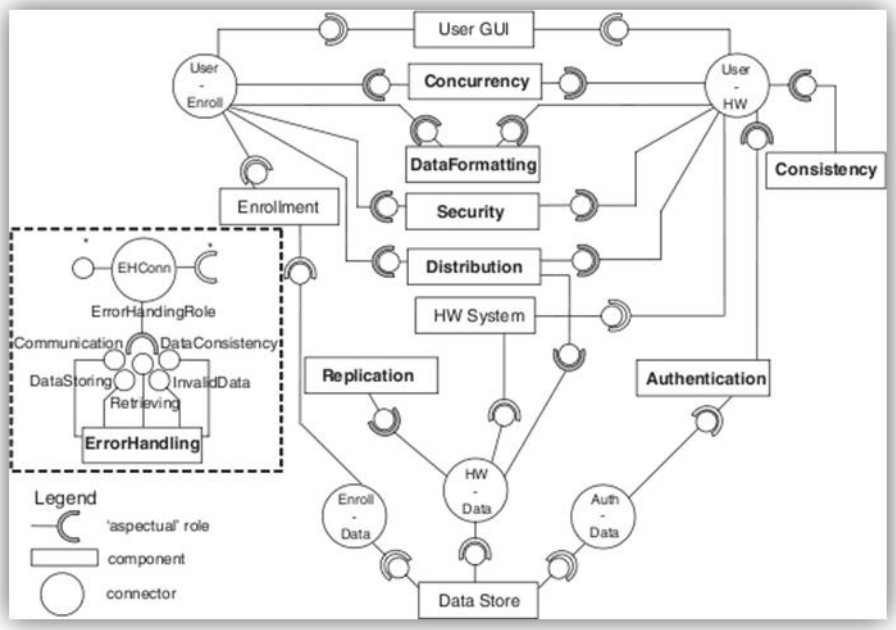

Figura V.2 - Arquitectura propuesta por Pinto y otros (arquitectura 2)

A continuación se muestra el razonamiento efectuado del por qué fue agregado cada atributo de calidad en la Tabla V.10 y, por la tanto, se considera un QA del sistema HWS.

Usability

El atributo de calidad *Usability* se expresa en varios requerimientos del SRS. Básicamente, se hace referencia a que la interfaz de usuario debe ser sencilla y fácil de usar para cualquier persona.

Los módulos "*GUI*" y "*Data Formatting*" de la arquitectura 1 (Figura V.3) y los módulos "*User GUI*" y "*Data Formatting*" de la arquitectura 2 (Figura V.4) muestran ciertas responsabilidades relacionadas con usabilidad.

En el módulo "GUI" ("Graphical User" Interface) de la arquitectura 1 se provee la interfaz de usuario del sistema HWS. Adicionalmente, el módulo "Data Formatting" es invocado cada vez que la información es mostrada al usuario.

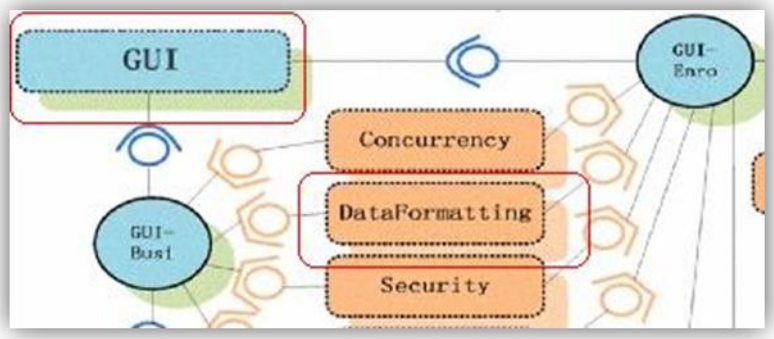

Figura V.3 - Módulos "GUI" y "Data Formatting" de la arquitectura 1 (Zhang)

La arquitectura 2 también cuenta con un módulo denominado "Data Formatting" que es invocado cada vez que la información del sistema es mostrada al usuario. A su vez, el módulo "User GUI" representa la interfaz gráfica para los dos tipos de usuarios del sistema HWS: ciudadanos y empleados.

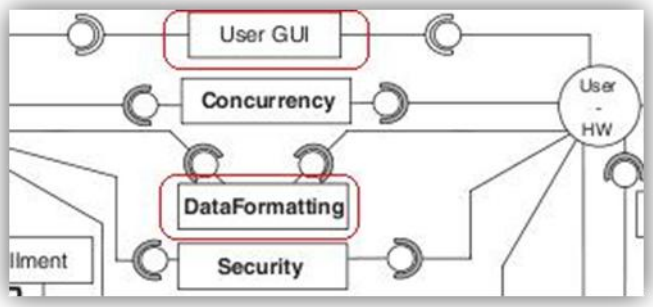

Figura V.4 - Módulos User GUI y Data Formatting de la arquitectura 2 (Pinto)

Availability

Dentro de la especificación de requerimientos del sistema HWS se aclara que se espera que el mismo esté disponible las 24 horas del día, los 7 días de la semana.

Ambas arquitecturas cuentan con, principalmente, un módulo que implementa ciertos mecanismos arquitectónicos que se relacionan con el atributo de disponibilidad. En la arquitectura 1 éste se llama “Replication” (Figura V.5) cuyo propósito es almacenar una copia de la información envida por el sistema HWS al modulo Data Formatting. La arquitectura 2 posee un módulo homónimo cuya funcionalidad es similar.

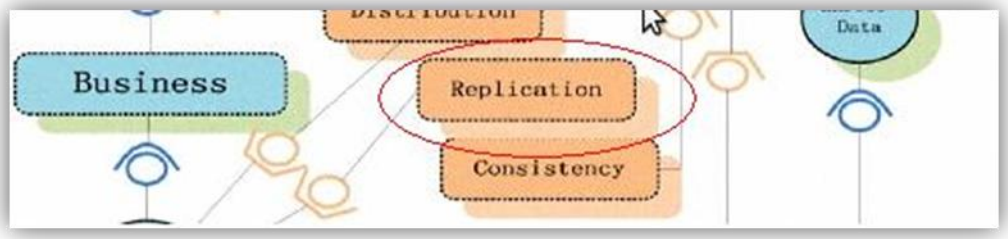

Figura V.5 - Módulo "Replication" de la arquitectura 1

Adicionalmente, en los papers que describen las arquitecturas también se identifica este atributo. Por ejemplo, en el trabajo de Zhang [9] se identifica el siguiente concern:

> *“Dependability and Availability: A serve or a processor can fail, usually in various ways, and must be built reliable using internal redundancy so that the register service remains available”*

Mientras que en el trabajo de Pinto [40] se afirma lo siguiente:

"In order to provide Availability, the Replication component stores a copy of the data the HWS system sends."

Performance

El atributo de performance es un requerimiento específico en el SRS del HWS. En el mismo se detalla que el sistema debe ser capaz de manejar 20 usuarios simultáneamente. Además, el tiempo de respuesta no debe ser mayor a 20 segundos.

Este atributo está relacionado en ambos diseños mediante las responsabilidades del módulo denominado "Concurrency" el cual "*provee la habilidad para manejar usuarios concurrentes".*

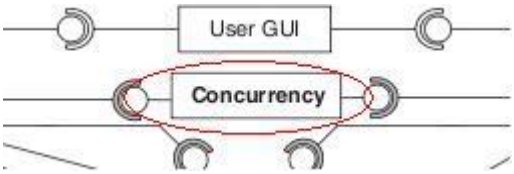

Figura V.6 - Módulo "Concurrency" de la arquitectura 2

Security

El atributo de calidad de seguridad está relacionado, principalmente, con las técnicas arquitectónicas presentes en dos módulos.

Ambos diseños poseen un módulo llamado "Authentication", cuya funcionalidad es similar en ambos casos. En el trabajo de la arquitectura 2, por ejemplo, la responsabilidad del mismo es "*interceptar la comunicación entre el empleado y el sistema HWS para chequear las credenciales del usuario cuando la operación lo requiera".*

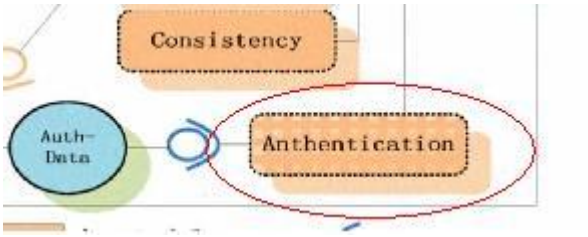

Figura V.7 - Módulo "Authentication" de la arquitectura 1

Tanto la arquitectura 1 como la 2 también poseen un módulo denominado "Security". En ambos trabajos la funcionalidad del módulo es *encapsular un protocolo para hacer segura la transmisión de datos a través de internet.*

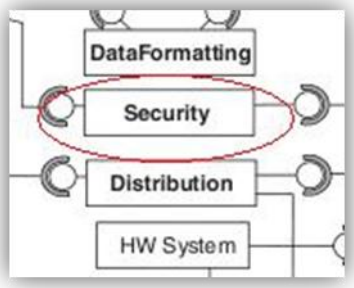

Figura V.8 - Módulo "Security" presente en la arquitectura 2

Adicionalmente, en la especificación de requerimientos del sistema HWS se detallan requerimiento de atributos de calidad de seguridad como, por ejemplo, el acceso restringido de los distintos usuarios en el sistema.

Scalability/ Distribution

En el SRS del sistema se menciona que el mismo es utilizado por ciudadanos dispersos en distintas áreas geográficas (ciudades, pueblos, etc.). En este sentido Zhang [9] afirma:

"HW system would be used by citizens from different area, and it must be designed to cater for variations"

En ambas arquitecturas se comprueba que está incluido un módulo llamado "Distribution". En la arquitectura 1 la funcionalidad de este módulo es hacer distribuibles los servicios del módulo "Bussines". Por ejemplo, cuando un ciudadano solicita un servicio a través del módulo "GUI", éste es interceptado por el módulo "Distribution" y direccionado al lugar correcto.

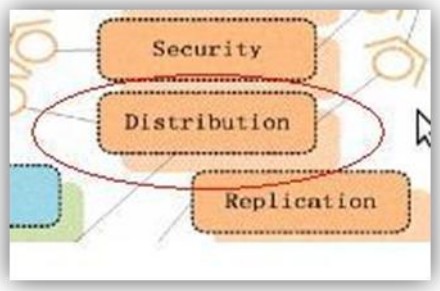

Figura V.9 - Módulo "Distribution", presente en la arquitectura 1

En la arquitectura 2 se indica que el componente representa la locación remota de los tres principales componentes de la arquitectura (User-GUI, HWSystem y DataStore).

Persistency/ Data consistency

La integridad de los datos es manejada por ambos diseños en un módulo denominado "Consistency". En la arquitectura 1 éste es invocado "ca*da vez que el sistema HW debe chequear la consistencia de los datos de entrada".* A su vez, en este mismo diseño, el módulo "Data Store" es el encargado de la persistencia de los datos manipulados.

Similar es el caso de la arquitectura 2, en donde se encuentran módulos homónimos, con funcionalidad similar.

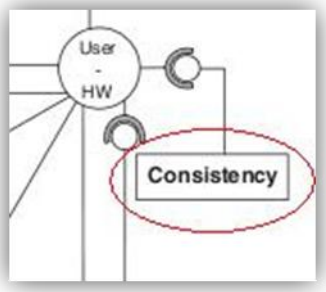

Figura V.10 - Módulo "Consistency", presente en la arquitectura 2

A su vez en el trabajo de Zhang (en el que se describe la arquitectura 1) se identifica el siguiente concern:

"Data is usually constructed and shared by citizen distributed in different locations. Therefore a major issue is the preserve the consistency of data in the concurrency and failure."

V.2.1.3. Aspectos tempranos identificados

Como ya se ha mencionado anteriormente, se utilizó la herramienta Aspect Extract Tool para la identificación de aspectos tempranos del sistema. Ésta toma como entrada los casos de uso del mismo. Al ser una técnica semi-automática, los resultados de la misma deben ser analizados por un analista experto en el dominio.

A continuación se muestran los seis aspectos tempranos detectados, junto con las palabras que los conforman. La herramienta Aspect Extract Tool especifica a cada aspecto mediante un conjunto de pares (verbo-sustantivo), o en algunos casos sólo verbos. Estos pares son los que la herramienta detecta como "crosscuting" entre todos los casos de uso y los agrupa según su similitud, formando un aspecto. Cabe mencionar que los nombres de los aspectos tempranos son arbitrarios, indicados por el analista experto en el dominio.

DATA FORMATTING: (perform, queries); (presented, user); (presented, user); (presented, user); (presented, user); (presented, user); (presented, employee); (presented, employee); (presented, employee); (presented, employee); (representing, unit); (representing, specialty); (representing, complaint); (representing, type); (creates, instance); (generates, assigns); (alters, data); (being, system)

PERSISTENCY: (stored); (stored); (stored, state); (store, information); (storing, complaint); (storing, complaint); (storing, data); (have, state); (have, HELP); (has, access); (retrieves, list); (retrieves, list); (retrieves, list); (retrieves, list); (retrieves, list); (retrieves, list); (retrieved); (retrieve, entry); (retrieve, information); (retrieve, type); (retrieving, data); (retrieves, details); (retrieving, details); (retrieving, complaints); (retrieves, data); (determine, type); (updated); (updated); (updated); (updated); (update); (informing, user); (described, login); (changed); (queried); (queried); (queried); (left, state); (left, state); (left, state); (updated, system); (updated, system); (updated, system); (saves, complaint); (saved, system); (saves, data); (saved); (stores, information); (stores, information);

CONSISTENCY: (ensures, information); (ensures, information); (ensures, information); (ensures, information); (ensures, information); (ensures, data); (ensures, data); (ensures, data); (ensures, data); (ensures, data); (ensures, data); (ensured); (ensured); (assured); (assured); (confirms, operation); (confirms, update)

DISTRIBUTION: (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (sent, server); (retrieved); (transmitted, server); (sending, data)

SECURITY: (validates, password); (validate, employee); (validated, system)

ERROR HANDLING: (ocurrs); (ocurrs); (ocurrs); (ocurrs); (ocurrs); (dealt, complaint); (verifies, data); (verifies, data)

El número de ocurrencias de cada palabra es tenido en cuenta por la técnica propuesta, y a través de filtro “ocurrencias”, explicado en el capitulo anterior, se deja constancia del número de veces que ocurre cada palabra dentro de un aspecto. Por ejemplo, en el aspecto temprano llamado “Error Handling” la palabra “ocurrs” tendrá una mayor relevancia que la palabra “dealt” a la hora de identificar el atributo de calidad, ya que la primera ocurre cinco veces y la segunda sólo una. Por lo tanto, es importante no descartar los pares <verbo, objeto directo> duplicados para contar con la mayor cantidad de información provista por el early aspect para el momento del análisis del mismo.

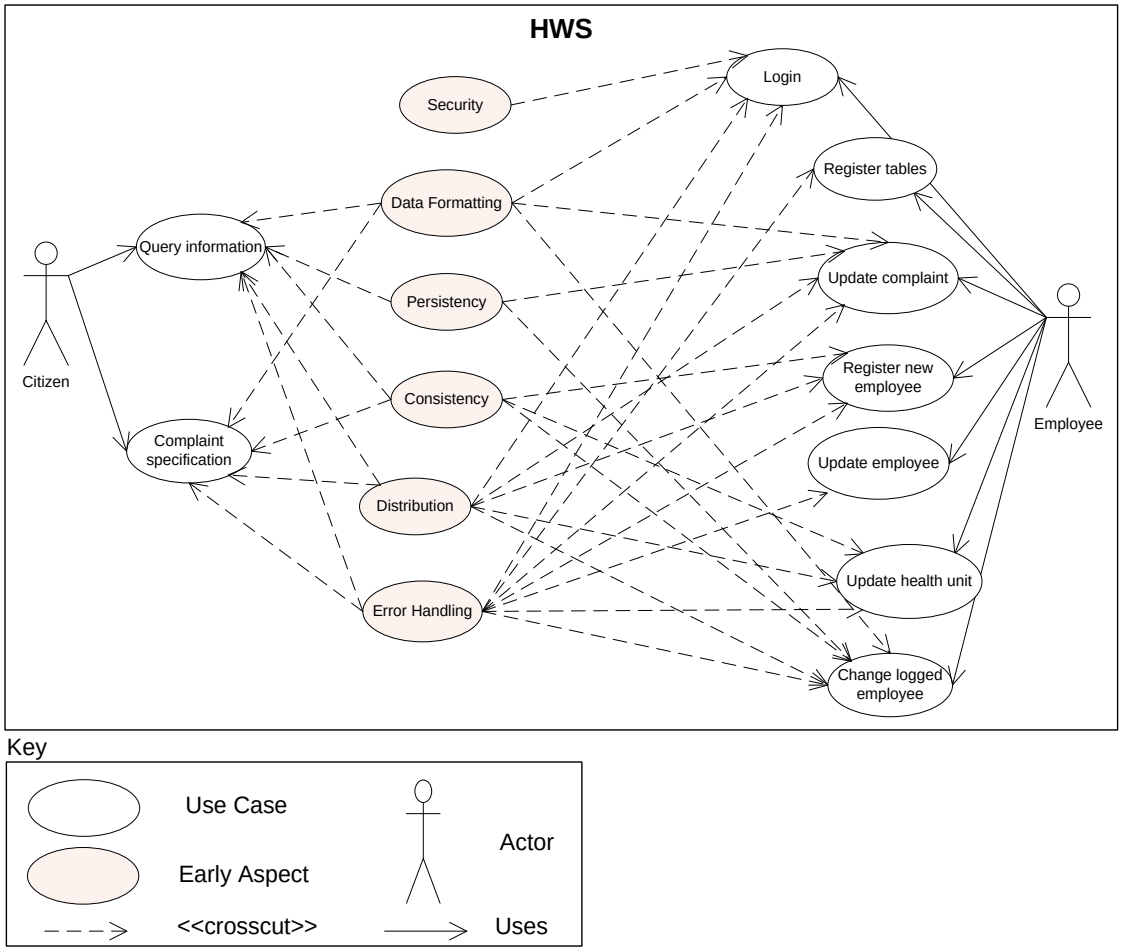

Figura V.11 - Diagrama de Casos de Uso y Aspectos Tempranos

Cada uno de estos aspectos tempranos identificados “atraviesa” un subconjunto del conjunto total de casos de uso. La herramienta Aspect Extractor Tool también brinda esa información que se resume en la Figura V.11.

V.2.1.4. Análisis con la técnica propuesta

A continuación se muestran los resultados obtenidos con la técnica propuesta para cada uno de los aspectos tempranos y sus casos de uso asociados. Vale la pena recordar que la herramienta *QA Miner* solo trabaja con seis atributos de calidad, que son los cargados en la ontología: security, availability, testeability, modifiability, usability y performance. Igualmente, la técnica no está limitada a estos atributos, ya que para poder identificar más QAs éstos se deberían cargar adecuadamente en la ontología.

Para el filtro de pesos se fijan los valores mostrados en Tabla V.11. También se muestran los resultados para distintos valores de k. Se debe recordar que, como se detalló en la sección IV.3.1.2, un valor de k igual a cero sólo toma en cuenta el mapa de tokens extraídos de los aspectos tempranos y un valor de k igual a 1 sólo toma en cuenta el mapa extraído a partir de los casos de uso. Un k igual a 0.5 hace una combinación de los dos mapas, multiplicando cada mapa por 0.5 y luego sumando ambos.

Se muestran los resultados de distintos valores de k para chequear qué resultados se obtienen con cada uno de ellos y, eventualmente, verificar si un valor en particular arroja mejores resultados que los otros.

Sección	Peso
<Nombre>, <Descripción>,<Prioridad>,<Actor>	1
<Flujo Básico>	2
<Flujo Alternativo>,<Trigger><Requerimientos especiales>,<Precondiciones>,<Postcondiciones>	3

Tabla V.11 - Ponderación de las secciones de los casos de uso

Debajo de cada tabla se hace un análisis acerca de los resultados obtenidos por *QA Miner* para cada aspecto. Básicamente, se analiza si la relación entre el aspecto temprano y los atributos de calidad que arrojaron mayores porcentajes es correcta o tiene sentido. Es decir, si por ejemplo un aspecto temprano tiene entre sus pares verbo-objeto palabras como encriptar, password, hacker, etc. se consideraría correcto que el atributo de calidad de seguridad obtenga el mayor porcentaje. Si en este caso el atributo de calidad de usability arrojara el mayor porcentaje o arrojara un porcentaje cercano al que arrojó seguridad, se consideraría que la relación es incorrecta, porque no hay ningún indicio entre los pares verbo-objeto del aspecto temprano que indique que se debería relacionar con ese atributo.

En el caso citado anteriormente se consideraría un falso positivo por la relación con usability (independientemente de que usability sea, o no, un QA del sistema). Adicionalmente, se consideraría como un “potencial” verdadero positivo a la relación con seguridad. Decimos que es “potencial” porque depende de que seguridad sea un QA real del sistema. En el caso de que lo sea, se consideraría como un verdadero positivo; caso contrario, un falso positivo.

El hecho de que el análisis anterior manifieste que la relación entre un aspecto temprano y un atributo de calidad es correcta, no significa que efectivamente ese sea un QA del sistema HWS. Solamente deja constancia acerca la correctitud, o no, de la relación entre el EA y el QA en cuanto a las palabras que lo conforman. Más adelante se corroborará que ese sea efectivamente un QA del sistema analizado.

La técnica propuesta relaciona un EA con cada atributo de calidad mediante un porcentaje. Es tarea del analista, dependiendo del contexto del aspecto, identificar el o los atributos de calidad que arrojan los mayores porcentajes para cada aspecto. En la técnica no se especifica un "umbral" que un porcentaje de un atributo de calidad deba alcanzar para ser considerado que se relaciona con el aspecto de origen. En los casos de estudio presentados se resaltan en negrita el o los atributos de calidad que consideramos que están relacionados con el aspecto (porque arrojan los mayores porcentajes), para cada valor de K.

Data Formatting

	K=0	K=0.5	K=1
Security	0.19	0.24	**0.28**
Availability	0.0	0.04	0.07
Testeability	0.08	0.09	0.11
Modifiability	0.0	0.05	0.09
Usability	**0.69**	**0.50**	**0.31**
Performance	0.04	0.09	0.14

Tabla V.12 - Resultados para el aspecto "Data Formatting"

Para el aspecto temprano "Data Formatting", el atributo de calidad *usability* es el que arroja los mayores porcentajes, para los tres valores de K. Adicionalmente, para K=1 *security* obtiene un alto valor.

Se considera que la relación con el atributo de calidad *usability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: present, represent, user, etc. A su vez, no se encuentran palabras que indiquen que el aspecto temprano se debería relacionar con *security,* por lo que esa relación es incorrecta.

A continuación (Figura V.12) se observa una captura de *QA Miner* luego de realizar el análisis para el early aspect "Data Formatting", con K=0. Los valores de la columna "Percent", de la tabla "Results" de la figura son los mismos que los de la tabla anterior, salvo que en este caso sólo se tienen en cuenta dos decimales.

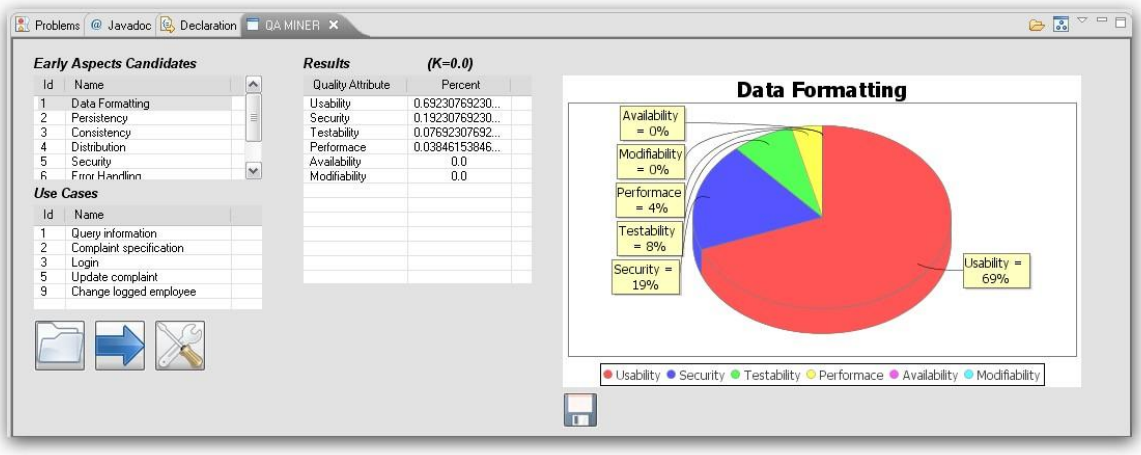

Figura V.12 - Captura QA MINER. EA: Data Formatting (K=0.0)

Persistency

	K=0	K=0.5	K=1
Security	**0.39**	**0.33**	**0.28**
Availability	**0.29**	0.05	0.07
Testeability	0.07	0.09	0.11
Modifiability	**0.26**	**0.18**	0.09
Usability	0.10	**0.20**	**0.31**
Performance	0.16	0.15	0.14

Tabla V.13 - Resultados para el aspecto "Persistency"

Para el aspecto “Persistency” los QAs *security, availability y modifiability* arrojan los mayores porcentajes para el valor de K igual a 0. Para un valor de K igual a 0.5 los atributos de calidad *security, modifiability y usability* obtienen los mayores porcentajes. Por último, para K igual a 1, *security* y *usability* obtienen los porcentajes más elevados.

Se considera que la relación con el atributo de calidad *modifiability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: update, change, etc. Adicionalmente, las relaciones con *availability, usability y security* se consideran incorrectas.

Igualmente, cabe mencionar que el aspecto temprano está conformado por varias palabras que no son específicas de ningún QA y podrían llegar a generar confusión en el análisis.

En la Figura V.13 se muestra una captura de la herramienta *QA Miner* para el early aspect "*Persistency*", con un K=0,5.

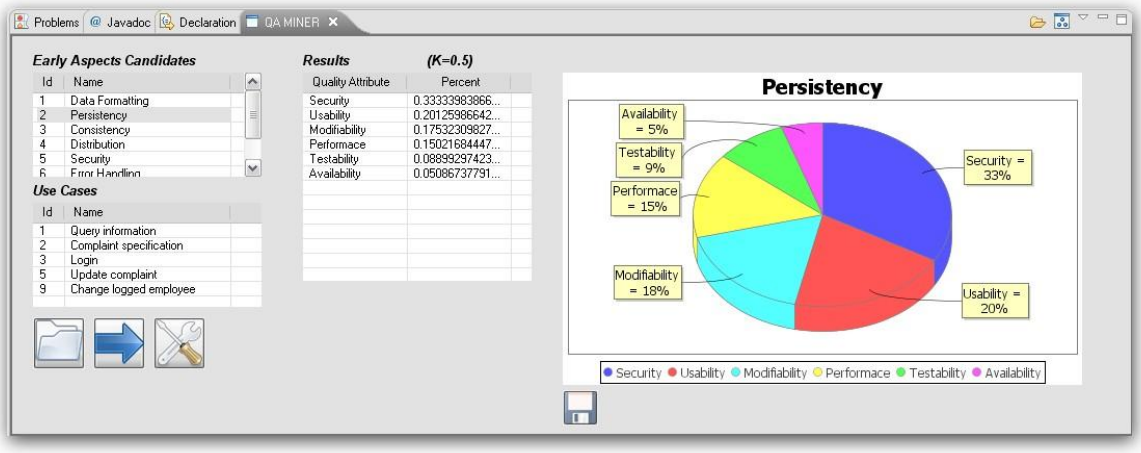

Figura V.13 - Captura QA MINER. EA: Persistency (K=0,5)

Consistency

	K=0	K=0,5	K=1
Security	**0.59**	**0.43**	**0.27**
Availability	0.02	0.04	0.06
Testeability	0.00	0.07	0.14
Modifiability	0.04	0.08	0.13
Usability	0.05	0.15	**0.26**
Performance	0.30	0.22	0.14

Tabla V.14 - Resultados para el aspecto "Consistency"

Para el aspecto temprano “Consistency” el atributo de calidad de *security* arroja los mayores porcentajes para los distintos valores de K. Adicionalmente, *usability* arroja un valor alto para un K=1.

La mayoría de los pares del aspecto contienen las palabras “ensure” y “data”. Esto da la idea de que el aspecto se refiere a la consistencia de la información, por lo que una relación con *availability* parecería la más adecuada. Es por ello que se considera incorrecta la relación con *security* arrojada por *QA Miner*. Más aún, teniendo en cuenta que *availability* arrojó resultados muy bajos en los distintos valores de K.

A continuación (Figura V.14) se observa una captura del plugin luego de analizar el early aspect “Consistency” con un K=1.

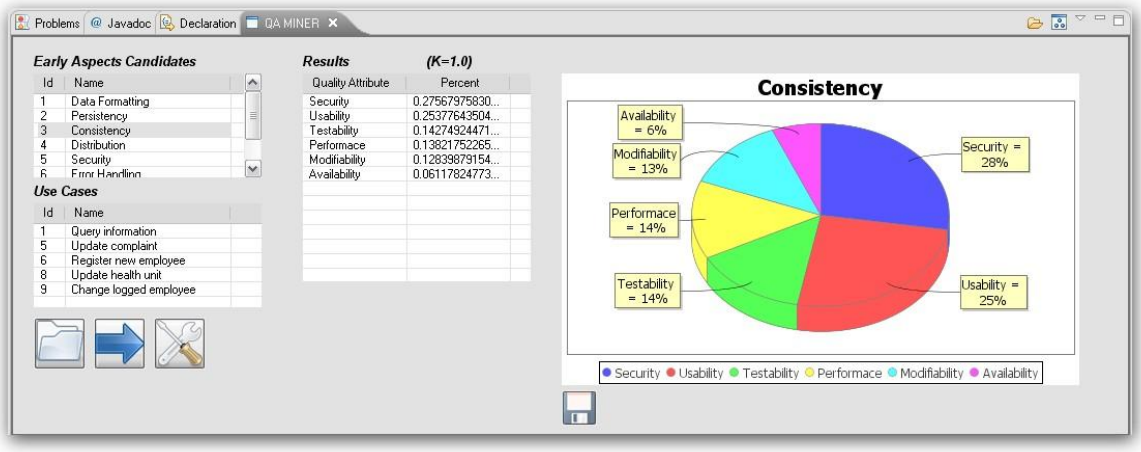

Figura V.14 - Captura QA MINER. EA: Consistency (K=1)

Distribution

	K=0	K=0.5	K=1
Security	0.05	0.17	**0.28**
Availability	**0.87**	**0.47**	0.06
Testeability	0.0	0.06	0.12
Modifiability	0.0	0.05	0.11
Usability	0.0	0.14	**0.28**
Performance	0.08	0.11	0.15

Tabla V.15 - Resultados para el aspecto "Distribution"

Para el aspecto temprano “Distribution”, el atributo *availability* arroja, por gran diferencia, los mayores porcentajes para un valor de K=1 y 0.5. Adicionalmente, para un valor de K=1 los porcentajes de *security* y *usability* son los mayores

Se considera correcta la relación del aspecto con availability. Las palabras del aspecto temprano (sent, server, transmitted) dan la idea de que existe una comunicación entre ciertas entidades para que el sistema pueda realizar su tarea. Además, se considera incorrecta la relación del aspecto con *security* y *usability.*

En la Figura V.15 se observa una captura del plugin luego de analizar el early aspect “Distribution”, con un K igual 0,5.

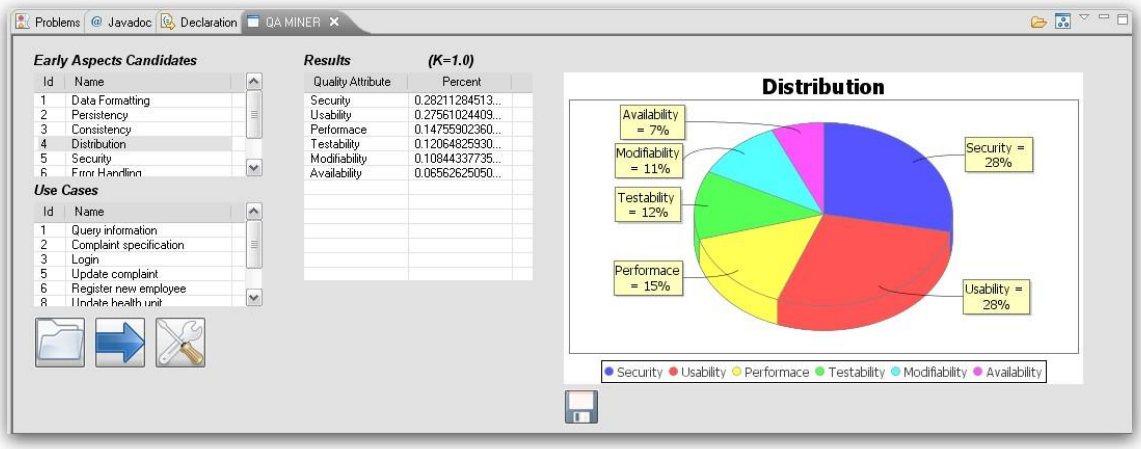

Figura V.15 - Captura QA MINER. EA: Distribution (K=0,5)

Security

	K=0	K=0.5	K=1
Security	**1.0**	**0.69**	**0.38**
Availability	0.0	0.04	0.09
Testeability	0.0	0.01	0.03
Modifiability	0.0	0.01	0.01
Usability	0.0	0.17	**0.35**
Performance	0.0	0.07	0.15

Tabla V.16 - Resultados para el aspecto "Security"

Para el aspecto “Security” el atributo de calidad *security* posee altos porcentajes para los distintos valores de K. Además, para K igual a 1, *usability* obtiene porcentajes altos.

La relación con *security* se considera correcta, teniendo en cuenta que palabras como “validates” y “password” conforman el aspecto temprano. Por otro lado, la relación con *usability* se considera incorrecta ya que no hay palabras que den indicios de esta relación.

En la Figura V.16 se observa una captura del plugin *QA Miner* luego del análisis del early aspect “Security”, con K=0.

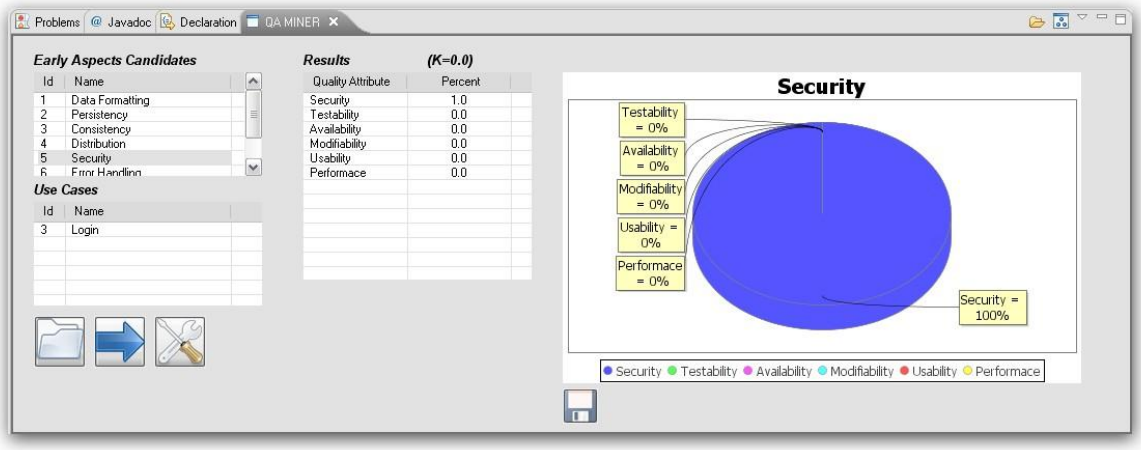

Figura V.16 - Captura QA MINER. EA: Security (K=0)

Error Handling

	K=0	K=0.5	K=1
Security	**0.7**	**0.50**	**0.29**
Availability	0.0	0.03	0.06
Testeability	0.0	0.05	0.11
Modifiability	0.0	0.06	0.12
Usability	0.2	0.23	**0.26**
Performance	0.1	0.13	0.15

Tabla V.17 - Resultados para el aspecto "Error Hadling"

Para el EA “Error Handling” el atributo de calidad *security* arroja los mayores porcentajes para los distintos valores de K. Adicionalmente, *usability* arroja un valor alto para un K=1.

Cabe mencionar que las palabras que conforman este aspecto no son muy descriptivas de ningún atributo, por lo que sería improbable que la herramienta *QA Miner* pudiera relacionar al aspecto con el atributo correcto. Es por ello, que la relación con *security* se considera incorrecta, y, además, en este punto, no se logra inferir claramente con qué atributo de calidad se debería haber relacionado.

A continuación (Figura V.17) se observa el resultado de *QA Miner* luego de analizar el aspecto *“Error Handling”,* con un K=0.0*.*

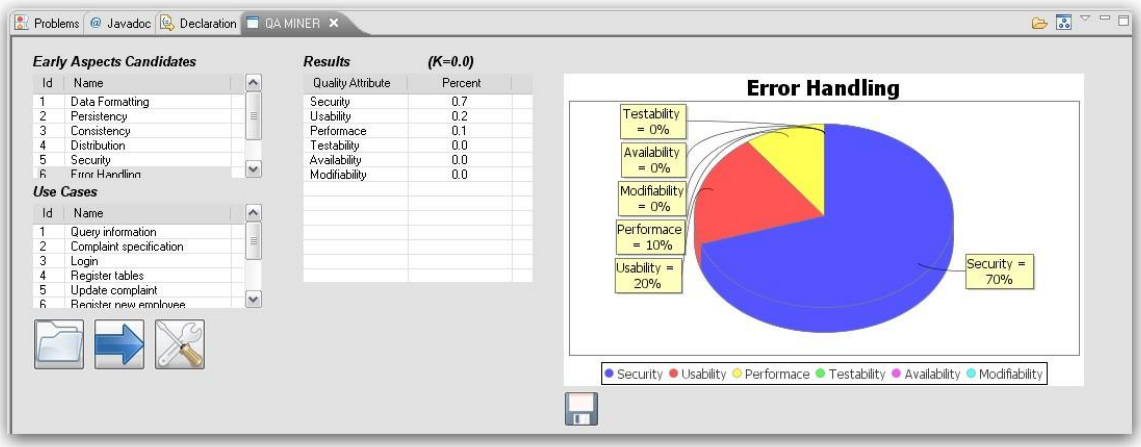

Figura V.17 - Captura QA MINER. EA: Error Handling (K=0.0)

V.2.1.5.Métricas

A continuación se mostrarán las métricas de Precision y Recall para los distintos valores de K. La idea de comparar a cada valor de k es corroborar si la técnica arroja mejores resultados con alguno en particular.

V.2.1.5.1. Métricas para K=0

La Figura V.18 muestra un resumen de los resultados para el valor de K=0. La primera columna muestra los aspectos tempranos detectados. Cada uno de estos aspectos se relaciona con un conjunto de atributos de calidad, como indican las flechas azules salientes de cada uno. En letras rojas aparecen los QAs que se consideraron incorrectamente relacionados, mientras que en

letras negras están escritos los QAs que se consideraron correctamente relacionados, para cada EA.

Los QAs correctamente relacionados forman un conjunto que se agrupa en la tercera columna mediante un círculo azul. A su vez, los QAs incorrectamente relacionados forman un conjunto que se ve en la parte inferior de la tercera columna, agrupado en un círculo rojo.

El conjunto de QAs correctos son los QAs que podrían ser considerados como verdaderos positivos. Para que esto último suceda, el QA de este conjunto debe formar parte de los QAs del sistema, que se muestran en la cuarta columna. Estos son los mismos que se mostraron previamente en la Tabla V.10 y que se consideran los QAs reales del sistema HWS.

Los QAs verdaderos positivos, en este caso, son Usability, Availability y Performance. El conjunto de QAs incorrectos (security, availability) se consideran falsos positivos. Adicionalmente, se consideran falsos positivos a los QAs correctos que no estén presentes dentro de los QAs del sistema (modifiability). Los falsos negativos son los QAs del sistema (cuarta columna) que no se encuentran relacionados con un QA correcto. En este caso los falsos negativos son performance, scalability y persistency.

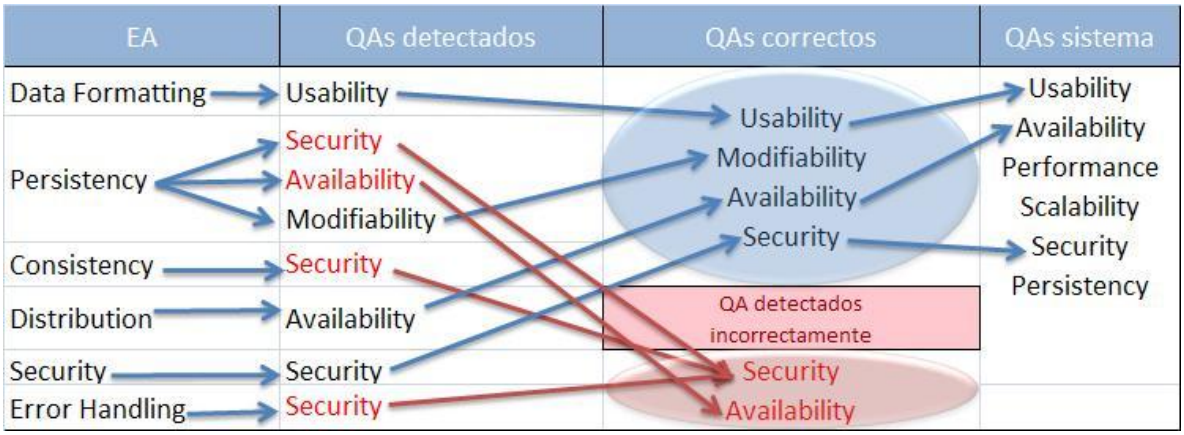

Figura V.18 - Resultados para K=0 (HWS)

En base a la Figura V.18 se calculan los QVP, QFP, QFN y QV para el caso de estudio HWS, para un valor de K=0. Esto se muestra en la Tabla V.18.

Caso de Estudio	HWS
QVP	3
QFP	3
QFN	3
QV	6

Tabla V.18 Valores de QVP. QFP, QFN y QV para HWS (K=0)

Teniendo en cuenta los valores de la Tabla V.18, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{3}{3+3} = 0{,}5$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{3}{3+3} = 0.5$$

V.2.1.5.2. Métricas para K=0,5

La Tabla V.19 muestra un resumen de los resultados para el valor de K=0.5.

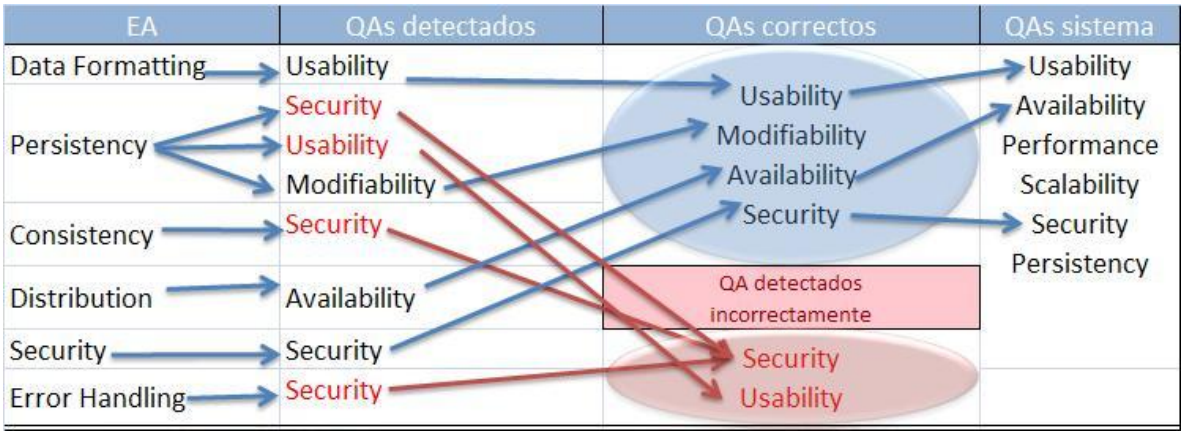

Figura V.19 - Resultados para K=0.5 (HWS)

A partir de la Figura V.19 se obtienen los valores de QVP, QFP, QFN y QV mostrados en la Tabla V.19 para un valor de K=0.5.

Caso de Estudio	HWS
QVP	3
QFP	3
QFN	3
QV	6

Tabla V.19 - Valores de QVP. QFP, QFN y QV para HWS (K=0.5)

Teniendo en cuenta los valores de la Tabla V.19, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{3}{3+3} = 0.5$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{3}{3+3} = 0.5$$

V.2.1.5.3. Métricas para K=1

La Figura V.20 muestra un resumen de los resultados para el valor de K=1.

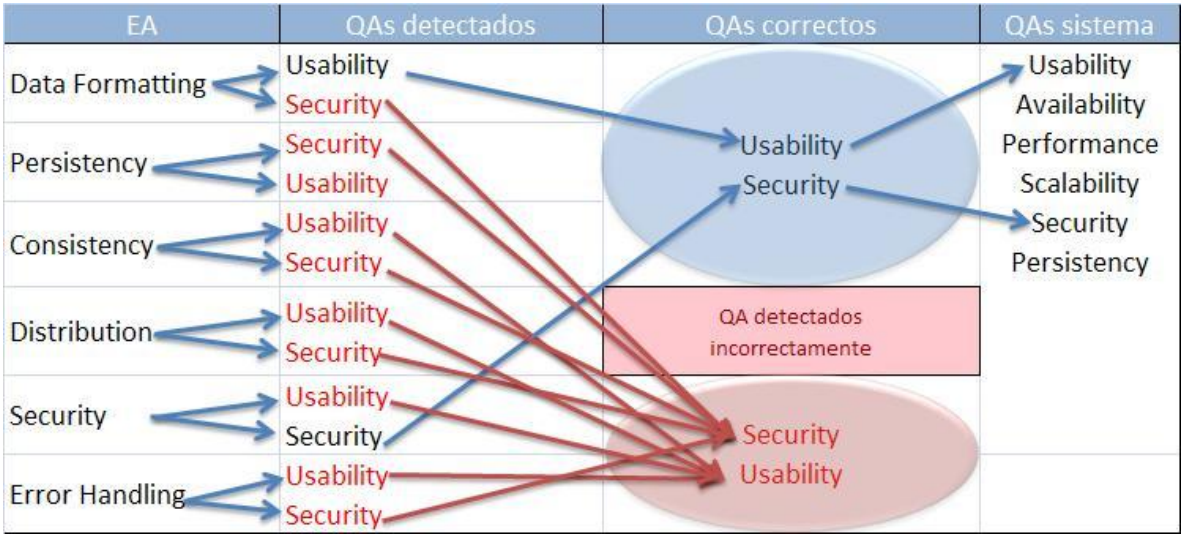

Figura V.20 - Resultados para K=1 (HWS)

A partir de la Figura V.20 se obtienen los valores de QVP, QFP, QFN y QV mostrados en la Tabla V.20, para un valor de K=1.

Caso de Estudio	HWS
QVP	2
QFP	2
QFN	4
QV	6

Tabla V.20 - Valores de QVP. QFP, QFN y QV para HWS (K=1)

Teniendo en cuenta los valores de la Tabla V.20, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{2}{2+2} = 0.5$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{2}{2+4} = 0.33$$

V.2.1.5.4. Resumen de las métricas

Los resultados de las métricas de precisión y recall se resumen en la Tabla V.21.

Métrico	K=0	K=0.5	K=1
Precision	0.5	0.5	0.5
Recall	0.5	0.5	0.33

Tabla V.21 - Valores de Precision y Recall para los distintos valores de K

En este caso de estudio, tanto para K=0 y K=0.5 se obtienen los mismos resultados para las métricas *recall* y *precisión*. Es por ello, que en este punto, no se podría indicar claramente que valor de K sería más conveniente utilizar.

Para el valor de K=1 se obtiene la misma precisión que en los casos anteriores, pero disminuye el valor del *recall* en un 33% aproximadamente. Esto indicaría que este valor de K sería menos eficiente que los otros dos.

V.2.1.6. Tiempo de ejecución

De manera de tener una noción de la velocidad de procesamiento de la identificación por parte de la técnica propuesta, en la Tabla V.22 se muestran los lapsos de tiempo necesarios para efectuar el análisis de identificación de atributos de calidad (medidos en milisegundos) para cada conjunto <aspecto temprano, casos de usos relacionados>.

La primera columna de la tabla muestra el aspecto temprano. La segunda columna indica el número de casos de uso que ese aspecto atraviesa. La tercera columna indica la cantidad de palabras que posee ese conjunto de casos de uso atravesados más las palabras que conforman el aspecto temprano. Finalmente, la cuarta columna indica el tiempo, en milisegundos, que tomó el procesamiento de ese aspecto temprano.

Aspecto temprano	Cantidad Casos de Usos Relacionados	Cantidad Palabras Casos de Usos y Aspecto	Tiempo de ejecución (en milisegundos)
Error Handling	9	2302	1438
Distribution	7	2114	1259
Persistency	5	1684	1246
Consistency	5	1717	1131
Data Formatting	5	1623	1108
Security	1	136	761
Promedio	5,33	1596	1157,16

Tabla V.22 - Tiempos de ejecución por aspecto temprano (HWS)

En general, se tienen tiempos de ejecución bajos, en donde no se superan los 1.5 segundos en ningún caso. El máximo valor registrado es cuando se analizó el aspecto "*Error Handling"*, con un tiempo de ejecución de 1438 milisegundos. El menor valor registrado se encontró en el aspecto "*Error Handling"*, con un valor de 761 milisegundos. El promedio de análisis de todos los aspectos tempranos arroja un valor de 1157,16 milisegundos.

Se puede notar que, en general, a medida que aumenta el número de palabras analizadas, aumenta el tiempo de ejecución. Igualmente, al tratarse de tiempos tan bajos, en diferentes ejecuciones se han observado variaciones en los mismos, no respetándose siempre el mismo orden que se muestra en la Tabla V.22. Esto indicaría que la cantidad de palabras o casos de uso analizados por aspecto temprano tiene un impacto mínimo sobre el tiempo total del análisis. Esto podría ocurrir por el hecho de que cuando comienza el análisis, tanto las palabras del aspecto temprano como de los casos de uso se encuentran cargadas en memoria, por lo que su procesamiento no sería significativo.

V.2.2. Sistema CRS (Course Registration System)

El segundo caso de estudio se denomina CRS (Course Registration System). Consiste en un nuevo sistema de tipo cliente-servidor para reemplazar el sistema anterior de una institución educativa, que está basado en mainframes. El nuevo sistema permitirá a los estudiantes registrarse en cursos y visualizar reportes de sus calificaciones. Adicionalmente, el mismo permitirá a los profesores registrarse en nuevos cursos e informar las notas de los estudiantes.

Al principio de cada semestre los estudiantes pueden solicitar un catálogo de cursos que contiene una lista de ofertas de cursos disponibles para el semestre. Por cada curso se incluye información sobre el profesor que lo dicta, departamento, y los requisitos previos que se deben cumplir para poder inscribirse.

El sistema permitirá a los estudiantes inscribirse a distintos cursos para el semestre. Adicionalmente, el estudiante indicará dos cursos alternativos, en caso de que no sea designado a su primera opción. Un curso que posea menos de tres estudiantes inscriptos será cancelado, permitiendo un máximo de 10. Una vez que se haya terminado el período de inscripción, el sistema enviará la información de cada estudiante a un sistema externo de facturación.

Al final del semestre, el alumno será capaz de acceder al sistema para ver un reporte electrónico de sus calificaciones recibidas. Dado que estos reportes representan información confidencial, el sistema debe emplear medidas de seguridad adicionales para evitar el acceso no autorizado.

V.2.2.1.Casos de uso

El presente caso de estudio está formado por ocho casos de estudio. Dada la extensión de los mismos, éstos se pueden chequear en el Anexo II. A continuación se presenta una breve descripción de cada uno, junto la cantidad de palabras que los conforman.

Nombre	[UC1] Login
Descripción	This use case describes how a user logs into the Course Registration System.
Cantidad de palabras	112

Tabla V.23 - Breve descripción del caso de uso "Login"

Nombre	[UC2] View Report Card
Descripción	This use case allows a Student to view his/her report card for the previously completed semester
Cantidad de palabras	136

Tabla V.24 - Breve descripción del caso de uso "View Report Card"

Nombre	[UC3] Register for Courses
Descripción	This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.
Cantidad de palabras	629

Tabla V.25 - Breve descripción del caso de uso "Register for Courses"

Nombre	[UC4] Select Courses to Teach
Descripción	This use case allows a professor to select the course offerings (date- and time- specific courses will be given) from the course catalog for the courses that he/she is eligible for and wishes to teach in the upcoming semester.
Cantidad de palabras	412

Tabla V.26 - Breve descripción del caso de uso "Select Courses to Teach"

Nombre	[UC5] Submit Grades
Descripción	This use case allows a Professor to submit student grades for one or more classes completed in the previous semester
Cantidad de palabras	276

Tabla V.27 - Breve descripción del caso de uso "Submit Grades"

Nombre	[UC6] Maintain Professor Information
Descripción	This use case allows the Registrar to maintain professor information in the registration system. This includes adding, modifying, and deleting professors from the system
Cantidad de palabras	386

Tabla V.28 - Breve descripción del caso de uso "Maintain Professor Information"

Nombre	[UC7] Maintain Student Information
Descripción	This use case allows the Registrar to maintain student information in the registration system. This includes adding, modifying, and deleting students from the system
Cantidad de palabras	471

Tabla V.29 Breve descripción del caso de uso "Maintain Student Information"

Nombre	[UC8] Close Registration
Descripción	This use case allows a Registrar to close the registration process. Course offerings that do not have enough students are cancelled. Course offerings must have a minimum of three students in them. The billing system is notified for each student in each course offering that is not cancelled, so the student can be billed for the course offering. The main actor of this use case is the Registrar. The Billing System is an actor involved within this use case.
Cantidad de palabras	425

Tabla V.30 - Breve descripción del caso de uso "Close Registration"

V.2.2.2. Atributos de calidad del sistema CRS

Los atributos de calidad del sistema CRS fueron identificados por nosotros a partir de la información presente en los documentos de especificación de requerimientos de [10] y [11]. La Tabla V.31 muestra los atributos extraídos, junto con el contexto de los mismos dentro del sistema CRS.

Atributo de calidad	Descripción
Usability	The user interface of the C-Registration System shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with no additional training on the System.
Availability	The C-Registration System shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time.
Performance	1. The system shall support up to 2000 simultaneous users against the central database at any given time and up to 500 simultaneous users against the local servers at any one time. 2. The system shall provide access to the legacy course catalog database with no more than 10 second latency. 3. The system must be able to complete 80% of all transactions within 2 minutes.
Security	The system must prevent students from changing any schedules other than their own, and professors from modifying assigned course offerings for other professors. 2. Only Professors can enter grades for students. 3. Only the Registrar is allowed to change any student information.
Modifiability	New Releases Downloadable. Upgrades to the PC client portion of C-Registration shall be downloadable from the UNIX Server over the internet. This feature enables students to have easy access to system upgrades.

Tabla V.31 - Atributos de calidad del sistema CRS

V.2.2.3. Aspectos tempranos identificados

A continuación se muestran los siete aspectos tempranos detectados por la herramienta Aspect Extractor Tool, junto con las palabras que los conforman.

Persistency: (completed, semester); (prepares, formats); (Add, sub-flow);(add, list); (Add, sub-flow); (Add, Offering); (adds, Student); (Add, Professor); (Add, Student); (Add, Student); (modify, selections); (modify); (modify); (modify); (modifying, students); (modify); (modify); (modify, schedule); (adding, courses); (added, system); (added, system); (teach, semester); (teach, semester); (teaching, offerings); (teach); (teach, registration); (teach, offering); (teach, offering); (updates, course); (updates, information); (updates, information); (taught, semester); (change, grade); (change, name); (change, name); (enables, students); (logs, System); (logged, system); (logged, system); (logged, system); (logged, system); (logged, system); (logged, system); (logged)); (register, offerings); (saves, schedule); (saves, schedule); (save, schedule); (saved, student); (registered, offering); (records, grade); (save, system); (save, system); (enrolled, offering); (enrolled, course); (delete, selections); (delete, schedule); (deletes, schedule); (deleting, professors); (deleted); (deleted, system); (deleted); (deleted, system); (deleted, system); (removes, professor); (take, system); (needed, registration); (require, space); (establish, offerings); (creates, assigns); (creates, assigns); (repeated, professor); (repeated, professor); (repeated, professor); (repeated, student); (repeated, student); (repeated, student); (create, professor); (create, student)

Entitlement: (logs, System); (logged, system); (logged, system); (logged, system); (logged, system); (logged, system); (logged, system); (logged); (register, offerings); (saves, schedule); (saves, schedule); (save, schedule); (saved, student); (registered, offering); (records, grade); (save, system); (save, system); (enrolled, offering); (enrolled, course)

External interface: (prompts, Student); (teach, semester); (teach); (signed); (signed, offering); (indicating, conflict); (notified, student); (communicate, System); (submit); (submit); (submit, grades); (returns, flow)

Security: (type, password); (type, number); (type, number); (acknowledges, message); (acknowledges, error); (acknowledges, message); (acknowledges, message); (resolve, conflict); (retrieves, information); (retrieves, professor); (retrieves, information); (retrieves, student)

Usability: (canceling, selection); (cancel, operation); (cancelled, student); (cancel, offering); (cancel, offering); (cancelled); (cancels, offering); (verify, deletion); (verifies, deletion); (verifies, Student); (verifies, offerings); (confirming, deletion); (confirming, deletion)

Data validation: (validates, password); (validates, data); (validates, data); (allows, Student); (allows, Student); (allows, professor); (allows, Professor); (allows, Registrar); (allows, Registrar); (allows, Registrar)

Performance: (process, course); (processed, semester); (designed, ease-of-use)

Cada uno de estos aspectos tempranos "atraviesa" un subconjunto de los casos de uso. Esta información se resume en la Tabla V.32.

Crosscuts	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
EA1 Persistency	X	X	X	X	X	X	X	X
EA2 Entitlement	X	X	X	X	X	X	X	X
EA3 External Interface			X	X	X			X
EA4 Security	X	X	X	X		X	X	
EA5 Usability			X	X	X	X	X	X
EA6 Data validation	X	X	X	X	X	X	X	X
EA7 Performance								X

Tabla V.32 - Crosscut de Aspectos en CRS

V.2.2.4.Análisis con la técnica propuesta

A continuación se muestran los resultados obtenidos con la técnica propuesta para cada uno de los aspectos tempranos. En este caso de estudio también se analizará si la relación entre el aspecto temprano y el atributo de calidad es correcta desde un punto de vista léxico. Por último, se tomarán las métricas con los resultados obtenidos.

Persistency

	K=0	K=0.5	K=1
Security	0.08	0.10	0.12
Availability	0.18	0.16	0.14
Testeability	0.04	0.04	0.05
Modifiability	**0.69**	**0.47**	**0.25**
Usability	0.00	0.15	**0.30**
Performance	0.01	0.07	0.14

Tabla V.33 - Resultados para el aspecto "Persistency"

Para el aspecto temprano "Persistency", con valores de K=0 y 0.5, es el atributo de calidad de *modifiability* el que arroja los mayores porcentajes. Con un K=1 los atributos de calidad *usability* y *modifiability* arrojan los mayores porcentajes.

Se considera que la relación con el atributo de calidad *modifiability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: add, modify, delete, remove, change, etc. Adicionalmente, se hubiese considerado correcta la relación con *availability* por las palabras require, establish, log, etc. La relación con *usability* se considera incorrecta.

Entitlement

	K=0	K=0.5	K=1
Security	0.12	0.12	0.12
Availability	**0.88**	**0.51**	0.14
Testeability	0.0	0.02	0.05
Modifiability	0.0	0.13	**0.25**
Usability	0.0	0.15	**0.30**
Performance	0.0	0.07	0.14

Tabla V.34 - Resultados para el aspecto "Entitlement"

Para el aspecto temprano “Entitlement”, con los valores de K=0 y 0.5, es el atributo de calidad *availability* el que arroja los mayores porcentajes. Con un K=1, el atributo de calidad *usability* y *modifiability* arrojan los mayores porcentajes.

Se considera que la relación con el QA *availability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: log, register, record, etc. Adicionalmente, la relación con *usability y modifiability* se consideran incorrectas.

External Interface

	K=0	K=0.5	K=1
Security	0.0	0.04	0.07
Availability	**0.5**	**0.37**	**0.24**
Testeability	0.0	0.01	0.02
Modifiability	0.0	0.10	0.18
Usability	**0.4**	**0.35**	**0.30**
Performance	0.1	0.14	0.19

Tabla V.35 - Resultados para el aspecto "External Interface"

Para el aspecto temprano “External Interface”, todos los valores de k arrojan altos porcentajes para los atributos *availability* y *usability*.

Se considera que la relación con el QA *availability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: communicate, flow, etc. Adicionalmente, la relación con *usability* se considera incorrecta.

Security

	K=0	K=0.5	K=1
Security	**0.4**	**0.26**	0.11
Availability	0.0	0.07	0.14
Testeability	0.0	0.02	0.05
Modifiability	0.1	**0.19**	**0.28**
Usability	0.2	**0.24**	**0.29**
Performance	**0.3**	**0.21**	0.12

Tabla V.36 - Resultados para el aspecto "Security"

Para el aspecto temprano “Security”, con valores de K=0 los atributos de calidad *security* y *performance* arrojan los porcentajes más altos. Con K=0.5, se observan porcentajes elevados para los QAs *security, modifiability, usability y performance.* Por último, para K igual a 1, los atributos de calidad con mayores porcentajes son *modifiability y usability.*

Se considera que la relación con el QA *security* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: password, message, acknowledeges, information, etc. Adicionalmente, las relaciones con *modifiability, usability y performance* se consideran incorrectas.

Usability

	K=0	K=0.5	K=1
Security	0.24	0.17	0.10
Availability	0.03	0.09	0.15
Testeability	0.0	0.02	0.05
Modifiability	0.24	0.25	**0.27**
Usability	**0.48**	**0.39**	**0.30**
Performance	0.01	0.08	0.14

Tabla V.37 - Resultados para el aspecto "Usability"

Para el aspecto temprano “Usability”, con los valores de K=0, 0.5 y 1, es el atributo de calidad *usability* el que arroja los mayores porcentajes. Con un K=1, el atributo de calidad m*odifiability* también arroja porcentajes altos.

Se considera que la relación con el QA *usability* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: cancel, select, operation, confirm, etc. Adicionalmente, la relación con *modifiability* se considera incorrecta.

Data Validation

	K=0	K=0.5	K=1
Security	**0.95**	**0.53**	0.12
Availability	0.0	0.07	0.14
Testeability	0.0	0.02	0.05
Modifiability	0.0	0.13	**0.25**
Usability	0.0	0.15	**0.30**
Performance	0.05	0.09	0.14

Tabla V.38 - Resultados para el aspecto "Data Validation"

Para el aspecto temprano “Data Validation” con los valores de K=0 y 0.5 es el atributo de calidad *security* arroja los mayores porcentajes. Con un K=1, los atributos de calidad m*odifiability* y *usability* son los que arrojan los mayores porcentajes.

Se considera que la relación con el QA *security* es correcta, ya que los pares (verbo, objeto) del aspecto contienen palabras como: allow, password, valídate, etc. Adicionalmente, la relación con *modifiability y usability* se consideran incorrectas.

Performance

	K=0	K=0.5	K=1
Security	0.0	0.06	0.12
Availability	0.0	0.11	**0.22**
Testeability	0.25	0.15	0.05
Modifiability	0.0	0.03	0.06
Usability	0.25	**0.28**	**0.31**
Performance	**0.5**	**0.37**	**0.25**

Tabla V.39 - Resultados para el aspecto "Performance"

Para el aspecto temprano “Performance” el QA *performance* arroja los mayores porcentajes para los distintos valores de K. Adicionalmente, con K=0,5 el QA usability también arroja un porcentaje alto, mientras que para k=1 los QAs *availability* y *usability* arrojan porcentajes altos.

Cabe mencionar que el aspecto está conformado por muy pocas palabras, por ello el análisis para considerar correcta la relación los QAs se basa en pocas palabras. Se considera que la relación con el QA *performance* es correcta por la palabra “proccess”, mientras que la relación con el QA *usability* también es correcta por la palabra “ease-of-use”. Adicionalmente, la relación con *availability* se considera incorrecta.

V.2.2.5. Métricas

A continuación se mostrarán las métricas de Precision y Recall para los distintos valores de k. La idea de comparar a cada valor de k es corroborar si la técnica arroja mejores resultados con alguno en particular.

Los QAs considerados como reales para el sistema CRS son los previamente mostrados en la Tabla V.31. Estos se consideran como los QAs del sistema y, en las siguientes figuras (Figura V.21, Figura V.22 y Figura V.23), son los mostrados en la cuarta columna.

V.2.2.5.1. Métricas para K=0

La Figura V.21 muestra un resumen de los resultados para el valor de K=0.

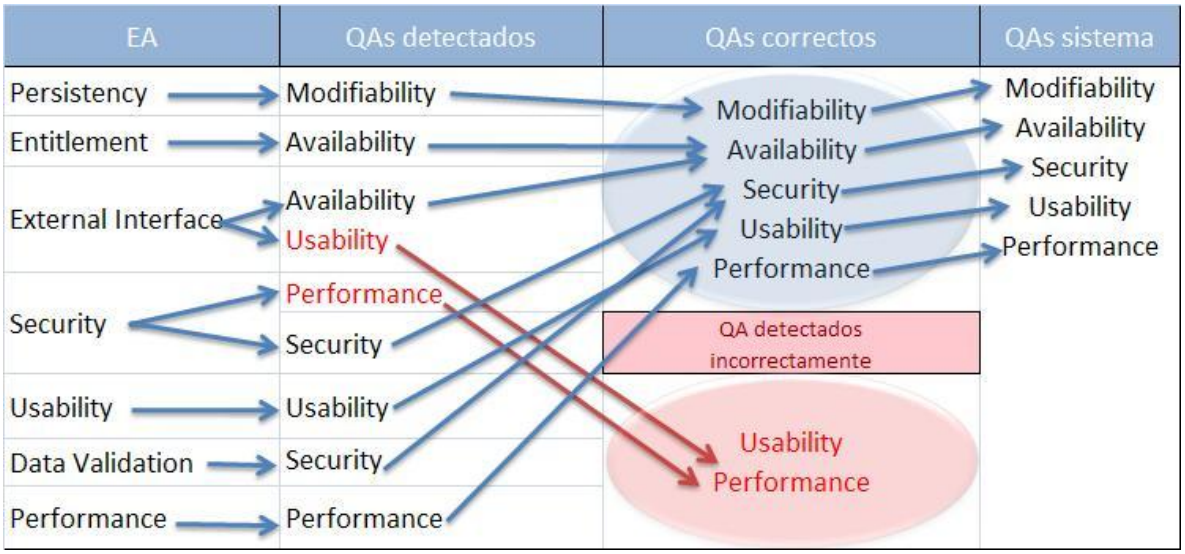

Figura V.21 - Resultados para K=0 (CRS)

A partir de la Figura V.21 se pueden calcular los valores de QVP, QFP, QFN y QV para un valor de k igual a 0. Estos se muestran en la Tabla V.40.

Caso de Estudio	CRS
QVP	5
QFP	2
QFN	0
QV	5

Tabla V.40 - Valores de QVP, QFP, QFN y QV para CRS (K=0)

Teniendo en cuenta los valores de la Tabla V.40, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{5}{5+2} = 0.68$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{5}{5+0} = 1$$

V.2.2.5.2. Métricas para K=0,5

La Figura V.22 muestra un resumen de los resultados para el valor de K=0.5.

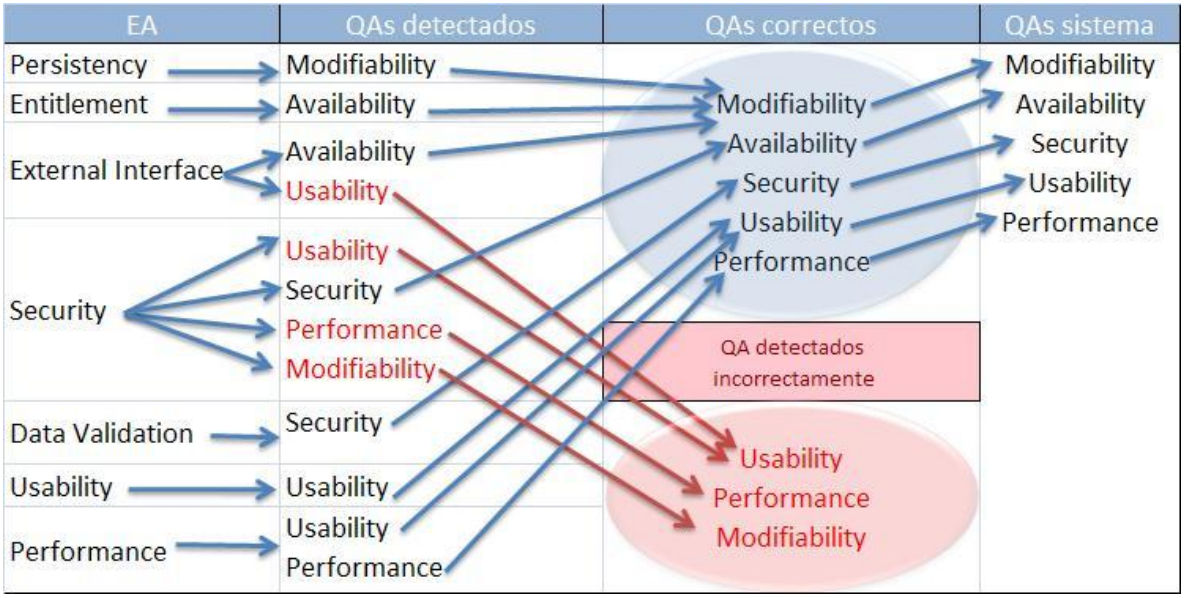

Figura V.22 - Resultados para K=0.5 (CRS)

A partir de la Figura V.22 se pueden calcular los valores de QVP, QFP, QFN y QV para un valor de k igual a 0.5. Estos se muestran en la Tabla V.41.

Caso de Estudio	CRS
QVP	5
QFP	3
QFN	0
QV	5

Tabla V.41 - Valores de QVP, QFP, QFN y QV para CRS (K=0.5)

Teniendo en cuenta los valores de la Tabla V.41, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{5}{5+3} = 0.625$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{5}{5+0} = 1$$

V.2.2.5.3. Métricas para K=1

La Figura V.23 muestra un resumen de los resultados para el valor de K=1.

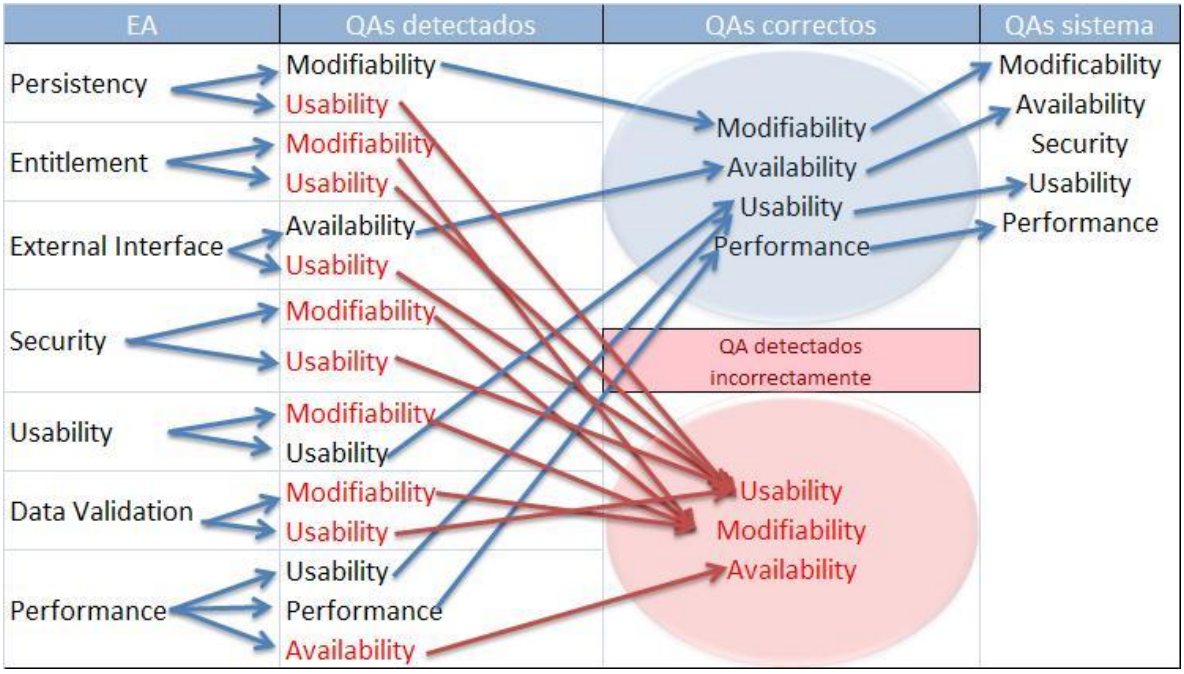

Figura V.23 - Resultados para K=1 (CRS)

A partir de la Figura V.23 se pueden calcular los valores de QVP, QFP, QFN y QV para un valor de K igual a 1.

Caso de Estudio	CRS
QVP	4
QFP	3
QFN	1
QV	5

Tabla V.42 - QVP, QFP, QFN y QV para CRS, con K=1

Teniendo en cuenta los valores de la Tabla V.42, se obtienen los siguientes resultados de Precision y Recall.

$$Precision = \frac{QVP}{QVP + QFP} = \frac{4}{4+3} = 0{,}57$$

$$Recall = \frac{QVP}{QVP + QFN} = \frac{4}{4+1} = 0.8$$

V.2.2.5.4. Resumen de los resultados

La Tabla V.43 muestra un resumen de los valores de Precision y Recall obtenidos para los distintos valores de K.

Caso de Estudio	K=0	K=0.5	K=1
Precision	0.68	0.625	0,57
Recall	1	1	0.8

Tabla V.43 - Valores de Precision y Recall para los distintos valores de K

Para un K=0 se obtienen los mejores resultados de precisión y recall. En este caso de estudio se consigue un recall igual a 1 tanto para un valor de K igual a 1 como para el valor de K igual a 0.5. En este último caso el valor de precisión disminuye ligeramente. Para el valor de K=1 ambas métricas disminuyen sus valores, obteniendo los peores resultados.

V.2.2.6. Tiempos de ejecución

La Tabla V.44 muestra los lapsos de tiempo necesarios para efectuar el análisis de identificación de atributos de calidad (medidos en milisegundos) para cada conjunto <aspecto temprano, casos de usos relacionados>.

Aspecto temprano	Cantidad Casos de Usos Relacionados	Cantidad Palabras Casos de Usos y Aspecto	Tiempo de ejecución (en milisegundos)
Persistency	8	3825	796
Data Validation	8	3686	781
Usability	6	3417	750
Entitlement	8	3703	719
Security	6	2913	701
External Interface	4	2228	656
Performance	1	476	500
Promedio	5,85	2892,57	700,42

Tabla V.44 - Tiempos de ejecución para el caso de estudio CRS

En general, nuevamente se tienen tiempos de ejecución bajos, en donde no se superan los 0.8 segundos en ningún caso. El máximo valor registrado es cuando se analizó el aspecto "Persistency", con un tiempo de ejecución de 796 milisegundos. El menor valor registrado se encontró en el aspecto "Performance", con un valor de 500 milisegundos. El promedio de análisis de todos los aspectos tempranos arroja un valor de 700,42 milisegundos.

Este análisis de tiempos se realizó sobre una maquina con un hardware más potente que en el caso de estudio HWS. A pesar de ello, también se detecta una cierta relación entre la cantidad de palabras analizadas y los tiempos de ejecución. En general, mientras más palabras se analicen mayores son los tiempos. Igualmente esta regla no siempre se cumple ("Entitlement" posee más palabras que "Usability" y, sin embargo, duró menos su ejecución), y solo sirve de referencia general. Nuevamente se notaron variaciones de los tiempos, para un mismo aspecto, entre una ejecución y otra, lo que indicaría que los valores de la tabla no siempre se cumplen.

V.3. Conclusiones de los casos de estudio

Se comprobó en ambos casos de estudio analizados que, para las métricas *recall* y *precisión*, se obtienen los mejores resultados utilizando un K=0. Igualmente, es cierto que con K=0,5 los resultados no sufren una gran variación, aunque en el caso de estudio CRS se obtuvieron, con una diferencia mínima, resultados ligeramente inferiores. Esto indica que los aspectos tempranos aportan información relevante, que por sí sola sería suficiente para identificar atributos de calidad del sistema.

Adicionalmente, en ambos casos se comprueba que un valor de K=1 obtiene los resultados de peor calidad. Esto indicaría que la información proveniente de los casos de uso no es suficiente para identificar correctamente los atributos de calidad del sistema.

En el segundo caso de estudio se obtuvieron mejores resultados que en el primero. Un factor que podría haber influido significativamente en este hecho es que dos de los QAs del sistema HWS (Persistency y Scalability) no se encuentran cargados en la ontología, por lo que nunca podrían haber sido detectados. Por esta razón, para el caso de estudio HWS, nunca se podría haber alcanzado un *recall* máximo. Esto último no ocurrió en el caso CRS, en donde todos los QAs de este sistema se encontraban modelados en la ontología, lo que permitía que se pudiera alcanzar un *recall* igual a 1, como luego sucedió.

En ambos casos de estudio se detectaron varios falsos positivos debido a relaciones incorrectas de QAs con EAs, lo que impactó negativamente sobre la *precisión*. Dos factores influyen en este hecho. El primer factor es que algunos aspectos tempranos no están correctamente formados, ya sea mostrando palabras relacionadas con la funcionalidad de un

sistema en particular (como por ejemplo: teach, dealt, etc.) o presentando pocas palabras. El segundo factor es que en algunas palabras de los EAs, a pesar de ser representativas de los aspectos, no se detectó correlación con ninguna instancia de la ontología. Lo primero se solucionaría mejorando la técnica de identificación de aspectos, mientras que lo segundo se podría mejorar extendiendo o completando la fuente de información que representa la ontología.

Con respecto a los tiempos de ejecución se notaron, en todos los casos, valores bajos. El segundo caso de estudio se ejecutó sobre un mejor hardware que el utilizado para medir los tiempos del primer caso. De esta manera, se pudo notar la influencia del mismo, ya que el promedio de los tiempos de ejecución disminuyó aproximadamente 0.45 segundos, a pesar de que los casos de uso poseían un mayor número de palabras. En general, se nota que, a medida que aumenta el número de palabras analizadas, aumenta el tiempo de ejecución. Sin embargo, esta no es una regla que siempre se cumpla. El tiempo de ejecución no pareciera ser una métrica en la cual se encuentren grandes diferencias para los diferentes casos de estudio, o que sea un factor que se deba mejorar para la técnica. Estos tiempos se consideran ampliamente aceptables para el analista que utilizará la herramienta.

Capítulo VI - Conclusiones

En la actualidad, varios investigadores han estudiado el problema de la identificación de atributos de calidad en etapas tempranas del ciclo de desarrollo de software, con resultados dispares. En particular, pocos enfoques han analizado la relación entre aspectos tempranos y atributos de calidad o, más precisamente, cómo los primeros podrían ser de utilidad para identificar a los segundos. En este trabajo se presentó una técnica semi-automatizada de identificación de atributos de calidad candidatos en casos de uso, que aborda la cuestión a partir del estudio de esa relación entre EAs y QAs.

Este enfoque se desarrolló tomando como entrada los aspectos tempranos detectados sobre un conjunto de casos de uso. La identificación de aspectos se realizó principalmente utilizando la herramienta Aspect Extractor Tool. Por cada aspecto temprano se consideraron tanto las palabras que lo conformaban, como el subconjunto de casos de uso que el aspecto afectaba. Se implementó una herramienta llamada QA Miner.

La técnica propuesta consiste en dos etapas: generación de tokens y análisis de tokens. La primera etapa itera sobre cada una de las palabras de entrada, generando una lista de tokens. Cada uno de éstos contiene atributos como peso, número de ocurrencias, etc. En la segunda etapa cada token es analizado para corroborar su relación con distintos atributos de calidad. Esta relación se calcula a partir de una ontología predefinida sobre atributos de calidad tales como disponibilidad, usabilidad, performance, modificabilidad, testeabilidad y seguridad.

Para comprobar la efectividad de nuestra propuesta, efectuamos una evaluación de la misma sobre dos casos de estudios, utilizando métricas de precisión, recall y tiempo de ejecución que nos permitieran analizar los resultados del QA Miner.

VI.1. Ventajas y desventajas

Luego de ejecutar la herramienta *QA Miner* sobre los distintos casos de estudio se identificaron diversas ventajas y desventajas de la técnica propuesta. A continuación se listan las ventajas encontradas, según diferentes criterios.

- Recall: el *Recall* de la herramienta mostró valores aceptables en el primer caso y muy buenos en el segundo. Los costos generados por la omisión de identificar un QA pueden llegar a ser muy altos. Por eso, se debe tratar de detectar la mayor cantidad de atributos de calidad posibles y, en consecuencia, conseguir el mayor *Recall* posible.

- Tiempo de ejecución: el análisis de cada aspecto mostró tiempos muy bajos, en ningún caso superando los 2 segundos. El número de casos de uso que atraviesa el aspecto tampoco demostró tener gran influencia sobre la duración del análisis, contabilizando prácticamente los mismos tiempos tanto con los aspectos que se relacionaban con pocos casos de uso, como los que se relacionaban con muchos.

- Nivel de automatización: la interacción con el analista es mínima, no siendo necesario que éste tenga que agregar o clasificar información durante el proceso. Además, el QA Miner es de fácil utilización. Este nivel de automatización y simpleza es deseable ya que facilita la tarea del analista, especialmente cuando se cuenta con un gran número de casos de uso.

- Extensión a otros documentos: la técnica propuesta puede ser de fácil adaptación para minar QAs de otros documentos basados en textos, como minutas, notas, entrevistas, etc. Es frecuente que, durante la elicitación de requerimientos, se produzcan una gran variedad de documentos. Con una adecuada selección de filtros y pocas modificaciones sobre *QA Miner*, se podría utilizar este enfoque para descubrir QAs sobre otros documentos textuales.

- Extensibilidad para la identificación de otros QAs: la ontología usada sólo contempla seis atributos de calidad. Sin embargo, este no es un impedimento teórico del enfoque propuesto. Extender *QA Miner* hacia la identificación de otros atributos de calidad sólo involucra su ingreso a la ontología.

A continuación se listan las desventajas encontradas, según distintos criterios.

- Dependencia de los aspectos encontrados: la técnica propuesta se basa en identificar atributos de calidad a partir de aspectos tempranos previamente detectados. Esta identificación de aspectos no es una tarea sencilla y, de hecho, hay varios enfoques para técnicas semiautomáticas para la identificación de EAs. Los aspectos deben estar correctamente formados, mediante palabras que los representen claramente. La herramienta Aspect Extractor Tool puede llegar a introducir “ruido” en los mismos, es decir, palabras que no se relacionan con los EAs identificados y, por lo tanto, dificultan la identificación de los QAs.

- Atributos de calidad no relacionados con aspectos tempranos: no todos los QAs de un sistema están siempre relacionados con un aspecto temprano. Por ejemplo, algunos QAs importantes provienen de los objetivos de negocio o directamente de los stakeholders. Estos QAs no serían detectados mediante este análisis.

- Definición de la ontología: la técnica requiere de una ontología como fuente de conocimiento. El modelamiento de la misma para cada atributo de calidad podría llegar a ser una tarea compleja. Incluso, aún con una ontología adecuada, puede haber casos en los cuales no sea posible, mediante la misma, la identificación de los QAs involucrados en un sistema. Adicionalmente, la técnica se encuentra limitada por el algoritmo que relaciona los tokens con las instancias de la ontología.

- Limitaciones del lenguaje: al procesar los casos de uso, la técnica posee las limitaciones propias del análisis basado en el procesamiento del lenguaje natural. Estas limitaciones son, entre otras, la sinonimia, la polisemia, la homografía, la ambigüedad, etc. Además, no todas las técnicas del contexto de la lingüística computacional, como sinónimos o etiquetado POS, han sido implementadas.

- Aprendizaje: *QA Miner* no prevé la interacción con un usuario o analista que pueda proveer cierta clase de feedback en base a los resultados obtenidos. Este tipo de información podría ser de ayuda para un mejor funcionamiento de la técnica.

VI.2. Trabajos futuros

La problemática de la identificación de atributos de calidad todavía es una tarea sin una solución óptima. Además, la técnica propuesta en este trabajo se basa en una idea que no ha sido demasiado explorada por los investigadores, por lo que hay varios puntos en los que se deberían profundizar. Los más relevantes son:

- Mejoramiento de la ontología: al ser la ontología una parte fundamental en el proceso, su mejora impacta directamente sobre el funcionamiento del mismo. La tarea de mejora cuenta con cuatro aristas. La primera, consiste en revisar los conceptos de la ontología, agregando nuevos o identificando nuevas relaciones sobre los existentes. La segunda, consiste en agregar, modificar y mejorar las instancias cargadas en la misma. La tercera, es incluir nuevos atributos de calidad y, por lo tanto, un conocimiento de esos atributos que permita ampliar el alcance de la técnica. La cuarta es implementar distintas estrategias de matching entre los tokens y la ontología.

- Atributos de los tokens: agregar más atributos a los tokens extraídos de los casos de uso puede agregar nuevas consideraciones a la técnica y, quizás, mejorar los resultados. Un filtro de “sinónimos” o uno de etiquetado POS podrían también mejorar los resultados.

- Minar QAs desde distintos tipos de documentos: la técnica propuesta pude ser extendida para el tratamiento de diversos documentos, en donde se consideren que se pueden identificar atributos de calidad de un sistema.

- Arquitecturas orientadas a aspectos: una vez terminado el trabajo se observó que la técnica propuesta podría ser de mucha utilidad en los casos en que se decida utilizar arquitecturas orientadas a aspectos. Ya que se tiene como resultado los aspectos de entrada, cada uno junto con el atributos de calidad relacionado, el arquitecto podría diseñar un modulo de la arquitectura por aspecto y aplicar a cada módulo las tácticas arquitectónicas que satisfagan el atributo de calidad relacionado.

Igualmente se debe mencionar que no necesariamente la identificación o análisis de los aspectos tempranos de un sistema conducen a arquitecturas orientadas a aspectos.

- Aprendizaje: sería de utilidad el desarrollo de alguna técnica de aprendizaje que pueda ser agregada para que la herramienta *QA Miner* pueda ser ajustada en base a interacciones con un analista. Es de esperar que, a partir de esta información provista, se puedan tener mejores resultados.

ANEXO I

En este anexo se describe el concepto de mapa y sus operaciones de una manera formal y detallada.

Definición de "mapa"

Se define un "mapa" como un conjunto de tuplas <K, V> , donde a K se lo denomina "clave" y a V se lo denomina "valor". Un mapa no puede tener dos tuplas con claves idénticas. Es decir, dado un mapa M no existe ningún par de tuplas <K1, V>, <K2, V2> tal que K1=K2.

En los mapas definidos en este trabajo se utiliza como "claves" los nombres de los atributos de calidad y como "valores" números reales entre 0 y 1.

Suma de dos mapas

Sean M1 y M2 dos mapas con igual número de tuplas, donde toda clave K que pertenece a alguna tupla de M1, también pertenece a alguna tupla de M2. Sea además cualquier valor de V perteneciente a M1 y M2 un número real. Definimos la suma M1+M2, como un mapa M3 en donde cada tupla que éste posee es el resultado de la suma de los valores de las tuplas de igual clave de M1 y M2.

Por ejemplo, si se tiene el mapa $M_1 = (\langle K_1, a\rangle; \langle K_2, b\rangle)$ y el mapa $M_2 = (\langle K_1, c\rangle; \langle K_2, d\rangle)$ se define $M_1 + M_2 = (\langle K_1, a + c\rangle; \langle K_2, b + d\rangle)$ siendo K1 y K2 claves; a, b, c y d números reales

División de un mapa por un número real

Sea M un mapa compuesto por un conjunto de tuplas <K, V> donde K es la clave y V son números reales. Sea x un número real distinto de cero. La división de M por x da como resultado el mismo mapa, con todos los valores V divididos por ese número.

Por ejemplo, si $M = (\langle K_1, a\rangle; \langle K_2, b\rangle)$ entonces $M/x = (\langle K_1, a/x\rangle; \langle K_2, b/x\rangle)$, siendo a, b y x números reales; M un mapa; K1, K2 claves del mapa.

Multiplicación de un mapa por un número real

Sea M un mapa compuesto por un conjunto de tuplas <K,V> donde K es la clave y V son números reales. Sea x un número real. La multiplicación de M por x da como resultado el mismo mapa, con todos los valores V multiplicados por ese número.

Por ejemplo, si $M = (\langle K_1, a\rangle; \langle K_2, b\rangle)$, entonces $M * x = (\langle K_1, a * x\rangle; \langle K_2, b * x\rangle)$, siendo a,b y x números reales; M un mapa; K1,K2 claves del mapa.

ANEXO II. Especificaciones de casos de uso completas

En este capítulo anexo, se describirán las especificaciones completas de los casos de uso que comprenden a los casos de estudio utilizados para efectuar la evaluación de la herramienta desarrollada en este trabajo.

HWS (Health Watcher System)

Nombre	[FR01] Query information
Descripción	This use case allows a citizen to perform queries. **Query Health Guide** The citizen might query: • Which health units take care of a specific specialty? • What are the specialties of a particular health unit? **Query Specialty Information** The citizen might query: ➢ Information about a complaint made by a citizen: ✓ Complaint details. ✓ Situation (OPENED, SUSPENDED, or CLOSED). ✓ Technical analysis. ✓ Analysis date. ✓ Employee that made the analysis. ➢ Information about diseases: ✓ Description. ✓ Symptoms. ✓ Duration.
Actor	❖ Citizen ❖ Employee

Disparador	-
Flujo Básico	1. The citizen chooses the type of query 1.1 In the case of query on specialties grouped by health units, the system retrieves the list of health units stored. 1.1.1 The system retrieves the details of each health unit such as its description and specialties. 1.1.2 The list of health units is presented to the user on their local display. 1.2 In the case of a query on health units grouped by specialties, the system retrieves the list of registered specialties. 1.2.1 The system retrieves the details of each specialty such as its unique identifier and name. 1.2.2 The list of specialties is presented to the user on their local display. 1.3 In the case of a query on diseases, the system retrieves the list of diseases. 1.3.1 The system retrieves the details of each disease type such as its unique identifier and name. 1.3.2 The list of disease is presented to the user on their local display. 2. The citizen provides the data for the query 2.1 In the case of a query on specialties grouped by health units; the citizen selects the health unit to be queried. 2.1.1 A unique identifier representing the selected health unit is sent to the server. 2.1.2 The system ensures the health unit information is consistent. 2.1.3 The unique identifier is used by the system to search the repository for the selected health unit. 2.1.4 The details of the selected health unit are retrieved including its specialties. 2.1.5 The specialties for the selected health unit are returned to the user. 2.2 In the case of a query on health units grouped by specialties, the citizen selects the specialty to be queried. 2.2.1 A unique identifier representing the selected specialty is sent to the server. 2.2.2 The system ensures the health unit information is consistent. 2.2.3 The unique identifier is used to retrieve the list of health units which are associated with the selected specialty.

	2.2.4 The details of the health units and specialties are retrieved. 2.2.5 The retrieved health units are returned to the user. 2.3 In the case of a query on complaints, the citizen provides the complaint code. 2.3.1 The unique identifier representing the complaint to be retrieved is sent to the server. 2.3.2 The system ensures the complaint information is consistent. 2.3.3 The unique identifier is used to retrieve the complaint entry. 2.3.4 The system must determine the complaint type as to retrieve the appropriate information. 2.3.4.1 If the complaint is a special complaint the complainer's age, education level and occupation are retrieved (in addition to the standard complaint information). 2.3.4.2 If the complaint is a food complaint the meal which was consumed, the number of people who ate the meal, the number of sick people, etc. are retrieved (in addition to the standard complaint information). 2.3.4.3 If the complaint is an animal complaint the animal species and the number of animals affected (in addition to the standard complaint information). 2.3.5 The complaint with all the appropriate information is returned to the user. 2.4 In the case of a query on diseases, the citizen selects the disease to be queried. 2.4.1 The unique identifier representing the disease type to be retrieved is sent to the server. 2.4.2 The system ensures the disease type information is consistent. 2.4.3 The unique identifier is used to retrieve the disease type to query. 2.4.4 The symptoms for the selected disease type are retrieved. 2.4.5 The complete disease information is returned to the user. 3. The query results are formatted and presented to the user on their local display.
Flujos Alternativos	1.x and 2.x: A communication problem occurs. 1. Raise an error message 1.x.1 and 2.x.4: A problem occurs retrieving the complaint data. 1. The system retrieves the available information.

	2. Raise an error message 2.3.3: An invalid complaint code is entered. 1. Raise an error message informing the user the complaint does not exist. 2.x.2: Consistent data cannot be assured. 1. The system abandons the data retrieval. 2. Raise an error message.
Precondiciones	The data to be queried must be registered on the system
Poscondiciones	The query result to the citizen
Requerimientos especiales	-
Prioridad	Important
Cantidad de palabras	776 palabras

Nombre	**[FR02] Complaint Specification**
Descripción	This use case allows a citizen to register complaints. Complaints can be: **Animal Complaint – DVA** • Sick animals. • Infestations (rodents, scorpions, bats, etc.) • Diseases related to mosquitoes (dengue, filariose). • Animal maltreatment. **Food Complaint - DVISA** • Cases where there is a suspicion infected food being eaten. **Special Complaint - DVISA** • Cases related to several reasons, which are not mentioned above (restaurants with hygiene problems, leaking sewerage, suspicious water transporting trucks, etc.). The three kinds of complaints have the following information in common: • Complaint data: description (mandatory) and observations (optional); • Complainer data: name, street, complement, district, city, state/province, zip

	code, telephone number and e-mail. All these fields are optional. • Complaint state (mandatory), which might be: OPENED, SUSPENDED or CLOSED. When a complaint is first registers its state must be OPENED. • The system must register the complaint registration date. In addition to the above data, each complaint type has its own specific data, including: **Animal Complaint – DVA** • Type of animal (mandatory), amount of animals (mandatory), date problem was observed (mandatory). • Problem location data: street, complement, district, city, state/province, zip code and telephone number. All of these fields are optional. Food Complaint - DVISA • Victim's name (mandatory). • Victim's data: street, complement, district, city (or closest one), state/province, zip code and telephone number. All of these fields are optional. • Amount of people who ate the food, amount of sick people, amount of people who were sent to a hospital and amount of deceased people. All mandatory. • Location where the patients were treated, suspicious meal. All optional. **Special Complaint - DVISA** • Age (mandatory), academic qualifications (optional), occupation (optional). • Street, complement, district, city, state/province, zip code and telephone number of the closest location to the complaint location. All optional.
Actor	❖ Citizen ❖ Employee
Disparador	
Flujo Básico	1. The citizen selects the kind of complaint. 2. The system shows the specific screen for each type of complaint. 3. The system registers the kind, date and time of the complaints. 4. The citizen provides the complaint specific data.

	5. The system saves the complaint. 5.1. The information entered by the user is sent to the server. 5.2. The system parses the data entered by the user. 5.3. The system creates a new instance of the appropriate complaint type. 5.4 The system generates a unique identifier and assigns this to the new complaint. 5.5. The complainers address is parsed and saved. 5.6. The common complaint information is parsed and stored with the OPENED state. 5.7. The specific complaint data is then extracted and stored accordingly. 5.8. The system ensures the data is left in a consistent state. 6. The unique identifier is returned and presented to the user on their local display.
Flujos Alternativos	5.1: A communication problem occurs. 1. Raise an error message. 5.2: Invalid data is entered by the user. 1. Raise an error message. 5.5-5.7: A problem occurs storing the complaint. 1. The complaint entry is rolled-back. 2. Raise an error message. 5.8: Data consistency cannot be ensured. 1. The complaint entry is rolled-back. 2. Raise an error message.
Precondiciones	None
Poscondiciones	The complaint saved on the system
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	525 palabras

Nombre	[FR10] Login
Descripción	This use case allows an employee to have access to restricted operations on the Health-Watcher System
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee provides the login and password. 2. The login and password are sent to the server. 3. The system retrieves the employee details using the login as a unique identifier. 4. The system validates the entered password. 5. The result of the login attempt is presented to the employee on their local display.
Flujos Alternativos	2: A communication error occurs. 1. Raise an error message. 3: A problem occurs retrieving the employee details. 1. Raise an error message. 4: The system cannot validate the employee. 1. Raise an error message.
Precondiciones	None
Poscondiciones	Password validated by the system
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	117 palabras

Nombre	[FR11] Register Tables
Descripción	This use case allows the registration of system tables. The following operations are possible: insert, update, delete, search and print. The available tables include: • Health unit (unit code, unit description). • Specialty (code and description). • Health unit / Specialty (health unit and specialty). • Employee (login, name and password).

	• Type of disease (code, name, description, symptom and duration). • Symptom (code and description). • Type of disease / Symptom (type of disease and symptom).
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee chooses the option to register (insert/update) in one of the tables. 2. The employee enters the data. 3. The system saves the data
Flujos Alternativos	
Precondiciones	Verified employee
Poscondiciones	Updated data on the tables
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	116 palabras

Nombre	**[FR12] Update complaint**
Descripción	This use case allows the state of a complaint to be updated.
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee selects the update complaint option. 2. The system retrieves the list of all registered complaints. 2.1. The complaint list is populated with general and complaint type specific data. 3. The list of complaints is returned to the employee. 4. The complaints are formatted and presented to the employee on their local display. 5. The employee selects the complaint they wish to update. 6. The complaint unique identifier is sent to the server. 7. The system ensures the complaint data is consistent. 8. The system retrieves the complaint entry. 9. The complaint is returned to the employee.

	10. The complaint is formatted and presented to the employee on their local display. 11. The employee enters the conclusion. 12. The conclusion is sent to the server. 13. The complaint status is set to close; the date the conclusion was entered is set in addition to the employee who dealt with the complaint. 14. The system ensures the complaint is left in a consistent state. 15. The complaint information is updated to store the new information.
Flujos Alternativos	2 and 8: An error occurs retrieving the registered complaints. 2. Raise an error message. 7 and 14: Data consistency cannot be ensured. 1. The complaint changes are rolled-back. 2. Raise an error message. 3, 6, 9, and 12: A communication error occurs. 1. Raise an error message. 15: An error occurs storing the updated complaint. 1. The complaint changes are rolled back. 2. Raise an error message.
Precondiciones	The complaint must be registered and have the OPENED state. Verified employee.
Poscondiciones	Complaint updated and with state CLOSED.
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	280 palabras

Nombre	[FR13] Register new employee
Descripción	This use case allows new employees to be registered on the system.
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee selects the insert employee option. 2. The employee provides the following information about the new employee: ✓ Name ✓ Login ID ✓ Password (with second password field for confirmation). 3. The employee confirms the operation. 4. The entered data is transmitted to the server. 5. The system verifies the entered data. 6. The system ensures employee data is consistent. 7. The system saves the new employee's data.
Flujos Alternativos	2: Incomplete data entered. 1. Show a message informing the employee of the missing/incorrect data. 4: A communication error occurs. 1. Raise an error message. 5: The employee is already entered. 1. Inform the employee that the new employee is already entered. 2. Abandon the entry. 6: Data Consistency cannot be ensured. 1. The employee entry is rolled-back. 2. The employee is informed the employee cannot be inserted. 7: An error occurs storing the new employee’s details. 1. The employee entry is rolled-back. 2. Raise an error message.

Precondiciones	Verified employee.
Poscondiciones	New employee registered on the system
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	185 palabras

Nombre	**[FR14] Update employee**
Descripción	This use case allows of the employee's data to be updated on the system.
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee chooses the update employee option. 2. The employee provides the data to be updated: ✓ Name ✓ New password (with second password field for confirmation) ✓ Current password 3. The employee confirms the update. 4. The entered data is sent to the server. 5. The system verifies the entered data. 6. The system ensures the employee data is consistent. 7. The system stores the updated employee information.
Flujos Alternativos	On step 3, in case the name or the current password is missing/invalid, an error message should be showed.
Precondiciones	Verified employee.
Poscondiciones	Employee's data updated on the system
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	117 palabras.

Nombre	[FR15] Update health unit
Descripción	This use case allows the health unit's data to be updated.
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee chooses the update health unit option. 2. The system retrieves the list of all health units. 3. The list of health units is returned to the employee. 4. The list of health units is formatted and displayed on the employee's local display. 5. The employee selects the health unit to be updated. 6. The unique identifier for the selected health unit is sent to the server. 7. The system ensures the health unit data is consistent. 8. The system retrieves the data for the selected health unit. 9. The data retrieved is returned to the employee. 10. The health unit data is formatted and presented on the employee's local display. 11. The employee alters the necessary data. 12. The updated information is sent to the server. 13. The system ensures the health unit data is left in a consistent state. 14. The system stores the updated health unit information.
Flujos Alternativos	2, 8: A problem occurs retrieving the health unit information. 1. Raise an error message. 3, 6, 9, 12: A communication problem occurs. 1. Raise an error message. 7 and 13: Data consistency cannot be assured. 1. Any health unit updates are rolled-back. 2. Raise an error message. 14: A problem occurs storing the updated health unit data. 1. Any health unit updates are rolled-back.

	2. Raise an error message.
Precondiciones	Verified employee.
Poscondiciones	Health unit's data updated on the system.
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	245 palabras

Nombre	**[FR16] Change logged employee**
Descripción	This use case allows the currently logged employee to be changed.
Actor	❖ Employee
Disparador	
Flujo Básico	1. The employee chooses the change logged employee option. 2. The system shows the login screen, and from this point on, the flow will follow the one described in [Login.FR10].
Flujos Alternativos	
Precondiciones	Verified employee.
Poscondiciones	• First employee signed out and new employee logged-in.
Requerimientos especiales	
Prioridad	Essential
Cantidad de palabras	59 palabras

Sistema CRS (Course Registration System)

Nombre	[UC1] Login
Descripción	This use case describes how a user logs into the Course Registration System
Actor	❖ Student ❖ Professor ❖ Registrar
Disparador	The use case begins when the actor types his/her name and password on the login form
Flujo Básico	**Login** 1. The system validates the actor's password and logs him/her into the system. 2. The system displays the Main Form and the use case ends.
Flujos Alternativos	*Invalid Name / Password* If in the basic flow the system cannot find the name or the password is invalid, an error message is displayed. The actor can type in a new name or password or choose to cancel the operation, at which point the use case ends.
Precondiciones	
Poscondiciones	
Requerimientos especiales	
Prioridad	High
Cantidad de palabras	112 palabras

Nombre	[UC2] View Report Card
Descripción	This use case allows a Student to view his/her report card for the previously completed semester
Actor	❖ Student
Disparador	The use case begins when the Student selects the "view report card" activity from the Main Form
Flujo Básico	**View Report Card** 1. The system retrieves the grade information for each of the courses the Student completed during the previous semester. 2. The system prepares, formats, and displays the grade information. 3. When Student is finished viewing the grade information the Student selects "close."

Flujos Alternativos	*No Grade Information Available* If in the basic flow the system cannot find any grade information from the previous semester for the Student, a message is displayed. Once the Student acknowledges the message the use case terminates.
Precondiciones	Log In: Before this use case begins the Student has logged onto the system
Poscondiciones	
Requerimientos especiales	
Prioridad	High
Cantidad de palabras	136 palabras

Nombre	**[UC3] Register for Courses**
Descripción	This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.
Actor	❖ Student
Disparador	The use case begins when the Student selects the "maintain schedule" activity from the Main Form
Flujo Básico	**Create a Schedule** 1. The Student selects "create schedule." 2. The system displays a blank schedule form. 3. The system retrieves a list of available course offerings from the Course Catalog System. 4. The Student selects 4 primary course offerings and 2 alternate course offerings from the list of available offerings. Once the selections are complete the Student selects "submit." 5. The "Add Course Offering" sub-flow is performed at this step for each selected course offering. 6. The system saves the schedule.
Flujos Alternativos	1 *Modify a Schedule* 1. The Student selects "modify schedule." 2. The system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester). 3. The system retrieves a list of all the course offerings available for the current semester from the Course Catalog System. The system displays the list to the Student. 4. The Student can then modify the course selections by deleting and adding new courses. The Student selects the courses to add from the list of available courses. The Student also selects any course offerings to delete from the existing schedule. Once the edits are complete the

Student selects "submit".

5. The "Add Course Offering" sub-flow is performed at this step for each selected course offering.
6. The system saves the schedule.

2 *Delete a Schedule*

1. The Student selects the "delete schedule" activity.
2. The system retrieves and displays the Student current schedule.
3. The Student selects "delete."
4. The system prompts the Student to verify the deletion.
5. The Student verifies the deletion.
6. The system deletes the schedule.

3 *Save a Schedule*

At any point, the Student may choose to save a schedule without submitting it by selecting "save". The current schedule is saved, but the student is not added to any of the selected course offerings. The course offerings are marked as "selected" in the schedule.

4 *Add Course Offering*

The system verifies that the Student has the necessary prerequisites and that the course offering is open. The system then adds the Student to the selected course offering. The course offering is marked as "enrolled in" in the schedule.

5 *Unfulfilled Prerequisites or Course Full*

If in the "Add Course" sub-flow the system determines that the Student has not satisfied the necessary prerequisites or that the selected course offering is full, an error message is displayed. The Student can either select a different course offering or cancel the operation, at which point the use case is restarted.

6 *No Schedule Found*

If in the "Modify a Schedule" or "Delete a Schedule" sub-flows the system is unable to retrieve the Student's schedule, an error message is displayed. The Student acknowledges the error and the use case is restarted.

7 *Course Catalog System Unavailable*

If, the system is unable to communicate with the Course Catalog System after a specified number of tries, the system will display an error message to the Student. The Student acknowledges the error message and the use case terminates.

8 *Course Registration Closed*

If, when the student selects "maintain schedule", registration for the current semester has been closed, a message is displayed to the Student and the use case terminates.

	Students cannot register for courses after registration for the current semester has been closed.
Precondiciones	Log In: Before this use case begins the Student has logged onto the system
Poscondiciones	
Requerimientos especiales	
Prioridad	High
Cantidad de palabras	629 palabras

Nombre	**[UC4] Select Courses to Teach**
Descripción	This use case allows a professor to select the course offerings (date- and time- specific courses will be given) from the course catalog for the courses that he/she is eligible for and wishes to teach in the upcoming semester. The actor starting this use case is the Professor. The Course Catalog System is an actor within the use case.
Actor	❖ Proffesor
Disparador	The use case begins when the professor selects the "select courses to teach" activity from the Main Form.
Flujo Básico	**Select Courses to Teach** 1. The system retrieves and displays the list of course offerings the professor is eligible to teach for the current semester. The system also retrieves and displays the list of courses the professor has previously selected to teach. 2. The professor selects and/or de-selects the course offerings that he/she wishes to teach for the upcoming semester. 3. The system removes the professor from teaching the de-selected course offerings. 4. The system verifies that the selected offerings do not conflict (i.e., have the same dates and times) with each other or any offerings the professor has previously signed up to teach. If there is no conflict, the system updates the course offering information for each offering the professor selects.
Flujos Alternativos	1 *No Courses Available* *If in the basic flow the professor is not eligible to teach any courses in the upcoming semester the system will display an error message. The professor acknowledges the*

	message and the use case ends. 2 *Schedule Conflict* *If the systems find a schedule conflict when trying to establish the course offerings the Professor should take, the system will display an error message indicating that a schedule conflict has occurred. The system will also indicate which are the conflicting courses. The professor can either resolve the schedule conflict (i.e., by canceling his selection to teach one of the course offerings) or cancel the operation, in which case any selections will be lost and the use case ends.* 3 *Course Registration Closed* If, when the Professor selects "select courses to teach", registration for the current semester has been closed, a message is displayed to the Professor and the use case terminates. Professors cannot change the course offerings they teach after registration for the current semester has been closed. If a professor change is needed after registration has been closed, it is handled outside the scope of this system.
Precondiciones	Log In: Before this use case begins the Proffesor has logged onto the system
Poscondiciones	
Requerimientos especiales	
Prioridad	High
Cantidad de palabras	412 palabras

Nombre	**[UC5] Submit Grades**
Descripción	This use case allows a Professor to submit student grades for one or more classes completed in the previous semester.
Actor	❖ Professor
Disparador	The use case begins when the Professor selects the "submit grades" activity from the Main Form
Flujo Básico	Submit Grades 1. The system displays a list of course offerings the Professor taught in the previous semester. 2. The Professor selects a course offering. 3. The system retrieves a list of all students who were registered for the course offering. The system also retrieves the grade information for each student in the offering. 4. The system displays each student and any grade that was previously assigned for the offering. 5. For each student on the list, the Professor enters a grade: A, B,C, D, F,

	or I. The system records thestudent's grade for the course offering. If the Professor wishes to skip a particular student, the grade information can be left blank and filled in at a later time. The Professor may also change the grade for a student by entering a new grade.
Flujos Alternativos	*1. No Courses Taught:* If in the basic flow, the Professor did not teach any course offerings in the previous semester the system displays an error message and the use caseends. *2. Course Cancelled:* If too many students withdrew from the course during the add/drop period and the course was cancelled after the beginning of the semester, the system displays an error message. If the Professor chooses to cancel the operation the use case terminates, otherwise is restarted at step 2 of the basic flow.
Precondiciones	Log In: Before this use case begins the Professor has logged onto the system.
Poscondiciones	
Requerimientos especiales	
Prioridad	High.
Cantidad de palabras	276 palabras

Nombre	**[UC6] Maintain Professor Information**
Descripción	This use case allows the Registrar to maintain profesor information in the registration system. This includes adding, modifying, and deleting professors from the system.
Actor	❖ Registrar
Disparador	The use case begins when the Registrar selects the "maintain professor" activity from the Main Form
Flujo Básico	Add a Professor: 1. The Registrar selects "add a professor." 2.The system displays a blank professor form. 3. The Registrar enters the following information for the professor: name, date of birth, social security number, status, and department. 4. The system validates the data to insure the proper data format and searches for an existing professor with the specified name. If the data is valid the system creates a new professor and assigns a unique system-generated id number. This number is displayed, so it can be used for subsequent uses of the system. 5. Steps 2-4 are repeated for each professor added to the system. When the Registrar is finished adding professors to the system the use case ends.
Flujos Alternativos	Delete a Professor: 1. The Registrar selects "Delete a Professor." 2. The system displays a blank professor form. 3. The Registrar types in the professor id number for the professor that's being deleted.

	4. The system retrieves the professor and displays the professor information in the form. 5. The Registrar selects "delete." 6. The system displays a delete verification dialog confirming the deletion. 7. The Registrar selects "yes." 8. The professor is deleted from the system. 9. Steps 2-8 are repeated for each professor the Registrar wants to modify. When the Registrar is finished deleting professors from the system, the use case ends. Professor Already Exists: If in the "Add a Professor" sub-flow, a professor already exists with the specified name, an error message, "Professor Already Exists", is displayed. The Registrar can either change the name, choose to create another professor with the same name, or cancel the operation at which point the use case ends. Professor Not Found: If in the "Modify a Professor" sub-flow or "Delete a Professor" sub-flow, a professor with the specified id number does not exist, the system displays an error message, "Professor Not Found". Then the Registrar can type in a different id number or cancel the operation at which point the use case ends.
Precondiciones	Log In: Before this use case begins the Registrar has logged onto the system.
Poscondiciones	
Requerimientos especiales	
Prioridad	Low
Cantidad de palabras	386

Nombre	**[UC7] Maintain Student Information**
Descripción	This use case allows the Registrar to maintain student information in the registration system. This includes adding, modifying, and deleting students from the system.
Actor	❖ Registrar
Disparador	The use case begins when the Registrar selects the "maintain professor" activity from the Main Form.
Flujo Básico	Add Student: 1. The Registrar selects "add student." 2. The system displays a blank student form. 3. The Registrar enters the following information for the student: name, date of birth, social security number, status, and graduation date. 4. The system validates the data to insure the proper format and searches for an existing student with the specified name. If the data is valid the system creates a new student and assigns a unique system-generated id number. 5. Steps 2-4 are repeated for each student added to the system. When the Registrar is finished adding students to the system the use case ends.

Flujos Alternativos	Modify a Student: 1. The Registrar selects "modify student." 2. The system displays a blank student form. 3. The Registrar types in the student id number he/she wishes to modify. 4. The system retrieves the student information and displays it on the screen. 5. The Registrar modifies one or more of the student information fields: name, date of birth, social security number, student id number, status, and graduation date. 6. When changes are complete, the Registrar selects "save." 7. The system updates the student information. 8. Steps 2-7 are repeated for each student the Registrar wants to modify. When edits are complete, the use case ends. Delete a Student: 1. The Registrar selects "delete student." 2.The system displays a blank student form. 3. The Registrar types inthe student id number for the student that's being deleted. 4. The system retrieves the student and displays the student information in the form. 5. The Registrar selects "delete." 6. The system displays a delete verification dialog confirming the deletion. 7. The Registrar selects "yes." 8. The student is deleted from the system. 9. Steps 2-8 are repeated for each student deleted from the system. When the Registrar is finished deleting students to the system the use case ends. Student Already Exists: If in the "Add a Student" sub-flow the system finds an existing student with the same name an error message is displayed "Student Already Exists". The Registrar can change the name, create a new student with the same name, or cancel the operation at which point the use case ends. Student Not Found: If in the "Modify a Student" or "Delete a Student" sub-Flows the student name is not located, the system displays an error message, "Student Not Found". The Registrar can then type in a different id number or cancel the operation at which point the use case ends.
Precondiciones	Log In: Before this use case begins the Registrar has logged onto the system.
Poscondiciones	
Requerimientos especiales	
Prioridad	Low
Cantidad de palabras	471 palabras

Nombre	[UC8] Close Registration
Descripción	This use case allows a Registrar to close the registration process. Offerings that do not have enough students are cancelled. Course must have a minimum of three students in them. The billing system is notified for each student in each course offering that is not cancelled, so the student can be billed for the course offering. The main actor of this use case is the Registrar. The Billing System is an actor involved within this use case.
Actor	❖ Registrar
Disparador	The use case begins when the Registrar selects the "close registration" activity from the Main Form.
Flujo Básico	Successful Close Registration: The system checks to see if a Registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress. For each open course offering, the system checks if three students have registered and a professor has signed up to teach the course offering. If so, the system closes the course offering and sends a transaction to the billing system for each student enrolled in the course offering.
Flujos Alternativos	Less Than Three Students in the Course Offering: If in the basic flow less than three students signed up for the course offering, the system will cancel the course offering. The Cancel Course Offering sub-flow is executed at this point. Cancel Course Offering: The system cancels the course offering. For each student enrolled in the cancelled course offering, system will modify the student's schedule. The first available alternate course selection will be substituted for the cancelled course offering. If no alternates are available, then no substitution will be made. Control returns to the Main flow to process the next course offering for the semester. Once all schedules have been processed for the current semester, the system will notify all students, by mail, of any changes to their schedule (e.g., cancellation or substitution). No Professor for the Course Offering: If in the basic flow there is no professor signed up to teach the course offering, the system will cancel the course offering. The Cancel Course Offering sub-flow is executed at this point. Billing System Unavailable: If the system is unable to communicate with the Billing System, the system will attempt to re-send the request after a specified period. The system will continue to

	attempt to re-send until the Billing System becomes available7: An error occurs storing the new employee's details.
Precondiciones	Login: The Registrar must be logged onto the system in order for this use case to begin.
Poscondiciones	
Requerimientos especiales	
Prioridad	High
Cantidad de palabras	425 palabras

Bibliografía

1. **Bass, Clements and Kazman.** *Software Architecture in practice.* s.l. : Addison-Wesley, 1998.

2. **Kazman, Clements and Kein.** *Evaluating Software Architectures. Methods and case studies.* s.l. : Addison Wesley, 2001.

3. **Barbacci, et al.** *Quality Attributes.* 1995. Technical Report. CMU/SEI-95-TR-021 ESC-TR-95-021.

4. **Zou, Joe and Pavlovski, Christopher.** Modeling Architectural Non Functional Requirements: From Use Case to Control Case. *IEEE International Conference on e-Business Engineering (ICEBE'06).* Octubre 2006.

5. **Hazeem Al Balushi, Taiseera, et al.** ElicitO: A Quality Ontology-Guided NFR Elicitation Tool. 2007.

6. **Casamayor, Agustin, Godoy, Daniela and Campo, Marcelo.** Identification of non-functional requirements in textual specifications:A semi-supervised learning approach. *Information and Software Technology.* 2010. Vol. 52, pp. 436-445.

7. **Noy, Natalya and McGuinness, Deborah.** Ontology Development 101: A Guide to Creating Your First Ontology Escenario. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.* Marzo 2001.

8. *Identificacion Temprana de Aspectos.* **Haak, B., Pryor, A. and Marcos, C.** 2005, Revista SCC (Workshop in SE), Vol. 6.

9. **Haimei, Zhang and Kerong, Ben.** Architectural Design of the Health Watch System with an Integrated Aspect. *201O International Conference On Computer Design And Applications.* 2010.

10. **Kutsick, Alex.** Course Registration System Supplementary - Version 2003. 2003.

11. **Gamble, S.** Course Registration System Version 1.0. Febrero 19, 1999.

12. **Baeza-Yates, R. and Ribeiro-Neto, B.** Modern Information Retrieval. s.l. : ACM Press, Addison-Wesley, 199.

13. **Clements, et al.** A Practical Method for Documenting Software Architectures. *Carnegie Mellon University.* Pittsburgh, Pennsylvania, USA : s.n., 2002.

14. **Reynoso, Carlos.** *Introducción a la Arquitectura de Software Version 1.0.* s.l. : Universidad de Buenos Aires, 2004.

15. *Survey of Architecture Description Languages.* **Clements, Paul.** 1996. Proceedings of the International Workshop on Software Specification and Design.

16. **Parnas, D. L.** On the criteria to be used in decomposing systems into modules. *Communications of the ACM.* Diciembre 1972. Vol. 15, pp. 1053-1058.

17. **Buschmann, et al.** *Pattern-Oriented Software Architecture: A System Of Patterns.* West Sus- : John Wiley & Sons, 1996.

18. *IEEE Standard 1061-1992. Standard for a Software Quality Metrics.* IEEE. 1992. Standard.

19. **Bass, Clements and Kazman.** *Software Architecture in Practice, Second Edition.* s.l. : Addison-Wesley, 2003.

20. **Berenbach, Paulish and Kazmei.** *Software & Systems Requirements Engineering: in practice.* s.l. : McGraw Hill Professional, 2009.

21. *Identifying Crosscutting Concerns in Requirements Specifications.* **Rosenhainer, L.** Vancouver : s.n., 2004. Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design , held in Function with OOPSLA.

22. **Sampaio, A.** EA-Miner: a Tool for Automating Aspect-Oriented Requirements Identification. *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering.* 2005.

23. **Baniassan, et al.** Discovering early aspects. *IEEE Software.* 2001. Vol. 23, 1, pp. 61-70.

24. **Cleland-Huang, Jane, et al.** Automated classification of non-functional requirements. Londres, UK : Springer-Verlag, Abril 2007. Vol. 12, 2, pp. 103-120.

25. IEEE 830: IEEE Recommended Practice for Software Requirements Specifications. 1998.

26. [Online] http://es.wikipedia.org/wiki/Caso_de_uso.

27. **Dörr, J., et al.** Eliciting Efficiency Requirements with Use Cases. *9th International Workshop on Requirements Engineering. Foundation for Software Quality, REFSQ '03. Pre-Proceedings.* 2003. Vol. 3, pp. 23-32.

28. **Moreira, Ana, Araujo, Joao and Brito, Isabel.** Crosscutting quality attributes for requirements engineering. *Proceedings of the 14th international conference on Software engineering and knowledge engineering.* 2002. Vol. 27, pp. 167-174 .

29. **Chung, L., Mylopoulos, J. and Nixon, B.** Representing and using nonfunctional requirements: a process-oriented approach. Junio 1992. Vol. 18, pp. 483 - 497. 0098-5589.

30. **Malan, M. and Bredemeyer, D.** Defininig Non-Functional Requirements. *http://www.bredemeyer.com/papers.htm.*

31. **Park, Sooyong, et al.** Using classification techniques for informal requirements. *Information and Software Technolog.* 2007. Vol. 49, pp. 1128-1140.

32. **Brill, E.** Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics.* Diciembre 1995. Vol. 7, pp. 543 -565.

33. Fundacion Eclipse. [Online] http://www.eclipse.org/.

34. *Early Aspect Identification from Use Cases using NLP and WSD Techniques.* **Rago, et al.** 2009. EA '09 Proceedings of the 15th workshop on Early aspects.

35. *UML Semantics version 1.1.* Rational Software Corporation. 1997.

36. [Online] http://es.wikipedia.org/wiki/Stemming.

37. Official home page for distribution of the Porter Stemming Algorithm. [Online] http://tartarus.org/~martin/PorterStemmer/.

38. OWL Web Ontology Language - Use Cases and Requirements. *W3C Recommendation .* [Online] 2004. http://www.w3.org/TR/webont-req/.

39. **Gruber, T. R.** A translation approach to portable ontology specfications. Junio 1993. Vol. 5, 2, pp. 199-220.

40. **Pinto, Monica, Gámez, Nadia and Fuentes, Lidia.** Towards the architectural definition of the Health Watcher system with AO-ADL. *Proceedings of the Early Aspects at ICSE: Workshops in Aspect-Oriented Requirements Engineering and Architecture Design.* 2007.

41. **Massoni, Tiago, Soares, Sérgio and Borba, Paulo.** Healt Watcher Requirements Document V_2. 2006.

42. **Shaw and Garlan.** *An introduction to software architecture.* CMU Software Engineering Institute Technical Report. 1994. CMU/SEI-94-TR-21, ESC-TR-94-21..

43. **Saleh, A. and Al-Zarouni, A.** Capturing Non-Functional Software Requirements Using the User Requirements Notation. *International Research Conference on Innovations in Information Technology (IIT2004), Dubai UAE. .* 2004.